software

MDPI

Article

Analysis of Faults in Software Systems Using Tsallis Distribution: A Unified Approach

Shachi Sharma

Department of Computer Science, South Asian University, New Delhi 110021, India; shachi@sau.int

Abstract: The identification of the appropriate distribution of faults is important for ensuring the reliability of a software system and its maintenance. It has been observed that different distributions explain faults in different types of software. Faults in large and complex software systems are best represented by Pareto distribution, whereas Weibull distribution fits enterprise software well. An analysis of faults in open-source software endorses generalized Pareto distribution. This paper presents a model, called the Tsallis distribution, derived using the maximum-entropy principle, which explains faults in many diverse software systems. The effectiveness of Tsallis distribution is ascertained by carrying out experiments on many real data sets from enterprise and open-source software systems. It is found that Tsallis distribution describes software faults better and more precisely than Weibull and generalized Pareto distributions, in both cases. The applications of the Tsallis distribution in (i) software fault-prediction using the Bayesian inference method, and (ii) the Goal and Okumoto software-reliability model, are discussed.

Keywords: software systems; fault distribution; maximum-entropy principle; Tsallis entropy; Pareto distribution; Weibull distribution; software reliability

Citation: Sharma, S. Analysis of Faults in Software Systems Using Tsallis Distribution: A Unified Approach. *Software* **2022**, *1*, 473–484. https://doi.org/10.3390/software1040020

Academic Editors: Junjun Zheng and Xiaoyi Zhang

Received: 20 September 2022
Accepted: 8 November 2022
Published: 11 November 2022

1. Introduction

The study of faults in software systems is important as it has a direct impact not only in determining its quality and reliability but also on overall management. The terms fault, bug, and defect are used interchangeably in the field of software engineering. Software fault is a problem leading either to its crash or undesirable output [1]. Thus, what should be considered as a fault varies and is primarily dependent on the requirements and standards of that software product. No software is free from faults, and hence their analysis has been an active area of research in software engineering [2]. One important aspect relates to the probability distribution of faults over modules in a software system, known as fault distribution. The module can be a class in an object-oriented system, a function in procedural languages, or a file in Python. The identification of fault distribution helps in prioritizing and analyzing those modules that have more impact on the overall quality of the software system. Additionally, the knowledge of the fault distribution facilitates developers to identify error-prone modules as early as possible while developing software in order to optimize resources and effective testing [3]. Moreover, this also helps in reducing software management costs post delivery because fixing a fault is far more economical during the earlier phases of the software development life cycle [4,5]. Knowledge of the underlying probability distribution of faults can be used to predict faults in future releases of the software as well [6].

1.1. Motivation

Finding an appropriate mathematical model that explains the empirical data of faults in a software system has a long history [3]. Some earlier investigations of faults in software systems favored exponential and logistic models. Later, the Pareto model became widely accepted, particularly in large [7] and complex software [8,9]. This essentially means

that 80% of the faults are contained in 20% of the modules. A replicated study establishes that the Weibull model describes software faults more effectively than the Pareto model [10]. However, another such analysis on complex software systems endorses the Pareto model [11]. The application of the bounded generalized Pareto model is evaluated in [3] for open-source software. The feasibility of the variant of Pareto models such as the double Pareto has also been explored [12].

In spite of the availability of numerous models for fault distributions, there is no single accepted model that can explain faults in a variety of software. It has been asserted that the distribution of faults in software systems depends on the environment and that only the replicated studies can validate a model [13]. Hence, there is a clear requirement for a generic mathematical model that can describe the underlying fault distribution across a variety of software systems viz. enterprise, open source, large, and complex. Such a model will eliminate the existing diversity in the domain of analysis of software faults and will help the software developers and management to focus their efforts on quality assurance rather than investing time in finding appropriate models.

1.2. Contributions

The paper makes the following contributions:

- A generalized mathematical model, called Tsallis distribution, is derived using the maximum-entropy principle.
- Tsallis distribution is fit to fault data sets of enterprise and open-source software, and it is found to be a generic model.
- Applications of the Tsallis distribution in software fault-prediction and the software-reliability model are also outlined.

The paper is organized into six sections. The related work on fault distributions is presented in Section 2. The methodology of the study along with the data sets is described in Section 3. The Tsallis distribution is derived using a maximum entropy framework in this section. Additionally, a procedure to estimate the parameters of the Tsallis distribution is developed. Section 4 contains the results of the experiments conducted on real data sets to validate the efficacy of the Tsallis distribution in describing software faults. The discussion of the results and the applications of the analysis is also presented. Threats to the validity of the analysis are discussed in Section 5. The last Section 6 contains the conclusion.

2. Related Work

The study of the distribution of faults has been a topic of importance in software engineering, primarily because of its role in predicting faults and hence in ensuring the quality and reliability of software [14]. One of the techniques adopted for software fault-prediction is Probit regression, which is a binary classification model [15]. Harter et al. [16] have applied Probit analysis to study the severity of faults with respect to the software improvement process. Many other research studies have a consensus on the applicability of the Pareto principle in explaining faults in software. This principle, also known as the 80-20 law, implies that the majority of the faults reside in a small number of modules. After analyzing the fault data of very extensive telecom software, Ericsson, Fenton, and Ohlsson [8] suggest that the empirical distribution of faults there obeys the Pareto principle . Later on, Andersson and Runeson [17] replicate this study and validate the applicability of the Pareto principle. The suitability of the Pareto principle in another similar study on Motorola's telecom software is verified in [18]. In a large inventory software system, Ostrand and Weyuker [7] establish the appearance of Pareto distribution in the number of faults. Another replicated analysis of fault distribution in a complex software system endorses the Pareto principle [13]. A comparison of Pareto, lognormal, Weibull, double Pareto, and Yale-Simon distributions in fitting the empirical distribution of faults in a proprietary complex system is carried out in [11]. Their study finds double Pareto distribution to be more efficient than others in explaining faults.

In the enterprise Eclipse software system, Zhang [10] observes that Weibull distribution provides a better fit to faults than Pareto in both pre-release as well as post release. Concas et al. [19] investigate the distribution of faults in six releases of Eclipse by using Yale-Simon, double Pareto, lognormal, and Weibull distributions and conclude that the generative model of the Yale-Simon distribution better describes faults than the others. The Weibull distribution is also found to perform well in explaining faults in four releases of the Windows operating system [20]. In the case of open-source software, Hunt and Johnson [21], after analysing faults in projects at sourceforge, endorse the Pareto distribution. However, Kuo and Huang [3] find that the bounded generalized Pareto distribution (BGPD) provides the best fit to the empirical fault distribution of many open-source software. In a recent study on fault distribution, Sriram et al. [12] examine the fault data of 74 versions of various open-source and proprietary types of software. They conclude that the double Pareto distribution outperforms all of the others in the case of proprietary software systems. For open-source software systems, the double Pareto, BGPD, and Weibull result in an almost similar performance with negligible differences.

It can be noted here that no single distribution can explain the faults in different types of software. In particular, it can be easily observed that the Pareto principle is applied to large and complex software, whereas Weibull distribution is useful for enterprise software. In contrast to both of these, BGPD is better for open-source software. A pertinent question then arises concerning whether it is possible to explain faults in these diverse software systems in a uniform way. If so, then a single model can be developed for predicting faults. This is the main motivation for this study.

3. Methodology

This section presents details of the methodology adopted in this study. The first step is the collection of the fault data of various types of software. The data-collection process and data sets are described in the following subsection.

3.1. Data Collection

The first data set used is from Eclipse 2.0, 2.1, and 3.0 pre-releases and post-releases. This data set was first presented in [22], and later on it was proven in [10] that the distribution of software faults in this data set is better explained by the Weibull distribution. Eclipse is an enterprise software. Additionally, the data from three consecutive releases of the same software helps in checking the persistence of the results in software. The details of Eclipse data set are presented in Table 1.

Table 1. Details of Eclipse software data set.

Software	Number of Modules	Number of Pre-Release Faults	Number of Post-Release Faults
Eclipse 2.0	376	4152	2049
Eclipse 2.1	433	2007	1394
Eclipse 3.0	431	3312	2151

Besides this, fault data of Equinox and KAA enterprise software, gcc, samba, Python, and Firefox open source are included in this study, as given in Table 2. The data have been gathered from [23–28] and are up to 18 February 2020. The status of all the included faults is resolved and fixed. These faults are reported by users and thus correspond to the user utilization phase of the software life cycle.

The next step is to identify the candidate probability distributions to be included in this study. The next subsection provides details of them.

Table 2. Details of other software data sets.

Software	Type	Number of Modules	Number of Faults
Equinox	enterprise	313	3120
KAA	enterprise	30	711
gcc version 10	open source	23	290
samba version 3.0	open source	35	2519
samba version 4.0	open source	19	2523
samba version 4.1	open source	133	2398
Python version 3.9	open source	74	841
Firefox version 2.0	open source	46	10,000
Firefox for Android	open source	29	10,000

3.2. Generalized Pareto Distribution

The 2-parameter generalized Pareto distribution has been employed to analyze the fault distribution of open-source software in [29]. If random variable X represents the number of faults then the probability distribution function (pdf) of the 2-parameter generalized Pareto distribution is given by

$$p(x) = \frac{1}{b}\left(1 + \frac{ax}{b}\right)^{-\frac{1}{a}-1}, \ a, \ b > 0, \tag{1}$$

where a is the shape parameter and b is the scale parameter.

3.3. Weibull Distribution

The importance of the Weibull distribution in modeling faults was first highlighted in [10] for enterprise Eclipse software. Thereafter, it has been part of many empirical studies [11,12,20]. The pdf of the Weibull distribution is

$$p(x) = \left(\frac{\mu}{\lambda}\right)\left(\frac{x}{\lambda}\right)^{\mu-1} e^{-(x/\lambda)^{\mu}}, \ x > 0, \tag{2}$$

where μ is shape parameter and λ is the scale parameter.

3.4. Maximum Entropy Tsallis Distribution

The notion of entropy is linked to the theory of statistical mechanics in physical systems. However, Shannon [30] developed a measure of randomness or uncertainty of a system in the context of communication theory, which is mathematically similar to the one in statistical mechanics, and called it Shannon entropy. For a system S with states $n = 0, 1, \ldots$ with probability p_n, the Shannon entropy is defined as

$$S = -\sum_{n=0}^{\infty} p_n \log p_n, \ \ 0 \log 0 = 0. \tag{3}$$

In the context of non-extensive dynamical systems, a generalized measure of entropy known as Tsallis entropy was proposed [31]

$$S_q = \frac{1 - \sum_{n=0}^{\infty} p_n^q}{q - 1} \tag{4}$$

with parameter q measuring the degree of non-extensivity in the system. Tsallis entropy reduces to Shannon entropy as the parameter $q \to 1$,

$$S_1 = \lim_{q \to 1} S_q = -\sum_{n=0}^{\infty} p_n \ln p_n. \tag{5}$$

Tsallis entropy has found applications in various domains [31]. It is to be noted that a software system can also be treated as a physical system [32].

It is imperative to mention that one of the ways to obtain a probability distribution when some prior information is available, usually in the form of moments, is through Jaynes maximum-entropy principle (MEP) [33]. MEP with Shannon entropy has been used to model component-size distribution in software systems [32,34]. MEP with Tsallis entropy has been applied to communication networks as well [35–38]. Recently, Sharma and Pendharkar [39] have employed Tsallis entropy to study software-component sizes.

In this section, a closed-form of Tsallis distribution in terms of Hurwitz zeta function is derived. Representing p_n as the probability of n faults in a software system S and assuming that the range of faults can be from 0 to ∞, the maximum entropy problem can be formulated as

$$\text{Max } S_q = \text{Max } \frac{1 - \sum_{n=0}^{\infty} p_n^q}{q-1} \tag{6}$$

subject to the mean number of faults

$$\sum_{n=0}^{\infty} n p_n = A \tag{7}$$

and the normalization constant

$$\sum_{n=0}^{\infty} p_n = 1 \tag{8}$$

as constraints. Defining the Lagrangian function

$$\phi_q = S_q + \alpha \left(1 - \sum_{n=0}^{\infty} p_n\right) + \beta \left(A - \sum_{n=0}^{\infty} n p_n\right) \tag{9}$$

where α and β are Lagrange's multipliers corresponding to the normalization and the mean number of fault constraints, and differentiating ϕ_q with respect to p_n and equating to zero results in

$$p_n = \frac{[1 + \beta(1-q)n]^{\frac{1}{q-1}}}{\sum_{n=0}^{\infty}[1 + \beta(1-q)n]^{\frac{1}{q-1}}}, \quad n = 0, 1, 2, \ldots \tag{10}$$

The finiteness of the normalization constant in (10) requires $\frac{1}{1-q} > 1$, i.e., $q > 0$. The probability distribution of the number of faults given by (10) can be rewritten as

$$p_n = \frac{\left[\frac{1}{\beta(1-q)} + n\right]^{\frac{1}{q-1}}}{\zeta\left[\frac{1}{1-q}, \frac{1}{\beta(1-q)}\right]}, \quad q > 0, \quad n = 0, 1, 2, \ldots \tag{11}$$

where $\zeta\left[\frac{1}{1-q}, \frac{1}{\beta(1-q)}\right]$ denotes the Hurwitz zeta function defined by

$$\zeta\left[\frac{1}{1-q}, \frac{1}{\beta(1-q)}\right] = \sum_{n=0}^{\infty} \left[\frac{1}{\beta(1-q)} + n\right]^{\frac{-1}{1-q}}. \tag{12}$$

The fault distribution given in (11) is regarded as a Tsallis distribution. The parameter β in (11) can be estimated from the constraint given in (7) as

$$A = \frac{\zeta\left[\frac{q}{1-q}, \frac{1}{\beta(1-q)}\right]}{\zeta\left[\frac{1}{1-q}, \frac{1}{\beta(1-q)}\right]} - \frac{1}{\beta(1-q)}, \qquad q > \frac{1}{2}. \tag{13}$$

Cumulative distribution of faults:

The cumulative distribution of faults can be obtained from (11) as

$$P(X \leq x) = \frac{1}{\zeta\left[\frac{1}{1-q}, \frac{1}{\beta(1-q)}\right]} \sum_{n=0}^{x} \left[\frac{1}{\beta(1-q)} + n\right]^{\frac{1}{q-1}} \tag{14}$$

Estimation of parameters:

For a given data set of faults, one can obtain the appropriate values of the Tsallis parameter q and Lagrange's parameter β so that the fault distribution (11) can be evaluated. For this, first q is varied over a range, and for a specific value of q, the parameter β is computed from (7) using the numerical method. The constraints (7) can be expressed as

$$\sum_{n}(n - A)p_n = 0 \tag{15}$$

Using (10) and (15), it can be rewritten as

$$\sum_{n}(n - A)[1 + \beta(1-q)n]^{\frac{1}{q-1}} = 0. \tag{16}$$

Equation (16) is of the form $f(\beta) = 0$ and can be solved through the Newton–Raphson method from an initial approximation of β and by replacing it at each iteration by $\beta + \Delta\beta$ where

$$\Delta\beta = \frac{-f(\beta)}{f'(\beta)} \tag{17}$$

which can be expressed as

$$\Delta\beta = \frac{\begin{array}{r}\beta(1-q)\left\{\zeta\left[\frac{q}{1-q}, \frac{1}{\beta(1-q)}\right] - \left[\frac{1}{\beta(1-q)} + A\right]\right. \\ \left.\zeta\left[\frac{1}{1-q}, \frac{1}{\beta(1-q)}\right]\right\}\end{array}}{\begin{array}{r}\zeta\left[\frac{q}{1-q}, \frac{1}{\beta(1-q)}\right] - \left[\frac{2}{\beta(1-q)} + A\right]\zeta\left[\frac{1}{1-q}, \frac{1}{\beta(1-q)}\right] \\ + \frac{1}{\beta(1-q)}\left[A + \frac{1}{\beta(1-q)}\right]\zeta\left[\frac{2-q}{1-q}, \frac{1}{\beta(1-q)}\right]\end{array}} \tag{18}$$

till the sequence of iterates converges. Once the value of β is approximated for a value of q, the fault distribution can be completely evaluated. Afterwards, the Kolmogorov–Smirov (KS) statistic [40]

$$D_n = \sup_x \mid F_n(x) - F(x) \mid \tag{19}$$

(where $F_n(x)$ is the empirical cumulative distribution, $F(x)$ is the fitted cumulative distribution, and $\sup_x$ is the supremum of the set of differences) is used to compute the difference between empirical and cumulative distribution of faults (14). Then, the value of q and corresponding β, which gives a minimum value of KS, is chosen. A similar method is proposed by Clauset [41] for fitting power law distributions. The procedure is outlined in Algorithm 1.

Algorithm 1 Algorithm for Fitting Tsallis Distribution to Empirical Dataset of Software Faults

Require: Empirical data
Ensure: Estimated values of q and β
Compute arithmetic mean A from the data
Compute empirical cumulative distribution of faults
Initialize Tsallis entropy parameter q
Give initial value to parameter β

while $q < 1$ **do**
compute $\Delta\beta$ using (18)
$\beta \leftarrow \beta + \Delta\beta$
repeat above two steps till β converges
compute cumulative distribution of faults using (14)
compute KS statistics
increment q
end while
Choose minimum value KS and corresponding q and β

The next section describes the results of the experiments conducted to validate the Tsallis distribution in modelling software faults.

4. Results and Discussion

The model developed viz. Tsallis distribution is validated by running experiments on several data sets using the procedure described in Section 3.4. For comparative analysis, generalized Pareto and Weibull distributions are also fitted. The goodness of the fit in all the cases is checked by *KS* test [40]. The results of the experiments for the enterprise software data set are presented in Tables 3 and 4.

Table 3. Comparison of fault distributions in enterprise software—Part I.

		Generalized Pareto			**Weibull**		
		KS	**h Value**	***p* Value**	**KS**	**h Value**	***p* Value**
Pre-release faults	Eclipse 2.0	0.1944	0	0.4603	0.3889	1	0.0059
	Eclipse 2.1	0.1667	0	0.8608	0.3750	0	0.0506
	Eclipse 3.0	0.1250	0	0.9868	0.2500	0	0.3873
Post-release faults	Eclipse 2.0	0.2353	0	0.6725	0.8824	0	0.2083
	Eclipse 2.1	0.9091	1	8.1868×10^{-7}	0.7083	1	4.0102×10^{-6}
	Eclipse 3.0	0.9412	1	1.0822×10^{-7}	0.5833	1	2.7336×10^{-4}
	Equinox	1.0000	1	1.3029×10^{-21}	1.0000	1	1.3029×10^{-21}
	KAA	0.0741	0	1.0000	0.0741	0	1.0000

Table 4. Comparison of fault distributions in enterprise software—Part II.

		KS	h Value	p Value	q	Tsallis β
Pre-release faults	Eclipse 2.0	0.0811	0	0.9995	0.71	1.2978
	Eclipse 2.1	0.1600	0	0.9896	0.75	1.6322
	Eclipse 3.0	0.1111	0	0.9713	0.71	1.7671
Post-release faults	Eclipse 2.0	0.0556	0	1.0000	0.72	3.1030
	Eclipse 2.1	0.0909	0	1.0000	0.82	2.9025
	Eclipse 3.0	0.1176	0	0.9994	0.76	2.6499
	Equinox	0.0435	0	1.0000	0.66	0.2850
	KAA	0.0741	0	1.0000	0.51	0.1250

In these two tables, the h value of zero indicates that the given distribution fits well to empirical data, whereas value one implies that the specified distribution is not a good fit. It can be observed that Tsallis distribution provides the lowest value of the KS statistic in all the cases. Simultaneously, the p value also needs to be given importance while deciding about the goodness of fit [41]. It can be noted that the p value remains close to 1 for eclipse pre-release faults for Tsallis distribution and is 1 for eclipse post-release faults. Thus, it can be concluded that Tsallis distribution is a better fit than generalized Pareto and Weibull distribution for the Eclipse data set.

To justify the efficacy of the Tsallis distribution further, the experiments are run on fault data of open-source software as given in Table 2. The results of the experiments are shown in Table 5.

Table 5. Comparison of fault distributions in open-source software.

Dataset	Generalized Pareto			Weibull			Tsallis				
	KS	h Value	p Value	KS	h Value	p Value	KS	h Value	p Value	q	β
gcc version 10	0.1429	0	0.9971	0.2857	0	0.5407	0.1429	0	0.9971	0.70	0.1857
samba version 3.0	0.1111	0	0.9936	0.1111	0	0.9936	0.1111	0	0.9936	0.71	0.0327
samba version 4.0	0.1500	0	0.9655	0.1500	0	0.9655	0.1000	0	0.9999	0.71	0.0178
samba version 4.1	0.9474	1	1.3431×10^{-8}	0.1053	0	0.9998	0.1053	0	0.9998	0.83	0.0158
Python version 3.9	1.0000	1	1.5659×10^{-9}	1.0000	1	1.5659×10^{-9}	0.1579	0	0.9563	0.56	0.6151
Firefox version 2.0	1.0000	1	1.3029×10^{-21}	1.0000	1	1.3029×10^{-21}	0.0652	0	0.9999	0.66	0.0143
Firefox for Android	1.0000	1	5.0391×10^{-14}	1.0000	1	5.0391×10^{-14}	0.1034	0	0.9961	0.57	0.0206

It can be again observed that Tsallis distribution provides same or better results than both generalized Pareto and Weibull distributions in this case as well. The results of the analysis can be summarized in Table 6.

Table 6. Comparative analysis of research work on distribution of faults.

Software Type	Pareto and Its Variants	Weibull	Tsallis
Enterprise	×	√	√
Open source	√	×	√

It can be easily noted that Tsallis distribution describes faults in both types of software viz. enterprise and open source successfully. Therefore, Tsallis distribution can be considered as a unified model that can explain faults in diverse types of software. One of the reasons behind this is the fact that maximum entropy Tsallis distribution is the best distribution satisfying the available information about the mean number of faults in software. As pointed out by Hatton [32], software systems can be treated as physical systems; thus, the theory of evolution of physical systems can be applied to them. Following this, software systems too tend to move towards configurations that have maximum entropy. This may be another reason behind the applicability of Tsallis distribution in modeling faults in diverse

types of software. Further, the Tsallis entropy parameter governs the interactions within the physical system. Identical or close values of parameter q signify that the underlying dynamics of this software is similar even if it belongs to diverse types .

Precise knowledge of fault distribution helps in predicting faults during early phases of the software life cycle in new similar projects. A framework based on the Bayesian inference method has been proposed by Rana et al. [42] where fault distribution from previous projects is used for prior fault prediction. The basis of Bayesian inference method is the formula

$$f(\kappa|\text{data}) = Cf(data|\kappa)f(\kappa) \tag{20}$$

which relates the distribution of the number of faults $f(\kappa|\text{data})$ that needs to be estimated from the available 'data' of the new project, also called posterior distribution, with the likelihood $f(data|\kappa)$ and prior $f(\kappa)$. Here, C is the normalization constant. The prior $f(\kappa)$ now follows the Tsallis distribution. The details of the method are presented in [42].

Another application of this work is in software reliability, specifically in describing software failures. One of the pioneering works in software reliability is the Goal and Okumoto model [43], which deals with modeling the number of failures observed in a time interval. Modeling the cumulative number of failures at time t, $N(t)$ as a non-homogeneous Poisson process (NHPP), Goal and Okumoto [43] derive an expression for the probability of the number of failures as

$$P[N(t) = z] = \frac{[m(t)]^z}{z!} e^{-m(t)}, \ \ z = 0, \ 1, \ ... \tag{21}$$

where

$$m(t) = a\left(1 - e^{-bt}\right). \tag{22}$$

Here, a is a random variable representing the number of faults to be detected in a software, and b is the fault-occurrence rate. The probability distribution of a can now be given by (11), thus modifying (22) results in

$$m(t) = \int_0^\infty m(t|a)f(a)da = \frac{\left(1 - e^{-bt}\right)}{\zeta\left[\frac{1}{1-q}, \frac{1}{\beta(1-q)}\right]} \int_0^\infty \frac{\left[\frac{1}{\beta(1-q)} + a\right]^{\frac{1}{q-1}}}{a} da \tag{23}$$

which, after simplifying, gives

$$m(t) = \frac{\beta^{\frac{q}{1-q}}}{q\,\zeta\left[\frac{1}{1-q}, \frac{1}{\beta(1-q)}\right]} (1-q)^{\frac{1}{1-q}} \left(1 - e^{-bt}\right), \qquad q > 1/2. \tag{24}$$

The quantity $m(t)$, the expected number of failures observed by time t, can now be in terms of the Tsallis distribution parameters. For a given fault-occurrence rate, the expected number of failures can now be easily computed. These results will be very useful to software management.

5. Threats of Validity

This section addresses various threats of validity related to this study. The first one is internal validity, which determines the causal relationship between two variables [11]. There are three possible threats of internal validity, such as many other empirical studies on fault distribution [3,8,11,19]. The data-collection process is a threat, especially for open-source software. These software are free, and sometimes the faults are not reported and documented properly. Therefore, there may be incomplete and imprecise fault data for open-source software. The other threat of internal validity is that the good fit of the Tsallis distribution may be due to chance. There are two reasons to reject this threat in our study: one, because the analysis has been carried out on different types of software and the

Tsallis distribution is found to be a best fit with statistically high p values; two, the Tsallis distribution is a maximum-entropy distribution, which is the most plausible one given the information about the number of faults. The last threat of internal validity relates to the fact that the Tsallis distribution may not be the generative model of faults even if it is a good fit. This is a complex question that even has not been handled in other such studies on fault distribution in the past [3,8,11,19].

The second threat of validity is construct validity, which ensures if the findings of one software instance are sufficient to ascertain the behavior across all instances. For this, a series of versions of the same Eclipse software are included in this study and are analyzed to check the consistency of the Tsallis distribution. The analysis confirms the consistency of the performance of the Tsallis distribution with a high p value both in pre-release and post-release fault data of Eclipse.

The last threat of validity is external validity, which guarantees that the results of the analysis are applicable across other software. It is believed that the Tsallis distribution is generic enough to explain faults in other large and complex, enterprise and open-source software. However, only further replications of this study can confirm this.

6. Conclusions

Despite the extensive research carried out on the analysis of the distribution of faults in computer programs, there is no single model accepted that can explain faults in different types of software. Using the maximum Tsallis entropy principle when information about the mean number of faults is available as a constraint, the distribution of faults in a software system is derived. A procedure to estimate the distribution parameters is also presented. The performance of the Tsallis distribution in describing faults in many tyoes of enterprise and open-source software is compared with popular generalized Pareto and Weibull distributions. The Tsallis distribution is found to perform the same or better than the other two, thus making it useful for various types of software, including open-source software.

Two applications of precise knowledge of fault distribution are discussed. The first one relates to predicting faults in new similar projects during the early phase of the software life-cycle, when limited fault data is available, using the Bayesian Inference method. The Tsallis distribution is prior there, and investigating its impact on the accuracy of fault prediction is an area of future study.

Expressing the probability distribution of a random variable representing the number of faults as Tsallis in the famous software-reliability model by Goel and Okumoto, a closed form of expression for the expected number of faults by time 't' has been derived in terms of Tsallis distribution parameters. Applying these new results to real data and analyzing the performance of the modified Goel–Okumoto model are tasks for future work.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this research will be made available on request from the corresponding author.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Jalote, P. *An Integrated Approach to Software Engineering*; Springer: New York, NY, USA, 2005.
2. Kaur, N.; Singh, H. An empirical assessment of threshold techniques to discriminate the fault status of software. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 6339–6353. [CrossRef]
3. Huang, C.Y.; Kuo, C.S.; Luan, S.P. Evaluation and application of bounded generalized pareto analysis to fault distributions inopen-source software. *IEEE Trans. Rel.* **2014**, *63*, 309–319. [CrossRef]
4. Boehm, B.; Basili, V.R. Software defect reduction top 10 list. *Computer* **2001**, *34*, 135–137. [CrossRef]
5. Ozakinci, R.; Tarhan, A. Early software defect prediction: A systematic map and review. *J. Syst. Softw.* **2018**, *144*, 216–239. [CrossRef]

6. Tanaka, K.; Tsuda, K. Methods to predict the number of software faults using Weibull distribution. In Proceedings of the IEEE 40th Annual Computer Software and Applications Conference, Atlanta, GA, USA, 10–14 June 2016; pp. 105–110.
7. Ostrand, T.J.; Weyuker, E.J. The distribution of faults in a large industrial software system. *ACM SIGSOFT Softw. Eng. Notes* **2002**, *27*, 55–64. [CrossRef]
8. Fenton, N.E.; Ohisson, N. Quantitative analysis of faults and failures in a complex software system. *IEEE Trans. Softw. Eng.* **2000**, *26*, 797–814. [CrossRef]
9. Vrankovi, A.; Grbac, T.G. Replication of quantitative analysis of bug distributions on open-source software systems. In Proceedings of the 7th Workshop of Software Quality Analysis, Monitoring, Improvement, and Applications, Novi Sad, Serbia, 27–30 August 2018; pp. 22:1–22:9.
10. Zhang, H. On the distribution of software faults. *IEEE Trans. Softw. Eng.* **2008**, *34*, 301–302. [CrossRef]
11. Grbac, T.G.; Runeson, P.; Huljenic, D. A second replicated quantitative analysis of bug distributions in complex software systems. *IEEE Trans. Softw. Eng.* **2013**, *39*, 462–476. [CrossRef]
12. Sriram, C.K.; Muthukumaran, K.; Murthy, N.L.B. Empirical study on the distribution of faults in software systems. *Int. J. Softw. Eng. Knowl. Eng.* **2018**, *28*, 97–122. [CrossRef]
13. Grbac, T.G.; Huljenic, D. On the probability distribution of faults in complex software systems. *Inf. Softw. Technol.* **2015**, *58*, 250–258. [CrossRef]
14. Pham, T.; Pham, H. A generalized software-reliability model with stochastic fault-detection rate. *Ann. Oper. Res.* **2019**, *277*, 83–93. [CrossRef]
15. Thapar, S.S.; Singh, P.; Rani, S. Using ordered Probit model to study the effects of component quality on reusability. *Appl. Math. Inf. Sci.* **2018**, *12*, 159–170. [CrossRef]
16. Harter, D.E.; Kemerer, C.F.; Slaughter, S.A. Does software process improvement reduce the severity of defects? A longitudinal field study. *IEEE Trans. Softw. Eng.* **2012**, *38*, 810–827. [CrossRef]
17. Andersson, C.; Runeson, P. A replicated quantitative analysis of fault distributions in complex software systems. *IEEE Trans. Softw. Eng.* **2007**, *33*, 273–286. [CrossRef]
18. Daskalantonakis, M.K. A practical view of software measurement and implementation experiences within motorola. *IEEE Trans. Softw. Eng.* **1992**, *18*, 998–1010. [CrossRef]
19. Concas, G.; Marchesi, M.; Murgia, A.; Tonelli, R.; Turnu, I. On the distribution of bugs in the eclipse system. *IEEE Trans. Softw. Eng.* **2011**, *37*, 872–877. [CrossRef]
20. Hribar, L.; Dula, D. Weibull distribution in modeling component faults. In Proceedings of the 52nd 52nd International Symposium ELMAR, Zadar, Croatia, 15–17 September 2010; pp. 183–186.
21. Hunt, F.; Johnson, P. On the Pareto distribution of sourceforge projects. In Proceedings of the International Workshop open-source software Develop, Orlando, FL, USA, 19–25 May 2002; pp. 122–129.
22. Zimmermann, T.; Premraj, R.; Zeller, A. Predicting defects for eclipse. In Proceedings of the Third International Workshop on Predictor Models in Software Engineering, Minneapolis, MN, USA, 20–26 May 2007.
23. Equinox. Available online: https://bug.inf.usi.ch/download.php (accessed on 30 October 2022).
24. KAA Platform. Available online: https://www.kaaproject.org/ (accessed on 18 Feburary 2020).
25. GCC. Available online: https://gcc.gnu.org/bugzilla/ (accessed on 18 Feburary 2020).
26. Samba. Available online: https://bugzilla.samba.org/ (accessed on 18 Feburary 2020).
27. Available online: https://bugs.python.org/ (accessed on 18 Feburary 2020).
28. Available online: https://bugzilla.mozilla.org/ (accessed on 18 Feburary 2020).
29. Kuo, C.; Huang, C.; Luan, S. A study of using two-parameter generalized Pareto model to analyze the fault distribution of open-source software. In Proceedings of the IEEE Sixth International Conference on Software Security and Reliability, Gaithersburg, MD, USA, 20–22 June 2012; pp. 88–97.
30. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, 27, 379–423. [CrossRef]
31. Gell-mann, M.; Tsallis, C. *Nonextensive Entropy: Interdisciplinary Applications*; Oxford University Press: Oxford, UK, 2004.
32. Hatton, L. Power-law distributions of component size in general software systems. *IEEE Trans. Softw. Eng.* **2009**, *35*, 566–572. [CrossRef]
33. Peterson, J.; Dixit, P.D.; Dill, K.A. A maximum entropy framework for nonexponential distributions. *Proc. Natl. Acad. Sci. USA* **2013**, *110*, 20380–20385. [CrossRef]
34. Sharma, S.; Pendharkar, P.C.; Karmeshu, K. Learning component size distributions for software cost estimation: models based on arithmetic and shifted geometric means rules. *IEEE Trans. Softw. Eng.* **2021**. [CrossRef]
35. Karmeshu, K.; Sharma, S. Power law and Tsallis entropy: Network traffic and applications. In *Chaos, Nonlinearity, Complexity*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 162–178.
36. Karmeshu; Sharma, S. q-Exponential product-form solution of packet distribution in queueing networks: maximization of Tsallis entropy. *IEEE Comm. Lett.* **2006**, *10*, 585–587. [CrossRef]
37. Sharma, S.; Karmeshu, K. Bimodal packet distribution in loss systems using maximum Tsallis entropy principle. *IEEE Trans. Comm.* **2008**, *56*, 1530–1535. [CrossRef]
38. Sharma, S.; Karmeshu, K. Power law characteristics and loss probability: finite buffer queueing systems. *IEEE Comm. Lett.* **2009**, *13*, 971–973. [CrossRef]

39. Sharma, S.; Pendharkar, P.C. On the analysis of power law distribution in software component sizes. *J. Softw. Evol. Process* **2022**, *34*, e2417. [CrossRef]
40. Massey, F.J. The Kolmogrov-Smirnov test for goodness of fit. *J. Am. Stat. Assoc.* **1951**, *46*, 68–78. [CrossRef]
41. Clauset, A.; Shallz, C.R.; Newman, M.E.J. Power-law distributions in empirical data. *SIAM Rev.* **2009**, *51*, 661–703. [CrossRef]
42. Rana, R.; Staron, M.; Berger, C.; Hansson, J.; Nilsson, M.; Meding, W. Analyzing defect inflow distribution and applying Bayesian inference method for software defect prediction in large software projects. *J. Syst. Softw.* **2016**, *117*, 229–244. [CrossRef]
43. Goel, A.L.; Okumoto, K. Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Trans. Rel.* **1979**, *28*, 206–211. [CrossRef]