Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 8 (2017) pp. 2411-2426

http://www.ripublication.com

Evenness and Test Coverage Requirements

Dr. Meenu Dave[1] and Rashmi Agrawal[2]

[1,2]Jagan Nath University Jaipur, Rajasthan, India.

Abstract

The paper is an empirical study of the diversity of requirements coverage by a given test suite. The aim of a test adequacy criteria is the full coverage of the requirements. The diversity is the study of the probability of occurrence of a particular requirement. The paper is a study of the diversity metrics and their relation to code coverage, and comparative analysis for a test suite generation. The main diversity indexes studied are Evenness and Entropy Index. The co-relation analysis shows the close relation between the two matrices and code coverage.

1. INTRODUCTION

Software Testing is an important process of defects detection and prevention [1]. Testing, a multilevel process, starts with white box testing and black box testing are performed at unit level, precedes with integration, followed by system and finally acceptance testing. A general testing cycle consisting of test case generation, execution, and output analysis is performed at each level. Test data generation (**TDGN**) is an important phase of testing [2]. The manual generation of test data is very laborious and time consuming, which becomes more complicated with increase in size of source code. The automatic generation can be seen as an alternate to the problem. **TDGN** and its execution can be automated, however, the generation of oracle requires human intervention. A quantifying measure or measures known as **criteria** or a completion milestone, helps to assess the testing process and mark it as complete or adequate [1].

Unit testing is performed at the basic single independent identity of software. The coverage is the part of the program that has been executed by a test suite which can be a heuristic to generate or enhance a test suit. The run time behavior of the software is collected in the form of *traces*. The coverage criteria are the base for strategic placement

of traces and the coverage of a test suite is adequate when all the traces are executed once by a test case. In this paper, we present a study of traces which is a statistical diagnostic analysis of the run time information and independent of the coverage criteria. The paper introduces the information theory based index for the analysis of diversity in **TDGN**. Entropy is a measure and unit of information theory initially derived from thermodynamics and first proposed by Shannon. *Diversity* is a term which is commonly used in ecology to measure the variation of the species and their distribution in a given area. *Entropy* is a syntax independent criteria [3] [4]. Suppose there are two test suites which cover different criteria, then these suites can be compared based on their information gain [3]. During the initial stages of development the behavior of software is highly uncertain. Testing software can be seen as cool down process to take out the randomness and uncertainty [3]. The diversity analysis is done through *Richness* and *Evenness*. Richness is the count of unique traces of software and Evenness is the distribution of these traces across the test suite. The density of traces for a test suite may change during the software evolution or in later stages due to the change in requirements, or fault detection, or enhancements in the later versions of the product. The diversity can guide the testers and the developers, on the abstract level of evolution or enhancement or requirement of enhancement of the test suite along with the software product development.

Suppose the test suite follows criteria C1 and the number of unique traces is n, then the Richness is n and Evenness are the count of each of the n traces in test suite. The Evenness index is derived from the entropy of information and the Theil's redundancy is the distance between the maximum Entropy index and the actual index. This can be the base for sorting and comparing test suites, irrespective of the criteria selected by testers and developers. The paper presents a brief description of three indexes and applies one to the **TDGN** process.

1.1 Objective

Objective: The aim of the paper is to a) study the diversity of traces for a test suite at various stages of enhancement b) application and analysis of various diversity index for a test suite which is adequate for a given criteria c) analysis of diversity index in other classes.

Input: Source code, trace details, details of code coverage criteria and test cases.

The outline of the paper is as follows. A brief description of Diversity and commonly used diversity index, **TDGN**, control flow graph criteria, Objectives and problem statement is presented in Section II. Work related to this domain is discussed in Section III. Section IV is the methodology to calculate the diversity followed by results in Section V. Section VI concluded the paper with directions for future work.

2. PRELIMINARIES

In this section we shall cover the concepts of Diversity, Richness, Evenness, Entropy Index, Automated **TDGN**, Search based **TDGN**, Control Flow Graph, Instrumentation, Objectives and Problem Statement.

2.1 Automated TDGN

Testing a program with some input with the aim of finding errors that might be present [5]. Broadly testing can be classified into two main strategies, black- box testing and white box testing [2]. Black box testing or Functional testing performed in the absence of source code based on the input output values. The requirements of the test case generation are derived from the specifications. White box testing or structural testing is performed with the source code. The number of possible test cases can be enormous and executing each test case is in-feasible [2] [5].The search for test data which is might not be optimal but near to optimal is known as the SBTG [6]. The input domain is searched in various dimensions and analyzed for best near optimal value which might not be visible to human mind .Test data selected from the input domain is analyzed against some fitness function to measure how close it is towards the goal. The aim of the search might be framed as the minimization (time, cost, resources) or maximization (coverage, fault detection) problem. The sorting of the test cases based on the fitness function helps easy selection or prioritization or guide for future search process. Reference [6] is the detailed survey of SBTG.

Before testing the code under test is analyzed for its structure and a control flow graph is drawn. Control Flow Graph is a graphical representation of a structure of a program for easy understanding of the logical flow [2] [7]. *Graph is a collection of nodes and directed edges with one entry and one exit point.* All the nodes are connected directly or indirectly to entry and exit nodes. The nodes are decision statements which may further split into two nodes true and false, however the path is decided run time depending on the input values. The edges are directed showing the path that can be executed at run-time. The input values are searched manually or automatically (Automated TDGN) to meet these requirements and measure the coverage.

A set of requirements are designed from the structure of program and set as *criteria* [2] [5] [7]. Test cases are executed with programs and the added to the test suite provided they are able to fulfill the requirement. A test suit is adequate when it has test cases for each and every requirement and the testing is said to be complete.

Function coverage, statement coverage and branch coverage are some of the commonly used criteria [7] [2]. Branch coverage has been briefly discussed with an example. The decision statements in a program can have two branches or paths to follow which is decided at run time. The test suite having test cases which execute these branches at-

least once and visiting the decision statement twice is adequate.

Example: following is code in Java

```
public void max (int a, int b)
        {

if (a>b)        {
System.out.println( A is max );
}

else        {
System.out.println( B is max );
}
}
```

The function max is the only function in a given class in java. A test case executes the function with values (5, 6). The function coverage is test case is 100%. Branch coverage is 1 branch covered. An additional test case (6, 5) is executed to cover the second branch and statement coverage. *Assumptions*: a) Oracle has been verified manually b) the traces have been placed strategically as per the requirements.

2.2 Diversity

The term diversity is commonly used in ecology to analyze the abundance of various species or families or types. The diversity index measure the *unpredictability degree* for any individual chosen randomly [8] [9]. The values are taken as probabilities (frequencies) of the traces and with equal weightage. Equal distribution of the all the traces gives the maximum index value which is the log of the total number of traces.

Redundancy is the distance between the maximum value and the actual entropy. In this paper we borrow the Theil's inequality from the economics and derive it from entropy of information theory. A trace is like a *household* in economics and a *household income* is the test suite with N_i/N as the probability. In economics mean is taken for Theil's inequality.

2.2.1 Richness (R)

The number of unique types or varieties or species or subjects under considera- tion is known as the Richness [8]. In this paper the maximum number of traces is taken as the Richness. Richness is independent of the coverage criteria and provides a syntax independent measure. We discuss three main diversity index, Simpson's, Shannon-

Wienner entropy and Renyi's. We will apply Shannon- Wienner entropy for our study.

Simpsons index (D) It is the probability based estimation for two individuals, se- lected randomly from a population of N individuals falling in to the same group where $n1$, $n2...nz$ are the individuals, grouped together [10] [11]. Simpson's index is given in equation 1.

$$D = [\sum n\ (n - 1)]/N\ (N - 1) \tag{1}$$

where, N = ∑n.

Shannon - Wienner Entropy (H)

In Information theory, Entropy also known as Shannon's Entropy is the measure of uncertainty [21], commonly known as Shannon-Wienner entropy in Ecology [10]. In this paper we refer Shannon - Wienner Entropy as Entropy *(H)*.

$$H = -\sum pi * log\ (pi) \tag{2}$$

where, i is the species from a set of species and p_i is the frequency or probability or portion from the population under study. The H is the probability of an individual belonging to certain species selected randomly [8].

Renyi's Entropy

The index is quite famous in ecology as diversity index [12]. It is an extension of Shannon's Entropy by generalizing it.

$$H_\alpha = \frac{1}{1} - \ \alpha * \ln(\sum_{i=1}^{S} p_i^\alpha) \tag{3}$$

Table 1: Example of Diversity Index for Trace coverage and test suite

	TS					
Traces	s1	s2	s3	s4	s5	Entropy
r1	X	X	X	X		0.387585
r3		X				0.163563
r4	X				X	0.260459
r5	X		X	X		0.332192
r6		X		X	X	0.332193
Coverage	3	3	2	2	2	1.475993

2.2.2 Evenness

Evenness or *Equability* ($H^{'}$) [13] is the distribution of individuals over species [10]. It is a numerical representation of distance between species in ecology. The evenness is the index of the Shannon's Entropy and the highest possible value of entropy ($H^{'}_{max}$). The commonly used Pielou's J' [10] is shown in 4.

$$J' = H/H_{max} = H/log(S) \tag{4}$$

where H is the number derived from the Shannon diversity index and H_{max} the maximum possible value of H (if every species was equally likely), equal to

$$H_{max} = -\sum_{i=1}^{S}\left(\frac{1}{S}\right) * \ln\left(\frac{1}{S}\right) = S \tag{5}$$

$J^{'}$is constrained between 0 and 1. The less variation in communities between the species (and the presence of a dominant specie), the lower J^{I} is and vice versa. S *i*s the total number of species.

2.2.3 Redundancy (Entropy Index)

Entropy Index also known as *Information Index* or *Theil's Index* is a popular inequality metric of economics to study the income inequality [14]. The Theil's index derived from the Information theory is the distance between the maximum value and the original value. In economics, the maximum value is when in a population of *N*, all earn equally, while, the minimum value is when a single person possess all the earnings. The entropy *H* of the population *y* is calculated as the maximum of the value is *log* (*N*) where *N* is the number of individuals in population.

Table 2: Example of Diversity Index for Trace coverage and test suite

Entropy	Hmax	Richness	Evenness	G
1.475993	2.584962	6	0.570992	1.108969

2.3 Problem Statement

Code coverage: Each program has a set of traces inserted strategically to collect the execution profile of each test case as per the requirements of the criteria. Let $C1$ be the criteria and R be the set of requirements $R = r1...r_n$ where $r1 - - - r_n$ are the

requirement and *n* is the total number of requirements. Let *pn* be the code coverage of test $t1$ where $t1 \in T$ and *T* is a test suite of test cases and *N is* the total number of traces. Then the *pn* as percentage (*p*) is $p = pn / N$.

Problem Statement: Test suite is a set of test cases and the execution profile of the test cases has the details of the traces covered. The test suite is a population of traces. The traces can be taken as the species and each occurrence as the individual of the species. *The distribution of traces within the test suite is analyzed for its diversity, evenness and redundancy*

3 RELATED WORK

Fisher proposed the diversity for the first time in 1943 [15]. Richness of a diversity was first presented by PRESTON EW [16] as log norm pattern of abundance. *Species evenness* a measure of how equitably abundances are distributed among species [17] [18]. The most commonly used Evenness index are Simpsons [11] and Shannon's index [19], Pielou's (1966, 1967) [20]. Theil's redundancy is derived from Information theory [21] [22] [23]. Details of Entropy for further studies can be found at [23]. Some Economics aggregate indices has been applied to software engineering [24] [25] [26] [27] [28]. Theil's redundancy has been introduced into software engineering by Serebrenik, Alexander, and Mark [24] [25] as software index for macro level. A preliminary investigation of the aggregate indices in software develop- ment at macro-level has been done by Bogdan Vasilescu, Alexander Serebrenik, Mark van den Brand [27]. Work by Vasilescu et al. is an experimental study of various diversity indices as aggregate software metric [29] [30]. Study of aggregate metrics in industry is presented in [31].

4 METHODOLOGY

To study the relation between the evenness and the coverage of software, we have instrumented the source code, by inserting the traces strategically for branch coverage. The instrumentation provided information in the form of an execution profile. Test cases are generated, executed and the execution profile is converted into diagnostic matrix.

Table 3: Details of variables of Algorithm 1

Variable	*Details*
P	Source code of Program under testing
S	Population of Test case inputs for program P
T	Test case input sets, initially empty
R	Set of random input data
D	Oracle
N R	Number of unique traces(Richness)
M	Size of set of test case inputs
brlog	branch log or structure details
trlog	log of probability for each trace
H	Entropy Entropy
H^I	Evenness
G	Evenness Index
Hmax	Maximum of H
pi	probability of trace
N	Number of traces, maximum

4.0.1 Diagnostic Matrix

This section explains the coverage data matrix and its diagnostics. Program P has N traces.

All the traces are covered with M test cases. The $N * M$ matrix shows the coverage of traces. The traces have been labeled as 1, when exercised by a test case, else 0. The count of the number of traces visited in total is the frequency, while the number of traces covered by a test case is its coverage. The number of unique traces in a test suite is the Richness and the *Population = Richness * M*. Table 1 is an example of coverage of test suit TS, where $S1...S5$ are the test cases. For each requirement to be covered, a trace r_i is inserted the code through instrumentation. In table 1 the traces are shown as $r1...r6$, which need to be executed by atleast one test case. Each trace executed is marked X to show as covered. The cardinality of a test case is the number of traces it visit or the code coverage. The cardinality of a trace is the number of test cases that execute it. This *cardinality* forms the base for the calculation of *probability* and *entropy* of the test suite, showing the *diversity* of the traces in a given population.

Algorithm 1: DiversityEvenness

Input: $brlog$, $R = \emptyset$, NR=0, pi,S, P, $T = \emptyset$, $trlog$, $D = \emptyset$,M =0, H_{max}, N

Output: H',G

Variable: Random numbers for the parameters of P

Initialization : Generate R randomly

```
for target ← brlog do
    for t_i ϵ R do
        NR = 0
        tracet_i = getTrace(t_i)
        if tracet_i.contains(target) then
            T = T ∪ t_i
            D = D ∪ P(t_i)
            pi = getProbability(tracet_i) (// get probability of
             trace)
            pi = pi + 1 (// probability updation)
            Update trlog
            M = M + 1
H_max = log_2(N) (// maximum entropy)
S = NR * M (// population)
H = 0
for tr ϵ trlog do
    pi = getProbability(tr)
    H = H + (pi * log_2(1/pi)) (// Entropy)
H' = H/H_max (// Evenness)
G = H_max − H (// Entropy Index)
```

TS	Cov	S	NR	H_{max}	H	G	H'	Min	Max	Mean	Mode
50	34	1700	34	5.08746	2.9381	2.1493	0.5775	2	51	25.7059	22
100	34	3400	34	5.0875	2.7313	2.3562	0.5369	2	101	47.2941	36
150	36	5400	36	5.1700	2.5507	2.6191	0.4934	2	151	64.4444	50
200	37	7400	37	5.2095	2.3451	2.8642	0.4502	2	201	76.8378	60
250	39	9750	39	4.9069	2.2568	2.6501	0.4600	2	251	90.9488	3
300	39	11700	39	5.2854	2.2925	2.9929	0.4337	3	301	111.3077	3

Table 4: Details of Metrics with the evolution of test suite

4.1 Algorithm DiversityEvenness

In this section we discuss the algorithm 1. The algorithm variable details are given in table 2. The algorithm provides the step-by-step procedure to calculate the diversity index G Entropy index and Evenness H for the population of traces Sin a test suite T

generated and Richness *N*. Initially a set of random numbers is generated as a set of test inputs. Each test set is executed with *P* and an oracle is generated and the probability for each trace is updated. Function *getProbability()* returns the details of the trace t_i. The maximum entropy *Hmax* is calculated, followed by *S* and *H* is initialized to 0. Then for each details in *trlog*, *H'* and *G* are calculated and returned.

5 EXPERIMENTS

To conduct empirical study of the diversity, we conducted some experiments. The subjects were benchmark java classes [34] and [32]. The source code was instrumented in two stages. The first stage of instrumentation was done with etoc [33]. The second stage instrumentation was done manually as per the criteria of coverage. At this stage the trace details inserted were mapped as database for future reference. The inputs were generated randomly. Each input set was run with the subjects and the traces were recorded. The experiments were conducted with all the java classes and the relation between the coverage and diversity were studied. The diversity index Pielou's Evenness and Entropy Index, both derived from the entropy of Information theory are analyzed.

6 RESULTS

This section presents the results of the empirical study conducted. The results of the experiments may reveal more information than analyzed in this paper. All the classes studied are from the literature of testing.

a. Test Suite Evolution and Diversity Index

The artifact of the study is the benchmark Triangle classification problem from the thesis of Sthamer [32]. The program is quite popular in research community and has attained the status of benchmark. The program takes three numeral values as input and checks if the triangle can be drawn and if yes, then what kind of triangle.

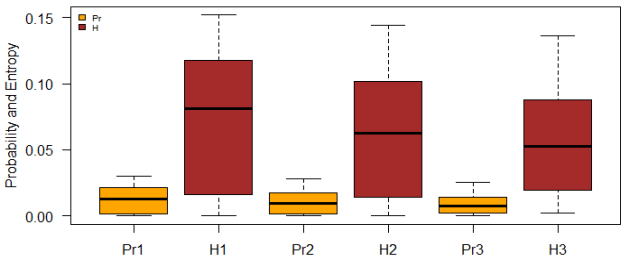

Figure 1: Trace Specific Entropy For Test Suite Evolution

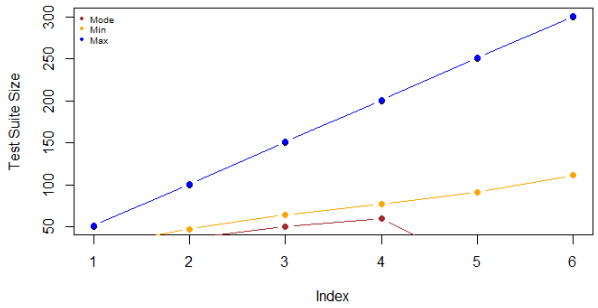

Figure 2: Min Max Median Mode

The particular program is the choice for detail study as it has a good hierarchy of decision statements and the test data generation explores all the possible decision statements as per the given requirement. This brings us close to the problem statement, **TDGN** is repetitive process of search of test data such that even the hardest trace can be executed. The analysis of the experiments primarily resolves around the relation between the diversity index and the diversity of the covered traces.

The other classes for study have been taken from the experiment repository of Paolo [33] [34] which is quite famous with research community. Table 4 shows the details of the test suite evolution. TS Size is the test suite size where 50,100,150,200,250 and 300 is the cardinality. Table shows the Coverage in terms of traces (*Cov*), Population (*S*) for each, Richness (*NR*), Maximum entropy (H_{max}), Entropy (*H*), Entropy Index (*G*), Evenness (*H'*), Minimum of Cov (Min), Maximum of Cov (Max), Mean and Mode of Cov.

Given a population S, the Evenness is the maximum entropy divided by the actual entropy as in equation4. Actual entropy is calculated from the probability of a trace for a given population by equation 2. Frequency of each trace for a given test suite is presented in figure 1, where the data is collected for a set of 300 test inputs. The traces are shown as the $b1, b2 \ldots bn$. The difference in the frequency of the traces clearly shows that the few traces are visited by all traces while, some are hard to execute.

The evolution of the test suite is the addition of traces by adding the unique traces covered to the Richness and updating the probability. Figure 1 shows the entropy of all the traces in three generations of test suite. The test suits evolves and the data for 50 sets of test inputs, 150 sets of test inputs and finally 250 sets is shown. Entropy of these generation are shown as Entropy50, Entropy150 and Entropy250. Some interesting trends can be seen as how the entropy changes, especially when a particular trace frequency is 0, to gaining a decent frequency. Some traces show consistency in all the three generations. The figure clearly depicts the uncertainty regarding the

program behavior and reduction in it with the testing process. The three generation differ in Richness and Population. Figure 3 shows the entropy for the generations with same Richness. Figure 3.a shows the generation Entropy 50 and Entropy 100 and their respective Evenness. 3b shows Evenness for generation Entropy 250 and Entropy 300 with same Richness.

Table 5: Correlation between Evenness, Richness, Diversity Index For Class Triangle Classification

Correlation	Entropy Index	Entropy	Richness	Mean
Evenness	-0.9759	0.9783	-0.9120	-0.9584
Entropy	-0.9095		-0.9425	-0.9471

Table 2 shows correlation between the Entropy Index, Entropy, Richness and Mean with Evenness and Entropy. It can be seen that Evenness has positive correlation with Entropy and negative with rest. Entropy has negative relation with all. The correlation is collected for all the test suite sizes and their respective matrix from table 4.

Figure 4 shows the trends of Richness with diversity index, and Entropy. Figure 4.a show an inverse relation between the Richness and Evenness for the respective generation in the test suite evolution. Similar trend is seen in 4.b between Richness and Entropy Index. Figure 4.b shows the trends of the Min, Max, Mean and Mode for traces for the various generations of test suite evolution. Generations are shown on X axis, while the traces are shown on axis. It can be seen that there is rise in Max and Mean. Min remains constant. Mode first rises, then falls down.

b. Java Classes and Diversity Index

In this part of analysis, we discuss the results of the trends for other java classes. The details of the result are presented in table 4. Table shows the details for

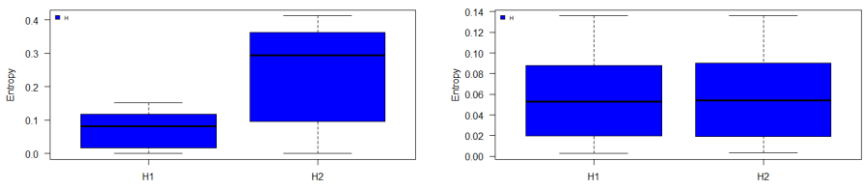

Figure 3. a Figure 3. b

Figure 3: Trace Specific Entropy for Population with same Richness

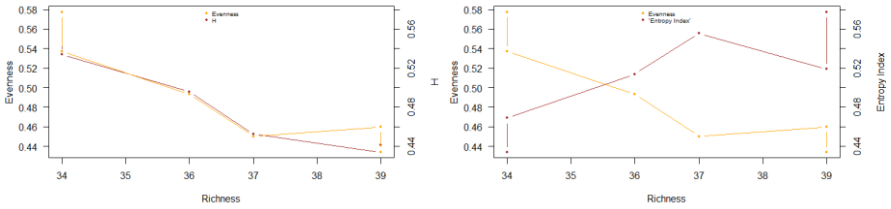

Figure 4. a Figure 4.b

Figure 4: Analysis of Richness of test suite for Diversity Index

Class	H'	NR	G	H	H_{max}	Min	Max	Mean	Mode	TSuite	LOC
Triangle	0.4600	39	2.6501	2.2569	4.9069	2	251	90.9488	3	250	90
Bisect	0.9130	5	0.2021	2.1199	2.3220	16	26	20.4	26	25	65
Bubble Sort	0.9172	9	0.2625	2.9074	3.1680	2	36	32.2222	36	35	40
Find	0.9340	16	0.2639	3.7361	3.7361	19	31	27.25	31	30	100
FourBalls	0.9620	8	0.3876	2.8860	3	7	16	14.25	16	16	65
Mid	0.8689	17	0.5360	3.5514	4.0874	9	36	29.0588	36	35	80

Table 6: Details of Metrics For all the classes

Table 7: Correlation between Evenness, Richness, Diversity Index For All Classes

Correlation	Entropy Index	Entropy	Richness	Mean
Evenness	-0.9881	0.4561	-0.9400	-0.9833
Entropy	-0.4277		-0.1355	-0.3808

six java classes for Evenness ($H^{'}$), Richness ($N\ R$), Entropy Index (G), Entropy (H), Maximum Entropy ($Hmax$), Minimum (Min), Maximum (Max), Mean and Mode for the traces for each class and No of Lines of Codes (LOC). Table 5 shows Correlation of Entropy Index, Entropy, Richness and Mean for Evenness and Entropy for all the six classes. It can be seen that Evenness and Entropy has a positive Correlation, and negative with Entropy Index, Richness and Mean.

7. CONCLUSION AND FUTURE WORK

In this paper we present the study of diversity index Evenness, Entropy Index along with statistics mean, mode, minimum and maximum at unit level testing. The diagnostic matrix of the test suite evolution is the base for entropy and diversity matrix. The study shows that given any set of requirement, a test suite can be analyzed for all the above metrics and hence creating a base for comparison between the test suites for same

version, for different versions and might be a ground for test suites adequate for different requirements. Analysis of entropy might help in fault localization, non-reachable code segment and code complexity. The probability of traces can be a heuristic for testing based on prioritization of code. Future work includes extension of analysis to industry software products with different versions of same software.

8. REFERENCES

[1] Huizinga, Dorota, and Adam Kolawa. Automated defect prevention: best practices in software management. John Wiley & Sons, 2007.

[2] Mathur, Aditya P. Foundations of Software Testing, 2/e. Pearson Education India, 2008.

[3] Yang, Linmin, Zhe Dang, and Thomas R. Fischer. "Information gain of black-box testing." Formal aspects of computing 23.4 (2011): 513-539.

[4] Yang, Linmin, et al. "Entropy and software systems: towards an information-theoretic foundation of software testing." Proceedings of the FSE/SDP workshop on Future of software engineering research. ACM, 2010.

[5] Pressman, Roger S. Software engineering: a practitioner's approach. Pal- grave Macmillan, 2005.

[6] McMinn, Phil. "Search-based software test data generation: A survey." Soft-ware Testing Verification and Reliability 14.2 (2004): 105-156.

[7] Zhu, Hong, Patrick AV Hall, and John HR May. "Software unit test coverage and adequacy." Acm computing surveys (csur) 29.4 (1997): 366-427.

[8] Jost, Lou. "Entropy and diversity." Oikos 113.2 (2006): 363-375.

[9] Jost, Lou. "The relation between evenness and diversity." Diversity 2.2 (2010): 207-232.

[10] Heip, Carlo HR, Peter MJ Herman, and Karlin Soetaert. "Indices of diver- sity and evenness." Oceanis 24.4 (1998): 61-88.

[11] Simpson, Edward H. "Measurement of diversity." Nature (1949). doi:10.1038/163688a0

[12] Rnyi, Alfrd. "On measures of entropy and information." Proceedings of the fourth Berkeley symposium on mathematical statistics and probability. Vol. 1. 1961.

[13] Mulder, C. P. H., et al. "Species evenness and productivity in experimental plant communities." Oikos 107.1 (2004): 50-63.

[14] Theil, Henri. "Economics and information theory." (1967).

[15] Fisher, Ronald A., A. Steven Corbet, and Carrington B. Williams. ”The relation between the number of species and the number of individuals in a random sample of an animal population.” The Journal of Animal Ecology (1943): 42-58.

[16] Preston, Frank W. “The commonness, and rarity, of species.” Ecology 29.3 (1948): 254-283.

[17] Wilsey, Brian J., and Catherine Potvin. “Biodiversity and ecosystem functioning: importance of species evenness in an old field.” Ecology 81.4 (2000): 887-892.

[18] Polley, H. Wayne, Brian J. Wilsey, and Justin D. Derner. “Do species evenness and plant density influence the magnitude of selection and complementarity effects in annual plant species mixtures?.” Ecology Letters 6.3 (2003): 248-256.

[19] Shannon, Claude Elwood. “A mathematical theory of communication.” ACM SIGMOBILE Mobile Computing and Communications Review 5.1 (2001): 3-55.

[20] Pielou, Evelyn C. “The measurement of diversity in different types of biological collections.” Journal of theoretical biology 13 (1966): 131-144.

[21] Cover, Thomas M., and Joy A. Thomas, Information theory and statistics, Elements of Information Theory (1991)

[22] Conceio, Pedro, and Pedro Ferreira. “The young person’s guide to the Theil index: Suggesting intuitive interpretations and exploring analytical applications.” (2000).

[23] Iceland, John, “The multigroup entropy index (also known as Theils H or the information theory index).” US Census Bureau. Retrieved July 31 (2004): 2006.

[24] Serebrenik, Alexander, and Mark van den Brand. “Theil index for aggrega- tion of software metrics values.” Software Maintenance (ICSM), 2010 IEEE International Conference on. IEEE, 2010.

[25] Serebrenik A, van den Brand MGJ. “Theil index for aggregation of software metrics values”. In Int. Conf. on Software Maintenance. IEEE, 2010; 19.

[26] Vasa R, Lumpe M, Branch P, Nierstrasz OM. “Comparative analysis of evolving software systems using the Gini coefficient”. In Int. Conf. on Software Maintenance. IEEE, 2009; 179188.

[27] Vasilescu, Bogdan, Alexander Serebrenik, and Mark GJ van den Brand. “Comparative study of software metrics aggregation techniques.” Proceedings of the International Worskhop Benevol 2010 (2010).

[28] Goeminne M, Mens T. "Evidence for the Pareto principle in Open Source Software Activity". In Proc. Intl Workshop SQM 2011. CEUR-WS workshop proceedings, 2011.

[29] Vasilescu, Bogdan, Alexander Serebrenik, and Mark van den Brand. "You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics." Software Maintenance (ICSM), 2011 27th IEEE International Conference on. IEEE, 2011.

[30] Vasilescu, Bogdan, Alexander Serebrenik, and Mark Van den Brand. "By no means: A study on aggregating software metrics." Proceedings of the 2nd International Workshop on Emerging Trends in Software Metrics. ACM, 2011.

[31] Mordal, Karine, et al. "Software quality metrics aggregation in industry." Journal of Software: Evolution and Process 25.10 (2013): 1117-1135.

[32] Sthamer, Harmen-Hinrich. The automatic generation of software test data using genetic algorithms. Diss. University of Glamorgan, (1995)

[33] Tonella, Paolo. "Evolutionary testing of classes". ACM SIGSOFT Software Engineering Notes. Vol. 29. No. 4. ACM,pp. 119-128 (2004)

[34] Repository: http://www.inf-cr.uclm.es/www/mpolo/stvr/