Estimating *t*-way Fault Profile Evolution During Testing

D. Richard Kuhn[1], Raghu N. Kacker[1], Yu Lei[2]

[1]*National Institute of Standards and Technology*
Gaithersburg, MD 20899, USA
{kuhn,raghu.kacker}@nist.gov

[2]*Computer Science & Engineering*
University of Texas at Arlington
Arlington, TX, USA
ylei@uta.edu

Abstract**: Empirical studies have shown that most software interaction faults involve one or two variables interacting, with progressively fewer triggered by three or more, and no failure has been reported involving more than six variables interacting. This paper introduces a hypothesis for the origin of this distribution, with implications for removal of interaction faults and reliability growth.**

Keywords – combinatorial testing; software fault; testing

I. Introduction

Empirical studies have shown that software interaction faults involve 1 to 6 variables, with no failures involving more than six reported. Interaction faults are denoted as *t*-way faults when *t* factors or variables induce the fault. For example, if a fault occurs when $x > 10$ *and* $y < 55$, this is a 2-way fault. Table 1 and Fig. 1 show the cumulative percentage of failures at different interaction *t* values, for a variety of applications, with the average indicated in Table 1 (headings keyed to references). For consistency, single factor faults are denoted 1-way faults. Thus for the various applications, the proportion of failures caused by 1-way or single factors ranged from 9% to 67%, and the proportion caused by either 1-way or 2-way faults ranged from 47% to 97%.

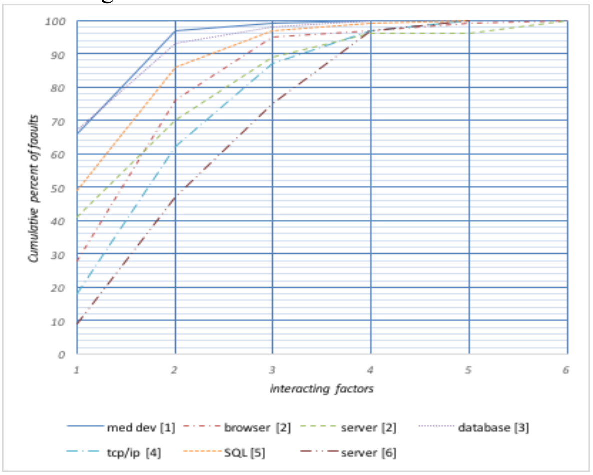
Table I. Cumulative Percent of failures at *t* = 1..6

t	[1]	[2]a	[2]b	[3]	[4]	[5]	[6]	average
1	66	28	41	67	18	9	49	39.71
2	97	76	70	93	62	47	86	75.86
3	99	95	89	98	87	75	97	91.43
4	100	97	96	100	97	97	99	98.00
5		99	96		100	100	100	99.00
6		100	100					100

The fault distributions were derived from failure reports for fielded software products, including medical devices [1], browser [2]a and server [2]b, TCP/IP [4], server [5], and SQL [6]. An additional distribution is from initial testing of a large distributed database application [3]. Empirically derived fault distributions such as these have provided the rationale for advances in the field of combinatorial testing over the past decade. While the distributions have been documented and analyzed thoroughly, relatively little is known about why the distributions have this consistent form or how they evolve as systems are tested and used. While it seems natural for more complex faults to be less common than simpler faults, we want to go beyond such a simple qualitative hypothesis and develop a model for estimating how the proportion of *t*-way faults varies with *t*, as testing or use progresses. We propose an explanation based on the two assumptions below.

- *t*-way faults occur in proportion to *t*-way conditions in code
- *t*-way faults are removed in proportion to *t*-way combinations in inputs

Fig. 1. Distribution of failures at *t* = 1..6



II. Analysis

For an estimate of the proportion of *t*-way conditions in code, we use the distribution of conditions in a collection of 7,685 branching statements from four avionics applications [7]. Note from Table II that this distribution is relatively close to the distribution of t-way faults discovered in initial testing in a database system described in reference [3].

Table II. *t*-way conditions, branch statements vs. Initial Test

t:	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Branch cond %	74.1	19.6	4.5	1.2	.3	.1	.1	.1
Initial test [3]	67	26	5	2	0	0	0	0

As software is tested or used, interaction faults will be discovered when a *t*-way combination that triggers a fault occurs in a set of inputs. Each set of inputs includes $C(n,t)$ combinations at each level of *t*, for *n* variables, where $C(n,t) = n!/t!(n-t)!$. For variables with *v* values each, the total number of combination settings is $v^t \times C(n,t)$, so each test or input set can cover $1/v^t$ of the total number of settings. The number of values, *v*, must of course be at least 2, but may be larger. As *t* increases, the proportion of combinations covered in each test is reduced, i.e., the proportion of $(t+1)$-way combinations covered is $1/v$ of the proportion of t-way combinations covered.

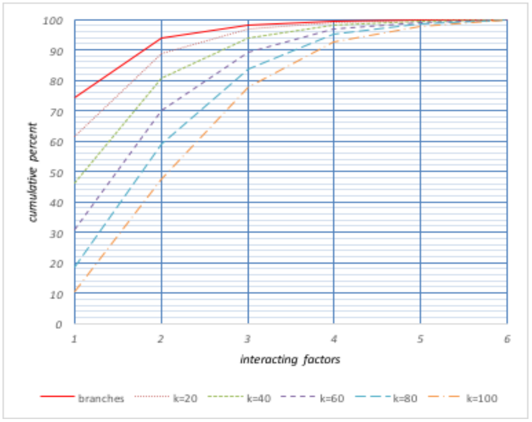
We make the simplifying assumption that 1-way faults are removed at rate r for some number of test sets, and the proportion remaining after k sets will be $(1-r)^k$. Since the discovery of a t-way fault depends on the presence of t-way combinations in input, and the proportion of $(t+1)$-way faults is $1/v$ of t-way faults, the fault discovery rate will be reduced by this proportion, or r/v for 2-way, r/v^2 for 3-way, etc. We can consider 2 the minimum value of v, and boolean or binary variables are also extremely common in practice. Then for k test sets we would have $(1 - r)^k$ 1-way faults remaining, $(1 - r/2)^k$ 2-way faults, $(1 - r/4)^k$ 3-way faults, and so on.

Now consider the evolution of the fault distribution as tests are run. Table III shows an example starting from the assumption that t-way faults occur approximately in proportion to the occurrence of t-way conditions in branching statements. For a fault detection rate of $r = .05$ per test set, $k = 48$ sets will produce a nearly matching value for the proportion of 1-way faults in the average of Table I. But the distribution of faults for $t = 2..6$ is also quite close to the average, as would be predicted if these faults are removed in proportion to $r/2$, $r/4$, $r/8$ etc. For example, starting with 74.1 1-way faults, after 48 test blocks, we would have $74.1(1 - .05)^{48} = 6.3$ 1-way faults; $19.6(1 - .05/2)^{48} = 5.8$ 2-way faults, $4.5(1 - .05/4)^{48}$ 3-way faults, etc. Normalizing this to 100%, we have the distribution shown in Table III, line (2), which is quite close to the average (3).

TABLE III. FAULTS REMAINING AT $t = 1..6$ AFTER 48 SETS OF TESTS, $r = .05$

t:	*1*	*2*	*3*	*4*	*5*	*6+*
Orig distrib %	74.1	19.6	4.5	1.2	0.3	0.3
After 48 sets	39.9	36.7	15.6	5.6	1.6	0.6
Avg, Tbl 1	39.7	37.6	15.5	6.6	1.0	1.0

FIG. 3. FAULT DISTRIBUTION FOR $t = 1..6$ AS TESTING PROGRESSES.



Notice that in Table III, after 48 test sets the proportion of faults for lower levels of t declines, while the proportion for higher t increases. Because an individual test contains a higher proportion of 1-way combinations than 2-way, the 1-way faults decline faster than others, and thus represent a smaller proportion of total remaining faults after testing. In general, t-way faults will decline faster than u-way faults for any $u>t$. This is consistent with intuition, as 2-way faults are in some sense "simpler" than 3-way faults, and thus likely to be found more quickly. Thus experience suggests that as testing progresses, the proportion of simpler faults should be reduced faster than more complex faults, shifting the distribution curves down at lower levels of t. This shift can be seen clearly in Fig. 3, which shows the proportion of faults at each level of t left after sets of tests for $r = .05$.

Fault reduction continues as bugs are detected in fielded products, and this process would result in different distributions of faults at each level of t, depending on how extensively a product is used. Data reported in two studies allow us to consider this model for a specific product. Both [2] and [5] report bug data for the Apache server, for two periods: 2001 – 2002 [2], and 2002 – 2006 [5], although some variation is likely introduced as versions were changed. Comparing columns [2]b and [5] in Table I, it can be seen that the proportion of less complex (lower t-way) faults is reduced over the time period, as expected. Starting from the distribution in [2], with $r = .05$ and $k = 54$ test sets, the distribution evolves as shown in Table IV.

TABLE IV. FAULTS REMAINING AT $t = 1..6$ AFTER 54 SETS OF TESTS, $r = .05$

t	*1*	*2*	*3*	*4*	*5*	*6+*
Rpt [2]	41	29	19	7	0	4
Rpt [5]	9	38	28	22	3	0
54 test sets	9.1	26.2	34.1	17.8	0	13.0

III. CONCLUSIONS AND IMPLICATIONS FOR TESTING

Preliminary results suggest that the model described in Sect. II is relatively successful in reproducing the fault distributions observed in empirical data. Additional empirical data will be needed to evaluate validity thoroughly.

The most significant implication for testing is that t-way interaction faults for $t = 4, 5, 6$ are exceedingly difficult to discover without tests specifically designed as covering arrays to include all t-way combinations at these levels.

IV. REFERENCES

[1] D.R. Wallace, D.R. Kuhn, "Failure Modes in Medical Device Software: an Analysis of 15 Years of Recall Data", *Intl J. Reliability, Quality and Safety Engineering*, vol. 8, no. 4, 2001.

[2] Kuhn, D.R. and Reilly, M.J., An investigation of the applicability of design of experiments to software testing. *27th Annual NASA Software Engineering Workshop, 2002.*. (pp. 91-95). IEEE.

[3] Kuhn, D.R., Wallace, D.R. and Gallo Jr, A.M., 2004. Software fault interactions and implications for software testing. *IEEE Trans Soft Eng,30*(6), pp.418-421.

[4] Bell, K.Z. Optimizing Effectiveness and Efficiency of Software Testing, PhD Diss, North Carolina State University, 2006.

[5] Cotroneo, D., Pietrantuono, R., Russo, S., & Trivedi, K. (2016). How do bugs surface? A comprehensive study on the characteristics of software bugs manifestation. *J.Systems and Software*, *113*, 27-43.

[6] Z. Ratliff, R.Kuhn, R. Kacker, Y.Lei, K. Trivedi, The Relationship Between Software Bug Type and Number of Factors Involved in Failures, submitted to Intl Wkshp Combinatorial Testing, 2016.

[7] Chilenski, J. J. *An investigation of three forms of the modified condition decision coverage (MCDC) criterion*. FAA. 2001.