MÄLARDALEN UNIVERSITY
SWEDEN

MDH reference: 2018/0071

Model of Fault Distribution in Complex Safety-Critical Systems

Abstract – **The suggested model is developed for the purpose of investigating the relationship between test coverage and its effect on a given fault distribution in large complex safety-critical n-parameter software systems. The faults are represented by subspaces of the entire volume which represents the entire input space of the system. The behavior of the system is considered to be either correct or incorrect. Inside the subspaces the system behaves erroneously. The shape of the subspaces have no meaning only the size of its volume. A uniform distribution of test points leads to predictable and quantifiable fault detection.**

Keywords **– Model of input space, system, fault distribution, test coverage, software, complex, safety-critical, system, uniform distribution**

I. Introduction

Modelling and visualizing the *input space* **and** the *faults* of a complex system, with many parameters, is quite a challenge. A solution to the multi-dimensional problem not only needs to be theoretically accurate, it also needs to visualize the information in a way that is comprehensive and understandable. Attempts to go beyond three dimensions typically struggle with either having to adjust the data or the visualization in some clever way.

A model in 3D is suggested, where the (multi-dimensional) input space of the system is represented by the volume of a cube 1x1x1. In this cube, the faults are represented by rectangular prisms, see Fig. X. These smaller volumes or subspaces are randomly distributed, both in size and in their placement. The volumes of the prisms correspond to their respective probability of occurrence, provided that the input space is randomly sampled with a uniform distribution. Meaning that a prism volume of 0.01 corresponds to a 1% probability of occurrence if any random point from the full volume is selected. The fault occurs for all points that lie within or on the limits of the subspace. The approach to view probabilities as *fault sizes* has been suggested before [7].

This paper elaborates on the research question;

Is such a 3D-model a viable representation of a fault distribution in a multi-dimensional system?

The proposed model aims at aiding visualization and further analysis of the prediction of the effect of a given test effort, in terms of fault detection.

Questions of interest in this context; Is it worthwhile to increase the number of performed tests? What can be expected in terms of fault detection for a given test effort?

II. Complex Safety-Critical Systems

Large complex software systems are considered to be hard to test as the number of system states prohibits exhaustive testing [2][3]. This is problematic especially for safety-critical applications where a failure of the system can result in severe consequences, e.g. loss of lives.

The assumptions made in this paper are;

1. *The SUT (System Under Test) behaves deterministically.*
2. *All parts of the code is reachable (during test).*
3. *All parameters are defined.*
4. *The size of the system makes it virtually impossible to test exhaustively.*

This well describes many safety-critical systems. Assumptions 1, 2 and 3 are consequences of how this type of software is developed. Strict development standards, e.g. DO-178C [4] and ISO 26262 [5], regulate the development process. Emphasis is placed on determinism (of the code) and removal of unreachable code. If the code, executable or not, cannot be tested it must be removed.

In this safety-critical context, faults are not tolerated. The faults are therefore not classified in to categories of severity. All detected faults must be dealt with.

III. Probabilities as fault sizes/volumes and tests as points?

If a system is assumed to contain faults with certain probabilities, they can be modelled as volumes or subspaces of the larger input space, see Fig. 3. Subsequently, the testing process, which samples the input space, can be represented by a number of (test) points in a 3D-plot. Consequently, if the entire input space volume was filled with such points, all the subspaces would contain test points as well – meaning that all faults have been detected. This model can describe the relationship between the faults detected and the number of test points.

An unbiased uniform distribution is implemented as the default test point generation, in the model. A uniform distribution that gradually saturates the input space, with test points, gives predictability. Smaller and smaller volumes can go undetected (remain hidden) in the swarm of points, as its density increases. The model allows for quantification of this relationship.

Other test point distributions can be implemented as well. However, in the context of safety-critical software, the goal is to identify and remove all faults (regardless of severity). Alternative distributions of test points would require further discussions on the feasibility, of the now random, placement of the faults. When an independent and uniform fault

distribution is used **together** with an independent and uniform test point distribution, the question of where the faults are located is bypassed. If all fault subspaces were to be placed in one half of the input space volume the number of remaining and detected faults would not change. This provided that the level of overlap does not change. For (realistic) smaller fault volumes this effect is negligible. All inserted points have the same probability of encountering a fault.

If for instance a normal distribution of test points were to be used (for the purpose of simulating a test focus on a more executed part of the code), then the question of whether the faults are "in the middle" or near the edges must be dealt with.

IV. Why Fault Modelling in 3D?

In general, a main advantage of a 3D-model is that it is easy to visualize and understand. However, there are other aspects to using a three dimensional representation. Empirical data shows that 89-99% of all faults are caused by three parameters or less [1]. The remaining part is triggered by no more than six parameters. This means that a 3D-model can be seen as a feasible and in many cases realistic representation of a fault's expanse in the input space, where three parameters cause a fault.

The suggested model assumes that faults are not (primarily) present in the input space as isolated points but rather as small volumes, subspaces.

This concept can be exemplified by an Automatic Braking System, for cars, that fails under certain conditions. Assume that such a system, for some reason, is malfunctioning when the speed of the car is 50 km/h and traffic in front of the car is 20 m ahead and moving at a slower speed of 15 km/h. Under these circumstances the Automatic Braking System should engage. If this problem also occurs when the parameters are changed a little, e.g. 49.2 km/h, 20.6 m and 16.3 km/h, a subspace emerges, where the problem happens.

Assume that this problem occurs when the parameters are within the ranges listed in Table 1.

In a 3D-visualization of the entire input space where these three parameters constitute the axes, the fault can be represented by subspace, as shown in Fig. 2.

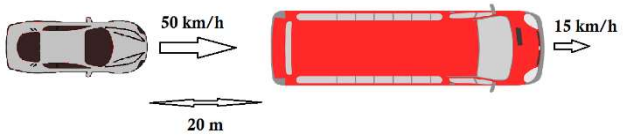

Fig. 1 Automatic Braking System, failure scenario.

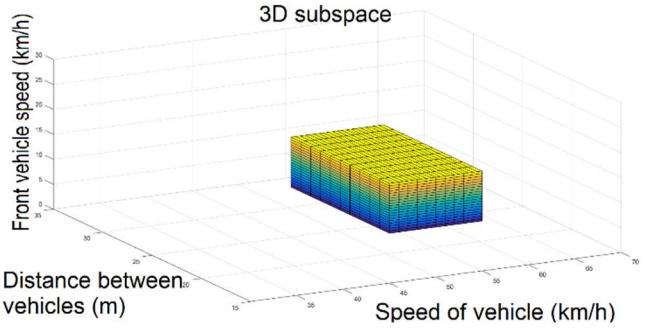

Fig. 2 Three parameters forming a subspace.

The model represents fault probabilities as corresponding volumes. As long as the proportions are correct the shape of the fault volume is not important. The model does not aim on giving a realistic representation of such aspects.

TABLE I. Parameter ranges for fault

Parameter	Range where the fault occur
Speed of the vehicle	45 – 55 km/h
Speed of the vehicle ahead	10 – 20 km/h
Distance between the vehicles	15 – 25 km/h

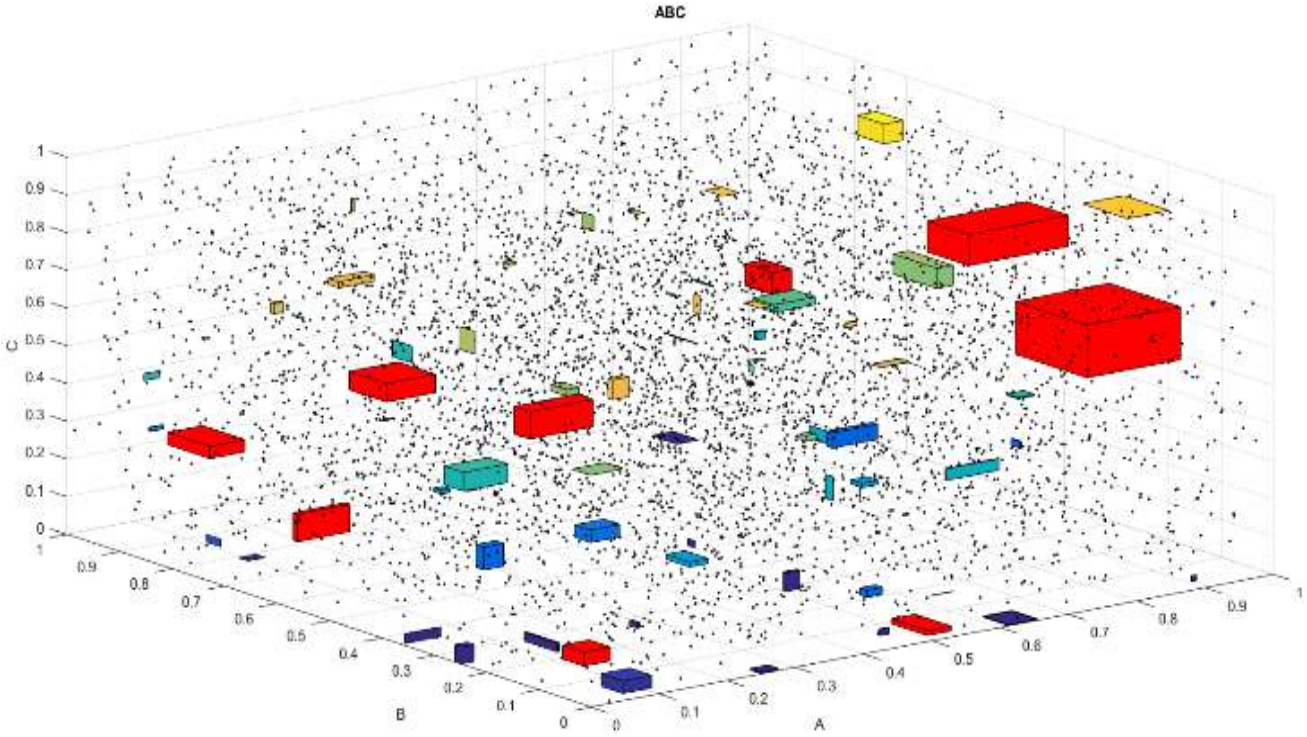

Fig. 3 An input space of a system with three parameters. There are 99 errors comprising 0.5% of the input space. The points represent 5000 tests that have been performed. The red prisms contain points and are detected.

V. How can a 3D-model represent n-parameter systems?

The 3D-model can represent complex systems with many parameters. Considering that empirical data show that 89-99% of all errors are caused by three parameters or less [1]. The remaining part is triggered by no more than six parameters.

In a combinatorial view, each test point represents that all possible combinations are tested. This holds true for a system with many parameters as well. If a system has a higher number of parameters the number of tested combinations, in each test point, is increased even more.

It does not matter which combination of parameters or which of their values that cause the system to crash or perform incorrectly. For a given relative proportion (percentage) and number of faults the model will yield a statistically correct result, in terms of test coverage vs fault detection.

A. Example: Faults in a four parameter system

Assume a system with four parameters: A, B, C and D. Here the test points, see Fig. 4, represents that all four possible three-way combinations have been tested, i.e. ABC, ABD, ACD and BCD.

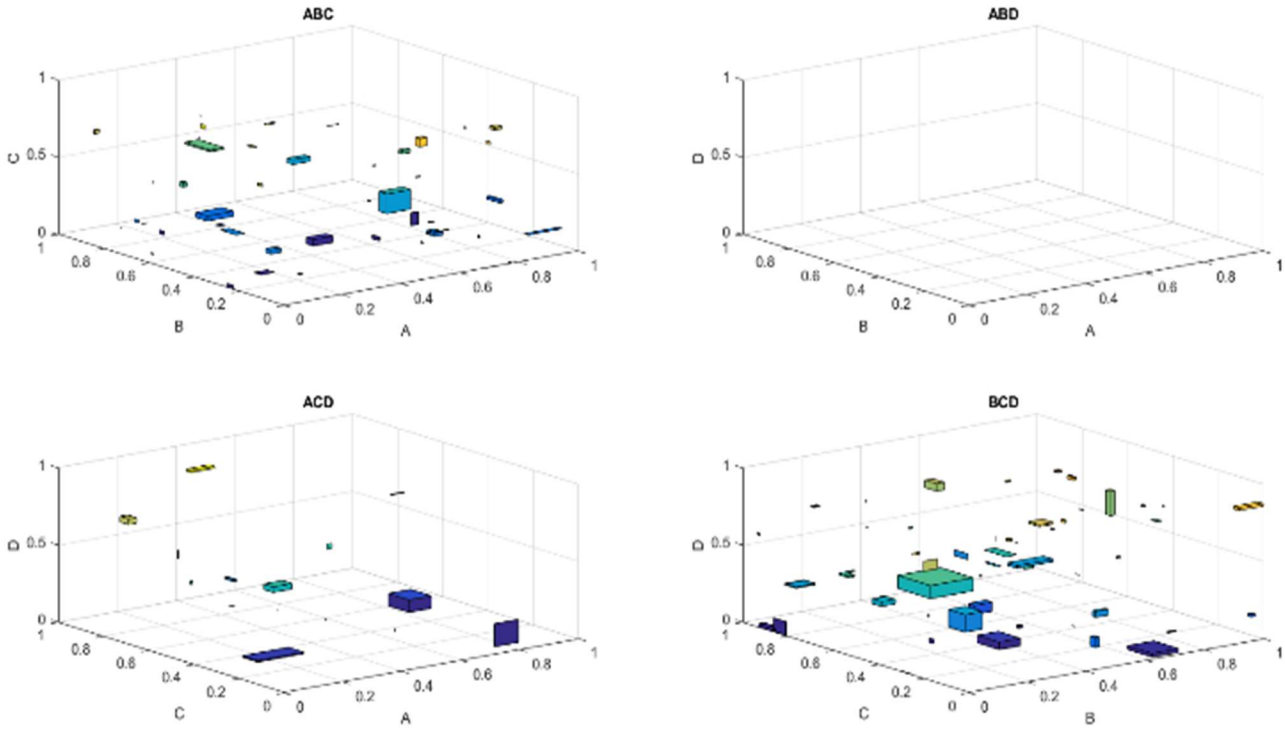

Fig. 4 The four possible three-way combinations of A, B, C and D. Random unique faults have been placed in each subspace. ABD is fault free.

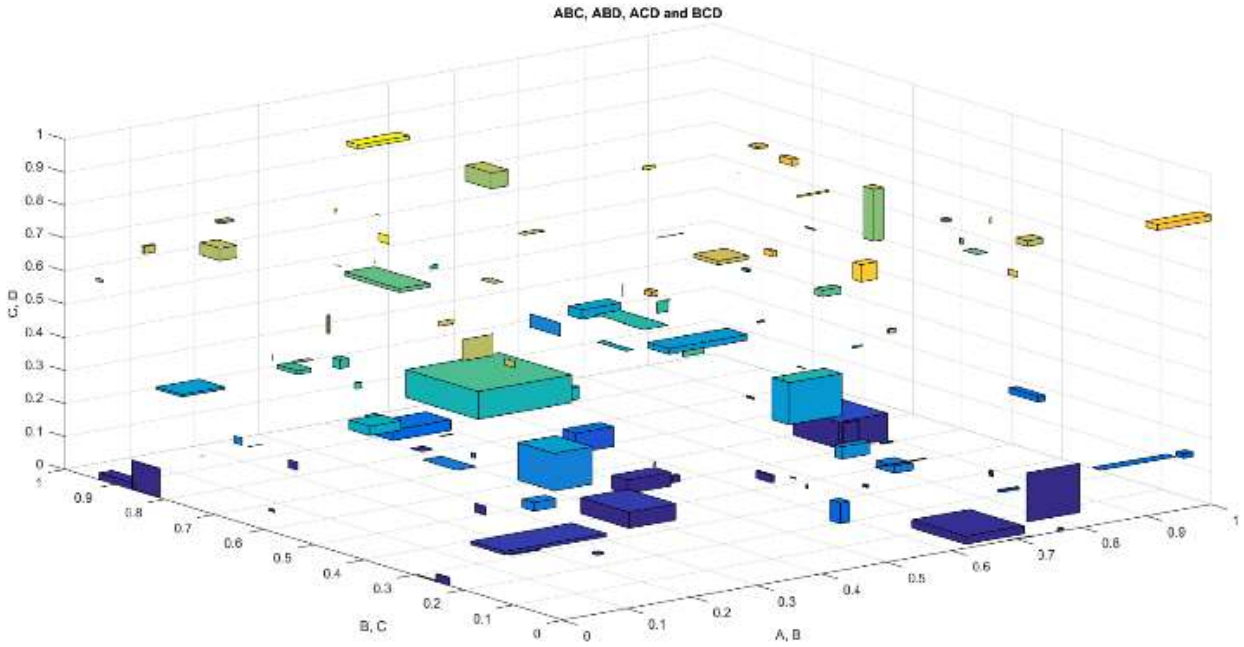

Fig. 5 The aggregate input space of the subplots in Fig. 4.

In Fig. 4, the four different combinations are shown. For each subplot the full volume represents the entire input space of the system projected on the three axes. Each point in the volumes of the subplots in Fig. 4, belongs to the input space of the entire system.

For a point p in the ABC space, $\forall p_{ABC} \in \mathbb{R}^3_{ABC}$, there are corresponding points in the other combinations, i.e.

$\forall p_{ABC} \exists p_{ABD}$ ($p_{ABD} \in \mathbb{R}^3_{ABD}$)

$\forall p_{ABC} \exists p_{ACD}$ ($p_{ACD} \in \mathbb{R}^3_{ACD}$)

$\forall p_{ABC} \exists p_{BCD}$ ($p_{BCD} \in \mathbb{R}^3_{BCD}$).

This means that the subspace of each combination overlaps the other, i.e. $\mathbb{R}^3_{ABC} \cup \mathbb{R}^3_{ABD} \cup \mathbb{R}^3_{ACD} \cup \mathbb{R}^3_{BCD}$.

The faults shown in each of the subplots, in Fig. 5, are unique and only appears for one specific combination of parameters. Since all subspaces overlap the fault volumes from different combinations can be aggregated.

The sum of the fault volume for a subplot:

$$\mathrm{E}\mathbb{R}^3 = \sum_{k=1}^{ne} \varepsilon\mathbb{R}^3{}_k$$

ne is the number of faults.
$\varepsilon\mathbb{R}^3$ is the volume of an individual fault.

Hence, the sum of all fault volumes in the system:

$$Etot\mathbb{R}^3 = \sum_{i=1}^{nc} \mathrm{E}\mathbb{R}^3{}_i$$

nc is the number of three-way combinations.

B. Representation of n-parameter faults

When a fault is caused by other than three parameters they can still be represented by in 3D. All faults have a likelihood of occurring. As long as the probability corresponds to a volume of the same relative proportion, the representation is correct.
Assume a system X with n parameters. One of the parameters is a boolean b that is either TRUE or FALSE. The boolean b is defined for all inputs of X. When b is TRUE the system X fails. If b is just as likely to be TRUE as FALSE, X will fail in 50% of the cases, regardless of the number of other parameters (n). The input space for b completely overlaps the input space of X.
Fig. 6, shows the input space for a boolean that causes a fault whenever it is TRUE. Provided that no other faults exist the input space of the complete system would look the same, i.e. a fault volume covering 50% (failing in 50% of the possible inputs).

VI. Test coverage model

The model is written as a function in MATLAB, version R2016b, see appendix A. The inputs are percentage of the input space that are erroneous, number of tests and a parameter used to control the granularity of the fault distribution.

A typical function call:

randomTestingFaultDetection3d (0.5,10000,0.9)

which means

0.5% of the input space is erroneous
10000 test points
0.9 as the "granularity factor"

1) *The input space of the system:*
   The volume 1x1x1 represents the entire input space of the system or the input space being tested.
2) *The colored blocks:*
   The faults are represented by blocks. Only the volume of the block have a meaning representing the size of the fault. The shape is <u>not</u> representing a realistic expansion of a fault. All blocks are independent of each other and can overlap. If a test point lies within a block, the block is colored red, meaning the fault has been detected.
3) *The axes:*
   Above system has three parameters A, B and C. The range of the parameters have been normalized to values between 0 and 1.
4) *The points:*

The points represent performed tests, test points. If a point lies inside a block the fault has been detected, i.e. a perfect test oracle is assumed. The generation of test points are uniformly distributed.

A. Generation of fault distribution

The generation of the random fault distribution is done by a while loop. The user input are the *granularity factor* and the *total fault probability*. The while loop can (somewhat simplified) be written as;

```
while generated fault probability < total fault probability

    rep = remaining fault probability;
    rnd  = random number; % 0-1, (uniform)
    newFault = rep * rnd * granularity factor;
    allFaults = allFaults + newFault;
    generated fault probability = sum(allFaults)

end
```

A granularity factor of 1 gives approximately 50 faults and 0.1 about 56.000.
This allows for flexible generation of faults, see Fig. 7 and Fig. 8. When many faults are generated (by a low granularity factor) the distribution turn into a geometric distribution. This can describe a number of phenomena [6]. Moreover, it can be seen as a conservative alternative, as it has a long "tail". This long tail represent smaller faults with a low probability of occurring.
After each simulation the fault distribution is presented in a plot as well as summary of which faults have been detected.

The model is available as executable MATLAB m-files. Please contact the author of this paper.

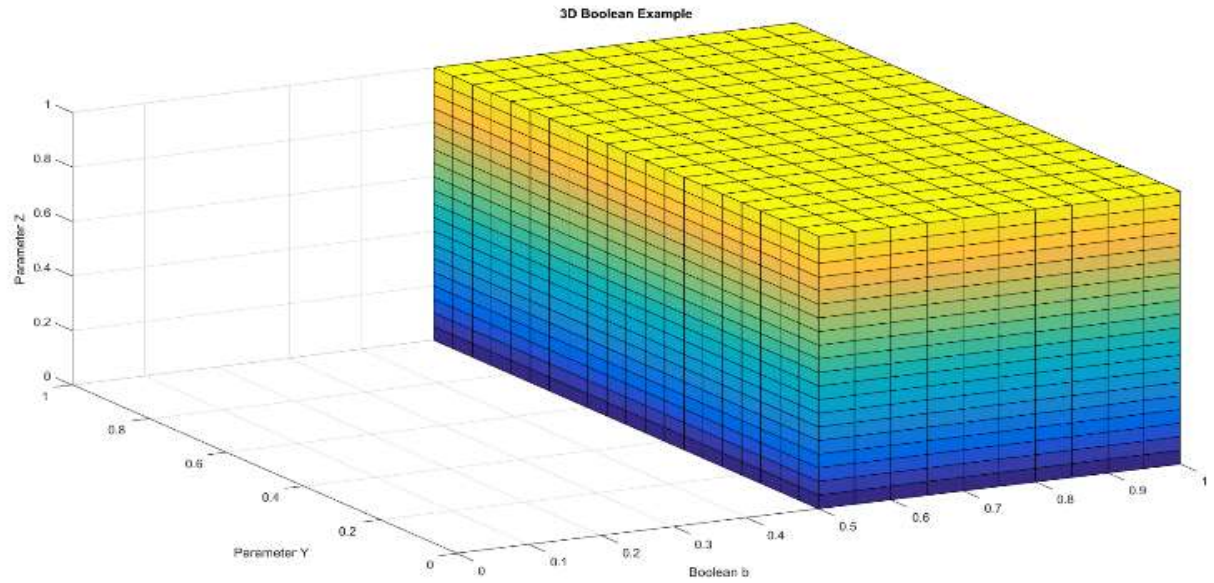

Fig. 6 A system that fails when boolean b is TRUE. Here the fault comprise 50% of the total volume

VII. CONCLUSIONS

The suggested 3D-model can provide a simplified representation of a fault distribution in a complex safety-critical system. Such a system can be assumed to behave deterministically and to have no unreachable code [4],[5]. The model uses a uniformly distributed random placement of the fault volumes/subspaces. Together with a uniformly distributed random selection of test points the placement and shape of the faults become unimportant. This permits quantification of the effect, in terms of fault detection, of a given test effort. If the shape of the faults, or their placement were to be changed, the result would be the same. The same amount of fault probability would be detected and same amount would remain undetected.

The purpose of the model is to gain a better understanding of the relationship between the number of tests and fault detection.

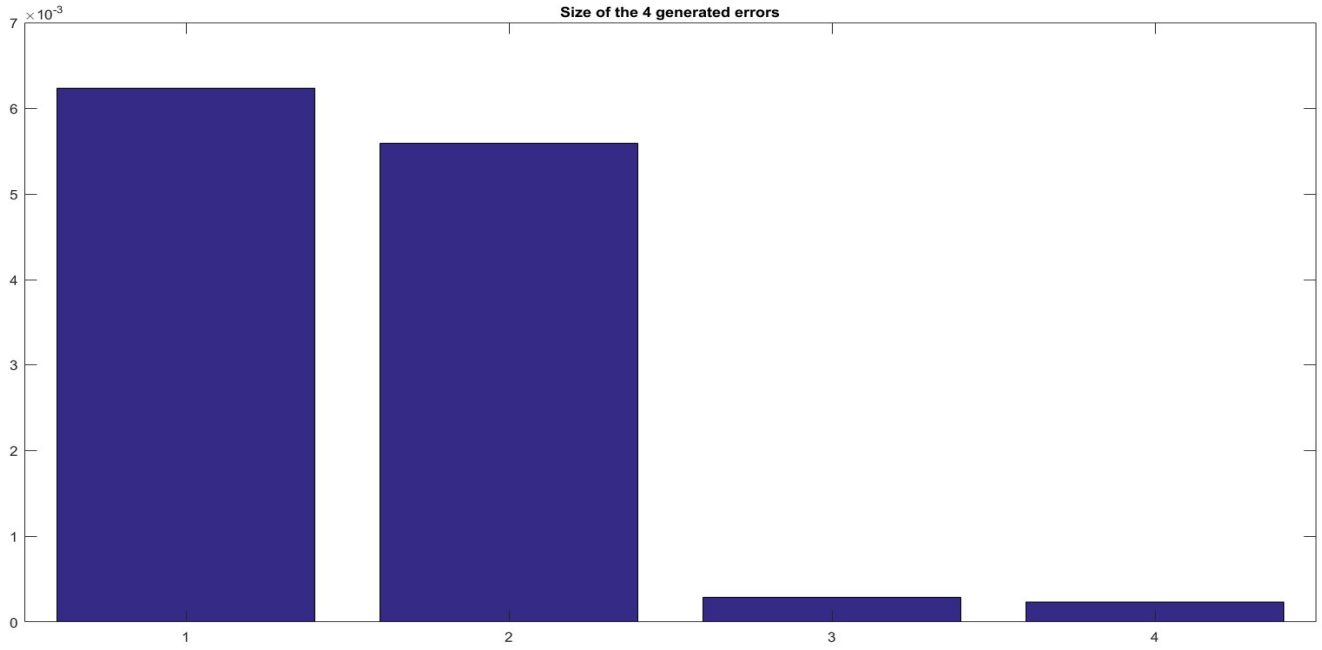

Fig. 7. A low number of faults generated. Y-axis represent the probability of each fault. X-axis shows the number of each fault.

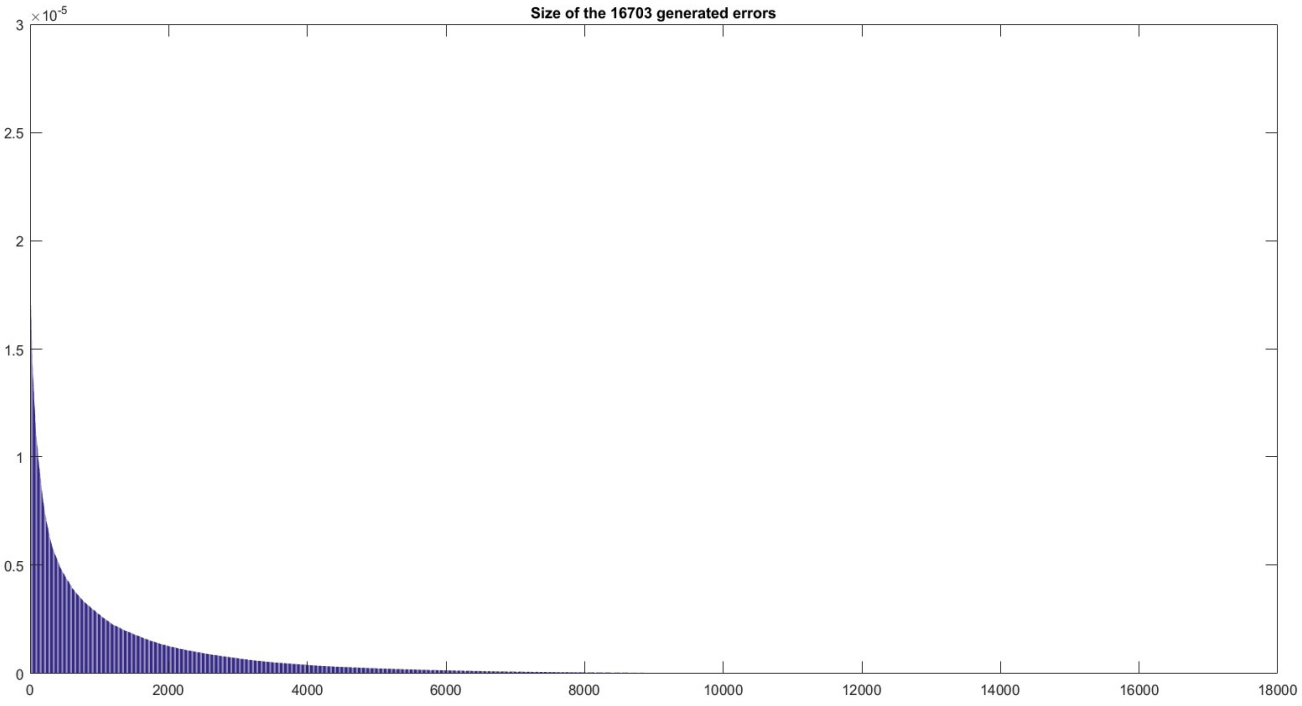

Fig. 8. A high number of faults generated. Y-axis represent the probability of each fault. X-axis shows the number of each fault.

REFERENCES

[1] Computer Security Resource Center homepage. [Online]. Available: https://csrc.nist.gov/projects/automated-combinatorial-testing-for-software/interactions-involved-in-software-failures

[2] J.C. Knight, Safety critical systems: challenges and directions,
Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on Software Engineering.

[3] David L. Parnas, A. John van Schouwen, and Shu PO Kwan, "Evaluation of Safety-Critical Software," *1991 ACM Nineteenth Annual Computer Science Conference*.

[4] RTCA/EUROCAE, "DO-178C, Software Considerations in Airborne Systems and Equipment Certification", 2011.

[5] International Organization for Standardization (ISO), "ISO 26262, Road vehicles – Functional safety", 2011.

[6] W. Feller, "An Introduction to Probability Theory and Its Applications", third ed., vol. 1. Wiley, 1968.

[7] Jeffrey M. Voas, Christoph C. Michael* Keith W. Millert.
"Confidently Assessing a Zero Probability of Software Failure".
1993 SAFECOMP '93, The 12th International Conference on
Computer Safety, Reliability and Security.