Software, the Urn Model, and Failure

Peter Bernard Ladkin
Version 3 of 2015-02-25

Abstract: IEC 61508-7:2010 Annex D explicates some features of the statistical evaluation of software through its operational history. But it raises many questions. One is: how does the traditional urn model apply, if at all, to software? A second is: to what kinds of software does the reasoning in Annex D apply? A third is: do the confidence-interval numbers apply to all distributions of inputs? Recently, I have experienced reliability-engineering experts giving wrong answers to the second and third questions. It seemed worthwhile to explain correct answers in terms understandable also by non-professionals. This paper attempts to do so.

IEC 61508-7:2010 Annex D includes a brief (and, some would say, obscure) introduction to the statistics of evaluating the reliability of software. Annex D relies on software execution being construed as a Bernoulli process, which is a sequence of Bernoulli trials (a "random" operation resulting in one of two outcomes, traditionally denoted"success" or "failure", with a certain probability of "success") with fixed probabilities of success for each trial [BedCoo01 Chapter 12]. This conception of software reliability has been developed in the software engineering literature for over forty years, and has had success in assessing the reliability of software for UK nuclear power plant protection systems using statistical methods [Lit10]. It is the basis for some of the most well-known literature in the power and pitfalls of assessing software statistically for ultra-high reliability [LitStr93,LitStr11].

The figures in Annex D Table D.1 are based upon calculations considering software failure as a Bernoulli process, defined to be a sequence of independent Bernoulli trials with a constant probability of success per trial [Lit10]. The salient information from Table D.1 is reproduced below in Tables 1 and 2. They are discussed further in [LadLit15].

Acceptable probability of failure to perform design function on demand	Number of observed demands without failure for a confidence level of 99%	Number of observed demands without failure for a confidence level of 95%
$< 10^{-1}$	4.6×10^{1}	3×10^{1}
$< 10^{-2}$	4.6×10^{2}	3×10^{2}
$< 10^{-3}$	4.6×10^{3}	3×10^{3}
$< 10^{-4}$	4.6×10^{4}	3×10^{4}

Table 1: Example Operational-History Requirements for On-Demand Functions

Acceptable probability of failure to perform design function per hour of operation	Number of observed hours of operation without failure for a confidence level of 99%	Number of observed hours of operation without failure for a confidence level of 95%
$< 10^{-5}$	4.6×10^{5}	3×10^{5}
$< 10^{-6}$	4.6×10^{6}	3×10^{6}
$< 10^{-7}$	4.6×10^{7}	3×10^{7}
$< 10^{-8}$	4.6×10^{8}	3×10^{8}

Table 2: Example Operational-History Requirements for Continuously-Operational Functions

The so-called “urn model” is often used as an example of a Bernoulli process, along with tosses of a single coin [Fel68]. One reason for this is that the urn model was the basis for Bernoulli’s considerations on his eponymous process, in his posthumous 1713 manuscript *Ars Conjectandi* [Ber1713].

The urn model, Bernoulli trials and Bernoulli processes

Consider an urn containing M balls of two colors, white and black. There are w white balls, and thus (M-w) black balls. A Bernoulli trial consists of drawing a ball “at random” (that is, without any evident selection criteria) from the urn, and (correctly) observing the color.

There are two phenomena to be commented here. One is what is meant by a “random” selection. The second is observing the color.

What is meant by “random” is that each individual ball has an equal likelihood of being selected by a single trial. Obviously many physical confounding factors could enter in to vitiate this hypothesis, for example, the urn could be too narrow for all the balls to lie flat, and those nearer the top of the urn may be more likely to be selected by a lazy selector than those near the bottom. But the Bernoulli analysis is based upon the hypothesis being true.

We have the following situation:

- each ball has an equal likelihood of being selected;
- such selections are mutually exclusive (if one ball is selected, all others are thereby not selected),
- there are M balls; and
- it is certain that some one will be selected (likelihood = 1 conventionally),

Suppose we represent likelihoods numerically. It is nowadays generally accepted that the Kolmogorov axioms pertain [Hac01]. This entails that the likelihood of any one of a finite set of mutually exclusive events occurring is equal to the sum of the individual likelihoods of those events. We have M mutually-exclusive events of equal likelihood, of which one must occur. So the sum of the equal likelihoods is 1. It follows that each individual likelihood is 1/M.

Second, the color must be correctly recorded. Say a blind person performs the trial, and randomly chooses to report the result as “white” or “black” independently of the true color of the selected ball. Suppose that, in fact, there are M white balls and no black balls in the urn. The blind reporter will report, over the long term, equally many “white” selections as “black” selections. Whereas in fact only white balls have ever been selected. The reports are no guide to the reality of the selection.

A sequence of Bernoulli trials forms a Bernoulli process when each trial is identical in set-up. Suppose one Bernoulli trial is performed on the urn. Say the result is white. Then there is one white ball selected, that is, outside the urn. There are (M-1) balls left in the urn, of which (w-1) are white, and (M-w) are black. Each ball left in the run now has a likelihood 1/(M-1) of being selected in a second Bernoulli trial. It follows that the two Bernoulli trials do not have the same parameters. The likelihood of selection of a white ball through a “random” selection is now (w-1)/(M-1), and that of a black ball is (M-w)/(M-1), which is larger than (M-w)/M. The likelihoods of “random” selection have changed. The two trials do not form a Bernoulli process.

If the selected ball is replaced in the urn after completion of the Bernoulli trial (and actions are undertaken, such as stirring the urn, to ensure the likelihood of selection of any ball is again equal to that of any other), then another Bernoulli trial on the same urn will have identical parameters to

that of the first trial. The two trials will form a Bernoulli process.

The Bernoulli trial for the purposes of executing a Bernoulli process is thus conceived, not solely as the "random" selection of a ball (selecting a ball in a manner in which selection of any ball is equally likely), but as "random" selection followed by replacement of the ball in the urn, along with actions, such as stirring the urn, to restore the likelihoods of selection of each ball.

Amongst other results, it is argued that, long term, the proportion of white balls amongst all balls selected in the Bernoulli process consisting of unbounded Bernoulli trials on the urn tends to converge to w/M. (The phrase "tends to converge" needs some explanation!) This is known as Bernoulli's Theorem, and is the special case of the Law of Large Numbers for a two-outcome series of trials. Bernoulli's Theorem and the urn model is discussed in more depth in [Hac01, Chapter 16, Stability]. Notice that the results described by Bernoulli's Theorem concern the true outcomes of the Bernoulli process, not the reported outcomes from, say, the blind person.

The reported outcomes from the blind person also form a Bernoulli trial, but a different one, to which the urn is only accidental – a stimulating artefact, one might say. The blind person chooses "randomly" from the urn to report white/success or black/failure. Such a report is a Bernoulli trial with M=2 and w = (M-w) = 1, so the process is a Bernoulli process with the same parameters, and Bernoulli's Theorem says that in the long term it becomes very likely that the proportion of reported-white and reported-black outcomes converge to approximately ½ each of the total outcomes.

The balls in the urn are reported as being selected with equal likelihood. We'll keep that property of the trials, but add some additional features. Consider now that the balls, as well as being white or black, are uniquely numbered with an integer, say with blue numbering. And let's say consecutively from 1. If each ball has equal likelihood of being selected in a trial, then we can consider that we have M "inputs" to the trial, numbered 1 through M, each "input" resulting in success (white) or failure (black).

The result of the trial is functional upon the selected ball: ball k is white, or ball k is black, and this remains so throughout the Bernoulli process – there is no repainting of balls. If we regard the ball numbers as input, and the ball colors as output, this relation is given by the function

$$\text{Color: } \{1,2,.....M\} \rightarrow \{\text{success, failure}\}$$

The scenario envisaged in a Bernoulli trial involves each ball having an equal likelihood of selection. So each input to Color has an equal likelihood of selection in the trial. We say that inputs of the function Color are *uniformly distributed* in the trial and in the process.

Now consider the following scenario. We have now S balls, $\{b_1, b_2, ... b_S\}$and each ball is labelled in blue with its item number as before (b_1 is labelled with a blue "1", b_2 with a blue "2", and so on). And let us suppose there are more balls: S > M. Now we add a red number, from 1 to M, which we call the intermediate number, to each ball. The function

$$\text{Distr: } \{1,.....,S\} \rightarrow \{1,2,...M\}$$

gives the labelling of each ball: the individual ball number in blue is input to Distr, and the output is its intermediate number in red. Further , let us assume the ball coloring, white or black, is consistent with the function Color on the intermediate number, namely,

Ball b_k is white if and only if (Color(Distr(k)) = white

We may perform Bernoulli trials, and thus construct a Bernoulli process, on the urn full of two-colored-numbered balls just as we may on the original urn with unnumbered balls. But now we have more that we can record, and we may use the numbers to give us more information about the process.

Consider the function

$$D(j) = \text{number of balls with intermediate number } j$$
$$= | \{m : Distr(m) = j\} |$$

The function D is called the *distribution* of the intermediate numbers in the urn. If each ball has an equal likelihood of being selected in a Bernoulli trial, then

Intermediate number j has a likelihood of being selected of D(j)/S

because D(j)/S is the proportion of balls with intermediate number j in the urn. Suppose we perform the Bernoulli process, and record at each trial not only the outcome ("success", "failure") but also the intermediate number. Then we have a Bernoulli process which is consistent with Color, but in which the intermediate numbers are distributed according to D. We can consider the Bernoulli process as giving us information about the function from red number to ball color.

Bernoulli's Theorem holds for any Bernoulli process; in particular it holds for the two-colored-labelled balls. This means that the long-term selection process tends to select balls with intermediate numbers approaching the distribution D (again with a caveat concerning the phrase "tends to").

Software execution and Bernoulli processes

Suppose we have a computer program P which takes input at various times and returns output. We assume that P is deterministic, that is, given a specific input i from input-domain I, the program always returns specific o from output-domain O: Alternatively, there is a mathematical function

$$F_P: I \rightarrow O$$

such that P inevitably returns the output value $F_P(i)$ when it terminates computing on input i.

Suppose also that there is a specification for P: SP, such that

$$S_P: I \rightarrow O$$

We define the *satisfaction function of P*, $Sat_P: I \rightarrow$ {"success", "failure"}as follows

$$SatP(i) = \text{"success" if } SP(i) = FP(i)$$
$$SatP(i) = \text{"failure" if } SP(i) \neq FP(i)$$

We may define a program trial as follows. P commences execution in initial state *Init.* P receives input i, and the trial records SatP(i).

I claim that a program trial can be construed as a Bernoulli trial as follows. Say you have an

unlimited supply of white and black unlabelled balls. For each time the program P is executed, choose a ball from the supply, label the ball in blue with a number: say, the first unused number in a register of consecutive numbers; strike the number through as "used" when the ball is labelled. Then label the ball "i" in red, where i is the input to this execution of P. Finally, ensure the ball is colored white if SatP(i) = "success", and black if SatP(i) = "failure". If the ball is the wrong color, then pick a new ball of the right color, inscribe it with the same blue and red numbers as the wrong-color ball, throw the wrong-color ball away, and put the right-color ball in the urn.

After a considerable time, the urn is filled with consecutively-blue-numbered balls. Given ball b_k, the function Distr(k) gives the input to program P on the k'th test, and the color of b_k records whether the execution on input Distr(k) was successful or failed. The distribution of inputs to program P is given by the function D.

The urn represents the operational history of program P. Suppose a Bernoulli process is now started on the urn. The salient characteristics of this process are exactly those derivable from the operational history of program P. Note that the distribution of inputs to the Bernoulli process is exactly the distribution of inputs to program P in the operational history of program P. The Bernoulli theorem holds; and chances of any events of interest are given by the usual centuries-old mathematics.

Practical Considerations

In order to construe P as deterministic, it suffices to ascertain that

- P executes a series of atomic actions with unambiguous semantics; that is, when P executes an atomic command, the resulting memory state is uniquely specified as a function of the preconditional memory state in which the atomic command was executed;

- P always commences execution from a defined memory state, the initial state *Init*.

P thus starts from a defined state, executes a sequence of atomic actions of which, for each action, the post-state is by hypothesis uniquely determined from the pre-state of the memory in which the action was executed along with input i.

We may without loss of generality consider the input i to be residing in a specific memory location at the start of the execution of P.

I believe that these considerations on P may be somewhat relaxed. For example, if P includes a nondeterministic atomic operation O, then the execution of P may be split into two halves: P1 up to and including the execution of O; and a number of programs PF1, PF2, ... following O, each one computing on one post-execution memory state following O. There will be a distribution of outputs to O. This distribution will be the activation distribution of PF1, PF2, etc.
Let us assume that O is the only nondeterministic operation in P. Then the output to P on input i will consist in the distribution of outputs to PF1, PF2, which is exactly the distribution of outputs to O.

These considerations may be generalised to any number of nondeterministic operations in P, but at risk of combinatorial explosion of the outputs to P. This would render the theoretical considerations here impractical to ascertain.

Furthermore, taking these nondeterministic operations into account in the way indicated here would require some decomposition of the operation of the program P in line with the architecture of P. This is not what is usually considered in "black box" evaluation of software as considered in IEC

61508-7:2010 Annex D.

A significant assumption in the notion of Bernoulli process is that the likelihoods of success or failure remain constant. This is obviously so for our mathematical construal of program execution. Crucial is that in each trial the program P starts from the same initial state. If it computed on memory, part of whose value was retained from previous computations of some sort, there would be no way of guaranteeing that the Bernoulli-process assumption of constant probability of success pertained.

It is further important that the success or failure of the program execution is correctly determined. Consider again the blind recorder. However, in practice failures may be overlooked (for example, masked, or just simply overlooked). The Bernoulli-process mathematics rests, however, on failure detection being perfect.

It is important to note that this is just one set of considerations on software, on programs P, that allow their behavior to be modelled by the urn and thereby as a Bernoulli process. For all I know, the considerations may be relaxed, maybe considerably, and the urn model thereby more widely applicable to more types of software.

Comments on a Proposal

In a recent note [Anon15], a reliability expert has claimed that the estimations of failure rate, and the non-failure statistics required for specified levels of confidence in those rates, included in Table D.1 of IEC 61508-7:2010 Annex D (and reproduced here as Tables 1 and 2) are valid only for the uniform distribution of input values.

Consider an arbitrary finite distibution D. Let us construct an urn as above, with red numbers exhibiting the distribution D. Selection from the urn is uniformly distributed amongst the balls.

Suppose it is legitimate to conclude somehow: *after X trials without selecting a black ball, I am entitled to infer, to a level of confidence Y, that the probability of selecting a black ball is Z.* Notice this is a property of the experiment along with the proportion of balls colored black versus white. It has nothing to do with blue or red labels.

The balls are uniformly distributed. [Anon15] suggests that the conclusion is legitimate (for appropriate values of X, Y, and Z). The selection of balls according to their blue labels is uniformly distributed (considering the mapping *blue-labels → {white, black}*). But the same experiment also gives us ball selection according to the distribution D, by paying attention to the red labels, to the the mapping *red-labels → {white, black}*. But the conclusion holds irrespective of which mapping I pay attention to: *after X trials without selecting a black ball, I am entitled to infer, to a level of confidence Y, that the probability of selecting a black ball is Z.* Thereby as true for D as for the uniform distribution.

Notice in particular that this conclusion holds whatever the preferred means of coming to the conclusion about the probability of selecting a black ball. It is entirely independent of interpretation, whether classical, frequentist, Bayesian or what-have-you.

Further, [Anon15] suggests that the only software to which the Bernoulli-process intepretation applies is software which makes no use of internal memory, so-called “stateless” software. Our considerations above show that the execution of deterministic software from an initial state constitutes a Bernoulli trial, and thus repeated execution a Bernoulli process. There is no condition on memory use arising from our construal of a the execution of a deterministic program P forming a

Bernoulli process, contrary to what [Anon15] suggests.

References

[Ber1713] J. Bernoulli, Ars Conjectandi, Basel, 1713.

[Anon15] Anonymous, Determining Software Safety: Understanding the Possibilities and Limitations of International Safety Standard IEC61508-7 Annex D , preprint presented to DKE AK 914.0.3 on 10 February 2015, dated January 2015.

[Fel68] W. Feller, An Introduction to Probability Theory and its Applications, Volume 1 Third Edition, John Wiley and Sons, 1968.

[Hac01] I. Hacking, An Introduction to Probability and Inductive Logic, Cambridge University Press, 2001.

[LadLit15] P.B. Ladkin and B. Littlewood, Practical Statistical Evaulation of Critical Software, submitted for publication.

[Lit10] B. Littlewood, personal communication, January 2010.

[LS93] B. Littlewood and L. Strigini, Validation of ultra-high-dependability for software-based systems, Communications of the ACM 36(11):69-80, 1993.

[LS11] B. Littlewood and L. Strigini, “Validation of ultra-high dependability...” - 20 years on, Safety Systems 20(3):6-10.