Estimating worst case failure dependency with partial knowledge of the difficulty function

Peter Bishop[1,2] and Lorenzo Strigini[1]

[1]Centre for Software Reliability, City University, London, UK
{pgb,strigini}@csr.city.ac.uk
[2]Adelard LLP, London, Exmouth House, London, UK
pgb@adelard.com

Abstract—For systems using software diversity, well-established theories show that the expected probability of failure on demand (*pfd*) for two diverse program versions failing together will generally differ from what it would be if they failed independently. This is explained in terms of a "difficulty function" that varies between demands on the system. This theory gives insight, but no specific prediction unless we have some means to quantify the difficulty function. This paper presents a theory leading to a worst case measure of "average failure dependency" between diverse software, given only partial knowledge of the difficulty function. It also discusses the possibility of estimating the model parameters, with one approach based on an empirical analysis of previous systems implemented as logic networks, to support pre-development estimates of expected gain from diversity. The approach is illustrated using a realistic safety system example.

Keywords. safety, software reliability, fault tolerance, failure dependency, software diversity, difficulty function.

1 Introduction

Software diversity has been advocated as a means of improving the reliability of safety related software and in particular safety systems that react to a demand, where a 1 out of 2 or a 2 out of 3 voting scheme can be used to ensure that some safety action is performed. This approach is used in industry (e.g. for railway interlocking), but development and maintenance is costlier than for a non-diverse system and it is not easy to predict in advance the likely safety improvement that can be achieved.

Theory as well as experimental studies indicate that failures of diverse implementations ("versions") are not necessarily independent [3, 7, 15]. The challenge is to determine *how much* improvement should be expected with diversity. Early theoretical work by Eckhardt and Lee [4] showed that variations in the degree of "difficulty" for different inputs (or "demands") will result in the expected *pfd* for a pair of diverse programs being greater than the product of the expected *pfd*s of the two programs and thus limit the effectiveness of diversity. Littlewood and Miller [8] later showed that, if diversity in development results in different "difficulty functions" for the two diverse

programs, the expected *pfd* for common failures of a diverse pair can also be *less* than the product of the expected *pfd*s of the single versions.

To quantify via these theories the improvement in expected *pfd* for a diverse safety system, one would need to specify both the difficulty for every demand, and the demand profile. Difficulty functions will normally be impossible to estimate for real projects (although *a posteriori* difficulty estimates have been obtained [1, 15] for some "toy" applications where many different versions were developed [12]). In addition, if the safety system is used in different operational contexts, the demand profile might also be different and this can change the expected *pfd*s.

In this paper we examine an approach for a more modest, but still useful, goal of estimating the worst case improvement in average *pfd*, by deriving a worst case demand profile that only requires knowledge about two points on the difficulty function rather that characterizing the whole function.

The paper will first summarize the theory underlying the difficulty function, then identify the worst demand profile that maximizes the expected *pfd* for a pair of diverse programs, relative to the expected *pfd* of a single version for a given difficulty function. We then consider what estimates for the expected *pfd* can be derived given different types of knowledge about the difficulty function. We also discuss means, and difficulties, for estimating the model parameters and tentatively suggest an approach for systems implemented as logic networks.

2 The difficulty function

In these models, the process that delivers a program, with its unknown faults (if any), is modelled as the random sampling of a program from a "population of all possible programs". The (unknown) probability of "drawing" each specific possible program depends on the specification, the development process, the development team, etc. Given these factors, the "difficulty function" $\theta(x)$ is defined as the probability that such a "randomly drawn" program will fail on a given demand x.

The mean *pfd*s of a single program (pfd_1) and of common failure for a pair of diverse programs (pfd_2) depend on the difficulty function $\theta(x)$ and the demand profile $p(x)$. For a difficulty function $\theta(x)$, the expected *pfd* of a single program version is:

$$pfd_1 = \sum \theta(x)\, p(x) \tag{1}$$

We consider the case in which the two versions for a 1-out-of-2 system are developed from the same process for the same specification: the two developments have the same "difficulty function" (Eckhardt and Lee model [4]). Thus pfd_1 is the same for both programs; among all the scenarios where this holds, this scenario, of identical difficulty functions, yields the highest (i.e., the worst) value of pfd_2.

Assuming conditional independence between failures of the versions for each demand [4] (i.e., the two developments are independent [11]), the expected *pfd* for a randomly drawn pair of programs is then:

$$pfd_2 = \sum \theta(\mathrm{x})\ \theta(x)\, p(x) \tag{2}$$

pfd_2 increases with the variance of the difficulty function $\theta(x)$. Intuitively, if the difficulty function is very "spiky", there is a high probability that the diverse programs will fail on the same, "difficult" demands: the average benefit from diverse programs will be lower than otherwise. Conversely if the difficulty function is "flat" (the same value for all demands), there is nothing that forces the diverse programs to fail on similar inputs, so the gain in average reliability is higher. In this case, pfd_2 does equal the product between the expected *pfd* values for the two versions. For brevity, when this equality holds we will say there is "independence on average"[1].

Clearly the mean *pfd* depends upon the demand profile $p(x)$ as well as the demand difficulty $\theta(x)$. In the next section we use this dependence on the demand profile to derive the worst case value of pfd_2 for a given value of pfd_1.

3 Estimating the worst case expected *pfd*

We compare the expected *pfd* for this system with that of a single-version (i.e., non-diverse) system used in the same function (hence with the same demand profile).

To determine the worst-case impact of the profile on the expected *pfd*, we choose the family of the most extreme profiles possible: the ones in which the only demand values with non-zero probabilities are $x=hi$ and $x=lo$ that have the highest and lowest values of the difficulty function, $\theta(hi)$ and $\theta(lo)$. For this profile, we can write:

$$pfd_1 = z\theta(hi) + (1-z)\theta(\mathrm{lo}) \tag{3}$$

$$pfd_2 = z\theta(hi)^2 + (1-z)\theta(\mathrm{lo})^2 \tag{4}$$

where $z = p(lo)$ and $(1-z) = p(hi)$. So pfd_1 and pfd_2 can vary between their minimum and maximum values depending on z, i.e.:

$$\min(pfd_2) = \theta(lo)^2, \quad z = 0 \tag{5}$$

$$\max(pfd_2) = \theta(hi)^2, \quad z = 1 \tag{6}$$

No other profile can achieve this range, as non-zero probabilities for demands with intermediate θ values would reduce the maximum and increase the minimum value achievable. There is also another sense in which these profiles are "extreme". Given a certain difficulty function, a given value of pfd_1 can in general be the result of many different profiles, only one of which is "extreme"[2]. This "extreme" profile is the one

---

[1] We underscore that "independence on average" is a property of expected values, not of individual program pairs. If independence of failures held for every pair of diverse programs, "independence on average" would also hold. However, "independence on average" could hold even if independence does not hold within each pair; and we can have $pfd_2 > pfd_1^2$ even in a population of pairs in which pairs with negative or zero correlation between failures of their component versions are more common than pairs with positive correlation [10].

[2] To be precise, we should consider the cases in which more than one demand values have values of θ equal to $\theta(hi)$ (or equal to $\theta(lo)$). The reasoning presented here remains valid: we can treat all the demands with an identical value of θ as one demand.

where pfd_2 is largest, i.e., the advantage of diversity is smallest. Under this profile, the expected *pfd* of a diverse pair, pfd_2, is a linear combination of $\theta(lo)^2$ and $\theta(hi)^2$ and hence a linear combination of (min pfd_1)^2 and (max pfd_1)^2. This can be compared with the "best case" profile that only selects demands x with exactly the same difficulty, i.e. where $\theta(x)=pfd_1$, so from (2) that $pfd_2=pfd_1^2$ which represents "independence on average". The expected *pfd*s for the best and worst profiles are shown in Fig. 1 below for the case where $\theta(hi)$=0.04, and $\theta(lo)$ takes different values from zero to 0.02.

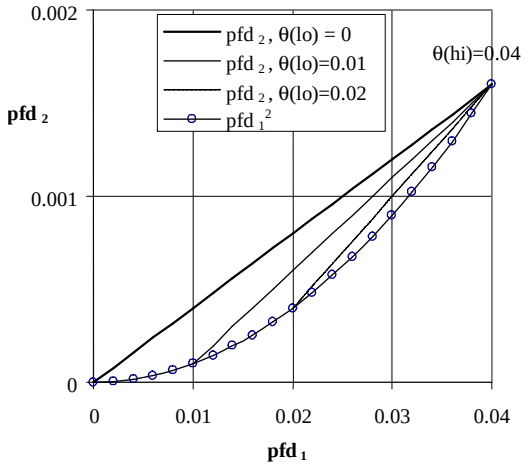

Fig. 1. Variation in pfd_2, given the extreme profile and $\theta(hi)$=0.04

To understand this graph, we consider that for a given value of $\theta(lo)$, the lowest possible value of pfd_1 is $\theta(lo)$, given by setting z=0 in equation (3), and the corresponding value of pfd_2 equals $\theta(lo)^2$ (equation 4). This is why the straight lines for $\theta(lo)$=*0.01* and $\theta(lo)$=0.02 do not continue to the left of these points. The maximum values of pfd_1 and pfd_2 are given by setting *z=1*, and correspond to the rightmost point in the graph, $pfd_1=\theta(hi)$, $pfd_2=\theta(hi)^2$. All the intermediate "extreme" profiles for the same values of $\theta(lo)$ and $\theta(hi)$ give the {pfd_1, pfd_2} pairs represented by the points of the straight line joining these maximum and minimum points on the pfd_1^2 curve.

We now study the effects of various levels of knowledge about $\theta(hi)$ and $\theta(lo)$.

3.1 Case where θ(hi) and θ(lo) are known

When θ(hi) and θ(lo) are known, the endpoints of the linear combination are known and changing z changes the ratio of pfd_2 to pfd_1, since from equation (3):

$$z = \frac{\theta(hi) - pfd_1}{\theta(hi) - \theta(lo)} \quad \text{and} \quad 1 - z = \frac{pfd_1 - \theta(lo)}{\theta(hi) - \theta(lo)}$$

From equation (4) we can calculate pfd_2 as a linear function of pfd_1, i.e.:

$$pfd_2 = \theta(lo)^2 + \frac{(\theta(hi)^2 - \theta(lo)^2)(pfd_1 - \theta(lo))}{\theta(hi) - \theta(lo)} \quad (7)$$

We note that if $pfd_1=\theta(lo)$ or $pfd_1=\theta(hi)$, then $pfd_2 = pfd_1^2$. This is not surprising: these cases select profiles where all points have the same difficulty value: effectively a flat difficulty function, which is known to imply "independence on average". It also follows that the reduction factor pfd_2/pfd_1 will vary between $\theta(lo)$ and $\theta(hi)$.

3.2 Case where only θ(hi) is known

If $\theta(lo)$ is unknown, then the worst case assumption is that $\theta(lo) = 0$, and hence equation (7) reduces to the following, known bound on pfd_2:

$$pfd_2 \leq \theta(hi)\, pfd_1 \quad (8)$$

3.3 Case where θ(hi) and θ(lo) are not known

The worst case assumption is now $\theta(lo) = 0$, $\theta(hi) = 1$ and hence equation (8) reduces to the extreme case where the bound is:

$$pfd_2 \leq pfd_1 \quad (9)$$

In this case, the mean *pfd* of a diverse pair could be no better than that for a single version. The worst case – the equality in (9) holds – would mean that the programs developed have fail on some specific demands and no others (with probability 1). Comparison with the previous cases highlights how knowledge about the difficulty function allows us to reduce the mean value of pfd_2 as a proportion of pfd_1.

3.4 Case where the ratio between θ(hi) and θ(lo) is known

In this case we only know the maximum "roughness" of the difficulty function, k, the ratio of $\theta(hi)$ to $\theta(lo)$, but not the absolute difficulty values. Hence

$$k = \frac{\theta(hi)}{\theta(lo)} \quad (10)$$

In this model, there is no constraint on the x axis endpoints $\theta(hi)$ and $\theta(lo)$ apart from the ratio k (and the fact that $0 \leq \theta(lo) \leq pfd_1 \leq \theta(hi) \leq 1$). The worst case value of pfd_2 lies on the chord between the two endpoints: as illustrated in Fig. 2 below.

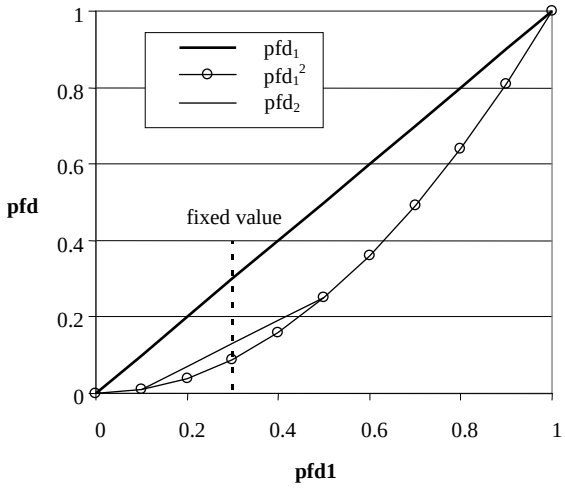

Fig. 2. Worst case given a known difficulty ratio *k*

To get the worst case for a given pfd_1, we effectively slide the linear combination chord for pfd_2 along the pfd_1^2 curve, while keeping the ratio between the endpoints of the chord (on the pfd_1 axis) equal to k. Then we choose the chord that gives maximum (worst) possible value of pfd_2 for a given value of pfd_1. Since pfd_1 is held constant, maximizing the ratio pfd_2/pfd_1 also maximizes pfd_2/pfd_1^2 (the worst case increase relative to independence on average). The analysis in Appendix A shows that pfd_2 is bounded by:

$$pfd_2 \le \frac{(k+1)^2}{4k} pfd_1^2 \tag{11}$$

So the worst case reduction factor for pfd_2 relative to pfd_1 is $pfd_1 \cdot (k+1)^2/4k$, rather than the factor of pfd_1 that would result from "independence on average".

This worst case bound equation is only applicable up to the point where $\theta(hi)=1$ (as we cannot slide the pfd_2 chord any further to the right). From the analysis in Appendix A, it can be shown that this limit is reached when:

$$pfd_1 \ge \frac{2}{k+1} \tag{12}$$

If pfd_1 exceeds this constraint, a variant of equation (7) has to be used instead where $\theta(hi)=1$ and $\theta(lo)=1/k$, i.e.:

$$pfd_2 = k^{-2} + \frac{(1-k^{-2})(pfd_1 - k^{-1})}{1-k^{-1}}, \qquad pfd_1 \ge \frac{2}{k+1} \tag{13}$$

The difficulty "roughness" parameter, *k*, could be a very large value or even infinity if $\theta(lo)$=0. Thus to forecast a *good* (low) worst-case pfd_2 we need $\theta(hi)$ to be not too high; counter-intuitively, we also need $\theta(lo)$ not to be too *low*: we need evidence that the probability of faults affecting any one demand will be "bad enough".

4 Numerical illustration

For some non-realistic programs, we have empirical difficulty data derived from an analysis of program versions from an archive of mathematical problems and solutions [12]. Analysis of a population of around 3000 initial releases of program versions for a relatively simple problem [1] yielded the difficulty surface (taking the observed frequencies as estimates of probabilities) shown in Fig. 3 below.

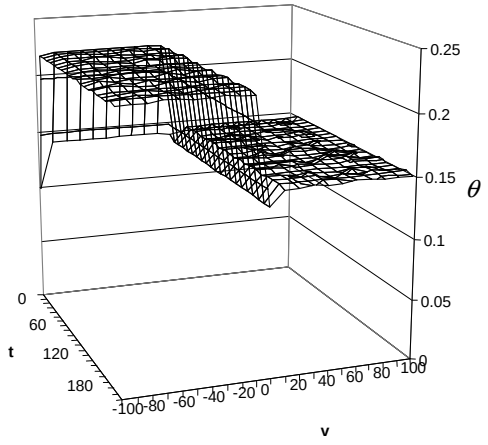

Fig. 3. Empirical difficulty function from around 3000 program versions [1]. These programs receive two inputs, t and v, hence the difficulty function is plotted as a surface.

Analysis showed that $\theta(hi)$=0.2282 and $\theta(lo)$=0.1012. That is, in this case k = 0.2282/0.1012 = 2.25.

Inserting these values into equations 7, 8 and 9, the relationship between these bounding equations is illustrated in Fig. 4.

The bound based on knowledge of both $\theta(hi)$ and $\theta(lo)$ (equation 7) represents the least pessimistic worst case bound on pfd_2. It can be seen that the curve based on an estimate of *k* (equation 9) touches the least conservative bound line at the maximum dependency point when pfd_1 = 0.140 (see Appendix A for details). Unlike equation 7, equation 9 allows the maximum dependency point that defines pfd_2 to be shifted if a new estimate of pfd_1 is made. This is equivalent to defining new values for $\theta(hi)$ and $\theta(lo)$ which retain the same ratio *k*.

The pfd_2 bound based on $\theta(hi)$ alone (equation 8) tends to be the most pessimistic, although it does intersect with the least conservative bound when pfd_1 = $\theta(hi)$

(0.2282) when equations 7 and 8 agree with the "independence on average" result when $pfd_2 = pfd_1^2 = \theta(hi)^2$.

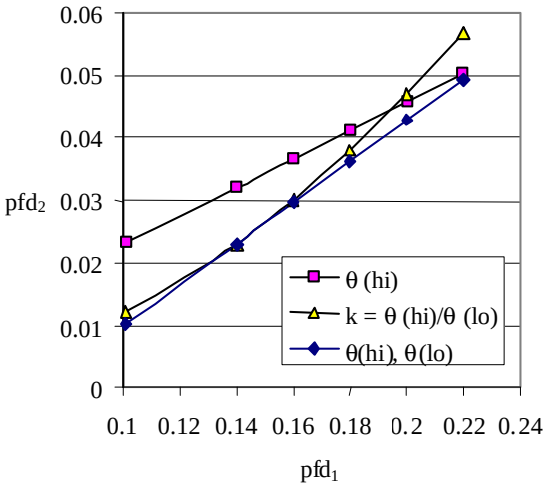

Fig. 4. Comparison of worst case bound equations. The labels for the three plots indicate which parameters are assumed known, with the values derived *a posteriori* for the study in **Fig. 3**.

5 Parameter estimation

In the analysis above we have identified alternative ways of deriving a worst case value for pfd_2 given pfd_1 by using different kinds of information about the difficulty function. To apply these models for prediction, as opposed to generic insight, we require realistic model parameters. At the current state of knowledge, there are no means for deriving credible difficulty values to support prediction (estimation of expected *pfd*) for a specific safety application. In the following subsections we examine some potential directions for deriving model parameters, and difficulties to be overcome for them to become applicable in the future.

5.1 Estimating *k*

Estimation of *k* could be based on relative complexity, if one assumed the demand difficulty for demand *x* proportional to some measure of demand logic complexity[3],

$$\theta(x) \propto c(x) \tag{14}$$

[3] We accept that this could only be a first-cut assumption. One of the possible objections is that higher complexity may lead to more effort to avoid or remove faults, so that the difficulty function might not be a linear function (or maybe not even a monotonic function) of a measure of complexity [5].

where $c(x)$ is a measure of the logic complexity required to process a given demand x. For example, the complexity measure could based on the number of lines of code or number of decision points in the code involved in a particular type of demand. Program analysis (such as code pruning) could be used to identify the relevant subset of code involved with each demand. Assuming such complexity measures can be derived, k would be estimated as:

$$k = \frac{\max_{x \in X} \theta(x)}{\min_{x \in X} \theta(x)} \approx \frac{\max_{x \in X} c(x)}{\min_{x \in X} c(x)} \tag{15}$$

This approach need not be restricted to conventional code. Many safety systems are represented as logic networks with a set of interconnected logic gates. Each type of demand requires different logic and sensor inputs to respond to different safety-related incidents on the associated plant. The complexity of the logic network could be used to estimate $c(x)$.

5.2 Estimating θ(hi) and θ(lo)

One means of estimating $\theta(hi)$ is to make use of the fact that, by definition:

$$\theta(hi) \leq p_{\text{faulty}} \tag{16}$$

where p_{faulty} is the probability that a program selected from the population of all possible program versions is faulty. The value of p_{faulty} could be estimated via a range of methods including an analysis of the development process [2] and the level of test coverage achieved [9], however they are not mature methods and confidence in their predictions will be especially low when few faults are expected to be present.

Alternatively, estimates of $\theta(x)$ could be made directly based on some expected fault density f [13] and an estimate of the amount of code or logic $n(x)$ needed to respond to a particular demand. There can be many demands x that exercise the same logic, but if we make a (strong) assumption that all demands are equally likely to fail if the logic is faulty (on average over the whole population), then:

$$\theta(x) = n(x)f \tag{17}$$

So the maximum and minimum values of $n(x)$ over all types of demand x can be used to obtain $\theta(hi)$ and $\theta(lo)$.

5.3 Empirical data analysis

Data sets that can be used for empirical parameter estimation are quite limited, but there is survey of programmable logic controller (PLC) logic faults undertaken for the UK Health and Safety Executive [14]. Such PLCs are not programmed using conventional code but as an interconnected set of logic elements (like AND, OR, NOT) or relay "coils" (that simulate the behavior of hardware relays). Field reliability data was collected from a range of industrial applications from the nuclear, chemical, oil and

gas and electrical industrial sectors (but predominantly nuclear). A full set of data was collected for 125 PLCs which included the number of:

- inputs
- outputs
- coils
- failures
- years of use

No PLC "platform" failures were recorded in over 600 PLC-years of operation, suggesting that PLC platforms are fairly reliable. We should also note that "coils" were used as the measure of the number of logic elements, but we cannot be sure the actual logic network contained coils or whether an "equivalent coils" estimate was calculated for alternative types of logic. In addition, the term "failures" in this data set might actually be a misnomer for "number of faults", as the failure counts are typically zero or one. We assumed "failures" indicate the number of different faults. This is conservative from the viewpoint of estimating maximum difficulty as it can only over-estimate the number of network logic faults. Furthermore, if over-estimation of faults were consistent, it would not affect the *k* ratio derived in Section 3.4.

On further analysis, we found that some of the PLCs contained identical logic network metrics and identical fault counts. This was interpreted to mean that the same logic network was installed on multiple PLCs, which could bias the result by counting the same logic network several times over. These duplicates were eliminated from the analysis, leaving size and fault data for 96 different PLC programs. The results are presented in Fig. 5 below.

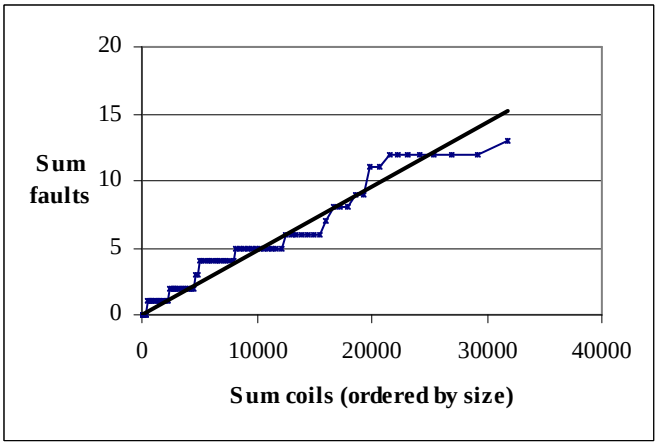

Fig. 5. Logic faults vs. network coils (cumulative)

As the fault data are relatively sparse, the graph presents a cumulative count of faults versus the number of "coils". The summation is performed in program size order, as this will reveal a non-linear relationship between PLC program size and faults by an increasing or decreasing gradient. As can be seen in Fig. 5, the relation-

ship does appear to be roughly linear (to 95% in a chi-squared test). Perhaps surprisingly, the slope seems to be slightly less for the larger PLC programs (at the right hand side of the graph).

A very similar graph was obtained when we used the sum of inputs and outputs (*io*) as the network size measure. Again, the correlation is better than 95% in a chi-squared test.

Given the evidence of a linear relationship between network size and logic faults, it may be legitimate to use an average fault density to estimate the number of faults in a logic network. The logic density estimates derived from the linear regression analysis of faults against the coil and *io* measures are shown in Table 1 below.

Table 1. Logic fault density estimates

Logic fault density measure		Value
f_{io}	(faults/io)	$3.0\ 10^{-4}$
f_{coil}	(faults/coil)	$5.0\ 10^{-4}$

Some caveats need to be placed on these fault density estimates, notably:

- The fault density is based on the number of fault *discovered* which could be an underestimate if some of the faults have yet to be found. The data in [14] shows the mean operating time is 5.9 years. It is not certain that all defects would have been detected over that period of time.
- Given the sparseness of failure data it cannot be demonstrated that linearity applies over the whole range of complexity. If the density is lower for small logic subsystems, *k* would be larger than predicted under the linearity assumption.
- We do not know which of the PLC faults were caused by errors in user requirements or implementation. Common user requirement flaws cannot be mitigated by diverse logic implementations. Section 6 provides an example where we assume the fault density estimates relate to diverse implementation faults.
- Fault density figures will differ depending on the application type and safety criticality.

So the empirically derived figures might be viewed as indicative “ball-park” figures, but should not be viewed as being applicable to a specific diverse system.

6 Example application

The application of the fault density quantification approach outlined in Section 5 is illustrated on an actual industrial safety system with two independent safety trains that control functionally diverse plant shutdown mechanisms[4]. The A and B train subsystems and logic element counts (taken from the logic specification sheets) are shown in Table 2 below.

[4] The details have been anonymised at the request of the system operators

Table 2. A and B Train subsystem network complexity measures

A Train subsystem	**I/O**	**Gates**
A1	24	16
A2	22	9
A3	49	28
A4	31	23
Total	126	76
B Train subsystem	**I/O**	**Gates**
B1	17	9
B2	28	16
B3	15	7
Total	60	32

We will use the data to illustrate the mean *pfd* estimation approach, using arbitrary (and quite extreme) assumptions about the variation of logic use with demands. Let us assume some logic subsystems are unnecessary for some demands. For example a standby electrical supply subsystem might be essential if connection to the grid power supply is lost but not otherwise.

So the extreme level of variation in number of subsystems with demand might be:

- One (the smallest) logic subsystem in A and B
- All logic subsystems in A and B

Using the *io* complexity measure we set an upper bound on the difficulty for a logic sub-network *x* containing $n_{io}(x)$ *io* elements as:

$$\theta(x) = f_{io}\, n_{io}(x)$$

Based on the $n_{io}(x)$ numbers assumed for the assumed maximum and minimum logic networks needed for a demand we derive the bounding values for θ and k shown in Table 3 below. Note that, in this particular example, the specified logic differs between trains A and B, so we are assuming that there are no common specification flaws and the difficulty values estimate the likelihood of implementation flaws only.

Table 3. A and B Train difficulty and difficulty variation estimates

Subsystem	A Train		B Train		Worst case
Demand *x* requires:	#io	θ(x)	#io	θ(x)	θ(x)
Smallest single subsystem	22	$6.6\ 10^{-3}$	15	$4.5\ 10^{-3}$	$6.6\ 10^{-3}$
All subsystems	126	$3.78\ 10^{-2}$	60	$1.8\ 10^{-2}$	$3.78\ 10^{-2}$
k	5.7		4		5.7

For the A and B trains, different logic is involved, so the values are not identical, but we conservatively assume the worst case value applies to both trains. In the following calculation, we assume that the expected value pfd_1 is 10^{-2} for each train. This is combined with the bound formulae for pfd_2 from Section 3 and the worst case values in

Table 3 above to yield the bounds on the expected value of pfd_2 shown in Table 4 below.

Table 4. Worst case A and B Train *pfd* estimates

Bound equation	pfd_2 bound
a) $pfd_2 \le p_{faulty} pfd_1$	$3.8\ 10^{-4}$
b) $pfd_2 \le \theta(hi)\ pfd_1$	$3.8\ 10^{-4}$
c) $pfd_2 \le (\theta(hi)+\theta(lo))\ pfd_1 - \theta(hi)\theta(lo)$	$2.5\ 10^{-4}$
d) $pfd_2 \le ((k+1)^2/4k)\ pfd_1^2$	$2.6\ 10^{-4}$

It can be seen that in this example the results are very similar. The results for a) and b) are identical because the $\theta(hi)$ estimate is based on all the logic in the train and hence is identical to p_{faulty} for the train. The results for c) and d) are also similar since the pfd_1 value chosen is close to the maximum dependency point.

7 Discussion

It should be noted that the analysis in this paper relates to *expected values*. The analysis seeks to estimate the improvement we might expect between the *averages* pfd_1 and pfd_2 by using diversity, under worst case assumptions about the operational profile. It *does not* predict the improvement ratio that will actually be achieved by a *specific pair* of program versions under a specific operational profile—clearly the actual reduction achieved could be better or worse than a model based on averages [10].

It is also important to note that this analysis describes the effects of those parts of development that are indeed diverse. For example, if there is a common specification, the possibility of a specification flaw would invalidate the conditional independence assumption required in the Eckhardt and Lee model [4]. Accounting for such common factors requires more complex models [11] and we have not checked how the approach would extend to those models.

The paper also explores possible directions for quantifying the model parameters. The analysis of PLC failure data in Section 5 illustrates how one might attempt to estimate ranges of difficulty. However the method discussed should be viewed as "work in progress", as it has not been validated or substantiated. In particular:

- It assumes that difficulty can be estimated from predicted fault density. This is not supported by deduction and could only be validated experimentally.
- There is an assumption of linearity between logic complexity and logic faults. This is not contradicted by the available empirical data but it would require far more empirical data to validate the assumption over the whole range of complexity.
- There are no fault density and complexity data for systems of higher criticality, so we have no support for an assumption of linearity for such systems, and we cannot derive "typical" fault density figures for such systems.

Clearly further empirical studies are needed to check the underlying assumptions and to derive realistic faults density values before the models can be applied to high criticality systems. For validating this approach to estimating the difficulty function, the main obstacle is that direct empirical assessment of difficulty functions means counting faults affecting each demand in a large population of programs developed independently for the same specification. The rarity of such populations, outside laboratory experiments on toy problems, is the usual problem in empirical software engineering. Given instead sets of realistic, but practically unique programs, a feasible form of weaker validation would be to select sets of programs that are assessed to have the same parameters for the estimation method used, and then, within such a set of programs, measure fault frequencies over sets of demands estimated to have the same difficulty.

More generally, the current models assume a common difficulty function and further work is needed to generalize this approach to the case where the difficulty functions differ for the two versions (the Littlewood and Miller model [8]).

We also note that these bounding methods also apply to the models by Hughes [6] that explain correlation between random failures in redundant hardware in terms of variation of failure rates with environmental stress. Variation of failure rates with environmental factors will typically be easier to assess than variation of difficulty between demands.

8 Conclusions

We have developed models for estimating the worst case reduction in mean *pfd* achievable by diverse program pairs that require only a partial knowledge of the difficulty function. These models can give qualitative insight, e.g. about the effects of development decisions thought to increase or to reduce maximum or minimum difficulty. We have also discussed ways for estimating the parameters required for these models, and their difficulties; in particular, an approach for estimating difficulty values and difficulty variation for applications defined in terms of a logic diagram.

We note that the methods discussed for deriving the difficulty parameters are very preliminary. Much more empirical support, especially for the high criticality systems where diversity is likely to be employed, would be needed before they could be applied for quantitative prediction (e.g. for pre-development decisions about diversity). Further work is also needed to generalize the theory to cases where the difficulty functions differ for the diverse program versions.

References

1. Bentley, J.G.W., Bishop, P.G., van der Meulen, M.J.P.: An Empirical Exploration of the Difficulty Function. In: Computer Safety, Reliability, and Security (SAFECOMP) 2004, pp. 60-71, Springer (2004)
2. Bloomfield, R.E., Guerra, A.S.L.: Process Modelling to Support Dependability Arguments. In: Dependable Systems and Networks (DSN) 2002, pp. 113-122, IEEE (2002)

3. Eckhardt, D.E. and Caglayan, A.K., et al.: An experimental evaluation of software redundancy as a strategy for improving reliability. IEEE Trans Software Eng 17(7), 692-702 (1991)
4. Eckhardt, D.E., Lee, L.D.: A theoretical basis for the analysis of multiversion software subject to coincident errors. IEEE Transactions on Software Engineering 11(12), 1511-1517 (1985)
5. Hatton, L.: Reexamining the fault density-component size connection. IEEE Software, 14(2), 89-97 (1997)
6. Hughes, R.P.: A New Approach to Common Cause Failure. Reliability Engineering 17(3), 211-236 (1987).
7. Knight. J.C., Leveson, N.G.: Experimental evaluation of the assumption of independence in multiversion software. IEEE Trans Software Engineering 12(1), 96-109 (1986)
8. Littlewood, B., Miller, D. R.: Conceptual Modelling of Coincident Failures in Multiversion Software. IEEE Transactions on Software Engineering 15(2) 1596-1614 (1989)
9. Malaiya Y.K., Denton, J.: Estimating the number of residual defects in software. In: Third IEEE International High-Assurance Systems Engineering Symposium, pp. 98-105, IEEE (1998)
10. Popov, P., et al.: Software diversity as a measure for reducing development risk. In: Tenth European Dependable Computing Conference (EDCC 2014), pp. 106-117, IEEE (2014).
11. Salako, K., Strigini, L.: When does 'Diversity' in Development Reduce Common Failures? IEEE Transactions on Dependable and Secure Computing 11(2), 193-206 (2014)
12. Skiena, S., Revilla, M.: Programming Challenges. ISBN: 0387001638, Springer (2003)
13. Sherriff M., Williams, L.: Defect Density Estimation Through Verification and Validation. In: The 6th Annual High Confidence Software and Systems Conference, Lithicum Heights, MD, pp. 111-117 (2006)
14. Wright, R.I., Pilkington, A.F.: An Investigation into PLC Reliability. HSE Software Reliability Study, GNSR/CI/21. Risk Management Consultants (RMC), Report R94-1(N), Issue B (1995)
15. van der Meulen, M.J.P., Revilla, M.A.: The Effectiveness of Software Diversity in a Large Population of Programs. IEEE Transactions on Software Engineering 34(6), 753-764 (2008)

Appendix A Worst Case *Pfd* Model Details

This analysis compares the mean *pfd* of a pair, pfd_2, with the "independence on average" value pfd_1^2 by defining a dependency factor D as pfd_2 / pfd_1^2

An extreme profile is assumed where only $p(hi)$ and $p(lo)$ can be non-zero. From the expressions of pfd_1 in (3) and pfd_2 in (4), the ratio of pfd_2 to pfd_1^2 is:

$$D = \frac{(\theta(hi))^2 z + (\theta(lo))^2 (1-z)}{(\theta(hi)z + \theta(lo)(1-z))^2} \quad (18)$$

By definition $\theta(hi) = k\theta(lo)$, so the dependency equation can be re-written as:

$$D = \frac{(k^2-1)z + 1}{((k-1)z+1)^2} \quad (19)$$

Differentiating with respect to z we obtain:

$$\frac{dD}{dz} = \frac{-(k-1)^2(z(k+1)-1)}{(z(k-1)+1)^3} \qquad (20)$$

Hence the differential is zero when either k=1 or z=1/(k+1). When k=1, the difficulty function is flat, $\theta(hi)=\theta(lo)=pfd_1=D$: the best case ("independence on average").

The z=1/(k+1) case is the situation where the dependency factor D is highest. Substituting z=1/(k+1) into (18), the maximum dependency value D can be shown to be:

$$D = \frac{(k+1)^2}{4k} \qquad (21)$$

This can be used the set the worst case value of pfd_2 relative to pfd_1^2, i.e.

$$pfd_2 = \frac{(k+1)^2}{4k}\, pfd_1^2 \qquad (22)$$

Substituting z=1/(k+1) into (3), this occurs when:

$$pfd_1 = \frac{2k}{k+1}\theta(lo) \qquad (23)$$

Or expressed in entirely terms of $\theta(hi)$ and $\theta(lo)$, it can be shown that the maximum dependency occurs when:

$$pfd_1 = \frac{2\theta(hi)\theta(lo)}{\theta(hi)+\theta(lo)} \qquad (24)$$

So that:

$$pfd_2 = \theta(hi)\theta(lo) \qquad (25)$$

Hence at the maximum dependency point, the ratio of the mean *pfd*s is:

$$\frac{pfd_2}{pfd_1} = \frac{\theta(hi)+\theta(lo)}{2} \qquad (26)$$