Reliability Engineering and System Safety 130 (2014) 61–68

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

A conservative bound for the probability of failure of a 1-out-of-2 protection system with one hardware-only and one software-based protection train

Peter Bishop, Robin Bloomfield, Bev Littlewood*, Peter Popov, Andrey Povyakalo, Lorenzo Strigini

Adelard and Centre for Software Reliability, City University, Northampton Square, London EC1V 0HB, UK

A R T I C L E I N F O

Article history:
Received 31 October 2012
Received in revised form
2 August 2013
Accepted 7 April 2014
Available online 16 April 2014

Keywords:
Software reliability
Redundancy and diversity
Probability of failure on demand
1-out-of-2 system
Protection system

A B S T R A C T

Redundancy and diversity have long been used as means to obtain high reliability in critical systems. While it is easy to show that, say, a 1-out-of-2 diverse system will be more reliable than each of its two individual "trains", assessing the actual reliability of such systems can be difficult because the trains cannot be assumed to fail independently. If we cannot claim independence of train failures, the computation of system reliability is difficult, because we would need to know the probability of failure on demand (*pfd*) for every possible demand. These are unlikely to be known in the case of software. Claims for software often concern its *marginal pfd*, i.e. average across all possible demands. In this paper we consider the case of a 1-out-of-2 safety protection system in which one train contains software (and hardware), and the other train contains only hardware equipment. We show that a useful upper (i.e. conservative) bound can be obtained for the system *pfd* using only the unconditional *pfd* for software together with information about the variation of hardware failure probability across demands, which is likely to be known or estimatable. The worst-case result is obtained by "allocating" software failure probability among demand "classes" so as to maximize system *pfd*.

1. Introduction

This paper presents an approach for estimating a conservative probability of failure on demand (*pfd*) that is applicable to a 1-out-of-2 diverse protection system where one of the protection trains is hardware-based and the other is computer-based.

The use of a protection system is an accepted strategy for hazardous industrial processes. The protection system independently monitors the industrial process and if it detects a departure from the safe operational envelope, it initiates some action that overrides the normal control system to place the process in a safe state.

The departure from the safe operational envelope is known as a demand on the protection system. Demands typically arise from different failures within the physical process and control systems. For example, in the nuclear industry, the underlying physical causes of demands on the protection system are known as postulated initiating events (PIE [1]), and the overall plant safety analysis will identify a set of PIEs that represent credible plant failures (such as loss of electrical grid connection or a rupture in the primary coolant circuit). As part of the design process, maximum rates are assigned for each PIE and, to reduce the risk of a PIE that occurs frequently, diverse means are used to detect the departure from normal operation (such as temperature and pressure). These are normally implemented in diverse trains of equipment using different types of sensors and with different means of achieving a safe state for the same PIE.

To improve its reliability, a protection train typically also has a high level of internal redundancy to tolerate hardware failure in the sensors, protection logic, actuators and plant components (like valves or pumps). Even so, if sufficient hardware sub-components fail, the train will be unable to respond to the demand triggered by the PIE. Depending on the PIE involved, different sets of hardware components of the protection train need to be able to respond successfully to demands initiated by different PIEs. Typically the hardware probability of failure on demand for a PIE is determined by fault tree analysis [2], where the minimal cutsets of failed sub-components are identified [3] that result in a demand failure. By quantifying and summing the minimal cutsets, the probability of failure per demand for a given PIE can be computed.

Probabilistic analysis of hardware-based systems is well established, but less work has been carried out on including the impact

* Corresponding author. Tel.: +44 20 7040 8420; fax: +44 20 7040 8585.
E-mail address: b.littlewood@csr.city.ac.uk (B. Littlewood).

http://dx.doi.org/10.1016/j.ress.2014.04.002
0951-8320/© 2014 Elsevier Ltd. All rights reserved.

of software failures in a computerized protection system. In this paper, we present a method that allows a conservative estimate to be made for the probability of failure on demand (*pfd*) for diverse 1 out-of-2 protection trains where one train is software based, given the form of claims commonly offered for software reliability.

2. Terminology and modeling approach

As discussed above, a protection system responds to demands – events that require its intervention. Whether the protection system will respond correctly or will fail on the demand depends on the characteristics of the demand. The protection system may fail – that is, fail to start the required safety action – if, for instance, some hardware component (or redundant combination of components) that is needed to respond correctly is permanently faulty, or suffers a transient fault, at the time of that demand; or due to a design defect in hardware logic or in software. We therefore can also identify a specific demand as a vector of values that together determine the likelihood of any of the protection trains failing

- since the protection system monitors the values of state variables of the plant (e.g. pressures, temperatures, measures of flows) and may fail or not depending on their values (especially software bugs may depend on the exact values of the data), this vector includes the sequences of values that are read during the demand event;[1]
- since hardware probabilities of failure are affected by environmental variables (e.g. temperature, humidity, atmospheric pressure, level of electromagnetic radiation, in the various part of the system), these variables are also parts of the vector. Probabilities of hardware failure are usually available for ranges of variables.

The demand is thus a (vector) random variable; processes in the protected plant and its environment determine the times of occurrence of each demand as an event and the value of the demand vector. In what follows we will use just the term “demand” when the meaning “event” or “vector” is clear from the context.

The demand vector includes all variables that have an effect on the success or failure of any part of the system. Thus, for instance, temperature values at a certain sensor in the plant are part of the demand even if only one of the protection trains reads them. But some components of the demand affect more than one protection train. Thus, for instance, an earthquake that affects two protection trains will increase the probability of failure of both, possibly (if the shock is way above their design limits) by 1. Thus we can model common causes of failure via demands which imply high probability of failure for both trains.

This form of modeling has been used in earlier work [4–6] to model in a consistent way failures due both to physical causes and to design, and thus both hardware and software failures. The basic idea here is one of variation of the probability of system failure between different demands, as an explanation for dependence in failure behavior between diverse trains to be used in a fault tolerant system (e.g. a 1-out-of-2 system). The idea is a simple one. The probability of a system failing on a demand will, in general, vary across demands. Since the component values of the demand vector together determine the likelihood of a protection train failing, and if the system's design is such that we can exclude failures of one train directly causing failures of the other, the failures of both trains on a specific demand (a specific value of the demand vector) can be assumed independent conditionally on the demand.

Interest then centers upon the covariance (across all demands) between the two functions (of the demand) that describe the probabilities of failure of the two trains of a 1-out-of-2 system. When there is positive covariance – roughly, when the demands that associated with a higher probability of failure for one train also tend to associate with a higher probability of failure for the other – then it is more likely that there will be positive correlation between the trains' failures on a random demand. In such a case, wrongly assuming independence of failures between the two trains will give an optimistic estimate of the reliability of the 1-out-of-2 system.[2]

An achievement of these conceptual models is their establishment of, and explanation for, the inevitability of dependence (positive or negative) of failure behavior between redundant, diverse systems, hardware or software. They thus support the empirical evidence for such dependence that comes, for example, from experiments: see e.g. [7,8]. No longer is it possible to claim that the two diverse protection trains of a 1-out-of-2 fault tolerant architecture will fail independently of one another, without making very strong claims: essentially that there is no variation of failure probability across demands for at least one of the trains. This means that the simple arithmetic of independence is not applicable for the computation of the system reliability as a function of the component reliabilities. Specifically, the *pfd* of a 1-out-of-2 system over all demands cannot simply be assumed to be the product of the *pfd*s of the two trains. Informally, it means that we need to know how dependent the failures of the trains will be.

Reliability estimation of such a 1-out-of-2 system is non-trivial as it seems to require a complete knowledge of how the probability of failure varies between demands.

However, we can reason by failure classes, defining a “class” as a set of demands such that the estimated *pfd* is the same for all demands in a class. For simple hardware, a “class” means a set of demands such that correct response to any of them requires the same set of subsystems to function correctly. If the demand classes for two protection trains do not exactly coincide, a demand class is defined as a set of demands such that each one of the trains has constant *pfd* over all the demands in the set. This will generally involve more demand classes for the system than that would be defined for each train alone. If this definition led to too many classes of demands, their number can be contained, and kept tractable, by merging classes and using the highest *pfd* among those classes thus merged. It can easily be shown that given conditional independence on each demand, and constant *pfd* across the class for at least one of the two trains, failures of the two trains are conditionally independent conditionally on the demand class [9]. Thus, the system *pfd* conditional on a certain demand class is obtained by multiplying the *pfd* values, for that demand class, of the two trains.

For software, things seem much more problematic: software faults are such that among two demands that are from all other viewpoints in the same “class”, the software may fail on one but not the other depending on the values of the inputs to the software. In fact, claims about software reliability are often limited to the marginal probability of failure on demand based on arguments of quality of production and verification, “proven in

[1] With software, one could imagine the protection sytem as realizing a deterministic function of the values it reads: the probability of demand could only be 0 or 1. In practice, whether it fails may depend on the software's past history. So a specific demand implies a probability of system/train failure, which is not necessarily 0 or 1, rather than deterministic failure or success.

[2] It is possible to do better than independence if there is negative association between the probabilities of failure over the demands.