Reasoning about the Reliability of Multi-Version, Diverse Real-Time Systems

A. Burns
Department of Computer Science,
University of York, UK.
Email: burns@cs.york.ac.uk

B. Littlewood
Centre for Software Reliability,
City University, UK.
Email: b.littlewood@csr.city.ac.uk

***Abstract*—This paper is concerned with the development of reliable real-time systems for use in high integrity applications. It advocates the use of diverse replicated channels, but does not require the dependencies between the channels to be evaluated. Rather it develops and extends the approach of Littlewood and Rushby (for general systems) by investigating a two channel system in which one channel, A, is produced to a high level of reliability (i.e. has a very low failure rate), while the other, B, employs various forms of static analysis to sustain an argument that it is perfect (i.e. it will never miss a deadline). The first channel is fully functional, the second contains a more restricted computational model and contains only the critical computations. Potential dependencies between the channels (and their verification) are evaluated in terms of aleatory and epistemic uncertainty. At the aleatory level the events "A fails" and "B is imperfect" are independent. Moreover, unlike the general case, independence at the epistemic level is also proposed for common forms of implementation and analysis for real-time systems and their temporal requirements (deadlines). As a result, a systematic approach is advocated that can be applied in a real engineering context to produce highly reliable real-time systems.**

***Keywords*-Real-Time Systems, High-Integrity Systems, Safety-Critical Systems, Diversity, Reliability.**

I. Introduction

For high-integrity real-time systems, it is imperative that all system deadlines are met on all occasions. A general approach to improve reliability in systems is to introduce redundant components. However, replication will not be effective if the sources of unreliability are defects in the way the system has been analysed, rather than in the implementation of the system. In this paper we investigate the properties of a dual channel system in which diverse forms of analysis (as well as diverse implementations) are explored. The paper follows the model introduced by Littlewood and Rushby[24] for general system reliability. A two-channel system is employed. The first channel (Channel A) is fully functional with its temporal behaviour being validated by a measurement-based approach. The other channel (Channel B) only implements the crucial software components, has a restricted software architecture and is validated by static analysis.

Channel B by virtue of its restricted complexity and formal analysis is open to a claim of perfection. If the claim for perfection is correct, then the channel will never miss a deadline. Static timing analysis linked to formal scheduling analysis is often claimed to provide an absolute guarantee. Here we interpret "absolute guarantee" to mean that the channel is *perfect* (in the temporal domain, which is the focus for this paper). A claim for perfection may of course not be valid; we denote by pnp_B the probability that channel B is in reality not perfect.

Channel A cannot claim perfection, but it can be engineered to a high level of reliability. We denote by fr_A the failure rate, e.g. one deadline miss per 10^3 hours of operation. At run-time this claim will be deemed to be valid if deadline misses are less frequent.

The use of a diverse argument (one channel is perfect, the other is reliable) is in contrast to the more usual employment of architectural diversity in which both channels claim a level of reliability and a limit to their dependency. Unfortunately it is hard/impossible to judge what the level of dependence is between the channels, and therefore to be able to compute overall system reliability [25]. However, as we shall show, it is possible to claim independence between diverse arguments.

In the Littlewood and Rushby paper [24] a distinction is made between real uncertainty "in the world", which they call *aleatory uncertainty*; and the inevitable uncertainty that comes from trying to estimate/measure the aleatory uncertainty. The latter is termed *epistemic uncertainty*. They demonstrated that for aleatory uncertainty it is possible to bound the unreliability of a two channel system to be $pfd_A \times pnp_B$ (where pfd_A is the probability of a failure on a demand for channel A). Epistemic uncertainty is however harder to evaluate as it requires an expert not only to provide values for these parameters but also to judge the level of dependencies between these evaluations. So although the events "A fails" and "B is imperfect" are independent at the aleatory level, estimates of their values may not be. In this paper we adapt the Littlewood and Rushby framework to timing failures and failure rates, and argue that there is also independence at the epistemic level.

The paper is organised as follows. In the next section a system model is provided for a two-channel system. This model first focuses on worst-case execution time (in Section II-A) and then (Section II-B) on full channel implementation (tasks, RTOS, software architecture etc), finally it gives in Section II-C a short introduction to timebands. Section III presents the adapted analysis framework and Section IV then derives an estimate of the overall reliability of the replicated real-time system. In Section V some alternative system models are

discussed, and Section VI considers related work. Conclusions are presented in Section VII.

II. System Model

We focus on a "one-out-of-two" system; although the arguments presented here may be applicable to other architectures. The two channels that make up the system execute on the same inputs and then produce channel-specific outputs. The channels have their own hardware (power supplies etc). System correctness requires that the right outputs are produced at the right time. Safety cases typically partition the overall case into domains (requirements, hardware, software, timing) and then argue for the required level of evidence in each domain[4], [6], [5]. In this paper we are only concerned with timing failures. We assume other elements of the overall safety case are satisfactory (to the same level of reliability).

On each channel a multi-tasking program is executing on a single processor (ie one processor per channel). Each task executes a series of invocations (*jobs*), each of which has a deadline by which it must complete its execution. There are differences between the software architectures of the two channels, but for each critical output there is a task/job in channel A with the same release condition and deadline as its corresponding task/job in channel B. The system suffers a timing failure only when both channels miss the same deadline. The channels do not vote, but they may exchange data to ensure they do not diverge (other than when a failure occurs).

The type of application that is considered here is a continuous control system as found in production control and avionic fly-by-wire configurations. For this type of system it is conventional to argue about *failure rate* rather than probability of failure per demand. The necessary level of reliability is therefore expressed with respect to a period of time. This could correspond to the duration of a mission (or flight) for a relatively 'short-lived' continuous system. For a 'long-lived' (non-stop) system a unit of time such as an hour of continuous execution would be more natural. Here we will use this latter interval, and define fr_A as the (real) failure rate, per hour, of channel A. Channel B is the one for which an argument of perfection is applied, with pnp_B denoting the probability that this is not in reality the case. We will revisit (in Section III-A) the derivation of the aleatory uncertainty for this (fr_A, pnp_B) vector.

We assume that the values of the deadlines are themselves correct (i.e. small enough for safe operation). It would clearly be a common-mode failure if each channel satisfied the specified deadline, but that these deadlines were too lax for, say, stable flight. The use of an incorrect deadline is viewed as a failure of the requirements process. Hence, here we are dealing with system failure rates conditional on the deadlines being defined appropriately.

Even in the most stringent of safety-critical software much of the code is actually not part of the critical functionality. Health monitoring, trend analysis, data gathering for performance evaluation etc. are all associated with the key code but are not strictly necessary for safe operation. Hence we require that only channel A has full functionality. Channel B, by comparison, only has the critical code. Its functionality is a strict subset of that of Channel A[1].

The implementation and analysis approaches for the two channels are as follows. Note these are examples of possible architectures, others within the same basic approach could be defined.

Channel A, the full functional channel, that can claim a high level of reliability, is characterised by:

1) Preemptive execution.
2) Event- (and time-) triggered job releases.
3) Scheduling by the Earliest Deadline First (EDF) policy with associated scheduling analysis.
4) Implementation on a full Real-Time Operating System (RTOS).
5) Exploiting advanced hardware features.
6) Execution times (RTOS and application code) analysed by measurement.

Channel B, that is open to a claim of perfection, but has only partial functionality is characterised by:

1) Non-preemptive (cooperative or deferred preemptive) execution.
2) Time-triggered job releases only.
3) Fixed-Priority (FP) scheduling with associated timing analysis.
4) Minimum RTOS (e.g. that provided by the Ravenscar profile[20]).
5) Restricted use of advanced hardware features.
6) Execution times (RTOS and application code) obtained via static timing analysis.

In a broad sense Channel A is analysed by a measurement-based approach, and Channel B by formal models. However, this is not a strict distinction as formal schedulability analysis may be undertaken on both channels. This issue will be returned to later (Section V). The channels also differ in the degree to which they exploit all the available hardware power. Channel A uses a number of optimal schemes (for example preemption and EDF scheduling) and efficient techniques (e.g. fully exploiting the hardware and using measurement for execution time estimation). Channel B characteristics are not optimal (in terms of resource usage), it is therefore necessary that channel B has a lower load (i.e. only the essential software).

Although there are a number of significant differences between the two channels it is in their approach to computing estimates of job execution times that the channels differ most. Timing analysis (i.e. the process of obtaining estimates of individual task's worst-case execution time) is therefore considered first before other aspects of the system model.

[1]Whether it is a subset of the actual code depends on the approach taken to functional reliability; diverse software, programming languages, compilers and hardware could all be employed to enhance the diversity between the two channels. From a timing point of view these issues are not significant as we assume adequate functional reliability.

A. Timing Analysis

Whatever the system model, it is always necessary to know, with a high level of confidence, what the worst-case execution time (WCET) is of each sequential thread of code within the application (and all code in the RTOS or equivalent). As the input space for the code is finite and the hardware's behaviour at the basic level is deterministic it is reasonable to argue that there is a *real* WCET, $WCET_{real}$, that is an upper bound on the execution time of the code for all possible execution behaviours. This value is, of course, in general unknown (and is potentially unknowable). Schedulability analysis of the entire system will use an estimate of each task's worse case execution time, C; with the assumption that $C \geq WCET_{real}$ for all tasks.

Within the research community of WCET analysis there is on-going informed debate about the use of static analysis (using a model of the code and a model of the hardware) or measurement (using a simpler model of the code and measurement of the hardware) to obtain an estimate of WCET. If we define $WCET_{stat}$ to be the estimate obtained by static analysis and $WCET_{obs}$ to be the maximum value observed during measurement then the following predicates are usually accepted:

$$WCET_{real} \ < \ WCET_{stat}$$

and

$$WCET_{obs} \ \leq \ WCET_{real}$$

Unfortunately for modern processors and conservative analysis, $WCET_{stat}$ can be substantially greater than $WCET_{obs}$.

The analytical approach can of course be in error (for example, a parameter such as the number of wait states for an instruction may be wrong in the processor manual, or the algorithm for branch prediction may not be complete – see Section IV-C). However it is possible to believe that the first predicate ($WCET_{real} \ < \ WCET_{stat}$) is indeed true – open to an argument of perfection in the terms used in this paper. When static analysis is used, some level of testing will also always be employed, and this will increase the confidence in the claim of perfection.

There are a number of different forms of static analysis. In general the simpler the model the larger the value of the resulting $WCET_{stat}$. But the simpler the model the greater the confidence in the claim of perfection, ie. the smaller the value of pnp_B. We shall return to the issue of estimating pnp_B in Section IV-C.

Those techniques that focus on measurement do not, in general, rely totally on the worst-case observed value. They never claim complete coverage and hence the real worst-case may not have been experienced during testing. Typically a "safety factor" is added to this value or, more analytically, some form of extreme value statistical approach is used [18], [17], [19] to construct a predicted value of WCET ($WCET_{pred}$) with $WCET_{obs} \ < \ WCET_{pred} \ < \ WCET_{stat}$, and the assertion:

$$WCET_{real} \ \leq \ WCET_{pred}$$

Extensive testing (measurements) can be used to gain considerable confidence in this assertion.

B. Scheduling Analysis

To assert that all deadlines for all tasks will be met in all situations, it is necessary to account for task interactions, context switches, RTOS overheads, interrupt load etc. The characteristics and techniques outlined above for Channel A are all mature engineering approaches and it is possible to undertake EDF-based scheduling analysis based on measured execution times of all components (RTOS and application code).

Testing of the complete system becomes more complicated and time consuming due to the difficulty in getting adequate coverage over the many different execution patterns that are possible with Channel A's flexibility. This is why measurement is made at the component level and schedulability analysis is undertaken at the system level.

The system model for Channel A allows for event-triggered computation, typically generated from interrupts. Schedulability analysis requires there to be a bound on the number of such events (from the same source) in any time interval. Usually this bound is obtained by assuming a minimum temporal separation between successive events. A measurement-based approach must therefore not only measure execution times but also event (interrupt) occurrences.

For Channel B, non-preemptive execution and only time-triggered computation leads to simple run-time behaviour and a minimum run-time support system (RTOS) that itself can be analysed for its worst-case behaviour. A schedulability analysis tool is not complex and its results can be checked manually. Fixed priority dispatching with non-preemptive (or cooperative/deferred preemption) execution is fully predictable and has the useful property that during an overload, if one did occur, the tasks with the larger priorities would be the last ones to miss their deadlines.

The methods of analysis outlined above are all established techniques described in standard textbooks [27], [12], [10]. Details are therefore not included here.

C. The Timeband Structuring Framework

A final aspect of the system model involves the use of a granulated time framework called *timebands*[8], [9], [32].

One characteristic of many real-time systems is that they are required to function at many different time scales (from microseconds or less, to days or more). In the timeband framework a system is assumed to consist not of a single time dimension but a finite set of partially ordered *bands*. Each band is represented by a *granularity* (expressed as a unit of time that has meaning within the band, e.g. the millisecond band) and a *precision* that is a measure of the accuracy of the time frame defined by the band.

A band is populated by *events* and *activities*. Events, in the normal way, are considered to be instantaneous (a cut of the time line) within the band of their definition. Activities by comparison have duration of one or more 'units' of the band's granularity. Behaviours across bands are formulated as *mappings* between events in one band and activities in a 'lower' band. So an activity that has duration in a lower (finer) band may *map* to an instantaneous event in a higher band. This property is employed in this paper to move from a discrete view of a typical execution cycle to a continuous view in which succeeding or failing cycles can be represented as points on a continuous time line.

III. Adapting the LR model for timing faults

In this section we reformulate the Littewood & Rushby (LR) analysis [24] as it applies to the timing faults of a real-time, continually executing, two-channel system. First, we show that at the aleatory level a simple multiplication of two channel-specific values is sufficient to obtain a measure of the reliability of the whole system. Then, in the next section, we consider how a domain expert (or possibly an assessor or certification agency) could estimate these parameters. This latter consideration of the epistemic uncertainty needs not only to provide meaningful estimates of the parameters but also has to assess the likelihood of dependencies between these estimations.

The LR paper addresses a serious problem that arises when multiple diverse software versions are used to achieve high reliability: specifically, the different versions cannot be assumed to fail independently of one another. So, for a 1-out-of-2 architecture (such as a protection system), even if it is known that each version has a probability of failure on demand (*pfd*) of 10^{-3}, it is not possible to claim a *pfd* of 10^{-6} for the system. In fact the true system *pfd* will generally be greater (ie. worse) than this – and often very much greater.

The evidence for these observations is now extensive. It comes from several carefully controlled experiments, most notably those of Knight and Leveson [21] and others (e.g. [15]), and from work on probability models that represent the joint failure processes of multiple versions [16], [22].

These observations should not be taken as a criticism of the use of multi-version diverse software as a good way of achieving high dependability. There is quite extensive evidence that high reliability can be achieved this way (albeit falling short of what could be achieved if the versions failed independently), and some evidence that it is superior to other means of doing so (e.g. heroic debugging). For example, even the Knight and Leveson experiment – which comprehensively rejected the hypothesis of failure independence – showed that on average the benefits from diversity (in their case 2-out-of-3 architectures) were considerable. There is also some evidence of efficacy from industrial applications where systems have been proven to be very reliable in extensive operation, e.g. some aircraft flight control systems, railway signaling systems. See Littlewood et al [23] for a more extensive discussion of the issues here.

As a design approach, then, the use of multiple diverse software versions as a means to achieve high reliability has strong attraction. The difficulty lies in the evaluation of what exactly has been achieved in a particular instance - i.e. in evaluating the achieved reliability in order to decide whether the system's operational behaviour will be acceptable. Being unable to claim independence poses a serious problem here. If the simple mathematics based on independence (informally, multiplying two small numbers together in order to arrive at a *very* small number) cannot be used, the assessor has to take account of *how* dependent failures of the diverse versions are. This problem is, essentially, as difficult as assessing the total system as if it were a black box.

The LR approach to this problem considers a special architecture: one in which there is a highly functional (and thus complex) channel A and a channel B of only basic functionality (and thus simple). The informal idea here is that a claim of *reliability* will be made about A (say as a *probability of failure on demand*), as above. But for channel B its simplicity means that it may be 'perfect' (i.e. will never fail), and so the claim about this channel will be in terms of its probability of *imperfection*. The main LR result is that the system *pfd* is (conservatively) just the product of these two probabilities - which contrasts with the situation above when the two *pfd*s could not simply be multiplied.

In what follows we show how this idea can be applied to a situation in which multiple versions are used to protect against timing failures, where 'failure' means failure to complete a task in the time available.

The account here follows that in LR, in particular in treating separately *aleatory* and *epistemic* uncertainty. Aleatory uncertainty can be thought of as "uncertainty in the world", and epistemic uncertainty as "uncertainty about the world". In much scientific modeling, the aleatory uncertainty concerns the unpredictability of systems, e.g. when they will fail. Epistemic uncertainty then often centres upon the parameters of the aleatory models, e.g. their failure rates.

A. Aleatory Uncertainty

We begin with aleatory uncertainty. We shall assume that channel A is the more complex channel, about which only reliability claims with respect to timing are feasible (i.e. it will eventually fail if it operates for a sufficiently long time); channel B is the simpler channel about which a claim of possible perfection can be made (i.e. there is a chance that it will never fail). Specifically, we shall assume that the failures of channel A occur in a Poisson process of rate

$$fr_A = \lim_{\delta t \to 0} \frac{P(A \ fails \ in \ time \ interval \ (t, t+\delta t))}{\delta t} \quad (1)$$

The treatment here reflects the fact that the failure process of such a system exists in a higher timeband (see Section II-C) than that of the successive execution/control cycles. So *failures* might be expected to occur only every few thousand hours, but the *execution cycles* might each be of the order of a few tens

of milliseconds in duration. So whilst the process at the lower time-band is inherently discrete – comprising the successive execution cycles – an observer at the higher timeband sees failures occurring in what is effectively *continuous* time. The reliability of the system is therefore naturally expressed as a failure rate, e.g. 0.0001 (10^{-4}) failures per hour, rather than a probability of failure per execution cycle.

Returning to the issue of system reliability, there is a system timing failure if and only if both A and B have timing failures:

$$\begin{aligned} & P(System\ has\ a\ failure\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \\ &= P(A\ and\ B\ both\ fail\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \end{aligned} \tag{2}$$

Note these statements about probabilities are conditional on knowing the values of fr_A and pnp_B. Now

$$\begin{aligned} & P(A\ fails\ in\ (t, t+\delta t)\ and \\ & \qquad B\ not\ perfect \mid fr_A, pnp_B) \\ &= P(A\ fails\ in\ (t, t+\delta t)\ and\ B\ not\ perfect\ and \\ & \qquad fails\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \\ &+ P(A\ fails\ in\ (t, t+\delta t)\ and\ B\ not\ perfect\ and \\ & \qquad succeeds\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \\ &\geq P(A\ fails\ in\ (t, t+\delta t)\ and\ B\ not\ perfect\ and \\ & \qquad fails\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \\ &= P(A\ and\ B\ both\ fail\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \end{aligned}$$

Substituting in Eqn(2):

$$\begin{aligned} & P(System\ has\ a\ failure\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \\ &= P(A\ and\ B\ both\ fail\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \\ &\leq P(A\ fails\ in\ (t, t+\delta t)\ and \\ & \qquad B\ not\ perfect \mid fr_A, pnp_B) \\ &= P(A\ fails\ in\ (t, t+\delta t) \mid \\ & \qquad B\ not\ perfect, fr_A, pnp_B) \\ & \quad \times\ P(B\ not\ perfect \mid fr_A, pnp_B) \end{aligned}$$

Now, knowing that B is not perfect tells us nothing about whether or not A will fail in a particular time interval (we know the failure rate of A and thus its chance of failure in that time interval). That is, "$A\ fails\ in\ (t, t+\delta t)$" and "$B\ is\ not\ perfect$" are independent. So we have

$$\begin{aligned} & P(A\ fails\ in\ (t, t+\delta t) \mid B\ not\ perfect, fr_A, pnp_B) \\ &= P(A\ fails\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \\ &= P(A\ fails\ in\ (t, t+\delta t) \mid fr_A) \end{aligned} \tag{3}$$

and

$$\begin{aligned} & P(B\ not\ perfect \mid fr_A, pnp_B) = \\ & \qquad P(B\ not\ perfect \mid pnp_B) = pnp_B \end{aligned} \tag{4}$$

thus

$$\begin{aligned} & P(System\ has\ a\ failure\ in\ (t, t+\delta t) \mid fr_A, pnp_B) \\ &\leq P(A\ fails\ in\ (t, t+\delta t) \mid fr_A) \\ & \qquad \times\ pnp_B \end{aligned} \tag{5}$$

Dividing by δt, taking limits and using the definition of fr_A from Eqn(1), we get the conditional, conservative system failure rate for the two channel system, fr_{AB}, to be bounded as follows:

$$fr_{AB} \leq fr_A \times pnp_B \tag{6}$$

Eqn(6) shows the minimal improvement in the failure rate that comes from the inclusion of the perfect channel:

$$fr_A / fr_{AB} \geq 1/pnp_B \tag{7}$$

So if pnp_B is 10^{-3} then the improvement in the failure rate is at least 1000.

The result above is *conditional* because it assumes the two parameters representing the failure rate of A and the probability of imperfection of B are known. It is *conservative* because it assumes that if B is imperfect, it always fails whenever A does: B brings no benefit if it is not perfect.

The value of this result, compared with the analysis sketched out above for the case where both channels have to be assumed to be fallible, is that the two parameters are sufficient to obtain the conditional conservative system failure rate. In the earlier case, because there is no independence result comparable to Eqn(3) above, the individual channel reliabilities are not sufficient alone to determine system reliability (we also need to understand the nature of the failure dependence).

The approach we have used here avoids the difficulties of modelling the dependence between failures at the lower timeband level, where time is discrete and each system has a reliability defined in terms of a probability of failure *per demand* (i.e. execution cycle). It also avoids the hard problems of conducting the modelling entirely at the higher timeband level, in terms of failure *rates* of channels operating in continuous time. Such an approach would treat the channel A and channel B failures as points on the time axis of zero duration. If these processes were *independent* stochastic point processes, for example, (such as Poisson processes) then simultaneous failures of A and B would be impossible without some further modelling assumptions.

The price paid in avoiding these difficulties, of course, is that our result is conservative, perhaps severely so.

This completes the aleatory modeling.

B. Epistemic Uncertainty

Estimates of the fr_A and pnp_B parameters must be made by those responsible for certifying or "signing off" on the system prior to its deployment – we used the term *assessor* for this role. In practice, of course, an assessor will not know for certain the values of the two parameters of this model. This is

where epistemic uncertainty comes in. Adopting a Bayesian approach, we can represent the assessor's beliefs about the two unknown parameters (fr_A, pnp_B) by the (subjective) probability distribution:

$$G(f_A, p_B) \quad = \quad P(fr_A \leq f_A, pnp_B \leq p_B)$$

This simply represents the assessor's posterior joint probability (i.e. based on all the evidence to hand) that the rate of A failures is smaller than f_A and the probability that B is not perfect is less than p_B. Using this distribution to take expectations over Eqn(6), we obtain the assessor's (conservative) posterior rate of system failures, FR_{AB}[2].

$$FR_{AB} \;=\; E(fr_{AB}) \;\leq\; E(fr_A \times pnp_B) \;=\; \int (f_A \times p_B) dG(f_A, p_B) \tag{8}$$

where E is the expectation operator. This is the conservative value that the assessor will respond with when asked "what is the failure rate of the system?".

IV. Estimating Aleatory and Epistemic Uncertainty

Eliciting from an expert their subjective beliefs, in terms of a distribution such as $G(f_A, p_B)$, is not an easy task, but there have been considerable advances in recent years in providing tools to assist this exercise (see O'Hagan et al [28] for a good introduction). Informally the assessor here needs to give values to their assessment of (belief about) fr_A and pnp_B, and must also judge if there is a dependency between these two beliefs. Here we will first address this dependency issue, and argue for independence (thus greatly simplifying the problem); then it just remains to assess the individual parameters.

A. Dependency between beliefs about fr_A and pnp_B

Dependency between beliefs about fr_A and pnp_B comes from commonalities in their derivation. Whether or not there is dependency between the assessor's beliefs about fr_A and pnp_B can be established, in principle, by asking the following (formally equivalent) questions:

- If the assessor knew the aleatory probability that channel B was not perfect, pnp_B, would it affect their belief about the value of fr_A?
- If channel A failure rate was known to have a particular value, would it affect their belief about the value of pnp_B?

It is reasonable to argue that the answer to both of these questions will be *no* for timing faults, the rationale being:

1) The common requirements (in terms of deadlines and periods) are assumed to be correct in this formulation – see system model in Section II.
2) The forms of timing analysis (measurement and static analysis) are diverse and share no common assumptions.
3) The forms of scheduling analysis are again diverse and share no common assumptions (other than those that derive from conservative assumptions for both channels, eg. tasks released at the critical instant).
4) Functional errors in the application software (causing for example the execution of an infinite loop) would manifest themselves in failures in the functional domain (considered separately, but assumed to be adequately rare).
5) Failures introduced by the compiler and associated tools (eg. linkers) would also manifest themselves in failures in the functional domain.

Together these strongly imply that the assessor's beliefs about the failure rate of channel A and the probability of (im)perfection of channel B are independent. In other words the assessor's (posterior) belief distribution $G(f_A, p_B)$ factorises into the product of the marginal distributions, $G(f_A)$ and $G(p_B)$.

Both of these distributions allow the assessor to state their beliefs about the parameter as confidence bounds. So, for example, $G(f_A)$ might imply that a failure rate of 10^{-5} corresponds to the (upper) 90% confidence level, 10^{-4} to the (upper) 99% confidence level, and so on. Often it will suffice to obtain the mean of the assessor's conservative posterior distribution of system failure, Eqn(8). Because of independence this is just the product of the means of $G(f_A)$ and $G(p_B)$, which we shall call FR_A and PNP_B, respectively. These parameters could be obtained via (partial) knowledge of their parent distributions, or they could simply be the expert assessor's *direct* estimates of failure rate and probability of imperfection.

The following sections look at the evidence that an assessor could use to come to separate judgements as to the values of FR_A and PNP_B. Obviously an assessment of a real system would involve significant amounts of evidence from the system itself. Here we consider, in general, what this evidence could provide and hence the type of claim that could be made about overall system reliability.

B. Judgement about fr_A

The obvious source of evidence upon which an expert could base his beliefs about fr_A, in order to arrive at his posterior mean FR_A, is the outcome of operational testing. Operational testing is a testing regime in which the test cases are generated in a way that exactly captures the statistical properties of real operation. That is, the probabilities of test cases, and sequences of test cases, are exactly the same in the test environment as they are in the real operational environment. Very simple statistical analysis then allows claims – e.g. confidence bounds – to be placed on parameters such as failure rate (for a continuously operating system), or probability of failure on demand (for a demand-based system).

In most timing applications of the kind considered here it will usually be more convenient to treat time as continuous,

[2]We are using Riemann-Stieltjes notation here because this allows the possibility of non-zero mass at points in the (f_A, p_B)-plane. This contrasts with the more usual notation involving probability *density* functions.

as we have done. Then, if a certain number, n, of failures is observed in an operational test of duration T, an assessor can compute his confidence, C, that the failure rate is smaller than 10^{-x}. In some safety critical applications (e.g. the UK nuclear industry) it is required that *no* failures are seen in operational testing. If a failure occurs, the software must be fixed whereupon it becomes a *new* program for which assessment must begin afresh. Clearly, in such a case, the longer the software survives its testing without failure, the greater the confidence an assessor will have that it has achieved a particular reliability.

For example, an assessor can be 99% confident that the failure rate is smaller than 10^{-3} per hour if 4605 hours of operational testing has produced no failures; the same claim (99% confidence in 10^{-3}) can be made if they had seen only 1 failure in 6638 hours of operational testing; and so on [26].

This kind of statistical analysis, of course, makes two important assumptions: that the operational environment is truly captured by the test environment, and the test oracle (that decides whether the output is successful or not) is correct. Any doubts here will contribute to epistemic uncertainty, and should be taken account of by the assessor's beliefs about fr_A. This can be difficult in some applications when the failures concern incorrect functionality, but we believe they might be less serious in the case of timing failures considered here. For example, measurement approaches to execution time estimation tend to be pessimistic in that paths through the program are explored that are not in practise feasible. For timing properties the testing environment is therefore likely to be a superset of the operational environment. Also the oracle here simply has to decide whether the execution has completed on time: this is typically much easier than deciding whether it is *correct*. It may be reasonable to assume that the oracle is perfect in such cases.

C. Judgement about pnp_B

In the real-time literature it is common to encounter the phrase "this analysis is pessimistic", ie. *safe*. Proofs are provided and simulation results used to evaluate the level of pessimism. It is far less common, however, to see any attempt to judge the probability that a specific system, when shown to be schedulable by a particular form of analysis, will indeed be safe.

Safety cases often make claims that all hazards and vulnerabilities have been identified and each of these has been mitigated so that the risks are ALARP (as low as reasonable practicable) [6], [5]. Completeness (ie. all threats have been identified) can never be formally proven but it is possible to provide a thorough evaluation. For channel B the claim for perfection (eg. all deadlines are always met) has the following vulnerabilities.

1) The model of the hardware, including its many parameters, is not accurate.
2) There are flaws in the theory on which the analysis is based.
3) There are bugs in the analysis tools.
4) Engineers apply the analysis/tool incorrectly (or indeed fail to apply it at all).

For timing analysis, if a simple model is sufficient to deliver schedulability, the behaviour of channel B can be reasonably straightforward to validate. But there may still be errors in the parameters that must feed into the analysis tool. For example, a system developed in 1993 was shown to have an error in the number of wait states defined in the processor handbook for a particular instruction [11].

In general, the 'theory' on which an analysis tool is based, once it has migrated to actual industrial use, will have been subject to considerable review and evaluation. As a result high levels of confidence can justifiably be assigned to its veracity. But again there are counter-examples. The theory used to analyze the non-preemptive behaviour of the CAN bus (a priority based arbitration bus employed initially in the automobile industry) was 'proven' correct in 1994 [29], [30] and then used extensively (at least 20 million cars have systems verified using this analysis). In 2007 a flaw in the analysis was identified that could lead to optimistic (ie. unsafe) results [7], [13].

The tools for timing and scheduling analysis are not complex. Certainly they compare favorably with the use of theorem provers for the analysis of the functional models of the software. Indeed they will often be less complex that the application software they are being applied to. Errors in tools of course do occur, but the theory on which these tools are based have nothing that would lead to particular problems.

The final threat comes from human error. There is nothing specific in the timing domain that would imply that any special problems are manifest here.

Mitigation comes from the very low (though admittedly not zero) likelihood of the above threats actually materialising. Also important is the natural robustness of channel B. If one task does indeed execute for longer than estimated there is a strong possibility that other components will be executing for less than their worst-case estimates. As a result local errors do not result in deadline failures. In both of the flaws identified above (error in processor timing table and error in the theory) it was possible to deduce that deadline misses would not result.

The threats due to tool bugs and human error can be mitigated by extensive testing (of the tool and the application) and by the use of redundancy. Here redundancy can be of the tool (there are a number of schedulability tools available) and of the humans involved using normal cross-checking and supervision procedures.

These mitigations are not absolute and hence some residual doubt on behalf of the assessor is to be expected. But for a simple model, quality tools and testing that reveals that all observed response times are considerably less than the predicted analytical value it is reasonable to argue that PNP_B is less than 10^{-3} and possible better than 10^{-4}.

It is inevitable that there is a substantial element of subjectivity in the derivation of these estimates. But at least we know here exactly what the various parameters are that need estimating within this formal model. The only practical

alternative is to let the experts simply say what they think about the overall system failure rate. The model presented here allows a third party to see what are the components of belief that went into the experts' bottom line *system* figure.

D. Final Estimation

From the above discussions and the independence of the assessor's beliefs about failure rates and potential perfection, it is possible to come to a final conservative numerical estimate of the reliability of the replicated real-time systems. With a real system the formulae $FR_A \times PNP_B$ can quite reasonably deliver a failure rate of less than 10^{-7} per hour of operation.

V. Alternative Models

In any particular application context, if the model presented in this paper is to be adopted, technology must be chosen for channel B for which an argument for perfection is sustainable. In Section II possible technology was identified. In particular fixed priority, non-preemptive, time-triggered dispatching on a minimal RTOS was advocated. In some contexts engineers may feel preemptive and event-triggered scheduling is acceptably safe; in others, perhaps only cyclic executives and table driven dispatching can be deemed acceptably safe. The choices are important but do not effect the basic model developed here.

Similarly, as hardware platforms become more complicated the use of static timing analysis becomes more problematic and complex models of behaviour cannot sustain an argument of perfection. In these cases simpler forms of analysis (with resulting pessimism) or conservative forms of measurement must be used for Channel B.

If diverse hardware is part of the system model then it might be possible to argue that Channel B should be constructed from more deterministic components, for example FPGAs for which effective timing analysis is much simpler - and therefore notionally perfect.

The system model used in this paper has the property that both channels have the same deadline for the corresponding jobs. But in many application contexts the deadline for safe operation may be significantly longer than the deadline for effective or efficient functional behaviour. In these circumstances Channel B's claim for perfection is strengthened by the use of an extended deadline. This can also be used to compensate for the need to use simple pessimistic static timing analysis. At run-time, Channel B will normally still produce output before Channel A's deadline (and indeed before the equivalent job in Channel A has finished), but the argument for perfection for Channel B is based on its extended deadline.

VI. Related Work

The use of multiple channels to improve reliability is a common architectural approach in high integrity applications. As indicated in the introduction to this paper, the majority of the methods used to assess the overall system reliability for these architectures involves combining the reliability measure of each individual channel and then compensating for any dependencies between the channels. These approaches are therefore not directly comparable with the scheme described here.

One aspect of the proposed scheme is however used in the analysis of mixed criticality systems [31], [3], [1], [14], [2]. In this work the worst-case execution time (WCET) for each single component (thread) is obtained in two (or more) different ways. The different verification techniques produce different levels of accuracy and confidence. If the component is part of a high criticality computation then the larger (more reliable) WCET value is used, if lower criticality then the smaller (less reliable) value is employed. Although this element of the approach is similar, the overall objective of this work is to support different levels of criticality on the same platform. By comparison the approach developed in this paper is concerned with just the highest level of criticality and the evidence that can be used in a safety case to argue that all the deadlines with be satisfied by the implementation.

VII. Discussion and Conclusions

High integrity (e.g. safety-critical) systems require diverse replication to survive faults. Typically this replication takes the form of two or more channels, all of which undertake the crucial computations. In the absence of faults only one of these channels would be required to meet the requirements of the system. For real-time systems one aspect of these requirements is the meeting of deadlines for all the hard periodic and sporadic tasks running on the computing resources.

A key need for these safety critical systems is not only to be highly reliable but also for this high level of reliability to be demonstrable. These systems are usually certified, and this certification is evidence-based. Judging the reliability of a replicated system as a single artifact is hard/impossible, and hence it is necessary to come to a judgement about each channel and then compute the overall system reliability. The difficulty with this approach is that the level of dependency between the channels is difficult to judge.

The main contribution of the original work by Littlewood and Rushby, lies in the fact that there is conditional independence at the aleatory level between failure of one channel (A), and (non-)perfection of the other channel (B). This considerably simplifies the analysis in obtaining the bound for the system failure rate. In this paper we have built upon this analysis and present the following distinct contributions:

- The adaptation of the analysis to timing failures.
- The use of an argument of perfection to relate proven timing and scheduling analysis to actual safe run-time operation.
- The extension to failure rates (as apposed to failures on demand).
- The use of the Timebands framework to move between discrete and continuous phenomena.
- The argument that an assessor's (epistemic) beliefs about probability of failure of channel A and the probability of imperfection of channel B may also be independent.

Although the model considered in this work delivers independence, it does so at the price of considerable conservatism

– specifically in the assumption that if B is imperfect it fails whenever A does. In reality, any imperfection in B is unlikely to affect many cycles per hour, and these in turn are unlikely to involve A failures.

We do not deny that the assessor still faces a difficult task in expressing their beliefs quantitatively about the two parameters of the model. But it is considerably easier than the task faced by an assessor in the conventional situation in which they must reason about two fallible channels, and the dependence between these channels.

REFERENCES

[1] J. Anderson, S. Baruah, and B. Brandenburg. Multicore operating-system support for mixed criticality. In *Proceedings of the Workshop on Mixed Criticality: Roadmap to Evolving UAV Certification*, San Francisco, CA, April 2009.

[2] S. Baruah, H. Li, and L. Stougie. Towards the design of certifiable mixed-criticality systems. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium (RTAS)*. IEEE, April 2010.

[3] S. Baruah and S. Vestal. Schedulability analysis of sporadic tasks with multiple criticality specifications. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, Prague, Czech Republic, July 2008. IEEE Computer Society Press.

[4] I. Bate and T. Kelly. Architectural considerations in the certification of modular systems. *Reliability Engineering and System Safety*, 81:303–324, 2003.

[5] R.E. Bloomfield and P.G. Bishop. Safety and assurance cases: past, present and possible future. In C. Dale, editor, *Proceedings of Safety-critical Systems Symposium*. Springer – to appear, 2010.

[6] R.E. Bloomfield, P.G. Bishop, C.C.M. Jones, and P.K.D. Froome. Ascadadelard safety case development manual. Technical Report ISBN 0-9533771-0-5, Adelard, 1998.

[7] R.J. Bril, J.J. Lukkien, and W.F.J. Verhaegh. Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited. In *ECRTS '07: Proceedings of the 19th Euromicro Conference on Real-Time Systems*, pages 269–279. IEEE Computer Society, 2007.

[8] A. Burns and I.J. Hayes. A timeband framework for modelling real-time systems. *Real-Time Systems Journal*, 45(1–2):106–142, June 2010.

[9] A. Burns, I.J. Hayes, G. Baxter, and C.J. Fidge. Modelling temporal behaviour in complex socio-technical systems. techreport YCS 390, University of York, 2005.

[10] A. Burns and A. J. Wellings. *Real-Time Systems and Programming Languages*. Addison Wesley Longman, 4th edition, 2009.

[11] A. Burns, A. J. Wellings, C.M. Bailey, and E. Fyfe. The olympus attitude and orbital control system: A case study in hard real-time system design and implementation. In *Ada sans frontieres Proceedings of the 12th Ada-Europe Conference, Lecture Notes in Computer Science 688*, pages 19–35. Springer-Verlag, 1993.

[12] G.C. Buttazzo. *Hard Real-Time Computing Systems*. Springer, 2005.

[13] R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[14] D. de Niz, K. Lakshmanan, and R. Rajkumar. On the scheduling of mixed-criticality real-time task sets. In *Real-Time Systems Symposium*, pages 291–300. IEEE Computer Society, 2009.

[15] D.E. Eckhardt, A.K. Caglayan, J.C. Knight, J.D. Lee, D.F. McAllister, M.A. Vouk, and J.P.J. Kelly. An experimental evaluation of software redundancy as a strategy for improving reliability. *IEEE Transactions on Software Engineering*, 17(7):692–702, 1991.

[16] D.E. Eckhardt and J.D. Lee. A theoretical basis of multiversion software subject to coincident errors. *IEEE Transactions on Software Engineering*, 11:1511–1517, 1985.

[17] S. Edgar and A. Burns. Statistical analysis of WCET for scheduling. In *Proceedings 22nd IEEE Real-Time Systems Symposium*, 2001.

[18] E. J. Gumbel. *Statistics of Extremes*. Columbia University Press, 1958.

[19] J. Hansen, S. Hissam, and G.A. Moreno. Statistical-based WCET estimation and validation. Technical report, SEI, Carnegie Mellon, 2009.

[20] ISO/IEC. Information technology - programming languages - guide for the use of the Ada Ravenscar Profile in high integrity systems. Technical Report TR 24718, ISO/IEC, 2005.

[21] J.C. Knight and N.G. Leveson. Experimental evaluation of the assumption of independence in multiversion software. *IEEE Transactions on Software Engineering*, 12:96–109, 1986.

[22] B. Littlewood and D. R. Miller. Conceptual modelling of coincident failures in multi-version software. *IEEE Transactions on Software Engineering*, 15:1596–1614, 1989.

[23] B. Littlewood, P. Popov, and L. Strigini. Modelling software design diversity - a review. *ACM Computing Surveys*, 33:177–208, 2002.

[24] B. Littlewood and J. Rushby. Reasoning about the reliability of diverse two-channel systems in which one of the channels is "perfect". *Provisionally accepted for publication in IEEE Trans. Software Engineering*, 2010.

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

[26] B. Littlewood and D. Wright. Some conservative stopping rules for the operational testing of safety-critical software. *IEEE Transactions Software Engineering*, 23:673–683, 1997.

[27] J.W.S. Liu. *Real-Time Systems*. Prentice Hall, 2000.

[28] A. O'Hagan, C.E. Buck, A. Daneshkhah, J.R. Eiser, P.H. Garthwaite, D.J. Jenkinson, J.E. Oakley, and T. Rakow. *Uncertain Judgements: Eliciting Experts' Probabilities*. Wiley, 2006.

[29] K. Tindell and A. Burns. Guaranteeing message latencies on controller area network (CAN). In *Proceedings 1st International CAN Conference*, pages 2–11, 1994.

[30] K. Tindell, A. Burns, and A. J. Wellings. Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169, 1995.

[31] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proceedings of the Real-Time Systems Symposium*, pages 239–243, Tucson, AZ, December 2007. IEEE Computer Society Press.

[32] J. Woodcock, M. Oliveira, A. Burns, and K. Wei. Modelling and implementing complex systems with timebands. *Secure System Integration and Reliability Improvement*, pages 1–13, 2010.