ON THE PROBABILITY OF PERFECTION OF SOFTWARE-BASED SYSTEMS

Xingyu Zhao

Centre for Software Reliability
School of Mathematics, Computer Science & Engineering
City, University of London

Submitted for the degree of Doctor of Philosophy

Supervisors: Bev Littlewood and Robin Bloomfield

November 2016

In loving memory of my father and Xiaoyu

DECLARATION

The work in this thesis is based on the research carried out at the Centre for Software Reliability, City, University of London. No part of this thesis has been submitted elsewhere for any other degree or qualification. All work is my own unless otherwise stated.

Signed: ____

Date: ____

Xingyu Zhao

ABSTRACT

The probability of perfection becomes of interest as the realization of its role in the reliability assessment of software-based systems. It is not only important on its own, but also in the reliability assessment of 1-out-of-2 diverse systems. By "perfection", it means that the software will never fail in a specific operating environment. If we assume that failures of a software system can occur if and only if it contains faults, then it means that the system is "fault-free". Such perfection is possible for sufficiently simple software. While the perfection can never be certain, so the interest lies in claims for the probability of perfection.

In this thesis, firstly two different probabilities of perfection – an objective parameter characterizing a population property and a subjective confidence in the perfection of the specific software of interest – are distinguished and discussed. Then a conservative Bayesian method is used to claim about probability of perfection from various types of evidence, i.e. failure-free testing evidence, process evidence and formal proof evidence. Also, a "quasi-perfection" notion is realized as a potentially useful approach to cover some shortages of perfection models. A possible framework to incorporate the various models is discussed at the end. There are generally two themes in this thesis: tackling the failure dependence issue in the reliability assessment of 1-out-of-2 diverse systems at both aleatory and epistemic levels; and degrading the well-known difficulty of specifying complete Bayesian priors into reasoning with only partial priors. Both of them are solved at the price of conservatism.

In summary, this thesis provides 3 parallel sets of (quasi-)perfection models which could be used individually as a conservative end-to-end argument that reasoning from various types of evidence to the reliability of a software-based system. Although in some cases models here are providing very conservative results, some ways are proposed of dealing with the excessive conservatism. In other cases, the very conservative results could serve as warnings/support to safety engineers/regulators in the face of claims based on reasoning that is less rigorous than the reasoning in this thesis.

Keywords: probability of perfection, Bayesian, 1-out-of-2 systems, conservative reasoning, software diversity, fault-tolerant, quasi-perfection, fault-free, software reliability.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisors, Professor Bev Littlewood and Professor Robin Bloomfield, for their helpful guidance, deep insight and collaboration since my joining CSR.

In addition, I would like to thank Professor Lorenzo Strigini, Dr. Andrey Povyakalo and Dr. David Wright for their continuous support as additional advisors of my PhD research and co-authors for publications.

- Chapter 4 has been published as (Zhao, Littlewood et al. 2015) with Professor Bev Littlewood, Dr. Andrey Povyakalo and Dr. David Wright as co-authors.
- Chapter 5 and 7 have been written up as two journal papers (for JSS and RESS respectively) which are under review. Professor Bev Littlewood, Professor Lorenzo Strigini, Dr. Andrey Povyakalo and Dr. David Wright are co-authors.

I intensively involved in the writing of these papers, with valuable advices from my supervisor Professor Bev Littlewood and co-authors.

- Chapter 3 extends the the earlier work (Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a) of Professor Bev Littlewood and Dr. Andrey Povyakalo.
- The initial "pruning idea" in Chapter 4 was proposed by Professor Bev Littlewood.
- Chapter 5 extends the work (Littlewood and Wright 1997) of Professor Bev Littlewood and Dr. David Wright.
- The initial quasi-perfection notion in Chapter 7 was firstly observed by Professor Lorenzo Strigini and Dr. Andrey Povyakalo (Strigini and Povyakalo 2013). The LR-single-qp model in Chapter 7 was firstly proved by Dr. Andrey Povyakalo.

I thank all of my colleagues at CSR for their support, questions, insight and advice concerning this work.

I thank Professor Minyan Lu for her help in my PhD application.

I deeply appreciate my mother who has been a constant source of encouragement and support.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

This chapter firstly reports the motivation of this thesis, i.e. why the probability of perfection is of interest. Then research objectives and research questions are discussed.

1.1. Rationale for the research

1.1.1. Requirement and assessment of ultra-high dependability

It is a commonplace that our current human societies highly depend on software-based systems which are widely used in an increasing number of applications. This is particular true in the safety critical areas, where the failures may bring serious monetary loss and/or human suffering. Examples include the control of aircraft, industrial plants, railway and air traffic, weapons and military units, banking and commercial transactions. As a result, the dependability requirements for such systems are often very high. One frequently quoted example is the flight-critical avionics systems in civil transport airplanes; the requirement is less than 10^{-9} probability of failure per hour of operation (FAA 1988). For demand-based systems, there are similarly stringent requirements: e.g. the claimed probability of failure on demand (*pfd*) for the combined control and instrumentation safety systems on the UK European Pressurised Reactor (UK EPR) is 10^{-9} (HSE 2011).

In the first example, the high reliability requirement is due to the *massive* exposure worldwide for a particular aircraft fleet, and thus for a particular critical flight-control system.

For instance, (Boeing 2015) shows that the Airbus A320 and its siblings had experienced some 80m flight departures by 2013. As world-wide flights increase in frequency, there is pressure to *improve* aircraft safety so that accident frequencies also do not increase. In the nuclear example, even though exposure measured in operating hours will be much more modest, the worst case accidents may lead to catastrophically greater consequences and thus need to be extremely unlikely (HSE 1992).

To *achieve* this kind of ultra-high dependability is not only a difficult task of design and implementation, but also poses even harder problems of *assessment*. In the assessment, we need a credible argument that would convince a regulator that the software-based system of interest is reliable enough to use in a safety-critical context *before* it is deployed in actual operation. However, (Littlewood and Strigini 1993; Butler and Finelli 1993) tells the fact of the difficulties to propose those credible arguments, e.g. direct black-box operational testing would require an infeasible time on test, prior to deployment, to support claims for the failure rates, or *pfds*, needed in practice.

In this thesis, to deal with the difficult problem of assessment, a new approach was proposed. The idea is that, instead of reasoning about *reliability* – failure rates, *pfd*s – we claim something about perfection. In the following sections, readers will find the definition of perfection and why we should believe such perfection is possible. Then, with two examples, the advantages of claiming perfection in the reliability assessment task are explained.

1.1.2. Definition of perfection of software-based systems

In this thesis, perfection of a software-based system means that a "perfect" system will never fail no matter how much operational exposure (associated with a specific operational environment/profile) it receives. If we assume that failures of a software system can occur if and only if it contains faults, then it means that the system is "fault-free". Of course, no one can be certain that a system is perfect in the sense of the definition. But they may be prepared to accept that such perfection is *possible*. To capture this uncertainty, we shall use "probability of perfection" as a measure.

1.1.3. Why perfection is possible?

In fact, as John Rushby pointed out in (Rushby 2009; Littlewood and Rushby 2012), the traditional processes of software assurance, such as those performed in support of DO-178B (the guidelines for the certification of safety-critical aircraft software), can be best understood as developing evidence of possible perfection, rather than ultra-high reliability. Indeed, claims for the perfection of some systems may be more intuitively plausible than claims for very high reliability, since the two would be based upon different types of evidence and reasoning. A claim for 10^{-9} probability of failure per hour seems to acknowledge that the system in question is unlikely to be perfect – for example because of the complexity of its functionality – and resulting assessment of an extremely small number may not be believable[1]. On the other hand, a claim for perfection may be based upon evidence that the design is simple enough that the designers had a chance of "getting it right". In (Bertolino and Strigini 1998), the authors illustrated the scenarios may occur that make claims of perfection very plausible. For instance, the program performs a very *simple* function (e.g., comparing a sensor reading against a threshold and producing a single-bit output). Then via a simple design, it could be extremely simple in its implementation, resulting in few lines of code with very few branches. Additionally with much effort in checking the program, the developer would have a good chance of making the program perfect.

1.1.4. What is the benefit of claiming perfection?

As stated earlier, claim something about perfection is believed to be able to facilitate the reliability assessment task of software-based system. There are at least two ways in which the new notion of possible perfection will be benefit.

Firstly, similar as the point in (Bertolino and Strigini 1998; Strigini and Povyakalo 2013), perfection can be seen as a *lifetime* claim which concerns the need to assess the chance of being free from failures during the whole lifetime of the system. An artificial example could be considered here to explain the point. Imaging we have a critical on-demand[2] system (e.g. emergency shut-down system for a nuclear reactor) for which we expect 100 demands in its

[1] For example, in an exceedingly long operational testing, doubts about the correctness of the testing oracle and testing profile may come to dominate.

[2] The point is also applicable to continuously operating systems.

lifetime. If we would like to be 99% confident that it will survive all the demands without any failure, we need its *pfd* to be no worse than 10^{-4}. When the number of expected demands increased to 1000, we would need a *pfd* smaller than 10^{-5} to reach the 99% confident of being free from failures. For 10,000 demands, a *pfd* of 10^{-6} is needed, and so on. As can be seen, the required *pfd* could be very demanding when the expected number of demands in the lifetime is getting big. However, we could be 99% confident of seeing no failures in *any* number of demands if we were 99% confident in perfection. If we could support such perfection claim somehow (which is the major research objective of this thesis), the difficulty of assessing an ultra-high reliability is avoided.

The second advantage concerns the assessment of the reliability of diverse channel software-based systems. Such fault tolerance architecture is widely used in safety-critical application, and some evidence from industrial applications shows that this kind of design diversity has been successful (Littlewood, Popov et al. 2002). For example, the safety-critical flight control systems of Airbus fleets have experienced massive operational exposure (Boeing 2015) with apparently no critical failure. The intuitive explanation is “two heads are better than one”, i.e. if we force two or more systems to be built differently, their resulting failures may also be different. So if, in a 1-out-of-2 system (1oo2 system), channel A fails on a particular demand, there may be a good chance that channel B will succeed.

But such industrial evidence is only available after the fact. While in the licensing of a safety-critical system, we need an assessment tells that such a design-diverse system is reliable enough before it is deployed. The key problem in assessing the reliability of diverse software fault-tolerant systems lies in estimating the level of dependence between the failure processes of the two (or more) channels. Both experimental work (Knight and Leveson 1986) and theoretical modelling (Eckhardt and Lee 1985, Littlewood and Miller 1989) have showed that we cannot naively assume independence between the diverse channels which are independently developed. This is due to the inherent variation of difficulty of the problem being solved. Such as, for a 1oo2 system, if channel A fails on a randomly selected demand, this may increase the likelihood that the demand is a “difficult” one and so increase the likelihood that channel B also will fail. Therefore, we cannot simply multiply the marginal *pfds* of the two channels (say pfd_A and pfd_B respectively) to get a system *pfd*, i.e. (1.1).

$$pfd_{sys} \neq pfd_A \times pfd_B \tag{1.1}$$

In recent work (Littlewood and Rushby 2012), the authors proposed a new way to reason about the reliability of a special kind of 1oo2 systems. Here the channel A is conventionally engineered and presumed to contain faults, and thus supporting only a *pfd* claim (say pfd_A). On the other hand, the channel B is extremely simple and extensively analysed, and thus is "possibly perfect"; the claim about this channel B is a probability of non-perfection (say pnp_B[3]). Then the Littlewood-Rushby model (LR model) tells:

$$pfd_{sys} \leq pfd_A \times pnp_B \tag{1.2}$$

The result depends on the fact that there is *conditional independence* between the events "A fails on a randomly selected demand" and "B is not perfect", *given* that the probabilities of these events, respectively pfd_A and pnp_B, are known. In reality, of course, an assessor would not know pfd_A and pnp_B with certainty: there is epistemic uncertainty about their numerical values. This imposes upon the assessor the need to express *bivariate* beliefs about these unknowns jointly: this is known to be a difficult task. In later work (Littlewood and Povyakalo 2013a) addressed this problem. Their results require only an assessor's (partial) *marginal* beliefs about the parameters. There is a large literature on the assessment of *pfd* from statistical analysis of operational tests (Littlewood and Wright 1997; Musa and Ackerman 1989; Miller, Morell et al. 1992), so the first of the parameters could be relatively easier assessed, e.g. in terms of a Bayesian posterior distribution. That leaves pnp_B as a research interest.

To sum up, the probability of perfection is not only of interest on its own as a lifetime claim, but also plays an important role in the reliability assessment of diverse-channel systems, which motives the research in this thesis.

1.2. Research questions and objectives

From the previous section, we know probability of perfection plays an important role in the reliability assessment, especially in dealing with the dependence issue of 1oo2 diverse

[3] In this thesis, "probability of perfection" is the research subject, which is of course simply 1-*pnp*.

systems. To improve our understanding of probability of perfection and apply it in reality, there are two overall research questions need to be answered:

1. Since the parameter pnp_B in the original LR and LP models is a population property (see explanations in the literature reviews), LR and LP models are essentially describing what happens on average in terms of a population of 1oo2 diverse systems. However, in most cases, assessors are interested in the reliability of the *specific* software on hand. So how to tackle the issue of dependence of a *single* 1oo2 diverse system in line with the original LR and LP models? Similarly, can we find out more ways to extend the LR and LP models to broaden the use of probability of perfection?
2. To use the models requiring beliefs on perfection, we need to know how we can claim perfection in a probabilistic form from the various types of V&V evidence. Ideally, is there any possible way to combine the various types of evidence to support a stronger perfection-related claim?

To answer the two overall questions, specific research objectives are listed below and form the overall structure of this thesis.

1. Research question 1 requires a comprehensive understanding of the dependence problem of diverse channel software systems. So literatures are reviewed in **Chapter 2** on the topic of software diversity: is diversity helpful; how to achieve diversity; and most importantly how to assess the diverse systems? All these questions lead to the problem of dependency and motivate the use of probability of perfection.
2. Essentially there are two types of probability of perfection, i.e. a subjective confidence (probability) in the perfection of the *single* program of interest (e.g. the first example in section 1.1.4) and an objective parameter of a population of programs (e.g. the second example in section 1.1.4). Via the population parameter pnp_B, the original LR and LP models are essentially describing what happens on average in terms of a population of 1oo2 diverse systems. **Chapter 3** discusses the relations and differences between the two different probabilities of perfection and extends the LR and LP models to tackle the issue of dependence of a *single* 1oo2 diverse system.
3. Failure free testing evidence seems the most common evidence for a possible perfect system in reality. To what extend can this type of evidence help us to increase our confidence in perfection? Topics on conservative claims for perfection via failure free runs evidence are discussed in **Chapter 4**.

4. One important source of evidence for the good quality of software-based systems is the "process evidence" which means the perfect behaviours of previous similar products in this thesis. Modelling via good process evidence (i.e. failure-free runs of a population of similar products) to learn about the probability of perfection is investigated in **Chapter 5**.
5. The evidence of formal verification seems a strong support to perfection claims. However, due to the uncertainties in a formal verification process, there are limitations of formal proof evidence to support perfection claims. In **Chapter 6**, how various sources of uncertainty affects the probability of perfect after seeing formal proof evidence is analysed. And what need to be elicited from assessors to get a quantitative posterior belief on perfection is discussed.
6. It has been noticed as a fact that "perfection" and "extremely small *pfd*" are effectively indistinguishable in practice as explanations for extensive failure-free working. The notion of "quasi-perfection" is introduced in **Chapter 7** to help modelling on the *pfd* of a 1oo2 diverse system. The advantages and limitations of "quasi-perfection" are discussed.
7. In **Chapter 8**, the work in previous chapters are summarised into 3 parallel sets of models spanning over 4 levels (aleatory, epistemic, learning and evidence levels). A possible framework to incorporate different kinds of evidence is proposed. Finally, contributions and limitations of each chapter and the whole thesis are discussed, as well as future work.

2. LITERATURE REVIEW

Since the diversity idea was first introduced as a fault tolerance means, four decades have passed. Research on software diversity has expanded in multiple directions: goals (fault tolerance, security, software engineering), means (managed or automated diversity), and analytical studies (quantification of diversity and its impact) (Baudry and Monperrus 2015). However, most of the recent achievement is essentially based on the classical techniques, experiments and modelling. In this chapter, literatures on the *classical* use of diversity for software are reviewed and summarised to answer questions like: what are the popular techniques based on the diversity idea? Is diversity useful? How to build diverse systems? Most importantly, how to assess diverse systems?

2.1. Techniques based on diversity

2.1.1. Design diversity

Design diversity is specifically used to tolerate design faults in programs arising out of wrong specifications and incorrect coding. Multiple versions of the software are independently developed by different development processes (e.g. different teams, different languages etc.) These variants are then used in a redundant manner to achieve fault tolerance. Popular techniques based on the design diversity idea for fault tolerance mainly include:

- **N-version programming**: Similar to the N-modular programming approach in hardware fault tolerance-version programming, N-version programming was first proposed (Avizienis and Chen 1977) in 1977. In this technique, two or more functionally equivalent programs are named versions which are independently developed. Then these N versions will be executed in a parallel manner, with an adjudication logic (e.g. majority voting) being used to compare the results produced by all the versions and provide a single adjusted output. Despite of the high cost of generating different versions and implementing the voting logic, this technique has been applied to a number of real-life systems such as railroad traffic control and flight control.

- **Recovery blocks:** This technique is analogous to the cold standby scheme for hardware fault tolerance, which was first introduced in (Horning, Lauer et al.1974). Similarly as N-version programing, multiple variants of software which are functionally equivalent are independently developed, but deployed in a time redundant fashion. An acceptance test is used to test the validity of the result produced by the primary version. Passing the test means the acceptance and use of the results from primary version. On the other hand, when the results from the primary version were rejected by the acceptance test, another version among the diverse versions would be invoked and executed to generate new results which would be tested by the acceptance testing again. This procedure would not stop until the acceptance testing is satisfied by one of the diverse versions or until the worst case that all the versions have been exhausted.

There are two significant differences between recovery blocks and N-version programming. First, only one version is executed at a time in recovery blocks approach; second, the validity of results is decided by an acceptance testing rather than by adjudication among the N outputs. The recovery block technique is the important basis for the distributed recovery block structure which has been applied to real-life command and control applications (Randell and Xu 1995).

There are other forms of techniques which are essentially the mixtures of these two basic ones, such as recoverable N-version blocks and N-self checking programming. The common feature of all these approaches is the independent generation of functionally identical program versions. Then the versions may be executed in parallel or sequentially, or mix of the two cases (Voges 1994).

2.1.2. Data diversity

Data diversity was first introduced by Amman and Knight (Amman and Knight 1987), and relies on the observation that a software sometime fails for certain values in the input space and this failure could be avoided if the minor perturbation in input data is acceptable to the software. There are two original data diverse software fault tolerance techniques developed by Amman and Knight (Amman and Knight 1988):

- **N-copy programming**: N-copy programing is the data diverse complement of the N-version programming in design diversity. Being different from N-version programing which provides fault tolerance capability via multiple diverse versions of the software written against the same specification, N-copy programing technique has N *copies* of a single program. The diverse-data system produces a related set of points in the data space, and then each copy runs on a different input set. An enhanced voting scheme (normally with more complex and precise algorithms than majority voting mechanism) is used to select the system outputs.
- **Retry blocks**: It is the data diverse complement of the recovery block technique. A watchdog timer is used and triggers the execution of a backup version if the original algorithm does not produce an acceptable result within a specified period of time. The primary version is executed using the original system input and then the results are tested by acceptance testing. If the results are not accepted, the input data will be "re-expressed" (by some built in re-expression algorithms). The primary version will be run using the re-expressed input data and tested again. By that analogy, the execution will end when a result passes the acceptance testing or the time period violates the watchdog timer. In the case of the time deadline expires, a backup version may be invoked to execute on the original input data.

In most of the real time control programs, since sensor values are usually noisy and inaccurate, data diversity techniques may be able to prevent a failure. However, data diversity based techniques have their own limitations; due to that equivalent input data re-expression might not be acceptable by all requirement specifications for some engineering tasks.

2.1.3. Environment diversity

Environment diversity is also an approach to tolerance fault in software (Trivedi and Vaidyanathan 2002). A dramatic instance of applying the environment diversity idea is restarting, as pointed by (Adams 1984) that restarting the software-based systems is the simplest and best approach to masking software faults. The environment diversity related techniques are based on the observation that most of the software failures are transient in nature. Transient failures are mostly due to the design faults in software which result in unacceptable and erroneous states in the operating system environment (Jalote, Huang et al. 1995). Thus environment diversity techniques attempt to provide a new or modified operating environment for the running software. Once seeing failures, the software will be executed in a different operating environment (e.g. a new OS environment state which is achieved by some clean-up operations).

Even though there seems no empirical experiment showing the benefits of environment diversity (like the ones for design diversity that you will see in later section), some theoretical analysis shows the potential usefulness of environment diversity in terms of certain type of faults (Trivedi and Vaidyanathan 2002).

Retry operation, restart application and rebooting (can be done on the same node or on another spare cold/warm/hot node) are all examples of environment diversity techniques. The new research area – software rejuvenation (Huang, Kintala et al. 1995;Cotroneo, Natella et al. 2014) is also a specific technique of environment diversity (Trivedi and Vaidyanathan 2002).

2.2. Is diversity design useful?

Diversity design (e.g. N-version programming) is the most classical and extensive application of software diversity. There had been some controversy about the benefits that the approach brings since it was first introduced. In earlier years, research work was mainly focusing on the effectiveness of software diverse design; or to be exact, whether the diversity design could bring us the benefit of allowing us to simply use the result assuming versions fail independently. To understand that, both empirical experiments and theoretical modelling work was conducted.

2.2.1. Experiments on software diversity design

Fault diversity

Clearly any common faults would limit the degree of reliability improvement, and certainly it would be unreasonable to expect failure independence in such cases. If diverse design results in diverse faults, then we should have higher confidence (than the case with common faults) in the fault tolerance capability of diverse systems.

Some of the earliest experiments (Dahll and Lahti 1979; Gmeiner and Voges 1980; Kelly and Avizienis 1983; Dunham 1986) centre on analysing the software faults diversity. One important common conclusion from all these experiments is that most of the faults are similar due to the low quality (incompleteness and ambiguity) of specifications. As the faults caused by design and coding tend to be discovered in the verification and validation stage, a high proportion of specification related faults were presented in the final program versions. For example in the experiment (Gmeiner and Voges 1980), 12 specification faults were found out of a total of 104. Then after acceptance testing, there are 10 specification-related faults out of a total of 18. Another extreme example is the Project on Diverse Software (PODS) (Bishop 1986), in which three diverse teams (in England, Finland and Norway) implemented a nuclear protection system under very good quality control. It turns out that all the faults were caused by omissions and ambiguities in the requirements specification, i.e. no design and implementation related faults after testing. Of course such specification related common faults will undermine the assumption of expected independence of failure.

In the experiments (Kelly and Avizienis 1983; Bishop 1986), diverse specifications from common requirements were introduced. In general, diverse specifications can potentially reduce specification related common faults, but the performance is uncertain and there is a risk that the specifications will not be equivalent. So the value of using diverse specifications in reality is still unclear.

In some of the experiments mentioned above, the impact of programming language had been checked. Programming language has impact on coding related faults which tend to be fixed before the final program. For specification and design related faults, the use of diverse programming languages has little effect on them.

Later, to deal with the specification related common faults, the "design paradigm" (which is a set of rigorous guidelines for the implementation of N-version programming) was proposed in (Avizienis, Lyu et al. 1988; Lyu and He 1993). In (Avizienis, Lyu et al. 1988) which is the very first experiment using this paradigm, only two specification-related common faults out of a total of 93 was found. In the experiment (Lyu and He 1993), no specification related faults were found after acceptance testing. More importantly, there was a big improvement of residual faults comparing to previous experiments and significant reduction in identical or very similar faults.

For "off-the-self" (OTS) products, Gashi conducted a study on the fault diversity of four OTS SQL servers in (Gashi, Popov et al. 2004). They found that very few bugs affected two of the four servers, and none caused failures in more than two servers. So it seems the diverse OTS servers (with diverse faults) have a good chance to deliver high reliability comparing with individual servers or any replicated configuration.

But there is still a gap between faults and failures. Even though the faults are diverse, the channels still could fail at the same time, which had been observed in (Bishop and Pullen 1987). So the reliability improvement could only be determined by the degree of *failure* dependency between the diverse channels. Experiments on observing the failure behaviours of diverse design systems are discussed in following.

Failure diversity

To directly evaluate the failure dependency, the highly referenced experiments (Eckhardt, Caglayan et al. 1991; Knight and Leveson 1986) had shown us that the *independence* of failures cannot be supported. For example, in Knight and Leveson's experiment, 27 versions of a program were implemented independently based on a common Missile Launch Interceptor specification at two universities, and then subjected to one million tests. The specification was claimed not to affect the outcome, based on the fact that the specification was built on the experience of earlier experiments and carefully independently reviewed. The major conclusions are the assumption of failure independency cannot be held, but the benefits may still be considerable on average. Even though disputes arose over the conclusions of the experiment, (e.g. the realism that it used students rather than professional software developers), later some theoretical analysis (see next section) supported the points of the experiments.

Rather than testing for the failure *independence* assumption, another experimental evaluation of the *degree* of failure dependency was made in a follow-up work to the PODS project, named STEM project (Bishop and Pullen 1987). The special part of this experiment is that all the known faults (containing both the specification-related and the implementation-related faults in PODS) could be switched on and off, so it is possible to measure the individual and coincident failure rates of all possible fault pairs, and then make comparisons with the independence assumption. For example, set one fault for channel A and test the failure rate say P_A, and set another fault for channel B and test the failure rate say P_B. Then assemble these two channels into a 2-version system and test the failure rate say P_{AB}. By comparing the $P_A \times P_B$ and P_{AB}, we could know the dependency of failures and how it was affected by the dependency of faults. The result was that random selected fault pairs were giving a distribution of the *degree* of failure dependency ranged from strong positive correlation to strong negative correlation. Therefore the simple independent assumption of failures between diverse channels was disproved again.

By looking into the details of the fault pairs (non-identical) in the STEM experiment, analyses (Bishop and Pullen 1991) have been carried out to explain the observation of both strong positive correlation and strong negative correlation. Specifically in the analysis of positively high dependency fault pairs, besides the well-known factor of common mode faults (e.g. in the requirements specification), there were some strange clusters of high dependency fault pairs observed which were of little or no commonality. Later by more examination, the "error mask" was discovered as a source of high dependency. The error masking is an inherent feature of the program functions. Any output whose computation relies on masking functions (e.g. AND gates, OR gates, MAX and MIN functions) is likely to exhibit dependent failures in diverse versions.

Error masking is not the only inherent feature of software that could result in two dissimilar faults leading to high levels of coincident failures. In a NASA experiment (Kelly, Eckhardt and el at. 1988), there is one function called the Fault Detection and Isolation (FDI) module. Faulty sensors would be detected by this module, then other subsequent modules can compute the input data which is from the remaining good sensors. Faults in FDI module will cause either too few or too many sensors to be used. In either circumstance, there would be "noise" in the collected data and thus cause a failure. So basically, any fault in FDI module (the reasons for the sensor diagnosis errors may differ) will change the status of the sensors which will finally cause a failure of the whole system.

All in all, the main lesson we learned here is that we cannot simply claim failure independence of diverse channels, due to both the common mode faults and dissimilar faults. The sources of common mode faults are mostly ambiguities and omissions in the specification and common implementation mistakes. Even though efforts are made (e.g. the diverse programming paradigm) to minimise the risk of containing common mode faults, the dissimilar faults would as well cause a burst of coincident failures due to some inherent features of software (e.g. error masking). However generally speaking, even if there is dependency of failures, the diverse approach used in the experiments can benefit us on average.

2.2.2. Theoretical models on software diversity design

The gap between laboratory and specific real-world projects can never be excluded. All the laboratory experiments mentioned above have their own limitations. Therefore theoretical modelling work had been proposed to support and extend the points of the experimental work.

The Eckhardt and Lee model (EL model)

Intuitively, it is possible that some programing tasks are intrinsically harder than others, so all programmers tend to make mistakes with greater probability in such circumstances. Based on that *difficulty variation* idea (which initially named as "intensity function" but later refined by Littlewood and Miller in (Littlewood and Miller 1989) as "difficulty function"), Eckhardt and Lee (Eckhardt and Lee 1985) proposed the first probabilistic model that tried to capture the nature of failure dependency.

First in the EL model, implementing a version of program is represented by randomly selecting a version from a population of programs. Then the key variable is the difficulty function $\theta(x)$, defined to be the probability that a program chosen at random will fail on a particular demand x. If we did a thought experiment that infinite programs have been selected (via the probability distribution over all possible programs) and tested, the proportion of those failed is the value of $\theta(x)$ for the demand x. This seems a natural definition of the intuitive notion of difficulty: the more difficult a demand, the greater we would believe the chance that an unknown program will fail. Difficulty should vary across the demand space, i.e. for two different demands x_1 and x_2, $\theta(x_1) \neq \theta(x_2)$.

If v_1 is a random selected version, then we have:

$$P(v_1 \text{ fails on a random demand}) = E_X\big(\theta(X)\big) \tag{2.1}$$

where X is the random variable representing demands.

For any given demand x, and two independent randomly selected versions v_1 and v_2, we have:

$$\begin{aligned} &P(v_1 \text{ fails}, v_2 \text{ fails} \mid \text{demand } x) \\ &= P(v_1 \text{ fails} \mid \text{demand } x) \times P(v_2 \text{ fails} \mid \text{demand } x) \\ &= [\theta(x)]^2 \end{aligned} \tag{2.2}$$

Then the probability that a randomly selected pair of programs (i.e. a diverse system) both fail on a randomly selected demand:

$$\begin{aligned} &\sum_X P(X = x)P(v_1, v_2\ both\ fail \mid demand\ x) = E_X([\theta(X)]^2) \\ &= \big[E_X\big(\theta(X)\big)\big]^2 + Var_X[\theta(X)] \end{aligned} \tag{2.3}$$

In the right hand of expression (2.3), the first term is simply the naive result using the independence assumption, i.e. multiply the two probabilities that randomly selected program fails on randomly selected input. And the second term $Var_X[\theta(X)]$ is always positive (or 0), so the real probability of failure of the diverse system is always bigger (or equal when $Var_X[\theta(X)] = 0$) than the one under the independence assumption. To be more exact, the more the difficulty varies (i.e. a bigger $Var_X[\theta(X)]$) between demands, the greater the failure dependence between the two channels. When there is no variation in difficulty of different demands, we indeed could claim independence of failures by this model, but this seems unlikely in practice. Instead, it is fair to claim that there will always be positive variance, and thus system reliability can always be expected to be lower than it would be if there was independence.

This EL model is important, as it is the first one trying to formally model the meaning of independent development of software versions and the nature of failure dependency. Here, as the programs are randomly selected from a population, so "independence" here seems to be essentially about process: the teams simply did not communicate with each other while developing their own versions. But in practice, this independent development is not realistic. Actually, another more plausible approach is to force diversity in the development processes of the different versions, e.g. deliberately use different teams, different programing languages and so on. Based on this point, the EL model later was refined by Littlewood and Miller, which will be introduced in next subsection.

The Littlewood and Miller model (LM model)

The LM model (Littlewood and Miller 1989) generalized the EL model by taking into account of the forced diversity approach in development process. So the difficulty functions for the two different develop methodologies[4] should not be the same. Use the similar notations in the EL model, say $\theta_A(x)$ and $\theta_B(x)$ as the difficulty function for a particular demand x in the methodology A and B respectively. Then the probability of a random selected version (say v_A) from the population A (i.e. developing a version via methodology A) fails on a random demand is $E_X\big(\theta_A(X)\big)$. And similarly, we have $E_X\big(\theta_B(X)\big)$ for the v_B.

For any particular demand x, the two randomly selected programs v_A and v_B (from their own population respectively) will fail independently:

$$\begin{aligned} &P(v_A, v_B \text{ both fail} \mid \text{demand } x) \\ &= P(v_A \text{ fails} \mid \text{demand } x) \times \mathrm{P}(v_B \text{ fails} \mid \text{demand } x) \\ &= \theta_A(x)\theta_B(x) \end{aligned} \tag{2.4}$$

Then the probability that a randomly selected pair of versions (i.e. a diverse system) both fail on a randomly selected demand:

$$\begin{aligned} \sum\nolimits_X P(X = x)P(v_A, v_B \text{ both fail} \mid \text{demand } x) &= E_X\big(\theta_A(x)\theta_B(x)\big) \\ &= E_X\big(\theta_A(x)\big)E_X\big(\theta_B(x)\big) + Cov_X\big(\theta_A(x), \theta_B(x)\big) \end{aligned} \tag{2.5}$$

It is easy to tell that the EL model is just the special case of the LM model when $\theta_A(x) = \theta_B(x) = \theta(x)$. Again, the first term in the right hand side of formula (2.5) is just the naive

[4] The set of constraints imposed on the development of a version is called methodology in the original LM model paper.

independent failure result, i.e. it is merely the product of the probabilities of failure of programs v_A and v_B. The innovative part is the second term in (2.5). Since a covariance could be either positive or negative, it is no longer certain that the probability of failure of both randomly selected versions (i.e. the result (2.5)) will be greater than in the independence case (i.e. $E_X(\theta_A(x))E_X(\theta_B(x))$). If the difficulty functions were negatively correlated, the reliability of a 1-out-of-2 system could be even better than the one under the independence failure assumption. The informal explanation is that what is difficult for A is coincidently easy for B, and vice versa.

Although negative correlation exists theoretically, it is very hard to justify it in practical applications. If we could assume negative correlation in some cases, then we would be able to claim the independence-based result as a conservative bound on the real system reliability. In (Littlewood and Povyakalo 2013b), the authors proposed some conservative bounds for the *pfd* of a 1-out-of-2 software systems, based on an assessor's subjective probability of "not worse than independence" which, only in special cases, we could know from assessors.

It is also worth mentioning that, if keeping the version reliabilities fixed in the EL and LM models, the LM model will always give a better result (a greater expected reliability of a diverse system) than the EL model. In this sense, the EL model can be seen as a worst case within the more general LM model, and it is reasonable to think that the EL result is unattainable.

The main lesson from the LM model to be learnt is that, though it is theoretically possible (by forced diversity in development process) to obtain a diverse channel system which exhibits a better than independent failure behaviour, we could not simply assume the better-than-independence case of channels in a diverse system. In fact there are possibilities of diverse channel failures from positive correlation and to negative correlation, and independence is just one very special case. There is no reason to expect failure independence is more likely than other cases without extra evidence.

2.2.3. Diverse versions or high reliability in a single version

From what we have learned above, even though independence between channel failures is unrealistic, it is reasonable to believe that diversity would contribute to reliability in an average sense. However, from the engineering point of view, we cannot ignore the cost diversity design idea brings, as the common sense that developing software is very expensive and time consuming. Is the diversity design method cost-effective? What if the same effort were applied on a single version system, will diversity still result in a more reliable system?

The cost of diversity has been discussed for example, in (Laprie, Arlat et al. 1990) and (Voges 1994). Basically, if we look at the cost of software diversity we need to distinguish which kind of diversity is applied. Taking the software life cycle to comprise Specification, Design, Coding, Testing and Use, the monetary costs for diversity are not the same for each stage in any particular circumstance. For example, if three teams independently construct a code from a common design, the cost of specification (ignoring the need to avoid ambiguities leading to discrepancies between the versions) and design are not affected, and the coding cost is simply triplicated, but the effect on the testing phase must be looked at in more detail: the possibility of testing the multiple versions back to back may reduce the cost of verification (compared to verifying N-versions separately). Last, the development of multiple versions has greater organizational costs than that of one version, in terms of coordination effort, cost of delays, and so on.

Hatton (Hatton 1997) in 1997 published a strongly argued paper in favour of the cost-effectiveness of design diversity basing on two empirical observations:

- On average, 2-out-of-3 systems are 45 times more reliable than the individual version.
- The "state of the art" development processes is costing much more than "ordinary" ones. And it delivers a 10 factor improvement in reliability over the ordinary processes.

Hatton's analysis crucially depends on the assumption that the gain of fault-tolerance will increase when the reliability of individual channels increase. It concluded design diversity is *always* more cost effective than improving the reliability single version software.

But later, this "10-vs-45" figure pair was questioned. First in the modelling work (Popov, Strigini et al. 2000), the authors refuted the ratios by showing that an increase in version reliability is not necessarily going to increase the gain from fault-tolerance. This refutation is based on both the experimental results (in section 2.2.1) and theoretical models (the EL and

LM models). And then more experimental examples (Littlewood, Popov et al. 2000a) were proposed to support the conclusion that we cannot simply assume the fault tolerant approach will be better. Actually, both trends could win and it depends on the constraints on the development process.

2.3. Achieve diversity

Even though we cannot claim failure independence, diversity is still a good thing to improve the reliability of software-based system in some average sense. Besides the very classical and basic use of diversity in section 2.1, diversity idea could be applied in any phase of the lifecycle in the software engineering process, for instance, the diverse organizations of development teams, diverse development environments, different tools and languages used at every level of specifications and coding, different algorithms for the implementation of functions, diverse V&V methods etc. As project engineers, we have to make decisions on seeking diversity within an acceptable cost. In this section, previous work on giving suggestions on achieving diversity in real projects is reviewed.

2.3.1. Diversity–Seeking Decision (DSD)

The term "diversity-seeking decisions" (DSD for brevity) was first proposed in (Popov, Strigini et al. 1999) to represent all the subtle decisions to achieve diversity in a software development process. Before that, there was no scientific guidance on how to achieve effective diversity for a given project and project managers were making decisions on rather "common-sense" advices, e.g. (Lyu and He 1993). The difficulties in DSDs were raised by questions like:

- How many DSDs are there? And what are they?
- How effective is a single DSD? And how effective is any particular combination of DSDs?
- What is the cost of each DSD? What is a cost-effective set of DSDs?

In (Popov, Strigini et al. 1999), the authors posed similar questions and first attempting answers by looking into the causal links from DSDs to failure diversity, as the cited Figure 1. DSDs firstly produce "process diversity". Visible differences in the structures and internal operations of versions were presumably caused by this "process diversity". And these different versions are "diverse products". One may hope that the "diverse products" were containing fewer numbers of identical faults than if the DSDs were not employed in the first place. This observation was termed as "fault diversity". Finally, if successful, the actual goal of "failure diversity" would be achieved.

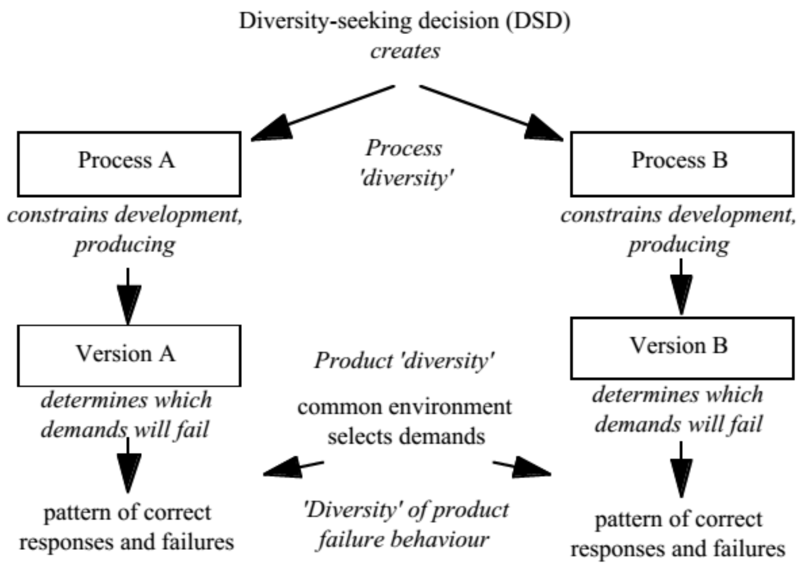
Diversity-seeking decision (DSD)
creates
Process A
Process 'diversity'
Process B
constrains development, producing
constrains development, producing
Version A
Product 'diversity'
Version B
determines which demands will fail
common environment selects demands
determines which demands will fail
pattern of correct responses and failures
'Diversity' of product failure behaviour
pattern of correct responses and failures

Figure 1 the causal links from DSDs to failure diversity

Knowing the mechanism how DSDs cause failure diversity, a complete list of DSDs was first introduced in (Littlewood and Strigini 2000) as below:

- Data diversity
    - Using random perturbations of inputs.
    - Using algorithm specific re-expression of inputs.
- Design diversity
    - Separate development (independent development)
    - Diverse development teams (forced diversity)
    - Diversity in description/programming languages and notations
    - Diverse requirements or specifications
        - Different expressions of substantially identical requirements
        - Different required properties implying the same behaviour
        - Requiring different behaviours from the diverse versions
    - Diverse development methods
    - Diverse verification, validation, testing
    - Automatic code transformation
    - Diverse development platforms:
        - Diverse tools.

        - Diverse compilers (also applicable to replicas of a single version).
    - Diverse support platforms: run-time platform
        - Separation and loose coupling. Diverse timing
        - Diverse hardware
        - Diverse operating systems or run-time executives
        - "Partial" diversity, limited to subsystems
- Functional diversity

For each item on the list, detailed analysis was carried out on:

- Mechanism of action, problems tolerated: this is relevant for matching DSDs to perceived threats, and checking that all threats against which diversity is the preferred defence are actually "covered".
- Considerations on cost, efficacy, and practical experience: Having identified some cost factors, predictions of detailed costs should then reflect the cost structures of the specific organisations involved.

Although the report (Littlewood and Strigini 2000) did not state any strong recommendations, it is useful to give indications for designers, project managers and dependability assessors in expecting on how effective these decisions will actually improve the delivered multi-version products.

2.3.2. Single DSD

Research work had been done on checking the efficiency of some items on the list. Even though the scope here does not cover the whole list, literatures on some representative DSDs are reviewed.

DSD – diverse programing languages

Meulen and Revilla in (van der Meulen and Revilla 2008) initiated a big range of empirical studies (covers 89,402 programs written to 60 specifications, on average 1,466 per specification) and reported some initial results about the effect of a single DSD, the diversification of programming language. They confirmed that programmers would make different kinds of faults with different languages. Therefore the diversity of programming language has positive effect on system reliability. And because of the size of their dataset, the authors stressed the statistical validity of their conclusions.

DSD – diverse development teams

For the DSD "Diverse development team (forced diversity)", Salako and Strigini showed that N-version programming may bring benefits even if the strict separation of development team is not achieved, and furthermore some communications between teams could bring us the benefit of "forced diversity" (Salako and Strigini 2013). Their work also gave three preference criteria between alternative ways of organising the development teams, using the expected system *pfd* as a suitable basis for decision-making.

DSD – diverse fault remove techniques

Another related work involving modelling the effects of diverse techniques used in software development to detect and remove software faults in the development of a single version of a program (Littlewood, Popov et al. 2000c), was inspired by the EL and LM model. With similar mathematical model as the EL and LM, the new model for diverse fault removal showed us some conclusions:

- When we keep repeating the application of a particular fault finding procedure, the effect of each iteration upon system reliability cannot be simply assumed statistically independent. And such an incorrect independence assumption will always lead to too optimistic results.
- Similar as one of the conclusions of the LM model, there is an intriguing possibility of *better than independent* behaviour of the diverse fault finding procedures. That requires a negative covariance between the targeted types of faults of the different fault removal procedures, i.e. whenever the 'difficult' faults for one procedure are the 'easy' ones for the other, and vice versa.
- If we could assume indifference among all fault finding procedures, then diversity should be applied as widely as possible. For example, AABCC is better than AAABC.
- As the parameters in the model seem practically attainable, a systematic investigation of these parameters for different fault-finding procedures - and different classes of software application domains - in industrially realistic situations was suggested.

DSD – functional diversity

From the sections above, on average, we know diversity is a useful way to improve reliability, even though we could not simply claim independence between failures of versions. Analysis implied the reason was that *difficult* demands tend to fail most the versions no matter how diverse they are. But what if we make the problem different, i.e. diversify the input? For instance, a diverse 1-out-of-2 system as a nuclear emergency shutdown protection system, the channel A makes its trip decision based on temperature inputs, and the channel B on pressure inputs. As there is no obvious source of common mode failures, it seems this higher level diversity could let us claim something no worse than independence of the channel failures.

The intuitive answer seems appealing and promising to the achievement of reliability in diverse system. Detailed work (Littlewood, Popov et al. 1999) was proposed to strictly answer how reliable could be claimed for the use of functional diversity. And specifically, could independence of channel failures be claimed?

The model used in (Littlewood, Popov et al. 1999) was a generalization of the earlier EL and LM model, which basing on the notion of variation of difficulty. The result turned out to be that failure independence between functionally diverse systems is rather unrealistic. Instead, just like the normal N-versions programming, functionally diverse systems will tend to exhibit positively correlated failures. The explanation of the result was similar to the ones executing the same inputs, showing that the difficulty functions for the developers of the two functional different versions must be considered.

The general conclusion here is that functional diversity, as a way of tolerating design faults and achieving high reliability, must be seen as a special kind of forced design diversity, requiring positive evidence for any claim of low correlation between failures.

2.3.3. Combination of DSDs

Extensive experiments about diversity were conducted (as mentioned in section 2.2.1), but few addressed alternative DSDs and only on small samples. Experimental evidence of the effectiveness of DSDs meant to force diversity among different versions, and in particular combinations of such DSDs, is rather rare. Meulen and Revilla in (van der Meulen and Revilla 2008) initiated a big range of empirical studies (covers 89,402 programs written to 60 specifications, on average 1,466 per specification) and reported some initial results about the effect of a single DSD, the diversification of programming language. Later in (Popov, Stankovic et al. 2012), the authors reused and extended Meulen's work into the consideration of two DSDs, diverse programing languages and diverse implementation algorithms. The general conclusions of these experiments are: first, some single DSD or combination of DSDs produces higher reliability than naive diversity maximization on average; second, the improvement varies significantly between the DSDs and the problems. So there are no simple universal rules, e.g. the more diverse the better, meaning the relationship between DSDs and the system reliability is complex and needs analysis under the specific circumstance.

2.3.4. Diverse arguments

A totally different new research line started about 10 years ago, looking into the efficiency of using diverse arguments to support dependability claims in a safety case.

The use of diversity in arguments, the so-called multi-legged arguments, is specifically used to reduce the doubt in the claims about the dependability of a system (e.g. a *pfd* claim). Examples are:

- A two-legged argument was used in (Hunns and Wainwright 1991) for the Sizewell B computer-based primary protection system, based on the UK's Safety Assessment Principles for Nuclear Power Plants (HSE 1992). One leg is about the product itself, basing on the V&V results and the other leg is about the quality of the development process.
- There are some standards and codes of practice (MoD 1997; CAA 2001) that suggesting the use of diverse arguments. For instance, the UK Def Stan 00-55 (MoD 1997) suggested that one leg be based upon logical proof of correctness, and the other upon statistical testing.

The examples shown above reflect the need of better understanding of the use of diversity in arguments. But before the paper (Bloomfield and Littlewood 2003), there was only informal justification for the use of multi-legged arguments for software. In (Bloomfield and Littlewood 2003), to model the diverse arguments, the authors introduced the notion of "confidence in claims" later refined in (Bloomfield, Littlewood et al. 2007). Then the efficacy of the multi-legged approach lies in how much confidence in a claim is increased by the multiple diverse arguments. With a primary model and some simple examples, the conclusion is that the efficacy depends crucially (as for design diversity) on the notions of dependence between arguments, e.g. between their assumptions.

The latest work (Littlewood and Wright 2007) looked in detail at a simplified and idealized 2-legged argument (comprising a verification leg and a testing leg) which centres upon a claim about the system *pfd*, represented by a parametrised Bayesian Belief Net (BBN). Via manipulating the parameters that defined its node probability tables, the authors showed that, in most cases, adding a diverse second argument leg will increase the confidence in a dependability claim. However, there are two counter-intuitive and unexpected subtleties:

- First, for a single leg, seeing definitely supportive evidence (i.e. correctly verified for the verification leg and no or very few failures for the testing leg) will *not always* increase our confidence in the dependability claim.
- Second, an entirely *supportive* second leg (i.e. a leg that increases the confidence in a dependability claim on its own) can sometimes undermine an original argument, resulting overall in less confidence than came from this original argument.

Even though there needs more work to look into the circumstance in which this counter-intuitive result will happen, the paper emphasised the importance of having this kind of formal reasoning about arguments and confidence.

2.4. Assess diverse software

The reliability assessment of software system is always a difficult problem, especially when we trying to claim high reliability and quantify it. Simply treating the fault-tolerant system as a black box and observe its failure behaviour on test is not feasible in the case that very high reliability required to be claimed (Littlewood and Strigini 1993; Butler and Finelli 1993). And this approach ignored the diversity feature which intuitively ought to be useful in the assessment activity. So, will the assessment task be easier for a software system with the diversity knowledge presented. Or could we design a diverse software system for which the purpose of better assessment is considered at the beginning. In this section, literatures on the assessment of the reliability of diverse software system are viewed.

2.4.1. The Popov and Strigini model

Even though the EL and LM models (section 2.2.2) provided useful insights into the reliability of diverse systems, they have obvious limitations. The difficulty function of each demand could never be known in practice, so they essentially described what happened on *average* reliability of versions and pair of versions. Their goal is to help our understanding of the problems instead of evaluating the reliability of a *specific* diverse system. But, in practice, we wish to know the distribution (or a bound with confidence) of the *pfd* of the *particular* system of interest. Popov and Strigini therefore extended the EL and LM models with some extra assumptions (Popov and Strigini 1998).

At the early stage of this Popov and Strigini model (PS model), the unrealistic assumption is that for each demand x, we know whether it would cause failure or not for both channel A and B, being described by two binary function $w_A(x)$ and $w_B(x)$. For example, $w_A(x)=1$ means that the demand x will certainly cause channel A fails. If we denoted $Q(x)$ is the probability that demand x will be the next one (depends on operation profile), then the probabilities of failure of versions A and B on a randomly selected demand X are:

$$P_A = \sum_{x \in D} P(X = x) P(\text{A fails}|\text{ demand } x) = \sum_{x \in D} Q(x) w_A(x) \qquad (2.6)$$

$$P_B = \sum_{x \in D} P(X = x) P(\text{B fails}|\text{ demand } x) = \sum_{x \in D} Q(x) w_B(x) \qquad (2.7)$$

For a specific demand x, the probability of both A and B fail is either 0 or 1(i.e. $w_A(x) \times w_B(x)$), then:

$$P_{AB} = \sum_{x \in D} P(X = x) P(\text{A and B both fail } | \text{ demand } x)$$

$$= \sum_{x \in D} Q(x) w_A(x) w_B(x) = P_A P_B + cov(\Omega_A, \Omega_B) \qquad (2.8)$$

where the random variables Ω_A and Ω_B are defined as the values taken by w_A and w_B on a randomly chosen demand.

The result (2.8) is similar to the result (2.5) of the LM model. The fundamental of LM model is the difficulty functions for two different development methodologies, which can take any value between 0 and 1 (representing the probability that a randomly chosen version, developed with that methodology, would fail on a given demand). However here, as the PS model is describing two known versions, the functions $w_A(x)$ and $w_B(x)$ can only take either 0 or 1, and the only uncertainty concerns the choice of the next demand x, described by the probability distribution $Q(x)$.

Of course, it is unrealistic to know the score functions $w_A(x)$ and $w_B(x)$ for any specific demand x and versions A and B. But some interesting properties of a particular diverse software system could be concluded:

- Diversity design would reduce the "tail" of the *pfd* distribution. That is, essentially, reducing the risk of unacceptably unreliable system.
- The efficacy of diversity varied greatly with the probabilities of individual faults – more than with their severities.

As very detailed knowledge about each demand for each version is required, it would be very hard to be attained the two functions $w_A(x)$ and $w_B(x)$ in reality. However, subdividing demands into classes (normally named as subdomains) is common practice for software engineers. This could be done in terms of modes of operation of a system or parts of the code being executed etc.

With this more practical innovation idea, Popov and Strigini divided the demand space into subdomains and extended the earlier PS model in (Popov, Strigini et al. 2003). Given the *knowledge on subdomains*, the new PS model were able to give some results on the upper and lower bounds of the system *pfd*. The practical use of the upper and lower bounds was also discussed. Their bounds are only useful if they are substantially "narrower" than the ones obtained by other means.

Even though the extended PS model did improve the ability of assessors and regulators to assess a *particular* diverse-channel system using information specific to that system, two main questions were still remained: first, useful *knowledge on subdomains* seems still difficult to get due to the fact that we usually deal with ultra-reliable versions. And the second is the gap of the differences between the demand – profiles under which the previous statistics data were collected and the one in the intended operational environment.

2.4.2. Bayesian inference for the reliability of diverse software

In (Littlewood, Popov et al. 2000b), Bayesian interference was first used to estimate the reliability of fault-tolerant software systems, and in particular, treating the system as a "white box". The intuitive idea here is that knowing the system is fault-tolerant will give us more confidence about its reliability than simply treat it as a “black” box. Thus, we could break through the limitation of the assessment approach of direct testing.

In the model, observation of not only the number of common failures but also the number of failures of each channel needs to be recorded. Use these 3 numbers of different failures as evidence, the prior– a joint distribution of pfd_A, pfd_B and pfd_{sys} – would be learned into a posterior 3 dimensional distribution.

Just like any Bayesian inference, the problem remains is giving a justifiable prior distribution. And in this particular case, this problem turns out to be surprisingly difficult. As this is a multivariate inference, which makes the difficulty of justifying the prior becomes even harder (Strigini 1994).

While in some circumstance, there is large amount of evidence from previous use that strong prior distributions could be justified, such as off-the-shelf software (OTS) and legacy software, the Bayesian approach to reliability assessment is practically applicable and thus valuable. A number of papers advocated a model of software of complex structure built with off-the-shelf software were proposed, e.g. (Kuball, May et al. 1999) which is under the assumption that the OTS components will fail independently, and Kuball asserted that under that assumption the predictions about system reliability are guaranteed to be pessimistic. Popov scrutinised their results and contrary to the assertion by demonstrating a counterexample of priors which leads to optimistic predications (Popov 2002). Later a new way of applying Bayesian assessment to legacy safety-critical systems upgraded with fault-tolerant off-the-shelf software was presented in (Popov 2013). A mechanism of “recalibration of coarse models” was used to deal with the problem of justifying the multivariate prior distribution.

2.4.3. Reliability of 1oo2 systems in which one channel is possibly perfect

In recent years, a new research line started on reasoning the reliability of divierse1oo2 systems in which one channel is “possibly perfect”. The big advantage of the new theory is that the need to estimate the dependence between diverse channels is avoided, *with the assumption that probability of perfection and probability of failure are independent.* Even though there is no empirical evidence or theory to justify the assumption to be universally true, we believe it is very plausible with some aid of random selection models (see later for detailed explanation). Here two fundamental papers (Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a) are viewed.

The Littlewood and Rushby Model (LR model)

In practice, there is one kind of 1oo2 diverse channel systems that one channel is simple enough to have some chance to be perfect. As the general idea showed in (Littlewood and Rushby 2012), extensive and desirable functionality is provided by the primary channel A which is conventionally engineered and presumed to contain faults; while the channel B has very restricted functionality, and thus is much simpler. Because of this simplicity, it might have been possible to claim that the channel B is perfect.

The perfection of a software-based system means that a “perfect” system will never fail no matter how much operational exposure it receives. If we assume that failures of a software system can occur if and only if it contains faults, then it means that the system is “fault-free”. Of course, no one can be certain that a system is perfect in the sense of the definition. But they may be prepared to accept that such perfection is *possible*.

There are two kinds of uncertainty involved in reasoning about the reliability of a 1oo2 diverse system. The first is aleatory uncertainty, or “uncertainty in the world,” the second is epistemic uncertainty, or “uncertainty about the world”. For more information about these two uncertainties, see (Oberkampf and Helton 2004). The LR model is that, at the aleatory level, it can be shown that there is conditional independence between the events “A fails on a randomly selected demand” and “B is not perfect,” given that the probabilities of these events, respectively pfd_A and pnp_B, are known. With that conditional independence, the LR model told us:

$$\begin{aligned} &\mathrm{P}(\text{system fails on randomly selected demand} | pfd_A = P_A, pnp_B = P_B) \\ &\qquad \le P_A \times P_B \end{aligned} \tag{2.9}$$

So with this LR result, when pfd_A and pnp_B are known, they are sufficient for computing an upper bound on the value of pfd_{sys}. It is believed to be a significant and useful result, because:

- We do not need to know about the dependence between the channel failures here, i.e. it avoided the long-standing problem (1.1) when assessing the reliability of software-based systems.

- It is useful because it provides a conservative numerical upper bound for the system *pfd* which is simply the product of two hopefully small numbers, which thus could be a very small number. In the case that pnp_B is not small (i.e. close to 1), the result is not practically useful due to close to the marginal reliability pfd_A. So it is worth mentioning that this new approach is not applicable to systems where perfection is less likely.

The Littlewood and Povyakalo Model (LP model)

In line with the LR model, the Littlewood and Povyakalo (LP) model (Littlewood and Povyakalo 2013a) was proposed to address the epistemic uncertainty of the two parameters – pfd_A and pnp_B. In reality, none of the values of pfd_A and pnp_B will be known with certainty, as a result of the imperfect knowledge of the assessors. This is where epistemic uncertainty comes in. Ideally, we plug in a two dimensional joint distribution of pfd_A and pnp_B into the LR model, but it is impractical that an assessor could express his whole joint belief about them. Rather, assessors are more likely to express partial information (e.g. a percentile) about the marginal distribution on pfd_A and pnp_B (such as the (2.10) and (2.11)). Therefore the aim of the LP model is to avoid the need to estimate the *epistemic dependence* between two parameters and requires only *partial* beliefs on the marginal distributions.

There are several theorems in the LP model, and each of them allows us to get a conservative expectation or confidence bound on the system *pfd*. For instance the theorem 1 in the LP model, if we know,

$$P(pfd_A < P_A) = 1 - \alpha_A \tag{2.10}$$

$$P(pnp_B < P_B) = 1 - \alpha_B \tag{2.11}$$

then we could know the mean system *pfd* is:

$$E(pfd_{sys}) = P_A \times P_B \times (1 - \alpha_B) + P_A \times \alpha_B + (1 - P_A) \times \alpha_A \tag{2.12}$$

As you can see from (2.12), the LP result solely relies upon assessors' marginal beliefs about the individual channel parameters – pfd_A and pnp_B – and do not require epistemic dependence between them to be estimated. There is a price paid, not surprisingly, for this simplification: further conservatism is introduced into the claims that can be made about the system *pfd*.

By now, the LR and LP models have reduced the problem of assessing the *pfd* of this kind of special 1oo2 system to one concerning simply marginal beliefs about the parameters pfd_A and pnp_B. There is a large literature on the assessment of *pfd* from statistical analysis of operational tests (Littlewood and Wright 1997; Musa and Ackerman 1989; Miller, Morell et al. 1992), so the first of these parameters could be relatively easier assessed, e.g. in terms of a Bayesian posterior distribution. That leaves pnp_B, which initially became the subject of my PhD research.

3. TWO DIFFERENT PROBABILITIES OF PERFECTION

It has been realised in the literature review that there are essentially two different probabilities of perfection, i.e. a subjective confidence in the perfection of a single program of interest (e.g. the first example in section 1.1.4) and an objective parameter of a population of programs (e.g. the second example in section 1.1.4). Firstly in this chapter, detailed explanation of these two different probabilities of perfection is presented. Then to complement the use of the subjective probability of perfection in reasoning the reliability of diverse 1oo2 systems, the original LR and LP models (Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a) are modified. Finally the advantages and disadvantages of them are discussed and recommendations are given on the practical ways to use them.

3.1. The two different probabilities of perfection

In section 1.1.4, the first example of using probability of perfection is about the *confidence* in perfection of a software system, i.e. a probability of an *objective event* that whether the software is perfect. While in the second example, probability of (not) perfection becomes an *objective parameter* of a population of programs – pnp_B (i.e. something existing in the world), and therefore assessors may have epistemic uncertainties about it. Essentially they are two different probabilities of perfection, which have not been explicitly distinguished and analysed in previous publications.

Note that this distinction does not concern the philosophical discussions on subjective probability vs objective probability. Rather, it is believed that they represent two paralleled sets of models which are built upon different fundamental setups. From the perspective of practical use and future research, it is necessary to explicitly distinguish them.

3.1.1. Probability of perfection as an objective parameter

The probability of (not) perfect as an objective parameter characterising a population property was first used in the LR and LP serial models (Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a) to reason conservatively about the reliability of diverse 1oo2 systems.

The useful classical interpretation of probability of perfection as an objective parameter can be obtained from the notion of software development as the random selection of a program from a population of programs, as first introduced in (Eckhardt and Lee 1985). The idea here is that, for a particular problem that a program is to solve, and a particular development process, there is a hypothetical population of all programs that could be written to solve the problem using the process, and a distribution over this population that determines the probability of selection for each member of the population. In this rather abstract conceptual model of software development, *pnp* is a property of a hypothetical *population* of programs: each program in this population will be either perfect, or not, for the problem being solved. The act of selecting a single program from the population – i.e. developing a program to solve the problem – results in a program that is either perfect or not. The parameter *pnp* is just the probability that a randomly selected program is not perfect. As the smaller *pnp* is, the better the developing process is, this objective parameter *pnp* is essentially an indicator of how good the development process is for the given problem.

Even though it is a property of a *hypothetical* population, such *pnp* parameter is believed to objectively exist “in the world” for each development process and problem. And of course no one would certainly know its value, so any confidence or doubt (expressed in probabilistic form) on its value is rational (i.e. captures the epistemic uncertainty). For instance, the confidence bound on pnp_B in the formula (2.11), which is mathematically a "probability of the probability of perfection". Note that this "probability of the probability of perfection" does not make sense in the case of probability of perfection as a subjective confidence, which will be discussed in next section.

3.1.2. Probability of perfection as a subjective confidence

Clearly, a single program is either perfect or not. There will be uncertainty about whether it is perfect. This *aleatory* uncertainty of the perfection of the single software could be captured in a single indicator parameter, *isimperfect*.

When *isimperfect* = 1, it means the software is imperfect, otherwise *isimperfect* = 0 means the software is perfect. If we expressed our confidence (i.e. the epistemic uncertainty) of the value of this indicator parameter in a probabilistic form, then *Prob*(*isimperfect = 0*) is the subjective probability of perfection.

Same as the objective probability of perfection concerning a population of programs, this subjective probability of perfection is also of interest in practical software dependability assessment activities. In section 1.1.4, it has been illustrated that the subjective probability of perfection of a specific piece of software could be used as its lifetime reliability claim, thus avoiding the difficulty of assessing ultra-high reliability. Besides, the subjective probability of perfection could also be practically useful in the reliability reasoning of a particular software system. Such as in (Strigini and Povyakalo 2013; Bishop, Bloomfield et al. 2011), Bayesian frameworks combining evidence of failure-free runs were proposed to claim the reliability of the software of interest. In (Rushby, Littlewood et al. 2014), a new theory of software *certification* was introduced that proceeds from the assessment of confidence in perfection to conservative prediction of reliability.

However, there is a fundamental difference from the objective probability of perfection case concerning a population of programs. It now does not make sense to talk about "confidence" in the probability of perfection which itself is simply a number representing the subjective belief of an assessor about an event, because we believe there should not be “confidence in confidence in an event”.

In the next section, the original LR and LP models are rewritten as new theories (named as LR-LP-single models) to reason about the reliability of diverse 1oo2 systems dealing with the case concerning the perfection of a specific B channel. This extension from LR-LP model to LR-LP-single model is similar to the extension from LM model (Littlewood and Miller 1989) to PS model (Popov and Strigini 1998), which is shifting the interest from an “average case” to a specific system.

3.2. Reasoning about the reliability of a specific 1oo2 system

3.2.1. New version of the LR model[5] *at aleatory level*

There are two *objective* parameters “in the world”:

- pfd_A for the channel A, and its range is [0,1].
- $isimperfect_B$ for the channel B. It could be either 0 or 1. $isimperfect_B = 1$ means the channel B is not perfect, otherwise is perfect.

As for the original LR mode, the interest here centres upon the system reliability expressed as a *pfd* of the system, then:

Theorem 0

$$Pr(\text{sys fails [on a random demand]}|pfd_A = P_A, isimperfect_B = I_B) \\ \leq P_A \times I_B \tag{3.1}$$

where I_B is an indicator variable.

Proof:

[5] Since it is for the single program of interest, “LR-single model” will be used as the abbreviation of this model later in the thesis.

$$\begin{aligned}
&Pr(\text{sys fails [on a random demand]}|pfd_A = P_A, isimperfect_B = I_B)\\
&= Pr(\text{sys fails}|A \text{ fails, B imperfect}, pfd_A = P_A, isimperfect_B = I_B)\\
&\times Pr(A \text{ fails, B imperfect}|pfd_A = P_A, isimperfect_B = I_B)\\
&+ Pr(\text{sys fails}|A \text{ succeeds, B imperfect}, pfd_A = P_A, isimperfect_B = I_B)\\
&\times Pr(A \text{ succeeds, B imperfect}|pfd_A = P_A, isimperfect_B = I_B)\\
&+ Pr(\text{sys fails}|A \text{ fails, B perfect}, pfd_A = P_A, isimperfect_B = I_B)\\
&\times Pr(A \text{ fails, B perfect}|pfd_A = P_A, isimperfect_B = I_B)\\
&+ Pr(\text{sys succeeds}|A \text{ succeeds, B perfect}, pfd_A = P_A, isimperfect_B = I_B)\\
&\times Pr(A \text{ succeeds, B perfect}|pfd_A = P_A, isimperfect_B = I_B)
\end{aligned} \tag{3.2}$$

The last three terms of the right hand side of (3.2) are 0, as the system does not fail if either A succeeds or B is perfect. While in the first term, the first factor could be *conservatively* replaced by 1, assuming if B is not perfect then it will fail with certainty whenever A fails. That is B brings no benefit when it is not perfect. Then:

$$\begin{aligned}
&Pr(\text{sys fails [on a random demand]}|pfd_A = P_A, isimperfect_B = I_B)\\
&\leq Pr(\text{A fails, B imperfect} \mid pfd_A = P_A, isimperfect_B = I_B)\\
&= Pr(\text{A fails}|\text{ B imperfect}, pfd_A = P_A, isimperfect_B = I_B)\\
&\times Pr(\text{B imperfect}|\ pfd_A = P_A, isimperfect_B = I_B)\\
&= P_A \times I_B
\end{aligned} \tag{3.3}$$

Here, a similar assumption as that in the original LR model is needed, i.e. the two events "A fails" and "B is imperfect" are *conditionally independent*, given the certain value P_A for pfd_A and I_B for $isimperfect_B$. The intuitive justification of the assumption is that whether or not B is imperfect tells us nothing about whether or not A will fail on a random demand.
QED

Result (3.3) is nicely paralleled to the original LR result (2.9). Essentially the aleatory uncertainty *pnp_B* of a population of programs in the original LR model collapsed into the aleatory uncertainty *$isimperfect_B$* of either perfect or not of a single program (i.e. assuming there was only one version in the population). So the work here is not simply a different interpretation of probabilities, rather a fundamentally different model built upon different objective parameters (capturing different aleatory uncertainties).

Ideally, an assessor would describe his epistemic uncertainty about these unknowns – pfd_A and $isimperfect_B$ – in terms of a complete bivariate distribution:

$$F_{pfd_A,isimperfect_B}(P_A, I_B) = Pr(pfd_A \leq P_A, isimperfect_B = I_B) \tag{3.4}$$

The *unconditional* probability of system failure is then:

$$\begin{aligned} &Pr(\text{sys fails [on a randomly selected demand]})^{6} \\ &= E_{pfd_A,isimperfect_B}\big(Pr(\text{sys fails } | pfd_A = P_A, isimperfect_B = I_B)\big) \\ &\leq E_{pfd_A,isimperfect_B}(pfd_A \times isimperfect_B) \\ &= \iint (P_A \times I_B) dF_{pfd_A,isimperfect_B}(P_A, I_B) \end{aligned} \tag{3.5}$$

In reality, it is unlikely that a real-world assessor would be willing or able to offer such a complete bivariate distribution to represent his beliefs about the unknowns of the model. In particular, it is known that people find it hard to express the dependence between their beliefs. In next section, some results based on only partial and marginal beliefs are obtained, and they parallel the earlier results in the original LP model.

3.2.2. New version of the LP model[7] at epistemic level

Conservative bounds on system *pfd* can be obtained.

Theorem 1

If the assessor could tell us:

$$Pr(pfd_A < P_A) = 1 - \alpha_A \tag{3.6}$$

$$Pr(isimperfect_B = 1) = 1 - \theta \tag{3.7}$$

then:

$$Pr(\text{sys fails}) \leq (1 - \theta)P_A + (1 - P_A) \times min\{\alpha_A, 1 - \theta\} \tag{3.8}$$

See Appendix A for proof.

Example 1

If the assessor is 95% confident that pfd_A is smaller than 10^{-5}, and 99% confident in the perfection of channel B, (i.e. $P_A = 10^{-5}, \alpha_A = 0.05$, $\theta = 0.99$) , then via (3.8):

[6] For simplicity, the notation $Pr(\text{sys fails})$ is used to represent this probability later in the thesis.

[7] Since it is for a single program of interest, “LP-single model” will be used as abbreviation later in the thesis.

$$Pr(\text{sys fails}) \leq (1-\theta)P_A + (1-P_A)\,min\{\alpha_A, 1-\theta\} = 0.01 \times 10^{-5} + (1-10^{-5}) \times 0.01$$
$$= 0.01$$

which is a very conservative result.

It is not hard to see that the result is dominated by the smaller one between two doubts – i.e. the doubt on P_A as pfd_A and the doubt on the perfection of channel B. However, human cannot express very small doubt which is considerably smaller than desired "reliability claims" (e.g. a 10^{-5} *pfd*). For instance, an assessor would not say he has a 10^{-5} doubt in something. So this theorem 1 is not practically helpful in this sense, but it does inspire us to think about what the minimum *helpful* partial and marginal beliefs are.

Theorem 2

Additionally to (3.6) and (3.7), if the assessor were able to tell a certain upper bound on the pfd_A:

$$Pr(pfd_A < P_A^U) = 1 \qquad (3.9)$$

then:

$$Pr(\text{sys fails}) \leq (1-\theta)P_A + (P_A^U - P_A)min\{\alpha_A, 1-\theta\} \qquad (3.10)$$

See Appendix A for proof.

Example 2

Same as the informal scenario of Example 1, but additionally with $P_A^U = 10^{-3}$, meaning the assessor is certain the *pfd* of channel A will be better than 10^{-3}, then via (3.10):

$$Pr(\text{sys fails}) \leq (1-\theta)P_A + (P_A^U - P_A) \times \min\{\alpha_A, 1-\theta\}$$
$$= 0.01 \times 10^{-5} + (10^{-3} - 10^{-5}) \times 0.01 = 10^{-5}$$

which is a much better result than the one in Example 1.

If drop the probability of perfection (i.e. θ) to 90%, we would have:

$$Pr(\text{sys fails}) \leq (1-\theta)P_A + (P_A^U - P_A) \times \min\{\alpha_A, 1-\theta\}$$
$$= 0.1 \times 10^{-5} + (10^{-3} - 10^{-5}) \times 0.05 = 5.05 \times 10^{-5}$$

which is still a useful result and it seems that θ is a crucial parameter.

Actually the result is dominated by the product of $P_A^U \times \min\{\alpha_A, 1-\theta\}$. Result (3.10) is potentially useful as it is basically a product of two smaller numbers in which one is a

"reliability claims" (i.e. the P_A^U which is essentially small and close to the desired reliability claims), and the other is a doubt (i.e. α_A or $1-\theta$) which could be afterwards learned (i.e. be further decreased) due to the subjective nature. For instance, do a Bayesian inference to update the doubt α_A when seeing some V&V evidence of channel A; and as well for the confidence θ of the channel B when seeing V&V evidence of it.
As the objective parameter *isimperfect*$_B$ is either 0 or 1, its marginal distribution is a 2-point one. It seems the assessor cannot vary his partial beliefs on this 2-point distribution, but simply give a probability, i.e. the (3.7). However, for the marginal distribution of *pfd*$_A$, the assessor may express his partial beliefs in various forms. Now, if he knew the first two *moments* of his marginal distributions of *pfd*$_A$, i.e. the mean and variance.

Theorem 3

$$\begin{aligned} Pr(\text{sys fails}) &\le E(pfd_A \times isimperfect_B) \\ &\le \sqrt{(1-\theta)\left(E(pfd_A)^2 + Var\,(pfd_A)\right)} \end{aligned} \tag{3.11}$$

See Appendix A for proof.

Example 3

If the assessor is 99% confident in the perfection of channel B (i.e. $\theta = 0.99$), and the expectation of *pfd*$_A$ is 10^{-4}. Besides, he also knows: $Var\,(pfd_A) \le 3E(pfd_A)^2$, then via (3.11):

$$Pr(\text{sys fails}) < \sqrt{(1-\theta)\left(E(pfd_A)^2 + Var\,(pfd_A)\right)} \le \sqrt{(1-0.99)(10^{-8} + 3\times 10^{-8})}$$
$$= 0.2 \times 10^{-4}$$

Again, this seems a useful result which is a product of two small numbers. One is associated with the doubt (i.e. $\sqrt{(1-\theta)}$) which could be afterwards updated when seeing more good evidence about channel B. The other one is associated with reliability claim i.e. $\sqrt{E(pfd_A)^2 + Var\,(pfd_A)}$. The first term is the expected value of *pfd*$_A$ which could be learned when seeing more evidence of channel A. While the second term is a about how spread out of the subjective distribution of *pfd*$_A$ around the mean, which seems not easy to be elicited from the assessors.

However, if the assessor was able to tell a certain upper bound on *pfd$_A$* (as the example 2), we do not need to elicit the second moment (i.e. the variance) of *pfd$_A$*, as we can calculate a conservative variance from the mean and the certain upper bound:

Theorem 4

For channel A, if the assessor could tell us a certain upper bound P_A^U of *pfd$_A$*, i.e. (3.9) and a mean, say $E(pfd_A) = M_A$; for channel B, a θ confidence in its perfection, i.e. (3.7), then:

$$Pr(\text{sys fails}) \leq E(pfd_A \times isimperfect_B) \leq \sqrt{(1-\theta) \times M_A \times P_A^U} \tag{3.12}$$

See Appendix A for proof.

Example 4

Same as the figures used in example 2 and 3, assume the assessor gives a certain upper bound $P_A^U = 10^{-3}$ and mean of *pfd$_A$* $M_A = 10^{-4}$ for channel A; and still 99% confident in the perfection of channel B (i.e. $\theta = 0.99$), then via theorem 4:

$$Pr(\text{sys fails}) \leq \sqrt{(1-\theta) \times M_A \times P_A^U} = \sqrt{10^{-2} \times 10^{-3} \times 10^{-4}} \approx 0.316 \times 10^{-4}$$

This seems a useful result. First, it is relatively easier to elicit numbers from the assessor, comparing to a variance in theorem 3. Second, the result (3.12) is essentially dominated by a product of 3 small numbers (one doubt and two reliability claims), in which θ and M_A could be individually updated when seeing good evidence from the two channels. And both of the individual afterwards learning of the two channels will make a contribution to the final 1oo2 system reliability.

3.2.3. Comparing with the "black-box" treatment

When we assess the reliability of a *specific* diverse software-based system, a more straight forward method is to treat it as a "black-box" and then collect the system V&V evidence to assess its reliability. So what benefit does the "clear-box knowledge" (i.e. knowing the system is a diverse 1oo2 one in which one channel is possibly perfect) bring us? And does this LR-LP-single approach use the "clear-box knowledge" in an effective way?

For the first question, if without the "clear-box" knowledge, it seems the assessor would only be able to express very modest partial prior beliefs. For instance, if without the knowledge that channel B is very simple, the assessor would never claim the possible perfection of a complex "black-box" system. Thus the reliability inference of the "black-box" system from available evidence will be conducted on very modest partial priors (e.g. only some percentiles of *pfd*), which leads to much worse results than the case with an *additional* prior belief in perfection (readers may find detailed numerical examples of the two cases in (Bishop, Bloomfield et al. 2011)). Similar observations could be generalised to other types of partial beliefs, e.g. expected system *pfd*. So at least, the "clear-box knowledge" does bring us benefit in terms of eliciting better partial priors.

However, the LR-LP-single models are clearly not the only approach to use the "clear-box knowledge". In Figure 2, the RHS route – a so called "whole-system-view" approach – is an alternative way to reason the reliability of a particular system based on its "clear-box knowledge". Its basic idea is to firstly get some partial beliefs about the *system pfd* from the partial and marginal beliefs about the two individual channels and then do the reliability assessment starting via partial beliefs about the whole *system*. For example, we could reason as follows:

- I have θ confidence in the perfection of channel B, therefore I should have at least θ confidence that the whole system reliability $pfd_{sys} = 0$.
- If I believe in a certain *pfd* upper bound (i.e. $Pr(pfd_A < P_A^U) = 1$) on channel A, then I should believe there is a same certain upper bound on the system *pfd*, i.e. $Pr\left(pfd_{sys} < P_A^U\right) = 1$.
- Then by the two pieces of priors above, we can easily know that $(1-\theta)P_A^U$ is an upper bound on the *pfd* of the whole 1oo2 system.

The first two statements are very simple and self-evident, by applying the principle of "the whole system's reliability should be better than any individual channel's reliability". Then the third one tells that, by the two partial beliefs about the whole system *pfd*, we can get a naive result of $(1-\theta)P_A^U$ as an upper bound on the reliability of the particular 1oo2 system. Generally, the third step should be done by some models on a single system, e.g. the ones in (Bishop, Bloomfield et al. 2011; Strigini and Wright 2014). It seems this "whole-system-view" approach is giving very similar results to LR-LP-single approach, which reduces the significant of LR-LP-single models.

This disappointing observation seems true for the current **theorem 1**, **2** and **4** (section 3.2.2) in the LR-LP-single models. But for **theorem 3**, which involves a variance of the *pfd* of channel A, the alternative "whole-system-view" approach does not work. Because, we cannot say something like "as I know the variance of the reliability of channel A, so I should know … about the whole system". In other words, we cannot do the reasoning to get partial beliefs about the whole system from marginal and partial beliefs of individual channels in this case, where the LR-LP-single model therefore brings benefit.

Even though there is only one theorem (i.e. the 3rd) showing the advantages of LR-LP-single models, we believe more theorems could be built to show the advantage when we looked into *more various* (and less minimal) partial and marginal beliefs of the two channels, where the steps of the alternative "whole-system-view" approach are hard and not self-evident.

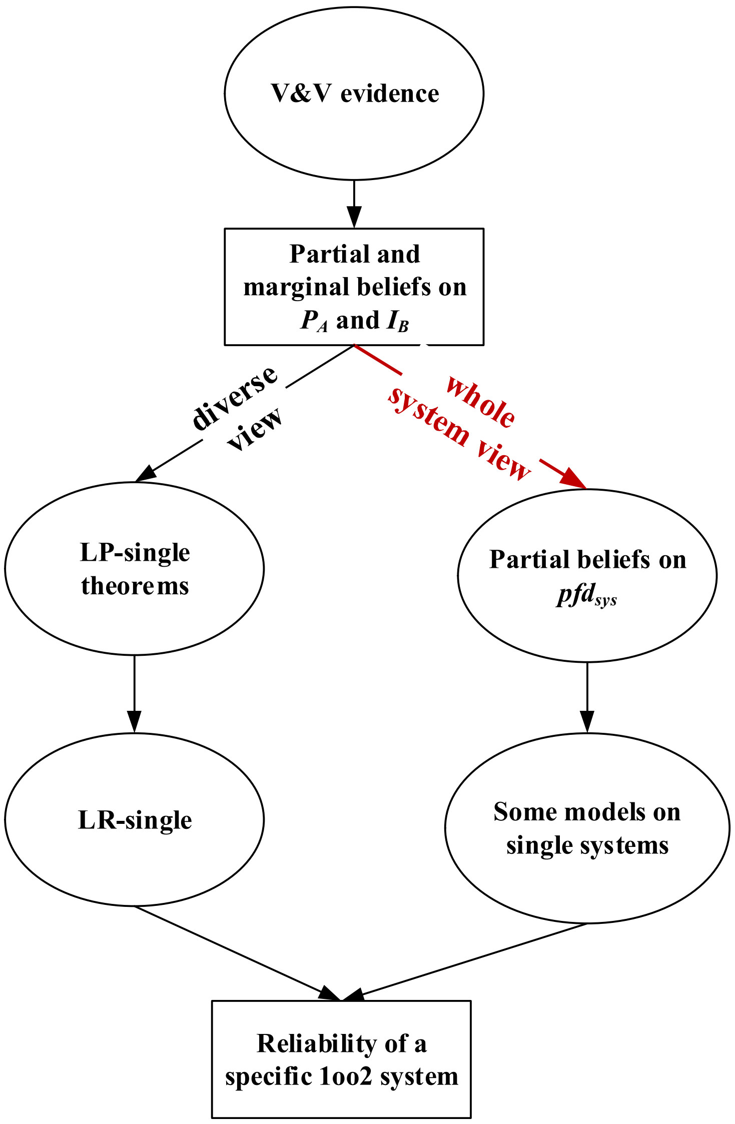
V&V evidence
Partial and marginal beliefs on P_A and I_B
diverse view
whole system view
LP-single theorems
Partial beliefs on pfd_{sys}
LR-single
Some models on single systems
Reliability of a specific 1oo2 system

Figure 2 two different approaches to use the "clear-box knowledge"

3.3. LR-LP models or LR-LR-single models?

Till now, two nicely paralleled sets of models – the original LR-LP models (in section 2.4.3) and the extended LR-LP-single models (in section 3.2) – are obtained and both of them are aiming to reason the reliability of diverse 1oo2 systems. Is one of them always superior to the other? Or more possibly, there might be no universal answer and it has to depend on the specific circumstances. If so, when should the assessor choose one over the other? This section is to answer questions above.

3.3.1. Model different aleatory uncertainties

At the aleatory level, the uncertainty being modelled for the channel A is pfd_A in both of the two cases.

- pfd_A is a true unknown property of the channel A, i.e. an *objective parameter* in the world about channel A. As a thought experiment, we could imagine executing a large number of demands, n, selected in a way that accurately represents the operational use of channel A, and allowing n to approach infinity: the proportion of failed demands would converge to the true (but unknown) pfd_A. In practice, assessors of course cannot be certain about the value of pfd_A via that infinite testing approach, hence the epistemic uncertainty.

While, for the possibly perfect channel B, two fundamentally different *aleatory* uncertainties are modelled in the two sets of models.

- In the original LR model, pnp_B is used as an *objective property* of a hypothetical population from which the B channel is selected. As the thought experiment described in section 3.1.1, the population is composed of all programs that could be written to solve the particular problem using the same development process of channel B. The pnp_B is the probability that a randomly selected program from that B population is not perfect. We can have a limiting relative frequency interpretation for it. Image we randomly select one program from the population, then do a "perfection test" (e.g. infinite testing) and mark it as perfect or imperfect based on the result. If we repeat that trial for idealized infinity times, then the pnp_B would equal to the "imperfect proportion" of the population. As the smaller pnp_B is, the better the developing process B is, this objective parameter pnp_B is essentially one of the important indicators of how good the development process is for the given problem. Please note that processes with lower pnp_B could not universally being deemed better than the ones with higher pnp_B. Other properties must be taken into account in that case.

- In the modified LR-single model, the aleatory uncertainty now is about whether perfect or not of the *specific* channel B, being captured by an indicator parameter $isimperfect_B$. Rather than an objective property of a population, $isimperfect_B$ is about the *objective event* of B's perfection. The limiting relative frequency interpretation for it is that if we do an infinite times test on channel B, then seeing any failure in the long sequence of tests would result in $isimperfect_B = 1$, otherwise 0. Even though this uncertainty containing only two possible states which is much simpler, it is still impossible to know it for certain in practice.

All the parameters mentioned above cannot be known for sure in reality, i.e. assessors have epistemic uncertainties about them. Normally, the epistemic uncertainties should not be independent for an assessor. Therefore, to use any two of the parameters, the assessor needs to express a subjective joint distribution of them. Via the original LP and extended LP-single models, assessors avoid the difficulty of expressing joint beliefs over the parameters of the two channels, rather something marginally and partially about the parameters pnp_B and $isimperfect_B$ are sufficient enough. Each model set addresses two aspects of *dependence*: at the aleatory level, and at the epistemic level, not surprisingly at the price of conservatism. It remains only to address the issues of expressing marginal and partial epistemic uncertainties.

3.3.2. Reduce different epistemic uncertainties

Aleatory uncertainty is "natural" uncertainty that is *irreducible* (Oberkampf and Helton 2004). For example, we would never be able to certainly predicate the result of tossing a coin. Even when we know the coin is "fair", prediction of tosses of the coin can only be expressed as probabilities, say P_{head} and P_{tail}, meaning this uncertainty of outcomes can never be eliminated. In the case of a "fair" coin, $P_{head}=P_{tail}=0.5$. However real coins cannot be fair, so we can never know the values of P_{head} and P_{tail} for sure, which is the epistemic uncertainty. The significant feature of epistemic uncertainty is that it is *reducible*. In that coin example, we could toss the real (thus unfair) coin very many times, and estimate P_{head} with the frequency of heads in the long sequence of tosses. And P_{head} will converge to the *true unknown* P_{head} when more tosses are done, i.e. *reducing* the epistemic uncertainty about the parameter P_{head}. Above is an example of frequentist's way to reduce epistemic uncertainty.

Similarly for a Bayesian, via the Bayes theorem, epistemic uncertainty could be reduced when collecting more evidence.

Similarly in the case of reasoning about the reliability of a diverse 1oo2 system, models have been built upon requiring only marginal and partial epidemic uncertainties about parameters pfd_A, pnp_B and $isimperfect_B$. These epistemic uncertainties of the 3 parameters could be estimated and then reduced via the various V&V evidence generated in software engineering.

However, the likelihood functions of the 3 parameters for a particular type of V&V evidence will be quite different, due to the nature that they are capturing different aleatory uncertainties. In this sense, some V&V evidence will be more "powerful" for one parameter than the others. As human beliefs could be incoherent, so when seeking *proper* evidence to reduce the epistemic uncertainty for a parameter, questions needs to be considered:

- Can a likelihood function be explicitly written down?
- What are the assumptions behind that likelihood function? Are they rational?
- Does the evidence sufficiently exist in practice?

For pfd_A which is an objective parameter for the single channel A, essentially any sufficiently presented V&V evidence of it could be used to reduce the epistemic uncertainty. But, as a reliability claim, a likelihood function could be easily and rationally written for testing evidence. Therefore, many existing models for *pfd* claims are built on statistical testing evidence, e.g. (Littlewood and Wright 1997). And it turns out the required amount of testing evidence is practical for modest *pfd* claims.

For perfection related parameters $isimperfect_B$ and pnp_B, as shown in the literature review chapter, many models built upon them could be found (Bertolino and Strigini 1998; Strigini and Povyakalo 2013; Bishop, Bloomfield et al. 2011; Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a), while no models is about how to *estimate* them. In other words, there is extensive literature about using perfection related claims in reasoning reliability rather than claiming perfection from available evidence, which initially motivates the work in later this thesis.

Despite the lack of existing work on claiming perfection, one could speculate that:

- For $isimperfect_B$, as it is a parameter of a single program, appealing evidence should include most of the "clear box" V&V evidence of the software of interest, e.g. static analysis, formal proof. Additionally as testing is an important and common evidence in reality, to what extend the failure-free testing evidence could help us to reduce the epistemic uncertainty of $isimperfect_B$ is also worth checking. In later chapters of this thesis, how these various kinds of evidence affect the claims on perfection will be modelled.
- For pnp_B: it is essentially a property of a *population*. Therefore it seems only "population evidence" would effectively reduce the epistemic uncertainty of it. Such "population evidence" could come from observing behaviour of some versions in the population. But the key to the efficiency would be the "coverage rate" of the population evidence over the population. In other words, it seems likely that seeing many versions of modest behaviour will be more supportive than seeing few versions with very good behaviour. However, this kind of population evidence is usually *luxury* in safety-critical area to reach a considerable "coverage rate" over the population. Same as the $isimperfect_B$, there is no actual work on modelling claims for pnp_B, which motivates a later chapter in this thesis to initially use the evidence of failure-free runs of similar products to reduce the epistemic uncertainty on pnp_B.

It seems the suggestions here are implying "using population evidence for population parameters" and "specific evidence for single software parameters". But population evidence is an important source to form priors for single software parameters. For example, via some population evidence, we could have a subjective distribution of the parameter pnp_B, and then the mean value of this distribution could be rationally treated as a good *estimation* of $isimperfect_B$. These ideas will be further discussed in later chapters.

3.3.3. Choosing between the two sets of models

By explicitly analysed what aleatory uncertainties have been modelled, and then what kinds of evidence could be useful to reduce the epistemic uncertainties, the decision on choosing between the two paralleled sets of models should rely on the specific contexts we are in (e.g. available evidence, the analysis object). Some recommendations can be concluded:

- In the sense of the "amount" of available evidence in practice, it seems the new modified LR-LP models are superior to the original LR-LP model, because clearly there is more considerable amount of evidence on single software than a population of software in reality. And this is especially true for safety-critical software systems. The marginal beliefs (epistemic uncertainty) on *$isimperfect_B$* could be continuously updated (reduced) when more and more evidence is collected, which is the major advantage over the original LR-LP models.

- However, in some special cases, there could be more available population evidence than specific evidence. For example, in the very *earlier* development stage of a program, the reliability *prediction* of this diverse channel system can only rely on the evidence from similar previous products. And thus the original LR-LP models are preferred.
- The choice may also depend on the potential users of the models. For instance, the policy makers would use the original LR-LP models and regulators would prefer the modified LR-LP models.
    - Policy makers normally work from a population view. When making new policies, one principle is to show new systems will no worse than currently existing systems. So there are essentially two populations – the population of existing systems and the population of "future" systems. Via the original LR-LP models, the *average* reliability of current existing systems could be calculated. Then based on that result, new policy of reliability requirements on future systems could be made, and even degraded to specific requirements on each channel via reversely use the original LR-LP model.
    - While, for a regulator, he/she usually concerns the specific software on hand, therefore the modified LR-LP models are more suitable. In this case, the regulator should pay attention to the marginal beliefs on *pfd_A* and *$isimperfect_B$*, i.e. the required inputs of the modified LR-LP models. The regulation task then becomes to justify the asserted confidence level of *pfd_A* and *$isimperfect_B$* from various presented evidence. For *pfd_A*, much work has been done to facilitate the task. While for *$isimperfect_B$*, non-existence work could be found, which motivates the work in this thesis. The final goal is to help the regulator understand how various types of evidence of the software of interest are used to build up the asserted level of confidence on *$isimperfect_B$*.

3.4. Chapter summary

The probability of perfection of software-based systems is found more and more useful in the dependability assessment activities in recent year. Although sharing the same English term, there are essentially two different probabilities of perfection out there. One is an objective parameter about a population property, and the other is a subjective confidence in the perfection of a single program (i.e. confidence in an objective event).

Both of them are of practical interest due to the fact that various models have been built on them to facilitate the dependability assessment activities. For example, via the original LR-LP models, the objective probability of perfection is used to reason the reliability of diverse 1oo2 systems. And here in section 3.2, the original LR-LP models are modified (in a nicely paralleled manner) in terms of subjective probability of perfection.

By the rewritten work in section 3.2, assessors now need to express his epistemic uncertainties on the truth of perfection of a *particular* software, rather than originally on an objective property of a population. In some cases, the former task seems of more interest and easier (due to the fact that there is more available evidence of a single program than a population of programs) for an assessor. Despite the limitation in terms of comparing with a simple "black-box" treatment (see section 3.2.3), the LR-LP-single models here is believed as a contribution to the existing reliability assessment methods of diverse 1oo2 systems.

However, there is no universal answer to the question on which set of models is better. Recommendations are given in section 3.3.3; basically showing it depends on what evidence you have and how you plan to use the result. More desirably, we would like to see a framework in which both of the two probabilities of perfection could be combined to enhance the reasoning of diverse system reliability.

A potentially feasible way is proposed here. That is to use the population evidence as a source to form priors for properties of any individual programs in the population, which seems a common sense solution of eliciting priors. In this instance, a subjective distribution of the parameter pnp_B can be obtained via some population evidence; and then the mean value of this distribution could be rationally treated as a good *estimation* of the confidence in *isimperfect*$_B$. This potentially feasible way to "stitch up" these two different probabilities of perfection will be further discussed later chapters in this thesis.

4. Claims on probability of perfection via failure-free testing evidence

In this chapter, the question on what can be claimed about probability of perfection from seeing many failure-free tests of the *single* software of interest is investigated. A probability model for this problem is built and illustrated with some numerical examples. The approach is Bayesian: the aim is to model the changes to this probability of perfection as seeing evidence of failure-free working. As the long-standing problem of any Bayesian model is to elicit a justified prior distribution, much of the chapter addresses some difficult issues of prior beliefs in the Bayesian framework: the approach is to support conservative claims for probability of perfection based on *limited* prior belief, but these should be no more conservative than is necessary.

4.1. An informal introduction to the approach

The approach to the problem is similar to that introduced in (Bishop, Bloomfield et al. 2011). However, in that work the interest centred upon the problem of obtaining conservative claims for system *pfd*; here the aim is to obtain conservative claims for probability of perfection which could be used as a lifetime claim on its own or in the reliability assessment for 1oo2 diverse system (as stated in section 1.1.4).

4.1.1. The objective parameter – pfd

The fundamental assumption in this work is that for every software-based system (or channel), there is a true unknown *pfd*, i.e. an *objective parameter* in the world. As a thought experiment, we could imagine executing a large number of demands, *n*, selected in a way that accurately represents operational use, and allowing *n* to approach infinity: the proportion of failed demands would converge to the true (but unknown) *pfd*. In practice, of course, there will only be a finite amount of evidence available, so the assessor will still be uncertain about the magnitude of the *pfd* after seeing this evidence. In the usual Bayesian terminology, this evidence will be used to update an assessor's prior beliefs about the unknown *pfd* to obtain his posterior beliefs: Bayes' Theorem modifies his uncertainty about the unknown *pfd* in the light of the evidence, but he does not arrive at certainty.

4.1.2. The difficulty of priors in Bayesian

If an assessor were able to specify a complete distribution to represent his prior beliefs about the *pfd*, it is a simple matter to use Bayes' Theorem to obtain his exact posterior distribution after he has seen the evidence. From this he could express his modified beliefs about quantities of interest such as the expected value of *pfd* (best "point" estimate), percentiles (confidence bounds for *pfd*), and so on.

A major – and often expressed – difficulty with the Bayesian approach is that assessors find it difficult, if not impossible, to express their prior uncertainty in terms of a complete probability distribution. This observation seems particularly pertinent for the kinds of software systems that are the subject of the present thesis. In contrast to some other applications of Bayesian statistics (e.g. some medical scenarios), where there is extensive previous empirical evidence that can be used to inform an assessor's prior judgments about this system, such evidence is often lacking, or very meagre, in software engineering applications. This is particularly true of safety critical applications.

Here the main purpose, then, is to show some ways that this problem of priors can be addressed when the interest centres on an assessor's confidence in the perfection of a software system. As stated earlier, the assessor cannot realistically be certain that the system is perfect; instead, he will have a prior probability of perfection, say *θ*. If use a variable *P* to represent the *pfd*, then *Pr(P=0)=θ*. Since perfection is not certain, the assessor must in addition specify his prior beliefs about the possible non-zero values of the *pfd*. Ideally, then, he would be able to express his beliefs about the system *pfd* in terms of a complete distribution (say *f(p)*)over the interval [0,1] that has probability mass at the origin: see Figure 3 for an idealized depiction of such a distribution. In the absence of such a complete prior distribution for the assessor's prior beliefs, what can be said? The approach to the problem is two-pronged.

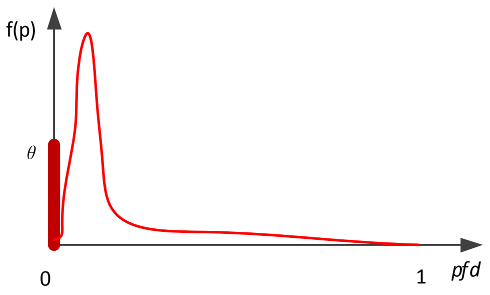

Figure 3 an idealized example of a *pfd* distribution

4.1.3. Conservative Bayesian inference with partial priors

Firstly, the reality is that an assessor may only be able to express extremely limited beliefs about the likelihood of the *pfd* taking particular values in the interval. Specifically, in a very simple case, it is plausible for the assessors to tell us a single percentile of the distribution of *pfd*, in addition to his expressed confidence in perfection. That is, he is only willing and/or able to express the following two precise beliefs:

$$Pr(P = 0) = \theta \tag{4.1}$$

$$Pr(P > y) = x \tag{4.2}$$

where he states the values $0<\theta, x, y<1$.

Of course, the limited constraints represented by (4.1) and (4.2) are far from sufficient to specify a single complete distribution. In fact there will be an infinite number of distributions satisfying these constraints (assessor expressed beliefs) for any particular vector of numbers $< x, y, \theta >$. By only expressing such limited prior belief, the assessor is implicitly accepting that none of these distributions has been ruled out as candidates to be his prior distribution for the *pfd*. The "implicitly" here means, of course, he cannot examine all these distributions to see whether some of them have characteristics that would result in his finding them unacceptable representatives of his prior beliefs.

The second part of the approach is now to choose the most conservative of these candidate distributions, that is the one (or many) that gives the most conservative (i.e. smallest) value for the quantity of interest, the posterior probability of perfection, following the observation of n failure-free demands. Denote this most conservative posterior probability as θ*.

The interpretation of this θ* probability is that it represents the lowest posterior confidence in perfection that the assessor could have, consistent with his only expressing prior beliefs, (4.1) and (4.2), and having seen n failure-free demands. This result is an attainable one, in the mathematical sense that there exists at least one prior distribution, in the assessor's infinite set of distributions that he has not ruled out via his expressed prior beliefs, that results in a posterior distribution (after seeing n failure-free demands) with mass θ* at the origin.

4.1.4. Solutions to the problem of being too conservative

Sometimes, the conservative Bayesian approach gives a result is that extremely conservative – so much so as to be of little practical interest. Therefore it might be asked whether it is too conservative for a "reasonable person" who holds beliefs (4.1) and (4.2). In other words, whether any prior distribution that gives this most conservative result would in fact be ruled out by him as representative of his beliefs if he were to examine it in detail.

The intuition behind this discussion is that an assessor may often hold unexpressed beliefs, in addition to the ones that are represented by his limited but precise expressions such as (4.1) and (4.2). One thing needs to be clear is that the reasoning here does not ask the assessor to change his prior beliefs in the face of embarrassingly conservative posterior consequences; that is, of course, unacceptable in the Bayesian framework. Rather, it is inviting the assessor to examine the infinite set of distributions initially allowed by (4.1) and (4.2), to see whether there are subsets of these distributions that he regards as unallowable (unbelievable) – and to do this before he sees the evidence from testing the *n* demands.

In this way, the new set of allowable distributions will be a subset of the original set of distributions. The assessor would then proceed as before: seeking the most conservative result from this more restricted (but still infinite) set of priors. The new conservative posterior probability of perfection obtained in this way would be expected to be less conservative than the one above obtained from the original, larger, set of allowable priors.

In some cases it may be possible to repeat this procedure by identifying other characteristics of priors that are not allowed by the assessor, thus further restricting the set of priors from which the most conservative will be selected. Indeed, given the weakness of the restrictions (4.1) and (4.2) imposed by the assessor's limited prior beliefs, it is unlikely the remaining set of distributions will all truly be allowable by the assessor. Informally, the aim here is to prune the set of allowable prior distributions to the extent that the assessor's extra expressed beliefs allow, so as to make the resulting conservative posterior beliefs less conservative. The expectation is that in this way the results will be useful, while at the same time still guaranteed to be conservative.

Of course this new conservative Bayesian approach here is not the way to tackle the problem of priors in Bayesian method. There are other alternative solutions, such as empirical Bayes, non-informative priors and etc. depends on the specific cases. We believe the new conservative Bayesian approach here is superior in the case of dealing with safety-critical systems, due to the fact that conservativeness is more desirable to some degree.

4.2. The probability model

Figure 4 shows an example of a potential prior distribution satisfying the assessor's conditions (4.1) and (4.2): it has point mass θ at the origin, and the remainder of the probability in (0,1], with probability x in the interval (y,1]. Note that the shape of the distribution in (0,1] in Figure 4 is an idealization for purposes of illustration only.

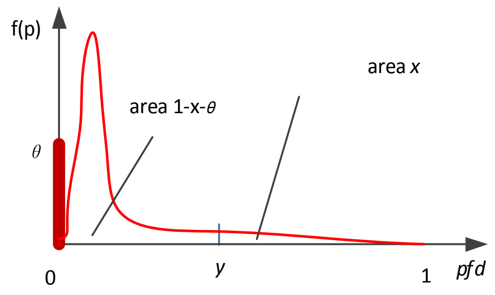

Figure 4 an idealized example of a distribution satisfying the assessor's expressed prior beliefs (4.1) and (4.2)

After seeing *n* failure-free demands, for any particular complete prior distribution *f(p)* we could calculate the resulting posterior distribution. Our interest centres on how the observation of failure-free working changes the assessor's belief in perfection. This posterior belief is:

$$\begin{aligned} Pr(P=0|\text{n failure free tests}) &= \frac{Pr(P=0 \text{ and n failure free tests})}{Pr(\text{n failure free tests})} \\ &= \frac{\theta}{\theta+\int_{0+}^{1}(1-p)^n f(p)dp} \end{aligned} \tag{4.3}$$

The problem now, as outlined in section 4.1.3, is to find the most conservative *f(p)*, i.e. the one that minimizes (4.3) subject to the constraints (4.1) and (4.2). The value of (4.3) at the minimum called θ*: this is the most pessimistic posterior belief in perfection consistent with the assessor's expressed minimal prior beliefs.

It turns out that the most pessimistic *f(p)* is a 3-point distribution (see the Appendix B for proof). As shown in the Figure 5, the distribution has probability mass at three points: at the origin with mass θ; at the P_1 point which is infinitesimally distant[8] from the origin with mass $1-x-\theta$; and at the P_2 point, at y, with mass x. With this worst case prior *f(p)*, the lower bound of θ, i.e. θ*, is obtained: see the result (4.4) below. This is the most conservative belief of the assessor about the probability of perfection of this system, after seeing n failure free tests, given his professed prior beliefs (4.1) and (4.2).

$$\begin{aligned} \theta^* = Pr(P=0|\text{n failure free tests}) &= \frac{\theta}{\theta+\int_{0+}^{1}(1-p)^n f(p)dp} \\ &= \frac{\theta}{1-x+(1-y)^n x} \end{aligned} \tag{4.4}$$

[8] To explain more carefully, the worst case posterior belief (4.4) is, under these constraints, approached as a *limit*, without being attained exactly by any individual prior *pfd* distribution. So to argue with full mathematical rigour entails an *infinite* set of distinct prior *pfd* distributions, each with its own $P_1>0$; with this set so constructed that zero is the infimum of all its member distributions' positive P_1s. Figure 5 is intended to depict this infimum property of this set of distributions. The infimum of the resulting set of Bayesian posterior perfection probabilities – each computed from one member of this set of priors – is easily show to be the RHS of equation (4.4).

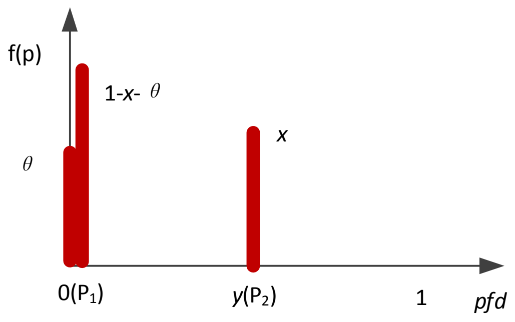

Figure 5 the most conservative prior distribution satisfying the assessor's expressed prior beliefs.

4.3. Illustrative numerical examples

Table I shows some numerical examples, where θ is the priori belief about perfection, x and y satisfy the assessor's prior belief (4.2), n is the number of failure-free tests that have been observed, θ^* is the posterior belief in perfection.

The final column of the Table I shows the factor by which the posterior beliefs improve on the prior ones. The proportional *reduction* in the *doubt* about perfection is chosen, since this seems better to reflect intuition here for the values of θ close 1 that engineers would anticipate in the critical applications that are our concern. So the ratio $(1-\theta)/(1-\theta^*)$ is the "doubt reduction factor". Clearly large values of this factor are preferred.

In each case there is some increase in the assessor's confidence in perfection as the number of failure free tests increases. However, this increase in confidence is very modest in all cases: the evidence from failure-free testing seems to be generally very weak in supporting claims about probability of perfection in this worst case.

Notice that the best results in support of perfection in Table I arise when θ and x are larger, i.e. $1-x-\theta$ is small. Now this is the probability mass arbitrarily close to 0 for the most conservative prior distribution satisfying the assessor's stated prior beliefs. The role this probability mass plays is best understood by seeing what happens when the number of failure-free demands, n, approaches infinity. The conservative bound on probability of perfection is then:

$$\lim_{n\to+\infty}\frac{\theta}{1-x+(1-y)^n x}=\frac{\theta}{1-x}=\frac{\theta}{\theta+(1-x-\theta)} \qquad (4.5)$$

Table I Numerical examples of the 3-point prior distribution

θ	x	y	n	θ^*	$(1-\theta)/(1-\theta^*)$
0.5	0.01	0.001	1000	0.503181641	1.006404032
0.5	0.01	0.001	10000	0.505050275	1.010203611
0.5	0.01	0.001	100000	0.505050505	1.010204082
0.5	0.05	0.001	1000	0.516323692	1.033749207
0.5	0.05	0.001	10000	0.526314538	1.055552767
0.5	0.05	0.001	100000	0.526315789	1.055555556
0.9	0.01	0.001	1000	0.905726953	1.060748572
0.9	0.01	0.001	10000	0.909090494	1.099994981
0.9	0.01	0.001	100000	0.909090909	1.10
0.9	0.05	0.001	1000	0.929382645	1.41608249
0.9	0.05	0.001	10000	0.947366169	1.899918692
0.9	0.05	0.001	100000	0.947368421	1.90

So even for an infinite number of failure-free demands, the assessor does not become certain of perfection. Informally, when we see an infinite number of failure-free demands, we should become certain that *pfd* cannot be greater than or equal to y; so the only possibilities remaining are perfection and $0<pfd<y$. In the case of the second of these, an infinite number of failure-free demands suggest that the probability mass in this interval will be concentrated at the extreme left of the interval. So, the infinite numbers of demands are failure-free because either (a) the program is perfect, or (b) it is not perfect but has an infinitesimally small failure rate. The probability that what has been seen is due to (a), rather than (b), is just the ratio of *Pr(a)* to *Pr(a or b)*, i.e. (4.5) above.

Informally, there is a limit to how much confidence in perfection can be obtained from failure-free demands, as we cannot tell whether this happened because of perfection or very high reliability.

If assessors were able to rule out the very high reliability *a priori*, this picture would change. Table II shows what happens if $1-x-\theta=0$, i.e. the expert is certain that the *pfd* is either 0 or greater than or equal to y. Here, relatively modest numbers of failure-free demands result in high confidence in perfection. These results are intuitively obvious because, for any value of *y*, the more failure-free working, the more confident we shall be that the system is "perfect" rather than "has a *pfd* worse than *y*". However, obtaining these stronger claims for perfection requires the assessor to rule out completely the possibility of the *pfd* lying anywhere between 0 and *y*. Such beliefs do not seem reasonable.

Table II Numerical examples of the case $1-x-\theta=0$ of the 3-points prior distribution

θ	x	y	n	θ^*	$(1-\theta)/(1-\theta^*)$
0.5	0.5	0.001	1000	0.7311569	1.85982084
0.5	0.5	0.001	10000	0.9999548	1106194.69
0.9	0.1	0.001	1000	0.9607485	2.547673337
0.9	0.1	0.001	10000	0.9999949	19607.84314

In the next section, what reasonable further restrictions can be placed on an assessor's prior beliefs and how these affect his posterior belief in perfection will be considered.

4.4. Refining the set of allowable prior beliefs

The results above arise from a situation in which an assessor has expressed some precise, but very limited prior beliefs, represented by (4.1) and (4.2). Because these beliefs are so very limited, they are satisfied by a large set of possible prior distributions. However, some of these distributions would be ruled out as possible priors by all "reasonable" assessors. The effect of this would be to "prune" the set of allowable priors (still satisfying (4.1) and (4.2), of course). An assessor would then proceed as above, but with this pruned – i.e. smaller – set of allowable distributions. That is, he would choose the most conservative of these to find a new posterior probability of perfection. But how should such pruning be carried out? Are there obviously unreasonable distributions in the set of priors that are allowable above?

4.4.1. How to prune? A Beta example

The conservative results represented by Table I and the limiting result (4.5) arise essentially because they are based on the most extreme priors that have probability mass at 0+ and *y*, as well as at 0 (the perfection point). The problem here – what is unreasonable "extremeness" of these priors – lies in their having positive support at *points* in (0,1].

For instance, a frequent situation is that the support for such prior beliefs comes from generic experience of the reliabilities achieved by similar development processes, in products that are comparable to the present one, e.g. in terms of application area, complexity and software engineering culture. Then, while it would seem reasonable for an assessor possibly to believe a non-zero probability of perfection – i.e. mass at 0 for his prior distribution for *pfd* – it does not seem reasonable for him to believe there is positive mass associated with any non-zero value of *pfd*. That is, statements like: "I think there is a 50% probability that the *pfd* is zero, i.e. that this program is perfect" seem reasonable; but statements like "In the event that this program is not perfect, I believe that there is a 20% chance that its *pfd* is exactly 0.1234" would *generally* not seem reasonable[9] . More precisely, it seems that assessors would usually think that the only beliefs they could reasonably hold would correspond to distributions that satisfy these conditions:

- There is no non-zero probability concentrated at *points* in (0,1], i.e. the only mass-at-a-point on the complete prior for *pfd* is that at 0, corresponding to perfection;
- There is no point in (0,1] for which the probability *density* is zero, i.e. no value is impossible.

These conditions suggest that the only distributions that assessors should normally consider as candidates to represent their prior beliefs should have probability mass only at 0, and have *continuous* density in (0,1]. Whilst imposing these conditions will eliminate many distributions from the set that simply satisfies (4.1) and (4.2), there will still remain a large set of candidate distributions: essentially any suitably renormalized continuous distribution on (0,1], together with a mass at 0.

To then analyse the implication of beliefs like those in (4.1) and (4.2), the next stage after this pruning would be to explore the implications of prior distributions in this restricted set;

[9] The word "generally" is used here, because of course there are exceptions. For example, it may be that there is a particular type of event that is associated with a known-to-be difficult (i.e. possibly fallible) operation, and these events occur at a known frequency. So-called "leap seconds", for example, have been known to cause system failures because of synchronization issues. The point here is that detailed and specific knowledge is needed for an assessor to hold such point-mass beliefs.

possibly eliciting more detailed approximate representations for, or bounds on, the distributions. Question of which representation would be most convenient in a specific case is not aimed to be addressed here. Instead, *purely for illustration*, an example in which the assessor is prepared to restrict the expression of his prior beliefs to a (suitably renormalized) Beta distribution, on (0,1], with parameters (*a,b*) is developed here. Later the reasonableness of such a restriction to this Beta parametric family will be discussed.

Table III shows the results of numerical optimization to find the most conservative prior within the new restricted set of allowable distributions. Once again, confidence in perfection increases slowly even for very large *n*. However, it is notable that in all cases in Table III both *a* and *b* are fractional, i.e. the most conservative priors in this refined set of allowable distributions are all "U-shaped" as shown in the Figure 6. In each case the probability density is infinite at both 0 and 1.

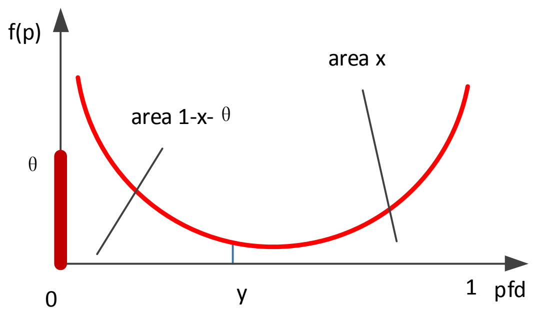

Figure 6 the most conservative prior re-normalized beta distribution

It seems that this kind of U-shaped distribution would not represent the beliefs of reasonable assessors. So it could further prune the set of allowable distributions by ruling out such U-shaped beta distributions: if *a* and *b* are not allowed to be fractional (i.e. a≥1, b≥1) that is an unimodal beta distributions[10].

[10] By "unimodal" here it includes those cases where the maximum of the density is at the end of the interval, and is not a turning point: this turns out to be the case for a=1 in Table V.

Table III Numerical examples for the Beta prior distribution with mass at origin, (*a, b*) unconstrained

θ	*x*	*y*	*n*	*a*	*b*	*θ**	(1-*θ*)/(1-*θ**)
0.5	0.01	0.001	1000	0.00019256	0.010154142	0.505055848	1.010214987
0.5	0.01	0.001	10000	0.000174069	0.009065462	0.505179928	1.010468306
0.5	0.01	0.001	100000	0.000226364	0.012069137	0.50532834	1.010771468
0.5	0.05	0.001	1000	0.001997401	0.0208142	0.526603047	1.056196068
0.5	0.05	0.001	10000	0.00058413	0.00547856	0.526730227	1.056479895
0.5	0.05	0.001	100000	0.000942005	0.009070504	0.527517591	1.058240457
0.9	0.01	0.001	1000	0.001371121	0.013629679	0.909150688	1.100723798
0.9	0.01	0.001	10000	0.000873768	0.008366982	0.909298034	1.102511938
0.9	0.01	0.001	100000	0.000771203	0.007326781	0.909423631	1.10404073
0.9	0.05	0.001	1000	0.007752825	0.008651361	0.94759435	1.908191213
0.9	0.05	0.001	10000	0.001784232	0.001829209	0.947621395	1.909176459
0.9	0.05	0.001	100000	0.002078817	0.002139551	0.947905689	1.91959541

Table IV Numerical examples for the Beta prior distribution (a≥1, b≥1) with mass at origin

θ	*x*	*y*	*n*	*a*	*b*	*θ**	(1-*θ*)/(1-*θ**)
0.5	0.01	0.001	1000	1.002865687	3919.454939	0.556728757	1.127977526
0.5	0.01	0.001	10000	1.000348556	3913.798799	0.780539772	2.278317141
0.5	0.01	0.001	100000	1.000219221	3914.753876	0.963720105	13.78173798
0.5	0.05	0.001	1000	1.001096111	2304.159264	0.589248898	1.217282187
0.5	0.05	0.001	10000	1.000879434	2302.601846	0.842539329	3.175396105
0.5	0.05	0.001	100000	1.001000815	2302.579322	0.978069486	22.79928354
0.9	0.01	0.001	1000	1.000493405	2302.371071	0.928116048	1.39113109
0.9	0.01	0.001	10000	1.000013517	2300.871174	0.97964033	4.911670911
0.9	0.01	0.001	100000	1.001562038	2304.529699	0.997518052	40.29093267
0.9	0.05	0.001	1000	1.002947755	695.838612	0.956506003	2.299167843
0.9	0.05	0.001	10000	1.000263576	693.2090557	0.992853625	13.99310937
0.9	0.05	0.001	100000	1.001273649	694.1325914	0.999239476	131.4883667

Table V Numerical examples for the Beta prior distribution (a=1) with mass at origin

θ	x	y	n	a	b	θ^*	$(1-\theta)/(1-\theta^*)$
0.5	0.01	0.001	1000	1	3910.066668	0.556688485	1.127875058
0.5	0.01	0.001	10000	1	3910.066668	0.780581515	2.278750575
0.5	0.01	0.001	100000	1	3910.066668	0.963735283	13.78750575
0.5	0.05	0.001	1000	1	2301.433608	0.589240023	1.217255887
0.5	0.05	0.001	10000	1	2301.433608	0.842398512	3.17255887
0.5	0.05	0.001	100000	1	2301.433608	0.97799837	22.7255887
0.9	0.01	0.001	1000	1	2301.433608	0.928112406	1.391060597
0.9	0.01	0.001	10000	1	2301.433608	0.979635914	4.910605966
0.9	0.01	0.001	100000	1	2301.433608	0.997506611	40.10605966
0.9	0.05	0.001	1000	1	692.8005492	0.95650425	2.299075183
0.9	0.05	0.001	10000	1	692.8005492	0.992852421	13.99075183
0.9	0.05	0.001	100000	1	692.8005492	0.999236102	130.9075183

Optimizing (numerically) over this further refined set of allowable distributions, results in Table IV are obtained. The worst case conservative results here give large increases in the posterior probability of perfection as n increases; this contrasts with the results we obtained for the less constrained sets of allowable priors, shown in Table I and Table III.

The shape of the worst case distributions in Figure 7 are similar to the idealized case of Figure 1: because a is close to 1 and b is large, in each case, they have a mode at $(a-1)/(a+b-2)$, far to the left and very close to the origin, but in all cases there is zero probability density at 0 and at 1 (because both a and b are not less than 1).

In fact, the "worst case" results of Table IV are only worst within the accuracy of the numerical optimisation we used. Even though it has so far not been able to prove, the asymptotic optimum – i.e. the true worst case – seems to occur when a=1, and to satisfy the percentile constraint $b = log_{1-y}\, x/(1-\theta)$. Results for this case are shown in Table V.

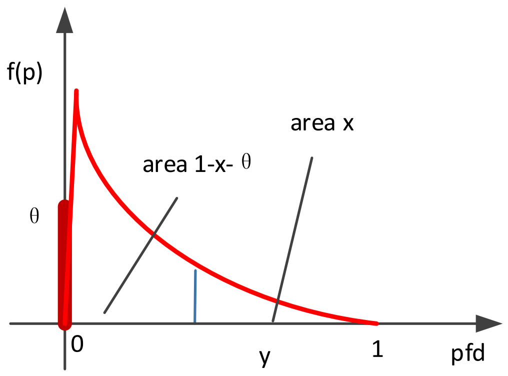

Figure 7 the most conservative prior re-normalized unimodal beta distribution after "pruning". This is an idealised representation: in reality the "spike" would be very large and close to the origin.

Within the "probability mass at 0 plus re-normalised Beta distribution" framework, these results are the most conservative with respect to the assessor's expressed prior beliefs, and (subject to our "prunings" of the original large class of priors satisfying only (4.1) and (4.2)) they can be thought of as "no more conservative than they need be".

The numerical values obtained above for the posterior probability of perfection seem potentially useful. But there is no claim for the practical plausibility of the vector of numbers $<x, y, \theta>$ that have been used for illustration. Note that in this case, as n goes to infinity, the assessor approaches certainty that the system is perfect.

4.4.2. Discussions on the use of a Beta

Given our interest in probability of perfection, readers may reasonably ask whether the "better" results have been obtained here are special to the Beta distribution assumption, or are more widely applicable. In particular, can it be expected to gain increasing confidence in perfection of software by observing large numbers of failure-free demands, under different assumptions? There is no definitive answer to this question. But it seems the Beta family assumption here is a rather weak one.

Using any particular parametric family of priors, as in our example using the Beta here, may be regarded as a "strong" assertion for an assessor. For example, in some safety-critical industries, a regulator with responsibility for oversight might reasonably respond: "You may represent your prior beliefs by this Beta family, but this seems unreasonably restrictive to me, so I cannot accept as reasonable the resulting claims you make for the probability of perfection.” Is it possible to impose weaker constraints on the form of the prior distribution – i.e. more “generally believable” ones – than the choice of a parametric family of distributions? One might, for example, try to state optimization constraints on only the general form of prior distribution, such as perhaps just its uni-modality on (0,1]. The Table IV and Table V show some results attained by unimodal priors which are constrained to be beta: what would be found if without the Beta family constraint but still with that of uni-modality? No clear answer currently. One can easily imagine other general classes of simple constraint for which similar questions can be asked: an upper or lower density bound, perhaps, or constraints involving distribution moments. These questions suggest possible avenues for further investigation into constrained worst cases priors which might help us to understand better the implications of what have been observed here with a Beta example.

4.5. Chapter summary

The work reported here was, of course, specifically addressing the problem of assessing confidence in perfection of software. However, to do this within the Bayesian framework, some general difficult problems concerning prior distributions have been considered. So the approach here may have wider applicability.

The Bayesian approach seems the most appropriate formalism for the assessment of the dependability of critical systems, but its use poses some difficulties for an assessor. The most obvious difficulties concern the need for the assessor to express his prior beliefs about his problem’s unknowns, formally and quantitatively, in order to feed into the formalism of Bayes’ theorem. It is well-known that this can be difficult, and has been used by some to argue that the Bayesian approach is impractical. Assessors are rarely able to provide a complete account of their uncertainty about the unknowns – in terms of the problem addressed in this work; they are unable to express their prior beliefs in a complete distribution for the unknown *pfd* of the system under study.

In the earlier work (Bishop, Bloomfield et al. 2011), a similar problem was considered. There the interest was in a system’s probability of failure on demand (rather than, as here, its probability of perfection, or non-perfection). It had been showed that useful results, guaranteed to be conservative, could be obtained even from very limited prior beliefs in that paper. The hope here is that something similar could be obtained in the case where interest centred on claims for perfection. However, an important result here – and a *disappointing* one – is that this seems not to be possible: see section 4.2. Even after observing large numbers of failure-free demands, confidence in perfection increases only modestly over an assessor's original prior belief. In fact he would not be certain of perfection even if he were able to see an *infinite* number of failure-free demands. The informal explanation for these disappointing results is that failure-free operation over many demands may be due to perfection or to a very small *pfd*. In section 4.4 ways in which it may sometimes be possible to rule out the second of these explanations were discussed.

The general approach here is in three stages. Firstly, although the assessor cannot provide a full probabilistic description of his uncertainty, he can express some partial but precise beliefs about the unknowns. Secondly, in the inevitable presence of such incompleteness, the Bayesian analysis needs to provide results that are guaranteed to be conservative (because the analyses is assumed to relate to critical systems). Finally, the assessor might seek to prune the set of prior distributions that he finds acceptable in order that the conservatism of the results is no greater than it has to be.

On the first point, the assessor's prior beliefs about system *pfd* are expressed in terms of just a prior probability that this *pfd* is zero (i.e. the system is perfect), together with a single percentile (because, in the event that the system is not perfect, its *pfd* may lie anywhere in the interval (0,1]): the assessor's prior beliefs are summarized in the numbers in (4.1) and (4.2). Of course, these are very "weak" beliefs. They fall far short of a complete distribution for the unknown *pfd*. The next step is to find the most conservative of the many possible complete prior distributions, i.e. the one that gives the most pessimistic value for the posterior probability of perfection. It turns out that this is very conservative – probably too much so to be of practical use.

Therefore the second stage is needed to "prune" the infinite set of initially allowable prior distributions, discarding those that it is argued all reasonable assessors would find unacceptable, and find the most conservative member of this new set of allowable priors. The corresponding most conservative posterior probability of perfection will be less conservative than the previous one.

There is a sense in which this pruning process makes the assessor's expressed prior beliefs more extensive than they were: it adds further assessor belief to those previously expressed. But note that, in contrast to the *specific and positive* assertions of a (hypothetical) real assessor represented by (4.1) and (4.2), this pruning process involves *general and negative* assertions about prior distributions: i.e. identifying properties that no prior distribution, for any reasonable assessor, should possess. So the pruning idea is an innovative method to elicit priors from assessors.

This is an important point that the pruning process described here does not mean that an assessor retrospectively changes his prior beliefs after he has seen evidence that produces embarrassingly inconvenient posterior consequences. That would be a perversion of a correct assessment process (and, of course, would be considered unacceptable within the Bayesian framework). Rather the claim is that some distributions can be discarded because they are unreasonable in general (for a broad class of situations, as discussed earlier in section 4.4), and are thus so in particular for this assessor.

The third stage was to analyze specific representations of the "pruned" set of prior distributions. Here an example was developed using a Beta distribution family for this continuous part of the prior for *pfd*. This choice can most certainly be questioned, but it was used for no more than a pure illustrative purpose[11].

Of course, failure-free working is not the only evidence that can be used to support claims about the reliability, or perfection, of software. Other software engineering measures and metrics have been proposed in the past to aid quantitative prediction of software reliability: see, e.g. (Li and Smidts 2000; Smidts and Li 2000). Whilst such evidence is generally not sufficient on its own to obtain accurate predictions of reliability (Cukic, Gunal et al. 2003), it seems that it may help assessors to provide the partial prior beliefs needed in the conservative Bayesian inference here, and perhaps to justify specific prunings. In particular, evidence from formal verification seems, on intuitive grounds, to be especially attractive (Littlewood and Wright 2007) as a means to support claims about perfection. Whilst this kind of evidence

[11] It might be argued that a Beta family is sufficiently flexible to represent most continuous beliefs over the unit interval, so that some member of the family may be a good approximation to an assessor's prior beliefs.

does not currently fit readily into the kind of Bayesian analysis here, an assessor might use it informally to support his (limited) subjective prior beliefs. More formal support for this kind of reasoning is clearly needed.

5. CLAIMS ON PROBABILITY OF PERFECTION VIA PROCESS EVIDENCE

Chapter 4 addresses the problem of assessing the *subjective* probability of perfection using extensive failure-free working of the single software in question, which could be used not only on its own (e.g. the first example in section 1.1.4) but also as the input of the theorems in the modified LP-single models in section 3.2.2. The difference in the two probabilities of perfection hints at the different ways we might learn about them. For subjective probability of perfection concerning a *particular program*, we would like to see the outcome (success or failure) of many randomly selected demands executed by that program. But for a *pnp* (i.e. an objective parameter) concerning a development *process* (population of programs), we would like to know, for each of many randomly selected programs, which of them are not perfect. Thus in this chapter, the evidence of extensive failure-free working of *previous similar products* (built using the same process) will be considered to address the problem of how to assess pnp_B which is the required input of the original LP model, i.e. beliefs on the *objective* probability of perfection. This is intuitively the right way forward, i.e. using the process evidence of a *population* of similar products to learn about a property (i.e. *pnp*) of that population. The learning approach here is Bayesian, as in previous chapters. An important feature, again, is that it takes account of the fact that experts can only provide limited prior beliefs about model parameters. It will be shown how to obtain useful but *guaranteed-conservative* results from such restricted prior belief.

5.1. A general model of “process quality” evidence

The model in this chapter is an extension of (Littlewood and Wright 1995), shifting the interest from centring upon reliability to claims about perfection. So we begin by describing the general model in (Littlewood and Wright 1995) for learning about the dependability of a system using evidence about the efficacy of the process used to build it, obtained from observing failure-free operation of previous “similar” systems, then later specialise this general model in order to support claims about perfection.

The informal scenario in mind is the following. We have built a new system. We want to predict its reliability (including its possibility of perfection). We have some information about this system, e.g. it has survived a number of tests without failure. We also have similar evidence from previous similar products. We wish to use all this evidence to predict the reliability of our current system. The informal idea here is that there is a common development process for all systems that are addressing “similar” problems. The process thus can be thought of as generating a possible *population* of systems, of which the current one – the one that interests us – is a typical example. Knowledge of the population thus informs our judgment of *this* system.

Slightly more formally, the “evidence” for each system is just the record of outcomes (success or failure) for the demands that it has experienced during its lifetime: see Figure 8 from (Littlewood and Wright 1995). This evidence thus supports the kind of informal claim that we have seen used by builders of safety-critical systems: “trust us to have built *this* system right because we have built similar ones in the past, and here’s the evidence of *their* success, i.e. evidence of the success of our design-and-build process that was used to build *your* system.” The aim is to formalize such reasoning, and to do so it shall begin by assuming the following *doubly stochastic* model.

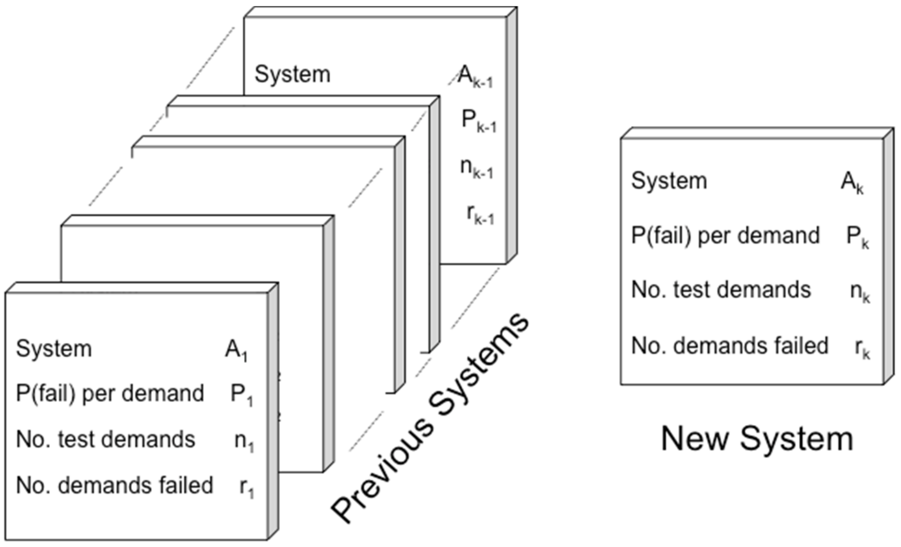

Figure 8 evidence from previous "similar" systems' behaviour to support a claim about the novel (but "similar") system of interest

At the first level of the general model, for each system, A_i, the successes/failures on demands will form a Bernoulli process. That is, successive demands of system A_i fail independently, with probability P_i, say. Thus, if system A_i has executed a known number n_i of test demands, and given $P_i=p_i$, the (random variable) number of failures would trivially have a binomial distribution.

The process generates different systems, A_i, and these different systems will have different true and unknown *pfd*s, p_i. In this second level of the doubly stochastic model, it is assumed that these system failure probabilities are sampled independently from a distribution, f_p. That is, the P_i are independent, identically distributed random variables: $P_i|\tau \sim f_p(p|\tau)$ given some parametric family of distributions $f_p(p|\tau)$ characterized by τ (possibly a vector).

The distribution f_p can be thought of as characterizing the development process, as far as its ability to produce reliable systems is concerned. More precisely, it characterizes the reliability variation between systems that come from the same development process. Thus if it were a highly concentrated distribution (i.e. has a very small variance), the process could be regarded as consistent, inasmuch its products would have reliabilities that differed little from one another. If most of the probability mass of f_p were near the origin, the system probabilities of failure would all tend to be small – i.e. the process would be "a good one", producing mainly reliable products.

In this reliability model, *building a system* can be thought of as randomly selecting a *pfd* from a population with distribution f_p. We shall be interested here in the case where there is a possibility of a chosen system being perfect – i.e. its *pfd* is zero. In fact the interest will centre on the probability of this event, which is just the probability mass of f_p concentrated at the origin.

The inter-product variation in this model is represented by an objective parameter τ. We can have a relative frequency interpretation of τ, by considering the following thought experiment. First, for each particular system A_i let $n_i \to +\infty$, it is easy to see that the ratio r_i/n_i converges to the true (but unknown, unobservable) probability of failure on demand, p_i. Now imagine generating k systems – i.e. randomly and independently selecting k probabilities of failure on demand, $\{p_i\}$, from f_p – and let $k \to +\infty$. This generates an infinite number of *systems*, and thus an infinite random sample of *pfd*s, p_i. A histogram of these p_is will converge to the true (but unknown) distribution $f_p(p|\tau)$, and we shall therefore know the value of τ with certainty.

The objective (i.e. true but unknown) parameter τ fully describes the shape of the population *pfd* distribution $f_p(p|\tau)$. Therefore, knowing τ with certainty will tell us our interest – the probability mass of f_p concentrated at the origin, i.e. the probability of perfection of the population. However, we could never know τ for certain in real life, rather learn about it in a Bayesian manner via collecting more and more evidence.

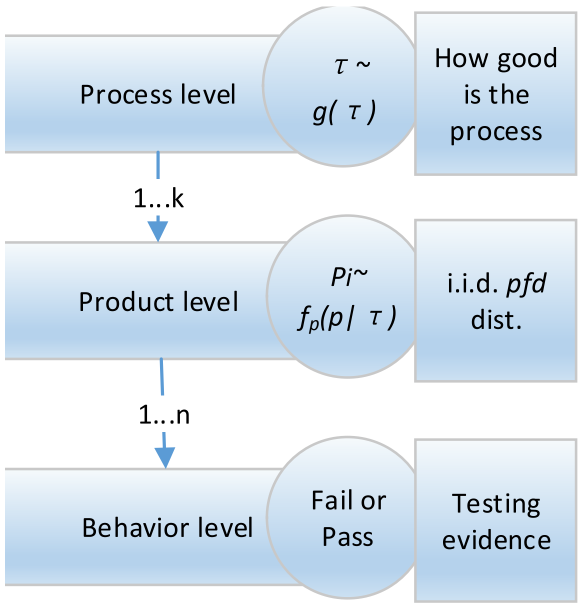

Figure 9 schema for the different levels of the general model

The Figure 9 is a overall schema of the different layers of this general model that need to be populated to enable its use.

At the bottom is the "behaviour level" that describes the *aleatory uncertainty* about failures or successes on demand of a system. Each system has a *pfd* that determines whether a randomly selected demand will result in failure (or not), and successive demands are assumed to fail independently. This *pfd* of a (randomly chosen) *system* is determined at the middle "product level" of Figure 9: for different products, these *pfd*s are independent and identically distributed random variables from a distribution $f_p(p|\tau)$, characterized by a vector of parameters, τ. This parameter, τ, is unknown. The statistical inference here– based upon evidence from the testing of several products – concerns this parameter at what has been called the "process level" of the Figure 9 schema. This statistical inference will proceed automatically, via Bayes Theorem, starting from the expert's prior distribution, $g(\tau)$.

Rather than continue this account with the full general model, a simple example to show the basic idea will be illustrated in next section.

5.2. A simple example using two-point distribution as f_p

In this section, we consider a simple case of a two point distribution of $f_p(p|\tau)$ with θ_{PP} mass at origin (representing the probability of perfection of the population) and $1-\theta_{PP}$ mass at π, as in Figure 10.

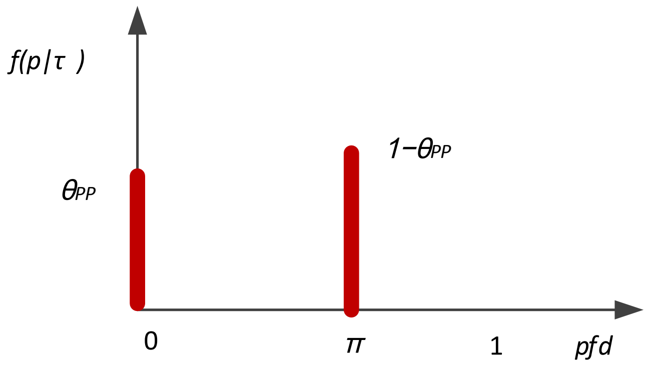

Figure 10 a simplified two-point f_p distribution

Here, for simplicity, π is assumed to be known. This is, of course, unrealistic, but will suffice for this simple illustration. It follows that the parameter θ_{PP}, representing probability of perfection of the population, on its own completely characterises the distribution f_p. That is,

θ_{PP} alone is the parameter τ in the previous section, at the process level of Figure 9, that determines the development process quality.

The interest is in the subjective *posterior* distribution of probability of perfection, say θ_{PP}^*, having seen evidence from the successful operation of several similar products. If an expert were able to provide a complete *prior* distribution, say $g(\theta_{PP})$, he can simply use Bayes theorem to obtain this posterior distribution. If, for simplicity, the *k* systems have each executed *n* demands without failure, we have:

$$\theta_{PP}^* \sim g(\theta_{PP}|\text{process evidence}) = \frac{L(\theta_{PP};\text{process evidnce})g(\theta_{PP})}{\int_0^1 L(\theta_{PP};\text{process evidnce})g(\theta_{PP})d\theta_{PP}} \quad (5.1)$$

where the likelihood function is:

$$L(\theta_{PP};\text{process evidnce}) = [\theta_{PP} + (1-\pi)^n(1-\theta_{PP})]^k \quad (5.2)$$

As argued elsewhere in this thesis, it is generally unreasonable to expect an expert to be able to provide a complete $g(\theta_{PP})$ as prior. But it is often feasible to express some *precise* but *partial* beliefs about the unknown θ_{PP}. Now examine an example of the very simple case of knowing only one percentile of θ_{PP}.

5.2.1. One percentile of θ_{PP} as partial prior belief

In this case, the assessor can only tell us one percentile of θ_{PP} as his prior belief, i.e. the confidence bound (y, α_θ):

$$Pr(\theta_{PP} < y) = \alpha_\theta \quad (5.3)$$

To use some of the results of original LR and LP models (Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a) – we need a *posterior* confidence bound. If the assessor would express a complete prior distribution, this would be:

$$Pr(\theta_{PP} < y|\text{process evidence}) = \alpha_\theta^*$$
$$= \frac{\int_0^{y-} L(\theta_{PP};\text{process evidnce})g(\theta_{PP})d\theta_{PP}}{\int_0^1 L(\theta_{PP};\text{process evidnce})g(\theta_{PP})d\theta_{PP}} \quad (5.4)$$

where $g(\theta_{PP})$ is the *prior* distribution as before, and L is the likelihood function as (5.2).

Now in general there will be an infinite number of prior distributions satisfying (5.3). The question is which of these gives us the most conservative (i.e. maximum[12]) α_θ^*.

It could be shown that Figure 11 is the most conservative prior of θ_{PP} (see Appendix C for proof), in which all the mass of $g(\theta_{PP})$ collapses to the point *y*. Therefore, starting with only one percentile prior (5.3), we cannot learn about θ_{PP} from process evidence, i.e. $\alpha_\theta = \alpha_\theta{}^*$.

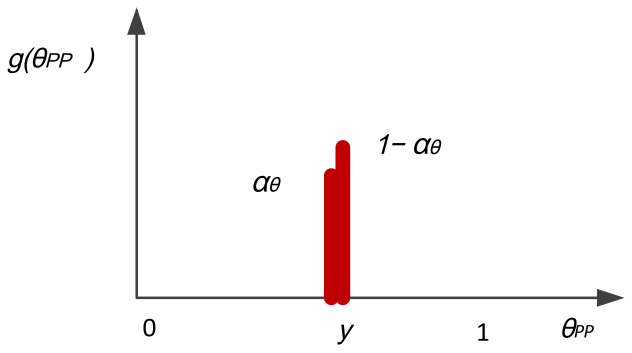

Figure 11 the most conservative prior of θ$_{PP}$

This case is, of course, unhelpful. The extremely minimal prior belief here is *too* minimal to provide useful learning from the multiple product evidence. Therefore the case in which the expert can provide *two* percentiles of θ_{PP} as prior belief will be considered next.

5.2.2. Two percentiles of θ_{PP} as partial prior belief

Consider the case where we have two percentiles of $g(\theta_{PP})$, i.e. the confidence bounds $Pr(\theta_{PP} < y_1) = \alpha_{\theta 1}$ and $Pr(y_1 \leq \theta_{PP} < y_2) = \alpha_{\theta 2}$, as in Figure 12.
The corresponding posterior confidence bounds in terms of a complete prior distribution $g(\theta_{PP})$ are:

$$Pr(\theta_{PP} < y_1 | \text{process evidence}) = \alpha_{\theta 1}^*$$
$$= \frac{\int_0^{y_1-} L(\theta_{PP}; \text{process evidnce}) g(\theta_{PP}) d\theta_{PP}}{\int_0^1 L(\theta_{PP}; \text{process evidnce}) g(\theta_{PP}) d\theta_{PP}} \qquad (5.5)$$

[12] $Pr(\theta_{PP}<y)=\alpha_\theta$ equals to $Pr(1-\theta_{PP}<1-y)=1-\alpha_\theta$, so α_θ is essentially the doubt that *pnp* (probability of non-perfection) is smaller than a certain bound. We want this doubt to be small, so here conservatism means to maximize it.

$$Pr(\theta_{PP} < y_2|\text{process evidence}) = \alpha^*_{\theta 1 \cap \theta 2}$$
$$= \frac{\int_0^{y_2-} L(\theta_{PP};\text{process evidnce}) g(\theta_{PP}) d\theta_{PP}}{\int_0^1 L(\theta_{PP};\text{process evidnce}) g(\theta_{PP}) d\theta_{PP}} \tag{5.6}$$

where $g(\theta_{PP})$ is the *prior* distribution as before, and L is the likelihood function as (5.2).

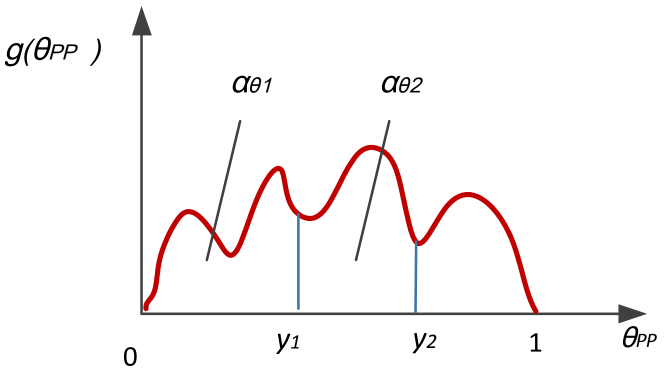

Figure 12 two percentile constraints of θ_{PP} as priori belief

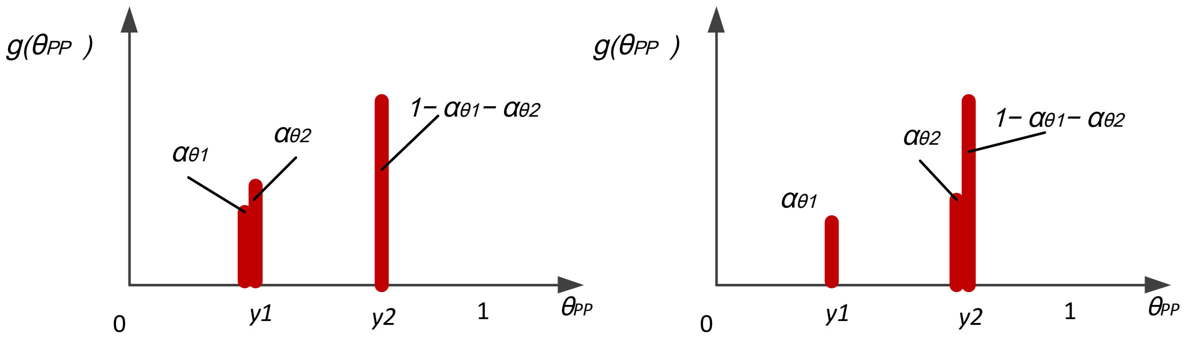

Figure 13 the most conservative prior distributions giving result (5.5) and (5.6) respectively

Appendix C proves that the most conservative prior distributions satisfying the two expert belief constraints are the point-mass distributions in Figure 13. The corresponding most conservative posterior confidence results, $\alpha^*_{\theta 1}$ and $\alpha^*_{\theta 1 \cap \theta 2}$ (keeping the same bounds, y_1 and y_2) are:

$$\alpha^*_{\theta 1} = \frac{\alpha_{\theta 1}}{\alpha_{\theta 1} + \alpha_{\theta 2} + \frac{L(y_2)(1-\alpha_{\theta 1}-\alpha_{\theta 2})}{L(y_1)}} \tag{5.7}$$

$$\alpha^*_{\theta 1 \cap \theta 2} = \frac{L(y_1)\alpha_{\theta 1} + L(y_2)\alpha_{\theta 2}}{L(y_1)\alpha_{\theta 1} + L(y_2)(1-\alpha_{\theta 2})} \tag{5.8}$$

Table VI are numerical results based on different values of the various model parameters.

Table VI Numerical examples of the case with prior beliefs expressed in terms of two percentiles

case #	y_1	$\alpha_{\theta 1}$	y_2	$\alpha_{\theta 1 \cap \theta 2}$	π	n	k	$\alpha^*_{\theta 1}$	$\alpha^*_{\theta 1 \cap \theta 2}$
1	0.9	0.05	0.99	0.1	0.001	10000	10	0.020540109	0.071473861
2	0.9	0.05	0.99	0.1	0.001	10000	50	0.000472913	0.053056233
3	0.9	0.05	0.99	0.1	0.001	1000000	10	0.02053921	0.071473018
4	0.9	0.05	0.99	0.1	0.01	10000	10	0.02053921	0.071473018
5	0.5	0.05	0.7	0.1	0.001	10000	10	0.001913788	0.054352684
6	0.9	0.05	0.99	0.1	0.000001	10000	10	0.04959824	0.099598419
7	0.9	0.05	0.99	0.1	0.000001	10000	50	0.048019803	0.098024151
8	0.9	0.05	0.99	0.1	0.000001	1000000	50	0.002897123	0.055239721
9	0.9	0.05	0.99	0.1	0.000001	1000000	10	0.029020956	0.079498839

Consider case #1, with some arbitrary but not unreasonable numbers for the parameters. Here we do learn something from the process evidence (10 similar products that each passed 10000 tests). However, even though the results are better than the "learn-nothing" results in section 3.1, the two posterior confidence bounds on θ_{PP} are still very unhelpful: the decrease of doubts from priors to posteriors (i.e. comparing the columns $\alpha_{\theta 1}$ with $\alpha^*_{\theta 1}$ and $\alpha_{\theta 1 \cap \theta 2}$ with $\alpha^*_{\theta 1 \cap \theta 2}$) is quite limited.

Cases #2 and #3 consider the impact of increases in *k* and *n*, respectively. The benefit of increasing *k* (from 10 to 50) is much greater – two orders of magnitude – than increasing *n* (from 10^4 to 10^6), even though the total number of tests (i.e. the number $k \times n$) for case #2 is 500000 which is much less than the 10^7 tests in case #3. The latter observation should not be taken as a guide to the relative "cost" of the two scenarios, however, since the number of test cases is a poor guide to the "cost" of gaining the evidence in these two cases: developing many systems is far more onerous than generating many test cases.

The importance of *k* here is not surprising. It seems intuitively obvious that "all things being equal" we learn more about the efficacy of the process by having evidence of good working from many products, than we do from massive exposure of only a few products. In practice, of course, it seems unlikely there will be available very many previous products to provide this kind of evidence.

Another observation that confirms intuition is that the weaker the claims are, the greater the confidence we could obtain from process evidence. Consider case #5: the claim $\theta_{PP} < 0.5$ is weaker than it is for the previous cases $\theta_{PP} < 0.9$, so seeing the same evidence (*n*=10000*, k* =10), more could be learnt in this case. Notice also that for each case the improvement for $\alpha^*_{\theta 1}$ is better than $\alpha^*_{\theta 1 \cap \theta 2}$. Again, this is because the claims (i.e. $Pr(\theta_{PP} < y_1)$) for $\alpha^*_{\theta 1}$ are weaker than the ones for $\alpha^*_{\theta 1 \cap \theta 2}$ (i.e. $Pr(\theta_{PP} < y_2)$).

We shall not pursue analysis of Table VI any further here, since this set-up with the two-point f_p distribution is unlikely to be a realistic representation of reality; it was meant only to be an illustrative aid to understanding.

However, the reader will note that, even for this unrealistically simple example, there are differences between the results of section 5.2.1 and 5.2.2. The *very* restricted prior beliefs of section 5.2.1 do not allow any useful learning from the evidence of failure-free working of multiple systems; the slightly more informative prior beliefs of section 5.2.2, in contrast, provide some modest learning. An important issue in this kind of study is identifying exactly *how* minimal prior beliefs can be – to aid the task of the assessor – without being *too* minimal to give useful results. In the following section, a more plausible model to investigate such issues will be introduced.

5.3. More practical assumption of an arbitrary f_p with mass at the origin

In the previous section, to illustrate the general approach, an analysis based on an *unrealistic* two-point-mass distribution f_p at the middle level of the schema Figure 9 was proposed; in this section, a more practical way to treat this middle layer of the model will be introduced.

Readers will note, however, that even with the unrealistic "product level" simplification of section 5.2.1 – where the only unknown quantity is θ_{pp} – there arise quite difficult problems concerning prior beliefs. What would be a reasonable assumption about f_p?

Clearly, this must have mass at the origin, θ_{pp}, since this is the centre of our interest. But what happens in (0,1]? An obvious "classical" approach then would be to assume a parametric family for this density. E.g. we could assume that p had a (conditional, given $p > 0$) Beta distribution with parameters (α, β). In that case the unknown model parameter at the top level of the Figure 9 schema would be $\tau =< \theta_{pp}, \alpha, \beta >$, i.e. a *vector* of parameters.

However there are problems with such an approach. Most importantly, it is hard to justify the choice of a Beta (or any other) parametric family. Additionally, experts would find it hard to express their prior beliefs about the vector $< \theta_{pp}, \alpha, \beta >$. In what follows, therefore, a different way forward will be proposed that does not rely on such a parametric assumption; in fact no assumptions are made about the shape of the distribution f_p for its non-zero part.

5.3.1. The introduction of R

Now consider an *unknown arbitrary* distribution f_p with some mass (θ_{PP}) at the origin. The objective function is:

$$\begin{aligned} Pr(\theta_{PP} < y|\text{process evidence}) &= \frac{Pr(\theta_{PP} < y \textit{ and } \text{process evidence})}{Pr(\text{process evidence})} \\ &= \frac{E_{f_p}\left(I_{\theta_{PP}<y}(f_p) \times Pr(\text{process evidence}|f_p)\right)}{E_{f_p}\left(Pr(\text{process evidence}|f_p)\right)} \end{aligned} \tag{5.9}$$

where $I_{\theta_{PP}<y}(f_p)$ is an indicator function using all possible f_p as inputs. When the mass at the origin of a f_p is less than y, $I_{\theta_{PP}<y}(f_p) = 1$, otherwise 0. E_{f_p} is the expectation over all possible f_p. The use of the indicator function $I_{\theta_{PP}<y}(f_p)$ ensures the mean value in the numerator only involves those possible f_p having a mass at the origin less than y.

And we could know:

$$Pr(\text{process evidence}|f_p) = \left[\theta_{PP} + \int_{0+}^{1} (1-p)^n f(p|\tau)\, dp\right]^k \tag{5.10}$$

So if we denote[13]:

$$R = \int_{0+}^{1} (1-p)^n f(p|\tau)\, dp \tag{5.11}$$

[13] Strictly, since it is a function of *n*, it would be more precise to talk of *R(n)* here and in what follows. *R* is used here instead just for notational simplicity.

then:

$$Pr\left(\text{process evidence}\middle|f_p\right) = [\theta_{PP} + R]^k \tag{5.12}$$

which is a function only of the pair $< \theta_{PP}, R >$. And in principle the joint distribution of θ_{PP} and R, say $g_{<\theta_{PP},R>}$, could be calculated from the $g(\tau)$ (i.e. the distribution of all possible f_p).

Then the objective function (5.9) turns into:

$$\begin{aligned} &Pr(\theta_{PP} < y|\text{ process evidence}) = \\ &= \frac{E_{f_p}\left(I_{\theta_{PP}<y}(f_p) \times Pr\left(\text{process evidence}\middle|f_p\right)\right)}{E_{f_p}\left(Pr\left(\text{process evidence}\middle|f_p\right)\right)} \\ &= \frac{E_{<\theta_{PP},R>}\left(I_{\theta_{PP}<y}(\theta_{PP}) \times [\theta_{PP} + R]^k\right)}{E_{<\theta_{PP},R>}([\theta_{PP} + R]^k)} \end{aligned} \tag{5.13}$$

which depends only on the joint distribution of θ_{PP} and R, say $g_{<\theta_{PP},R>}$.

This result is important for the following reasons. The problem of how to deal with f_p at the "product level" of Figure 9 has been transformed into a problem at the "process level". Most importantly, no assumptions about the shape of f_p for non-zero p have been made. Instead an expert "*only*" has to express joint prior beliefs about the pair of parameters θ_{PP} and R. Mathematically, R is sufficient to represent the non-zero part of f_p in terms of Bayesian learning about the probability of perfection[14].

Quotes for "only" in the previous paragraph are used because this remaining problem concerning prior beliefs is not a simple one. Certainly, it would be unreasonable to expect an expert to have a *complete* bi-variate distribution $g_{<\theta_{PP},R>}$ for his prior beliefs. As discussed elsewhere, experts have great difficulty expressing beliefs about *dependence*, and even providing complete *marginal* prior distributions.

In what follows are some results that require only *marginal* prior beliefs about the unknown model parameters, and these marginal beliefs will themselves be only *partial* – typically only

[14] Note the similarity of this observation to the concept of *sufficient statistic* in classical statistics. See, for example, https://en.wikipedia.org/wiki/Sufficient_statistic.

one or two percentiles. Of course, reducing the burden on experts in these ways introduces further conservatism in the results.

5.3.2. Bayesian learning with one marginal percentile of θ_{PP} and R respectively

As in section 5.2, firstly the case where an expert assessor is able to provide only very minimal prior beliefs about the unknown parameters is considered. Specifically he/she is only prepared to provide the four numbers that constitute a single marginal percentile for each:

$$Pr(\theta_{PP} < y) = \gamma_\theta \tag{5.14}$$

$$Pr(R < r) = \gamma_r \tag{5.15}$$

To illustrate the approach, only the case most likely to occur in practice will be considered here, in which $y + r \leq 1$. The reasoning in other cases is similar, but with some tedious differences of detail.

As the discussion in (Bishop, Bloomfield et al. 2011) and chapter 4 about the "conservative Bayesian inference", although a standard Bayesian analysis would here require an expression of full prior beliefs about a *pair*, now, of parameters θ_{PP} and *R,* we continue instead to pursue a less ambitious "conservative" treatment. This involves working from only the *partial* elicitation of this bivariate prior represented by constraints (5.14) and (5.15). Although not fully elicited, our approach nevertheless reasons with the *concept* of the potential bi-variate prior distribution. We shall denote such a hypothetical bi-variate joint distribution, of θ_{PP} and *R,* by $g_{<\theta_{PP},R>}$ illustrated in Figure 14. Because of the constraint $\theta_{PP} + R \leq 1$, this distribution would have non-zero density only beneath the broken line connecting (0, 1) and (1, 0). In this triangle, the probability masses associated with the four regions in the figure are label as M_1, M_2, M_3, M_4. Then we have $Pr(\theta_{PP} < y) = \gamma_\theta = M_1 + M_3$ and $Pr(R < r) = \gamma_r = M_1 + M_2$.

The interest lies in the posterior confidence bound for θ_{PP}:

$$Pr(\theta_{PP} < y|\text{ process evidence}) = \frac{\int_0^1\int_0^1 \left(I_{\theta_{PP}<y}(\theta_{PP}) \times [\theta_{PP} + R]^k\right) g_{<\theta_{PP},R>}(\theta_{PP}, R)\, d\theta_{PP}\, dR}{\int_0^1\int_0^1 [\theta_{PP} + R]^k g_{<\theta_{PP},R>}(\theta_{PP}, R)\, d\theta_{PP}\, dR} \tag{5.16}$$

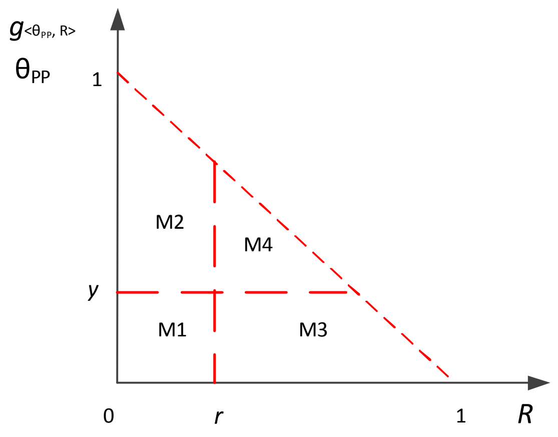

Figure 14 the prior constraints (5.14) and (5.15) on the joint distribution of θ_{PP} and R

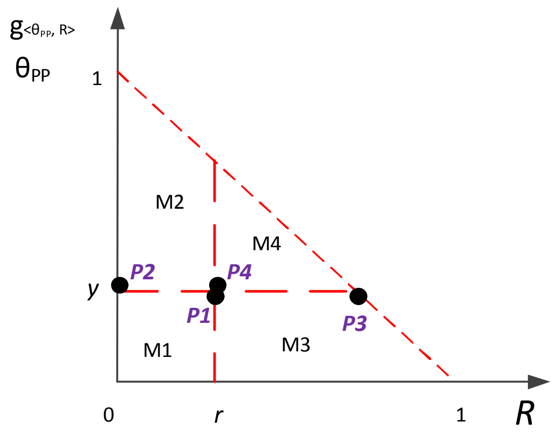

Figure 15 the most conservative joint prior distribution satisfying the prior constraints (5.14) and (5.15)

The most conservative joint prior, $g_{<\theta_{PP},R>}$, satisfying the expert's constraints (5.14) and (5.15) is the 4-point-mass distribution shown in Figure 15 (see Appendix C for proof). The dots, P_1, P_2, P_3, P_4, are representing the probability masses of the related value ranges. The corresponding most conservative posterior confidence bound for θ_{PP} is:

$$Pr(\theta_{PP} < y| \text{ process evidence}) \leq \frac{1}{1+\frac{y^k\gamma_r + [y+r]^k(1-\gamma_\theta-\gamma_r)}{\gamma_\theta}} = \gamma_\theta^* \qquad (5.17)$$

So γ_θ^* is the guaranteed-conservative posterior bound for the probability of perfection among the infinite number of prior distributions satisfying the expert's prior constraints (5.14) and (5.15), i.e. none results in a value for $Pr(\theta_{PP} < y| \text{ process evidence})$ smaller than γ_θ^*.

Table VII shows some numerical examples of the γ_θ^*.

Table VII numerical examples of the "counter-intuitive" results of the very limited priors

case #	y	γ_θ	r	γ_r	k	γ_θ^*
1	0.9	0.05	0.09	0.05	10	0.056729362
2	0.9	0.05	0.099	0.05	10	0.052166239
3	0.99	0.05	0.009	0.05	10	0.050696597
4	0.99	0.05	0.0099	0.05	10	0.050285647

In fact, Table VII shows some very *counter-intuitive* results: in all cases we find that $\gamma_\theta^* > \gamma_\theta$. That is, observing "good news" process evidence (failure-free working of k products) results in the posterior belief in perfection being *worse* than the prior belief. In the next section, the reasons for this result will be discussed in detail – essentially the reason is that the prior beliefs here are *too minimal* to be useful. This analysis suggests that priors be made sufficiently "partial" to make the expert's task feasible, but at the same time sufficiently informative to produce useful results.

5.3.3. The "counter-intuitive" results

The results above are "counter-intuitive" because seeing good evidence did not enhance our confidence in the positive claim about probability of perfection that interests us, on the contrary it increased our doubt. This result is not due to the choice of numbers in the examples of Table VII: for any valid numbers of the model parameters the "counter-intuitive" result (i.e. $\gamma_\theta^* > \gamma_\theta$) will apply.

To understand this result, consider the most pessimistic prior distribution (satisfying the percentile constraints (5.14) and (5.15)) in Figure 15, which is a bi-variate distribution with probability mass concentrated on four points: P_1 at (r-, y-)[15], P_2 at (0, y), P_3 at $(1 - \text{y}$, y-) and P_4 at (r, y). The corresponding masses at these points are M_1, M_2, M_3, M_4 respectively. The effect of seeing more and more process evidence is that in the posterior distribution the masses at P_2, P_1 and P_4 all gradually move to P_3. Since P_3 $(1 - \text{y}$, y-) is below the line $\theta_{PP} = y$, it follows that $Pr(\theta_{PP} < y|\text{ process evidence}) > Pr(\theta_{PP} < y)$ will always hold.

[15] Note the difference with (r, y): (r-, y-) is the coordinate infinitesimally close to (r, y) but smaller than r and y in both directions. The minus signs used here and later means smaller but infinitesimally close to the number.

What does this mean practically? The four points of probability mass in Figure 15 can be thought of as representing 4 different development processes:

- The points P_1 and P_4 represent very similar processes that produce *some* ($100y\%$) perfect, *some* ($100r\%$) reliable (but not perfect) and *some* ($100(1-y-r)\%$) unreliable versions.
- The P_2 process produces *some* ($100y\%$) perfect, *no* (0%) reliable (but not perfect) and *many* ($100(1-y)\%$) unreliable versions.
- The P_3 process produces *some* ($100y\%$) perfect, *many* ($100(1-y)\%$) reliable (but not perfect) and *no* (0%) unreliable versions.

As we accumulate evidence of only good working, we tend to believe more strongly that our development process is P_3, i.e. the one with no unreliable versions. That is, seeing good process evidence we tend to rule out processes producing unreliable versions. And in our worst case, the only process producing no unreliable versions is P_3. But *P_3's* ability to produce perfect versions is lower than what we want to claim (as the ordinate is y- which is smaller than y). Therefore, believing more strongly in the process P_3 means having increasing doubt that the probability that randomly selected software from the process is not perfect is lower than a claimed bound.

But why are there only these 4 development processes as candidates here? Ideally, for example, there could be a development process producing many perfect versions, based on the very best practices and design principles of utmost simplicity. If we had a prior belief that there exists this kind of process (even with very small probability), we would expect helpful support from the evidence to increase our confidence about the probability of perfection. Unfortunately, prior belief in the possibility of this kind of "perfection favoured" development process is ruled out due to the conservative nature of the method. That is, starting with the very minimal prior belief of only one marginal percentile of θ_{PP} and R respectively (i.e. constraints (5.14) and (5.15)), the most conservative joint prior distribution does not allow the possibility of "perfection favoured" process.

The counter-intuitive results, then, arise because in the pursuit of simplicity to aid the expert's task, we have allowed an expert to express partial prior beliefs that are *too* minimal.

In what follows we suggest to relax this minimalism slightly: the cost, of course, is a heavier burden on the expert in expressing prior beliefs.

5.3.4. Using less minimal prior knowledge to get useful results

As discussed above, the most conservative prior distribution in Figure 15 did not include the possibility of a development process producing many perfect software versions. Mathematically, this is due to the possibility for the marginal distribution of θ_{PP} to have zero variance (since only one prior percentile of θ_{PP} has been specified). Thus in the worst case, the marginal distribution of θ_{PP} will conservatively collapse onto one point (the *y* point of the vertical axis in Figure 15). Informally this collapsing means all our candidate development processes have the *similar* capability to produce perfect versions. Therefore the assessor will believe that the good process evidence is *only* due to high reliability property of the process. Unfortunately, in the worst case (i.e. Figure 15), the one having the highest reliability property is a one having slightly less capability of producing perfect versions than our interest.

By extending assessors' priors about θ_{PP} to at least two percentiles, he could rule out the possibility of a prior having zero variance and so stop the collapsing onto a single point of the marginal distribution of θ_{PP}. So, he would have candidate processes with different capabilities of producing perfect versions. Now perfection property might be thought to be the reason for seeing good process evidence. However, due to the conservative nature of the method, the "perfection favoured" process in the priors also tends to produce unreliable versions, and the "high reliability (but not perfect) favoured" process will never produce unreliable versions. So the "high reliability (but not perfect) favoured" process will always be preferred in the Bayesian learning. But this is obviously unrealistic, as no process will never produce unreliable versions. So there must be an upper bound on *R* to make room for the possibility of unreliable versions produced in all possible candidate processes.

Therefore, a new (less) minimal partial prior belief about the model parameters in terms of two percentiles for θ_{PP} and one percentile and a certain upper bound for *R* is proposed:

$$Pr(\theta_{PP} < y_1) = \gamma_{\theta 1} \quad (5.18)$$

$$Pr(y_1 \le \theta_{PP} < y_2) = \gamma_{\theta 2} \quad (5.19)$$

$$Pr(R < r) = \gamma_r \quad (5.20)$$

$$Pr(R < r_U) = 1 \tag{5.21}$$

The intention is that this set of prior beliefs may be sufficiently rich to produce useful results whilst still imposing upon an expert a manageable task in their expression. Figure 16 shows these prior beliefs, where the regions M_1, M_2,…M_6 contain non-zero probability mass. (Note that the position of the vertical line at r_U in Figure 16 is just for illustration; its precise location will be discussed in more detail later)

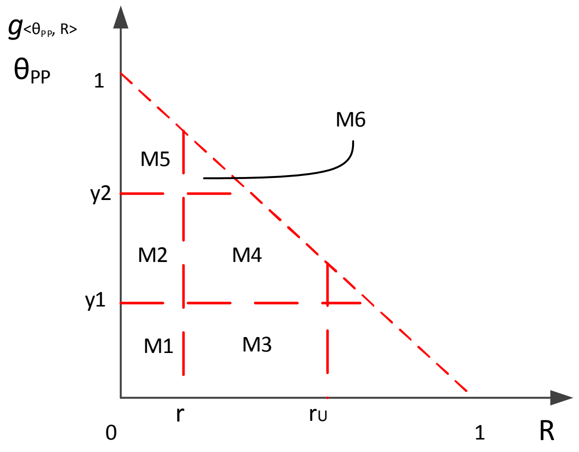

Figure 16 the minimum useful priori constraints on the joint distribution of θ_{PP} and R

From the definition of R, there is the constraint $\theta_{PP} + R \leq 1$ which derives the other two inexplicit constraints $y_1 + r < 1$ and $y_2 + r < 1$. The possible range of values of the certain upper bound r_U are: $1 - \mathrm{y}_1 \leq \mathrm{r}_U < 1$, $1 - \mathrm{y}_2 \leq \mathrm{r}_U < 1 - y_1$ and $\mathrm{r} \leq \mathrm{r}_U < 1 - y_2$.

As Appendix C, it can be shown that useful results can be obtained only when the objective function is $Pr(\theta_{PP} < y_1 |$ process evidence$)$[16] and r_U lies in the range $1 - y_2 \leq r_U < 1 - y_1$ or in the range $r \leq r_U < 1 - y_2$. The corresponding most pessimistic joint prior distributions are in Figure 17 and Figure 18 respectively. As before these are distributions with only a number of point masses.

The two most conservative prior distributions in Figure 17 and Figure 18 result in the same objective function as below:

[16] As there are two prior confidence bounds on θ_{PP}, i.e. (5.18) and (5.19), both of them could be updated and potentially used as the input required by the LP model. But by proof, only the former could be useful.

$$
\begin{aligned}
&Pr(\theta_{PP} < y_1 | \text{ process evidence}) \\
&= \frac{\int_0^1 \int_0^1 \left(I_{\theta_{PP}<y1}(\theta_{PP}) \times [\theta_{PP} + R]^k\right) g_{<\theta_{PP},R>}(\theta_{PP}, R) d\theta_{PP} dR}{\int_0^1 \int_0^1 [\theta_{PP} + R]^k g_{<\theta_{PP},R>}(\theta_{PP}, R) d\theta_{PP} dR} \\
&\le \frac{[y_1 + r_U]^k \gamma_{\theta 1}}{[y_1 + r_U]^k \gamma_{\theta 1} + [y_1 + r]^k \gamma_{\theta 2} + y_2{}^k \gamma_r + [y_2 + r]^k (1 - \gamma_r - \gamma_{\theta 2} - \gamma_{\theta 1})} \\
&= \gamma_{\theta 1}^*
\end{aligned}
\tag{5.22}
$$

where $\gamma_{\theta 1}^*$ is the most conservative posterior belief of interest.

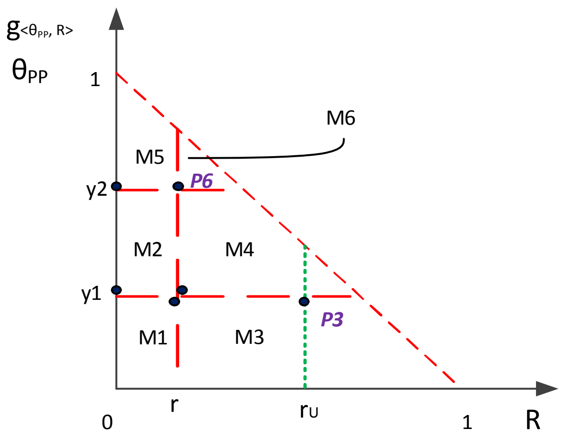

Figure 17 the most conservative prior satisfying the minimum useful priori constraints with ***1-$y_2 \le r_U < 1$-y_1***

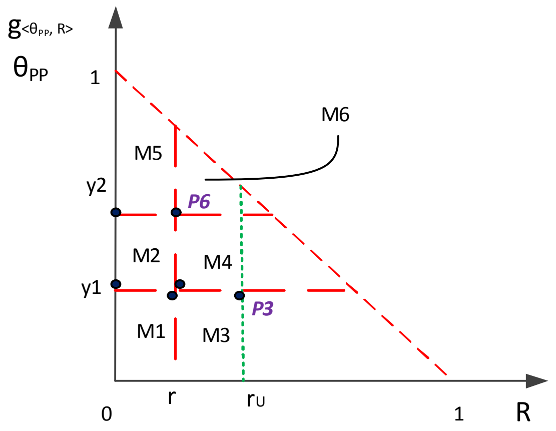

Figure 18 the most conservative prior satisfying the minimum useful priori constraints with ***$r \le r_U < 1$-y_2***

Thus, for example, the pair $(1 - y_1, \gamma_{\theta 1}^*)$ could be used as the input pair (p_B, α_B) of the original LP model, see (2.11) in section 2.4.3 for detail.

Table VIII shows some numerical examples. The first 4 cases are illustrating the results with constraint $1 - y_2 \le r_U < 1 - y_1$ and the rest of them are about the ones with constraint $r \le r_U < 1 - y_2$. The $\gamma_{\theta 1}^*$ column contains results that are both bigger and smaller than the prior value, $\gamma_{\theta 1} = 0.05$. It seems the "less minimal" prior constraints are indeed helpful in some cases compared with the counter-intuitive results in the 5.3.3, but not universally.

Table VIII Numerical examples using the "less minimal" prior knowledge

case #	y_1	$\gamma_{\theta 1}$	y_2	$\gamma_{\theta 2}$	r	γ_r	r_U	k	$\gamma_{\theta 1}^*$
1	0.9	0.05	0.99	0.05	0.009	0.05	0.0600	10	0.035391421
2	0.9	0.05	0.99	0.05	0.009	0.05	0.0999	10	0.052250881
3	0.9	0.05	0.99	0.05	0.009	0.05	0.0110	10	0.021265865
4	0.9	0.05	0.99	0.05	0.009	0.05	0.0600	50	0.007679250
5	0.9	0.05	0.99	0.05	0.009	0.05	0.0095	10	0.020925570
6	0.9	0.05	0.92	0.05	0.009	0.05	0.0790	10	0.082798474
7	0.9	0.05	0.99	0.05	0.009	0.05	0.0091	10	0.020835634
8	0.9	0.05	0.99	0.05	0.009	0.05	0.0095	50	0.000518818

The two counter-intuitive results happen in cases #2 and #6 in which the certain upper bound r_U is high. Mathematically, there is a competition between the points *P₃* and *P₆* in Figure 17 and Figure 18, in which the result depends on the specific choices of the parameters. The higher the certain upper bound r_U, the more likely *P₃* will be the result. The informal interpretation in that case is that we believe more in the *P₃* process which is producing many "ultra-reliable but not perfect" versions.

In contrast, when the result is the *P₆* process, we have our "perfection favoured" process. Cases #4 and #8 get practically useful results which are one and two orders of magnitude better that prior belief, respectively. Comparing with case #1 and #5 respectively where only the numbers of *k* vary, there is evidence that, for a certain set of prior numbers, collecting more versions in the process evidence is very helpful. This is consistent with the conclusion in the section 5.2.2, and is in accord with intuition. But comments about the feasibility of observing many previous products still apply.

Notice that, in contrast to *k*, we cannot say how the parameter *n* affects the result, since *n* is now hidden in the subjective beliefs about *R*. That said, from the definition of *R*, it is obvious that larger *n* means smaller *R*.

Comparing case #3 to case #1 and case #7 to case #5, the smaller values of r_U here indeed give better results; but its impact does not seem as great as that of k.

5.3.5. How to use the result? A possible negotiation model

Let's imagine a situation where there is to be negotiation between a regulator and the licensee of a system – say a nuclear plant – that contains a critical software-based subsystem. As part of the safety case for the plant, the licensee needs to make a claim for "probable perfection" of this subsystem. That is, he needs to convince the regulator to accept as reasonable his declared confidence that θ_{PP} is no smaller than some declared number y_1.

Clearly this cannot be done by mere assertion about the licensee's top-level claim here. It is agreed between the regulator and the licensee that their negotiation will take place within the framework of the approach outlined above. The model then allows the discussion between them to be a negotiation about the elements of the argument that underpin the regulator's claim: i.e. about numbers for the 7 parameters $< y_1, \gamma_{\theta 1}, y_2, \gamma_{\theta 2}, r, \gamma_r, r_U >$ [17] needed to obtain the conservative posterior bounding confidence (5.22).

Rather than say simply "trust me when I declare my confidence in perfection", the licensee says "here are the numbers I used to arrive at my confidence in perfection". *These numbers*, then, rather than simply the top-level claim, become the subject of discussion and negotiation with the regulator. Certainly these numbers are still difficult to reach an agreement in reality, but the negotiation framework proposed here eases the task to some degree. Below how such negotiation might be conducted is briefly sketched.

Begin with a simple monotonicity analysis of the impact of the seven parameters on $\gamma_{\theta 1}^*$. It is easy to see (by partial differentiation w.r.t. each parameter) that:

1. $\gamma_{\theta 1}^*$ is an increasing function in terms of $\gamma_{\theta 1}$.
2. $\gamma_{\theta 1}^*$ is a decreasing function in terms of y_2.
3. $\gamma_{\theta 1}^*$ is an increasing function in terms of $\gamma_{\theta 2}$.
4. $\gamma_{\theta 1}^*$ is a decreasing function in terms of r.
5. $\gamma_{\theta 1}^*$ is an increasing function in terms of γ_r.

[17] Recall that there are some constraints on the numerical values that these numbers need to satisfy.

6. $\gamma_{\theta 1}^{*}$ is an increasing function in terms of r_U.
7. The monotonicity of y_1 depends on specific choices of other parameters

The monotonicity analysis shows how changes to these parameters make claims about probability of perfection stronger or weaker: increasing parameters in 1, 3, 5, 6 make the posterior bound more conservative.

Note that 7 parameters are either "bounds" (y_1, y_2, r, r_U) or their corresponding "confidences" $(\gamma_{\theta 1}, \gamma_{\theta 2}, \gamma_r)$ which represent the licensee beliefs. It seems likely that agreement on the bounds will be easier, and negotiation will then concentrate on the related confidences. The bound y_1 is a special bound which is essentially the licensee's claim, his confidence in which he is trying to persuade the regulator is acceptable.

The negotiation could proceed in the following steps:

- They begin by agreeing on the "claim", y_1: this comes from higher-level system requirements. Then the licensee and regulator state their prior beliefs about this claim, i.e. values for $\gamma_{\theta 1}$.
- Second, licensee and regulator agree on y_2 and r. Then they state their beliefs about y_2 and r: values for $\gamma_{\theta 2}$ and γ_r.
- Finally, they negotiate on the r_U.

The following is a simple example of how this might proceed (the numbers are merely illustrative and not meant to be representative of those that would occur in a real negotiation about a critical system):

1. The claim concerning probability of perfection is *pnp*<0.1 (obtained from higher level requirements), so the y_1=0.9. For $\gamma_{\theta 1}$, the licensee's doubt that the probability of perfection is bigger than 0.9 is 0.05, and the regulator's doubt about it is 0.1. Then $0.05 \leq \gamma_{\theta 1} \leq 0.1$.
2. Then the two parties fix the $y_2 = 0.99$ and $r = 0.009$
    a. For $\gamma_{\theta 2}$, the licensee's doubt that the θ_{PP} is bigger than 0.99 is 0.1, and the regulator's doubt about it is 0.2. Then $0.05 \leq \gamma_{\theta 2} \leq 0.1$.
    b. For γ_r, the licensee's doubt that the *R* is bigger than 0.009 is 0.05, and the regulator's doubt about it is 0.1. Then $0.05 \leq \gamma_r \leq 0.1$.

3. The licensee does not believe that the R will be bigger than 0.0098, and the regulator does not believe that R will be bigger than 0.0095. Then $0.0095 \leq r_u \leq 0.0098$.

For each confidence/doubt here we have assumed that the regulator will be more conservative in his beliefs than the licensee. If the process evidence to form posterior beliefs is based on $k = 50$ versions, we have the following Table IX:

Table IX An illustrative numerical example of the negotiation model

y_1	$\gamma_{\theta 1}$	y_2	$\gamma_{\theta 2}$	r	γ_r	r_U	k	best $\gamma^*_{\theta 1}$	worst $\gamma^*_{\theta 1}$
0.9	[0.05, 0.1]	0.99	[0.05, 0.1]	0.009	[0.05, 0.1]	[0.0095, 0.0098]	50	0.00052	0.0012

The improvement/reduction in confidence/doubt about probability of perfection bound (i.e. from prior to posterior) is about the same for licensee and licensor in this example: around two orders of magnitude in each case. In the next example, Table X, there is significant variation (about an order of magnitude) between the two parties on parameter r_U, keeping the other parameters the same as before.

Table X A numerical example of the negotiation model with big variance on r_U.

y_1	$\gamma_{\theta 1}$	y_2	$\gamma_{\theta 2}$	r	γ_r	r_U	k	best $\gamma^*_{\theta 1}$	worst $\gamma^*_{\theta 1}$
0.9	[0.05, 0.1]	0.99	[0.05, 0.1]	0.009	[0.05, 0.1]	[0.0091, 0.098]	50	0.00051	0.110

Introducing this significant difference in the parties' views about r_U results in the best and the worst result being very different (the worst result even shows the "counter-intuitive" result). This suggests that the value of r_U is critical in negotiation.

Table XI shows some more results with different parameter values for each party. From this limited analysis it seems that prior belief in $\gamma_{\theta 1}$ also has a high impact on the final result. In contrast, wide variation in γ_r and $\gamma_{\theta 2}$ seems to have less impact.

From the limited evidence of these examples, it seems negotiation might centre upon $\gamma_{\theta 1}$ and r_U, since these seem to have the greatest effect on the result $\gamma^*_{\theta 1}$.

Table XI Numerical examples of the negotiation model with different variance on the parameters

y_1	$\gamma_{\theta 1}$	y_2	$\gamma_{\theta 2}$	r	γ_r	r_U	k	best $\gamma^*_{\theta 1}$	worst $\gamma^*_{\theta 1}$
0.9	[0.05, 0.1]	0.99	[0.05, 0.1]	0.009	[0.05, 0.1]	[0.0095, 0.0098]	50	0.00052	0.0012
0.9	[0.05, 0.1]	0.99	[0.05, 0.1]	0.009	[0.05, 0.5]	[0.0095, 0.0098]	50	0.00052	0.0015
0.9	[0.05, 0.1]	0.99	[0.05, 0.5]	0.009	[0.05, 0.1]	[0.0095, 0.0098]	50	0.00052	0.0024
0.9	[0.05, 0.5]	0.99	[0.05, 0.1]	0.009	[0.05, 0.1]	[0.0095, 0.0098]	50	0.00052	0.011

The examples here are meant to be only illustrative, so it would be wrong to draw strong conclusions from the particular numerical results above – there is no attempt to make these numbers typical of real systems. Rather the intention is to show how the modelling might provide a framework for negotiation. It does this by allowing the discussion to take place about the *components* of the parties' arguments that support their different claims about perfection. It seems that, of the 7 parameters in total in the model, it is likely that the parties' differences will centre upon 4 of these parameters: the 3 parameters representing *confidence*, i.e. $\gamma_{\theta 1}$, $\gamma_{\theta 2}$, γ_r, together with r_u.

5.4. Chapter summary

The model have been presented here allows evidence of the efficacy of development process(es) to be taken into account when assessing the possibility of perfection of a software-based system. Informal arguments in support of system dependability have long used this kind of evidence: "we have in place a wealth of experience and good processes, as evidenced by the many similar systems we have built in the past that have experienced lots of operational exposure without failure – this makes us confident that *this* system will operate with high dependability." Such claims about track records are, of course, generally attractive: e.g. we are confident flying in a new aircraft type built by Boeing or Airbus, because we have seen the excellent safety record of previous types. The work reported here is an attempt to put this kind of argument onto a formal basis. In particular, it allows *quantitative claims* to be made – in this case, about possible perfection – and it does this in ways that are *guaranteed to be conservative*.

The modelling here depends upon the reasonableness of notions of "similarity" between different products, and the way that these are represented mathematically. The claim of a "common development process" could be supported by seeing that the methods, tools or principles used in key activities (i.e. design, V&V and etc.) are similar, that the same prescriptive standards are used, and so on. Justification of such assumptions of similarity in particular cases is, of course, outside the direct scope of this study, and will need to be made on rather informal grounds by the assessor of the system in question. There is no intention to claim "complete similarity" in all aspects of process and problem: we recognize that differences will be present, and these will result in differences in the resulting products. They will not have identical dependability, for example. In particular, they will not all be perfect (or not perfect). The model takes account of this residual variation between products, indeed this is what drives the uncertainty about the top level claim about perfection.

In this section, some of the issues that arise in using the model to make claims about perfection of real systems will be discussed. There are, as the attentive reader will have noticed, some difficult problems that an expert assessor will face in using the model. They will be discussed, and ways forward to address the issues are proposed.

5.4.1. What is R, and how could experts express beliefs about it?

The general model is complex, and at its heart lies a distribution for *pfd* which has been called f_p: see the middle level of Figure 9. Specifying this distribution is difficult. As argued earlier, adopting some parametric family for f_p and then incorporating its unknown parameter values into the Bayesian analysis does not seem a plausible way forward. It would be hard to justify a particular choice of family; given such a family, it would be unreasonable to expect an expert to be able to express even partial beliefs about its parameters.

The result of section 5.3.1 is a way round these difficulties. The key result here shows that knowledge of the complete (non-zero) shape of f_p does not need to be specified, because knowledge of R is "sufficient" (with θ_{pp}), in a precise mathematical sense, for statistical inference about perfection. This notion of sufficiency is similar to the idea of "sufficient statistic" in classical statistics.

Whilst the use of R is a considerable simplification, compared with the need to specify a complete distribution f_p, it requires an expert to express (at least partial) prior belies about R. How could he do this?

From section 5.3.1, $R = \int_{0+}^{1}(1-p)^n f(p|\tau)\,dp$. Here $f(p|\tau)$ is the distribution of the *pfd* of a randomly selected software system from the population produced by the common development process for this and similar problems. Note the range of p excludes the perfect versions, so R means *the probability that a randomly selected version from the population produced by the development process is not perfect but passes* n *tests.*

As a shorthand, we shall say that R means "reliable and not perfect"; where by "reliable" we mean "passes n tests without failure". Note the presence of n in this: in fact in this treatment of the model here, n only appears in the Bayesian analysis via an expert's belief(s) about R.

Note also that R, like the probability of perfection θ_{PP}, is an *objective property* of the population software systems: it is an unknown "in-the-world" parameter. There will therefore be epistemic uncertainty for assessors about these two parameters – in fact this is the only epistemic uncertainty in the model now. The task of the assessor is to express his (limited, partial) beliefs, for example in constraints like (5.15). How should he go about this? – this task seems harder for R than the similar task he faces concerning θ_{PP}.

In the shorthand terminology we have:

$$R = Pr(\text{imperfect and reliable}) = Pr(\text{imperfect})Pr(\text{reliable}|\text{imperfect}) \qquad (5.23)$$

$Pr(\text{imperfect})$ here is just $1-\theta_{PP}$; The assessor is assumed to be able to express beliefs about θ_{PP}. More difficult is the second factor on the right hand side of (5.23), which is a conditional probability. It is well-known people find it hard to assess a conditional probability – in fact it is essentially the same, and thus as difficult, as assessing bivariate uncertainty.

Some help may come here from noting that $Pr(\text{reliable}|\text{imperfect}) \leq Pr(\text{reliable})$ from which it follows that $R \leq (1-\theta_{PP})Pr(\text{reliable})$ and the problem reduces to expressing beliefs about a product of unconditional probabilities, with all the difficulties related to epistemic dependence between them.

Clearly this remains a difficult problem, though, and is an obvious candidate for further thought.

5.4.2. Extent of process evidence: what values of n and k are feasible?

It is trivially true that the more versions and the more failure-free tests, the better. However, the numerical results here, whilst only intended to be illustrative rather than realistic, suggest – not surprisingly – that k seems more important than n. Massive operational exposure from only a very few previous products tells us less about the population of products than modest exposure of many products.

Unfortunately, the value of k is unlikely to be controllable. It is infeasible to build multiple versions just to evaluate the efficacy of a development process. Instead, users of a model like this must rely upon the evidence that is available. So, for example, within a company that has built several generations of safety protection systems, there may be evidence of their reliability in operational use that could be used for our model. Values of k in such cases are likely to be modest, however: they are unlikely to be as large as the value k= 50 in our earlier tables, which gave the most useful results.

One possible way out of this difficulty may be that the model is applicable to a wider class of problems. It has been described everything here in terms of *multiple similar products* developed using the same process (to tackle similar problems). In fact it seems the same model can be applied to the case where a *single product* is used in *multiple similar* (but different) *environments*. For example, some safety protection systems used in the nuclear industry for reactor shut-down have also been used extensively for emergency shut-down in dangerous process industries. If claims for "similarity" here can be supported – and this may be so in the case of a rather simple shut-down function – then k may be quite large. Thus the model here may be able to be used to support claims for perfection of a software-based system in a nuclear application by using evidence from the use of that system in other industries. However, a limitation needs to be noted here. As argued in chapter 1, there are two reasons for the interest in possible perfection: firstly that it is of value in its own right for making claims about lifetime reliability; secondly that it overcomes a basic hurdle in making reliability claims for fault-tolerant 1-out-of-2 systems via the result from original LR and LP models. This new interpretation of similarity fits well for the first use of probability of perfection, while whether it fits into the latter scenario or not needs to be double checked. Because some preliminary work has shown the original LR and LP models is invalid for the new interpretation.

There is a practical restriction on the allowable values of n: all the results assume that n takes the same value over all k systems observed. Clearly, this is a serious issue because it is unlikely to be true in practice. It arises because the parameter R involves a particular n. It would be infeasible to elicit beliefs about k different R_i corresponding to the different values of n_i for the k different products. A work-around is to conservatively choose the minimum n_i to be the n in our equations. Of course, this solution may be very conservative if the minimum n_i is much smaller than other n_i. In this case, we could simply ignore the evidence from this version, and carry out the model calculations using the next smallest n_i. The price, of course, is a reduction of k to $k-1$.

5.4.3. Why conservative Bayesian reasoning?

In this and other recent work on systems dependability we have adopted a Bayesian approach to the treatment of uncertainty. There is a growing consensus that this is the right way forward, but it presents some problems for potential users. Most notable is the difficulty of eliciting prior beliefs from experts who “own the problem” – in the terminology of this thesis, regulators and licensees – in the kinds of complex situations we are dealing with.

In the Bayesian literature, various approaches have been proposed to address this problem: e.g. empirical Bayes, non-informative priors. For many application domains, it is possible to obtain very extensive data, and the differences between such approaches and “proper” Bayes, involving informative priors, disappear. Unfortunately this is a luxury we do not have for the kinds of safety-critical applications we are considering. Indeed, in software engineering generally, large sample sizes are rare: e.g. random samples of *programs* are unknown, except in specialized small-scale experiments such as those described in (Knight and Leveson 1986; Eckhardt, Caglayan et al. 1991).

The method adopted in the face of these problems is novel. It allows the expert to populate the model with minimal partial prior beliefs. The procedure then is to obtain results for the claims of interest – in this case, confidence bounds for probability of perfection – that are guaranteed to be conservative. The trick here is to allow the user's prior beliefs to be as minimal as possible to aid his task, whilst not being *too* minimal to preclude useful results. Examples of prior beliefs that *are* too minimal to be useful have been shown, and also the prior beliefs that were only *slightly* more informative (i.e. that impose upon the assessor a task that is only *slightly* more onerous) could give useful results.

Whilst the model here is a complex one, here presented an example of how it might be used to inform the negotiation between a regulator and a licensee about a claim in part of a safety case. It tentatively shows the potential usefulness of the model here.

5.4.4. The counter-intuitive results

Here is a brief comment on the counter-intuitive results. It could be treated as a warning against placing too much trust in informal reasoning when dealing with these quite complex models for dependability. It has been observed here that "obviously good news" from operational testing of systems could – counter-intuitively – decrease probability of perfection.

In (Littlewood, Popov et al. 2000b), a similar counter-intuitive result was reported which obtained for the probability of system failure of a 2-channel system for which the channel *pfds* are assumed known with certainty. That paper started with distributions with mass concentrated in a few points, thus reducing the degree of freedom for the inference. These restricted priors may have unexpected consequences.

Another similar counter-intuitive result was found in (Littlewood and Wright 2007). There, a two-legged argument was used to support claims for the *pfd* of a system. The two legs used were, respectively, evidence of failure-free working on test (involving a possible fallible test oracle), and evidence of proof of correctness against a (possibly incorrect) formal specification. The counter-intuitive result there was that *more* failure-free working could, in certain circumstances; result in *lower confidence* in a small *pfd.*

In all these cases, of course, further analysis showed that the interpretation from the formal modelling was correct – i.e. eventually, the counter-intuitive results are reasonably explained in both formal and informal ways. Such examples are warnings against the use of unaided informal engineering judgment.

6. CLAIMS ON PROBABILITY OF PERFECTION VIA FORMAL VERIFICATION EVIDENCE

In previous chapters, testing evidence of the software of interest and its similar products (as process evidence) is modelled to support claims on probability of perfection, and it turns out that what can be claimed is quite modest if without any further information (e.g. less minimal partial priors and the "pruning idea" in Chapter 4). This is due to the nature of statistical testing evidence which can only show the presence of desired features, not the absence of bugs. Therefore, when based on some prior beliefs in perfection, the confidence in perfection could only be built very slowly when collecting more and more failure-free running behaviour evidence. It is appealing to think that formal verification provides a mathematical proof that a given system satisfies its specification, i.e. it exhibits all the desired properties, thus providing straight forward evidence to support perfection claims. But in reality, can we fully specify all the properties in terms of the "perfection" we would like to claim? Can we fully trust the prover? Due to the inevitable uncertainties hidden in a formal verification process, it seems we cannot naively believe our *proved* program is certainly perfect, rather than have some posterior confidence in its perfection.

In this chapter, the sources of uncertainties are analysed and undesired events that could ever happen in formal verification process are modelled to check how a successful formal proof can support the claims on probability of perfection.

6.1. An overview of formal verification

In the context of software-based systems, formal verification is used to prove the correctness of intended programs with respect to a certain formal specification, using formal methods of mathematics. Although in principle it can be carried out entirely by a human, in today's practice formal verification is usually conducted via mechanised automatic or semi-automatic techniques. Despite of the heated debate on the role of formal proof in earlier days (Fetzer 1988; De Millo, Lipton et al. 1979), it is increasingly used in real life projects nowadays. For instance, the control software for deep space probes was formally verified by NASA (Havelund, Lowry et al. 2001); and the verifying of protocols for medical procedures in (Hommersom, Groot et al. 2007).

Generally when we say formal verification, it can be approached from two extremes (Amjad 2004):

- The state-based approach – model checking, which is to exhaustively examine all possible states the system can ever be in. In each possible state, the desired properties (expressed in some assertion language, e.g. temporal logics) of the system should hold to pass the verification.
- The proof-based approach – theorem proving, in which the desired properties are derived as theorems in some formal mathematical logic. Then theorem provers are used to prove or reject the theorems.

The well-known drawback of model checking approach is the inability to deal with very large number of possible states, i.e. the "state explosion problem". While for the theorem proving, the required properties are expressively represented by formal logic as theorems, so there is no need to check each and every possible state. Even though the subtle difference between these two approaches is out of the scope of this thesis, the source of uncertainties may differ in terms of the modelling on claiming perfection. For instance, at least but not last, the "coverage rate" of the entire state space is an important source of aleatory uncertainty when applying model checking, while it does not exist (or certainly equals to 1) for the theorem proving approach. In this chapter, the modelling on formal verification mainly refers to the theorem proving approach, leaving model checking as future work.

Normally, in practice we wish to (or can only) prove a subset of the behaviour (i.e. some properties). So we would claim "freedom from critical defects" not "all defects", and put effort into defining "critical". Whilst in this chapter, the claim is about perfection which is defined in terms of failure free behaviours (in line with the one used in LR-LP models). So strictly speaking the claim here is "freedom from the defects causing any failure", and a failure is defined by the relations between the spaces of inputs and of outputs: i.e. any mapping outside the "correct-subset" of their Cartesian product which you will see detailed explanation later.

6.2. Source of uncertainty in formal verification

6.2.1. Proof cannot certainly prove perfection

Seeing a program passed formal verification, can we assert the software is certainly perfect for the specified use?[18]

The proofs of clock synchronisation and other properties of fault tolerant algorithms were originally done in (Melliar-Smith and Schwartz 1982) and subsequently found to contain errors, even though it had been peer reviewed before publication. This reminded us that human reasoning is quite flawed. Sometimes even the most carefully-crafted mathematical proofs are erroneous. That's why we would like to have computers to do the heavy mathematical task for us. And the use of machine in proofs indeed can increase the accuracy, e.g. in the example above (Melliar-Smith and Schwartz 1982), the errors in the proofs were found by automated theorem prover PVS when reformulated and machine checked. But note that any automated prover is merely another piece of software, i.e. another piece of artificial

[18] The answer is obviously no to any software engineers at first glance. Because if operating systems or compilers were used, the correctness of them would also have to be verified. But the scope of this thesis is about the perfection of "source code", so the uncertainties from compilers and operating systems etc. would not be discussed here.

design. The proof by machines could not be claimed certainly correct due to the reliability of the prover which is an important source of doubt in formal verification.

Besides, the outcome of a formal proof is really a statement that the formal specification and the program are equivalent or not. As the nature of formal proof, we can't prove more than the conformance of a program to a formal specification, and we have no guarantee that the formal specification is what it should be[19]. In the Darts diversity experiment (Smith, Wall et al. 1991), one channel was specified formally. An error – a sign inversion in one of the equations – persisted into test phase despite careful review. Even though engineers could do intensive reviews and/or testing on formal specification (Kemmerer 1985), there is still an uncertainty about the correctness of the formal specification. Again, this is due to the essence that any formal specification is a piece of artificial design (via some specification languages).

Here is not intended to exhaustively review of problems with formal proof in software assurance, rather to show the importance to consider the probability that the formal proof results might be misleading in claiming perfection of software. If we are to have high confidence in formal proof then the sources of doubt, and how we have addressed them, must form part of the assurance case.

To sum up, there are two inevitable sources of uncertainty ("out in the world", i.e. aleatory ones) in a formal proof:

- The proof process itself might be inaccurate. In the cases of using machine provers, this uncertainty lies in the reliability of the proof tools.
- The formal specification might not be what it should be. Developing a formal specification could be treated as writing another program using some specification language, therefore whether it is valid or not is uncertain.

[19] As the engineering specification could be incorrect, this does not mean the conformance to any particular engineering specification. In this chapter, the correctness of the engineering specification would not be discussed, so here assumes that the formal specification is built on some perfect specification in the engineers' "mind", i.e. what it should be rather than any specific engineering specification.

Then the next step is to find proper parameters to capture them. Without any loss of generality, an illustrative example using Frama-C (Cuoq, Kirchner et al. 2012) will be used to facilitate the justification of the parameters capturing the two sources of uncertainty, which is shown in Figure 19.

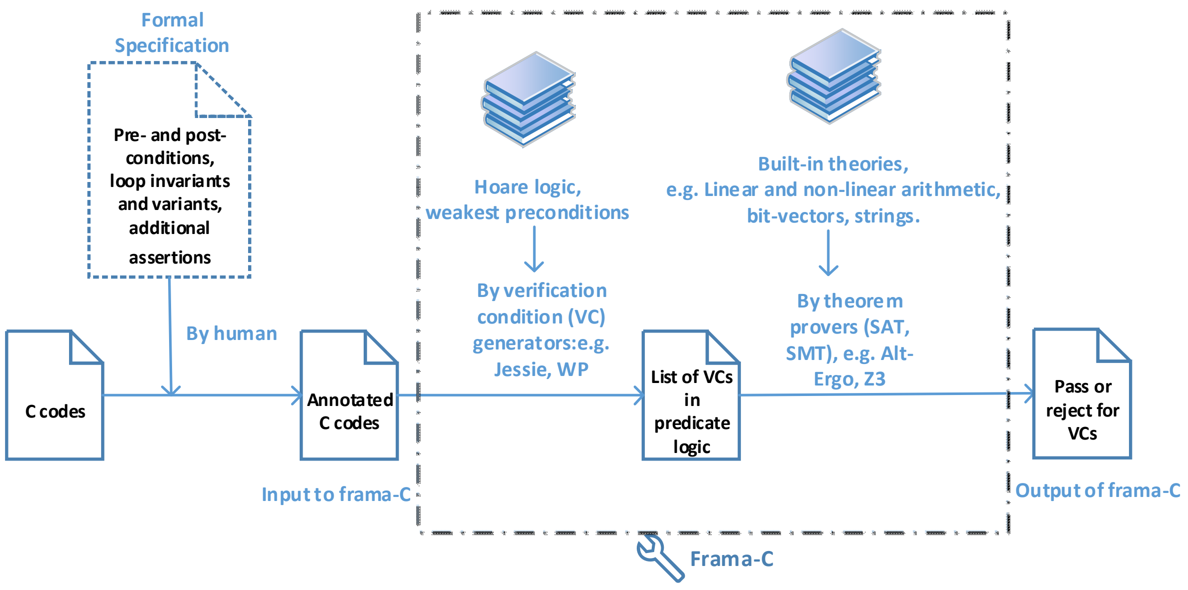
Formal Specification
Pre- and post-conditions, loop invariants and variants, additional assertions
By human
C codes
Annotated C codes
Input to frama-C
Hoare logic, weakest preconditions
By verification condition (VC) generators:e.g. Jessie, WP
List of VCs in predicate logic
Built-in theories, e.g. Linear and non-linear arithmetic, bit-vectors, strings.
By theorem provers (SAT, SMT), e.g. Alt-Ergo, Z3
Pass or reject for VCs
Output of frama-C
Frama-C

Figure 19 a schematic view of theorem proving via Frama-C

6.2.2. Capture the uncertainty in provers

As shown in Figure 19, Frama-C as a platform integrates several plug-ins to perform a formal proof. It takes annotated C codes (source code and formal specification) as input, and then translated it into verification conditions (VCs) by plug-ins e.g. Jessie and WP. The underlying theory is Hoare logic and weakest precondition calculus, which is a set of rigorous logical rules to translate annotated source codes to some statements of predicate logic, i.e. VCs. Then by theorem provers (e.g. Alt-Ergo), the truth of the VCs would be proved or rejected.

The serval plug-ins in Frama-C could fail in various ways, due to the inherent limitations of their fundamental theories or merely a bug in them. But if we treat Frama-C as a "black box", it could only fail in two ways due to its nature as a verifier – fail to prove true VCs (i.e. false alarm) and prove false VCs (i.e. fail to alarm).

We normally use *pfd* (e.g. for nuclear protection system) or probability of failure per hour (e.g. for flight-control avionics systems) as an "indicator" of reliability. The choice of "indicators" of reliability is normally based their feature, e.g. the *pfd* for demand-based systems which is working in the background, but not doing anything until a safety limitation is exceeded.

While in the case of the software is a verifier, two failures rates – the *false alarm rate*, say R_{t1} and *fail to alarm rate*, say R_{t2}– are believed to be sufficient to capture its reliability, therefore captures the aleatory uncertainty in the proof process itself of a prover.

6.2.3. Capture the uncertainty in formal specification

Developing a formal specification – writing function contracts with pre- and post-conditions, loop invariants and variants, and additional assertions using some specification language (e.g. ACSL in the Frama-C case) – could be as tedious as developing a program. So we could treat the formal specification as a "special" piece of program that aiming at rejecting imperfect software and passing the perfect software.

Geometry definition of perfect software

Essentially both programs and (formal/informal) specifications describe *relations* between the spaces of inputs and of outputs: i.e. a subset of their /..> (Popov and Strigini 2010). This relation could be shown in a "geometry" manner, citing the same example in (Popov and Strigini 2010) as Figure 20. It is an engineering specification that requires a program to calculate *y=sin(x)*, for *x* in the interval [0, 2], with a maximum error of 5% and satisfying the condition $sin(x) \leq 1$, i.e. the "correct-area" in Figure 20.

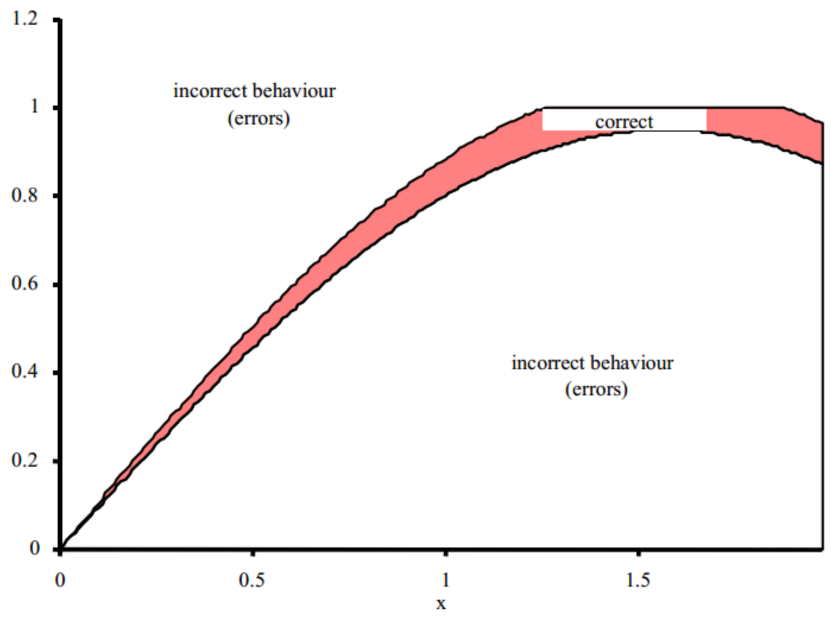

Figure 20 specification of a single-input, memoryless program as a subset of the Cartesian plane

Then essentially any program calculating an output *y* for an input *x* could also be represented by a "SW-area" in the figure. Then we can define the perfect programs for this particular engineering problem as the ones with a "SW-area" within the "correct-area", meaning the software {input, output} pairs is a sub-set of "correct" {input, output} pairs.

For the purpose of better illustration, more abstract visualized examples could be used to show the various possible programs to a particular engineering specification, as Figure 21.

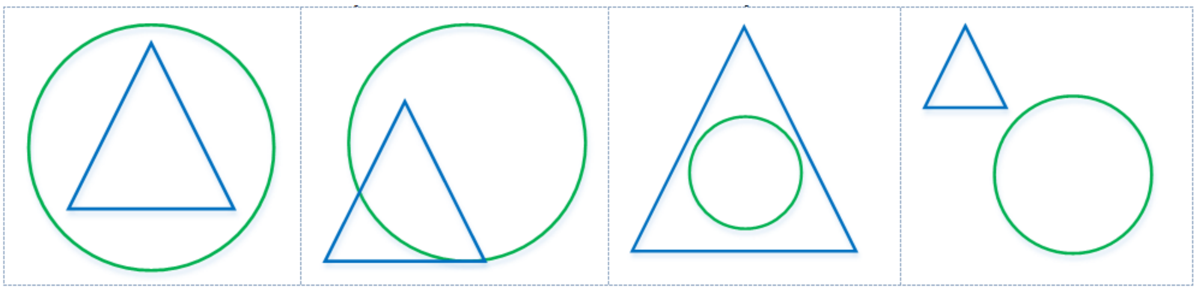

Figure 21 4 possible programs to a particular engineering problem. The green circle is the "correct-area" representing the engineering problem to solve, the blue triangle is the "SW-area" representing a program.

The first case in Figure 21 is a perfect program, as all "SW-area" is within the "correct-area". While, both of the second and third cases have some parts of the "SW-area" outside the "correct-area", meaning for some input, the {input, output} pair violates the relation defined by the engineering specification, thus they are imperfect programs. The fourth case represents totally irrelevant software which would not exist in reality (as any single test will refuse it), so this case would not be considered here.

Definition of the completeness and correctness of formal specifications

In the similar way above, we can check the logical completeness of a formal specification with respect to the engineering problem. That is if we were using a "FS-area" to represent a formal specification, then the ones with a "FS-area" within the "correct-area" are logically complete, thus defined as *complete* formal specification. Otherwise it is a *partial* formal specification.

Being complete is a *necessary* condition of being correct. We cannot simply define the *correctness* of a formal specification only in terms of its completeness. The aim of a formal specification is not to *solve* an engineering problem, rather to *verify* a program. A complete formal specification could reject a perfect program (due to being too restrictive), and such a complete formal specification is clearly not acceptable. So a correct formal specification should (3 criteria):

- never reject a perfect software;
- never pass an imperfect software; and
- complete to the engineering problem.

Different types of faulty formal specifications

With the certain definition of the correctness of a formal specification, we can find a possible type of a formal specification on the spectrum in Figure 22. Basically there are 3 "changeable factors": whether the software is perfect or not, whether the formal specification is complete or not and should (i.e. with a perfect prover) the software be proved by the formal specification or not. Each branch on that spectrum is explained with more illustrative examples in following figures.

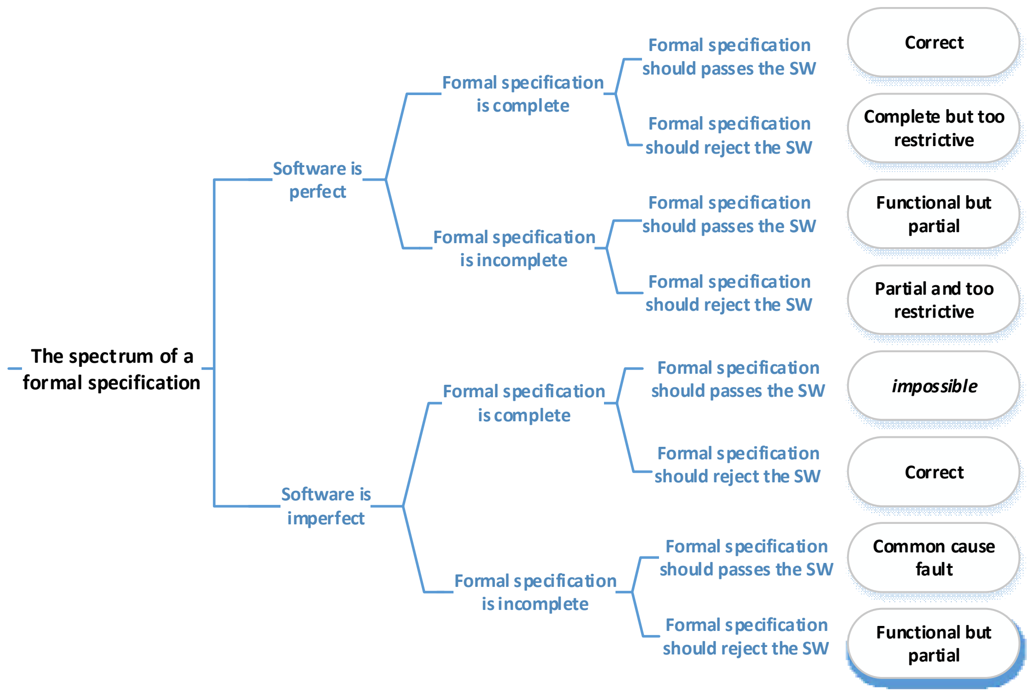
The spectrum of a formal specification
Software is perfect
Formal specification is complete
Formal specification should passes the SW
Correct
Formal specification should reject the SW
Complete but too restrictive
Formal specification is incomplete
Formal specification should passes the SW
Functional but partial
Formal specification should reject the SW
Partial and too restrictive
Software is imperfect
Formal specification is complete
Formal specification should passes the SW
impossible
Formal specification should reject the SW
Correct
Formal specification is incomplete
Formal specification should passes the SW
Common cause fault
Formal specification should reject the SW
Functional but partial

Figure 22 the 5 cases on the spectrum of a formal specification

The 3 cases in Figure 23[20] are "complete but too restrictive" formal specifications that only violate the 1st criteria of correctness. As the name, this kind of faulty formal specifications is complete but too restrictive for a correct program. More specifically, "too restrictive" might

[20] The meaning of the colours and shapes used in this figure and the ones later in this chapter are same.

due to too strong post-conditions and assertions, or too week pre-conditions. For instance, a formal speciation specifying a maximum 1% error (as a post-condition) in the example of Figure 20 is obviously too restrictive to pass a program with a maximum 3% error. Or when the formal specification is requiring an input interval [0, 4] (which is wider than the original [0, 2]) as a precondition, the correct program dealing input interval [0, 2] will be rejected due to failing to fulfil the task in (2, 4].

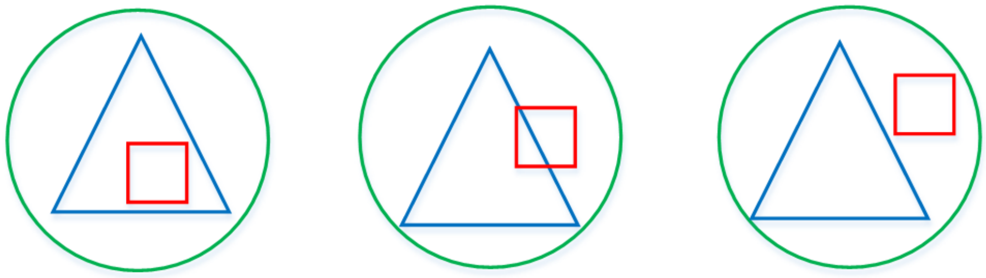

Figure 23 "complete but too restrictive" formal specifications that only violate the 1st criteria. The green circle is the "correct-area" representing the engineering problem, the blue triangle is the "SW-area" representing a program and the red square is the "FS-area" representing a formal specification.

Figure 24 shows "partial and too restrictive" formal specifications that violate the 1st and 3rd criteria of correctness. The name of this type of faulty formal specifications sounds contradictory at first glance, but it exists due to the fact that a specification normally has serval properties. Use the example of Figure 20 again, if the formal specification missed the constraint *sin(x)≤1*, and at the same time specifying a maximum 3% error of the output, then it is "partial and too restrictive" to any correct program of the problem.

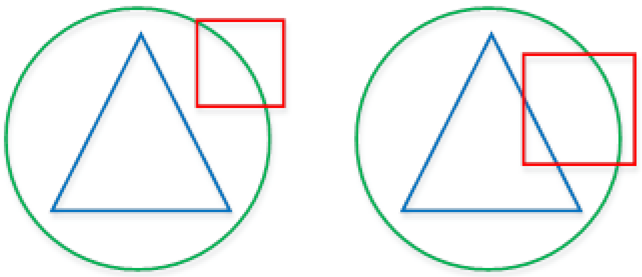

Figure 24 "partial and too restrictive" formal specifications that violate the 1st and 3rd criteria

When the software is perfect, there is a kind of partial formal specification would pass it, as shown in Figure 25. Even though the formal specification is functional (in the sense of passing a perfect program), it is faulty due to the incompleteness.

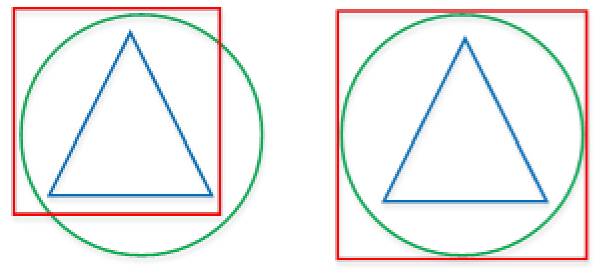

Figure 25 "functional but partial" formal specifications that only violate the 3rd criteria

By now, the 3 types of faulty specifications above are all associated with perfect programs to a particular problem. In Figure 26, the formal specifications are proving imperfect software. The key feature of this type of faulty formal specification is that the "incorrect part" in the "SW-area" is all within the "FS-area". Informally, this means the software and the formal specification is having same faults, and passed the imperfect software. For instance, both the program and formal specification missed the constraint *sin(x)≤1* somehow in the example of Figure 20. This undesirable situation happens due to some "common cause" reasons, e.g. lacking of understanding of the problem, ambiguity in project's documents.

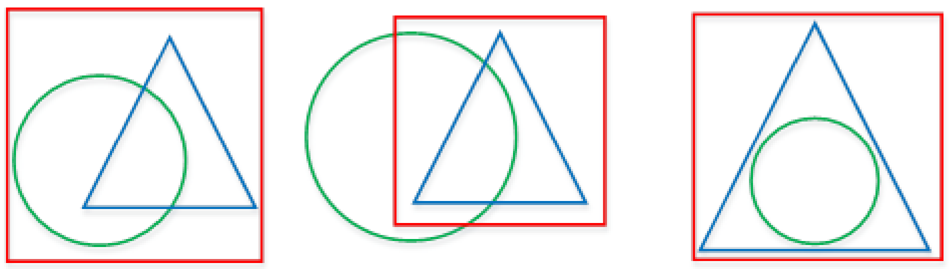

Figure 26 "common cause fault" formal specifications that violate the 2st and 3rd criteria

There is another type of "functional but partial" formal specification in the sense of rejecting an imperfect program, as shown in Figure 27.

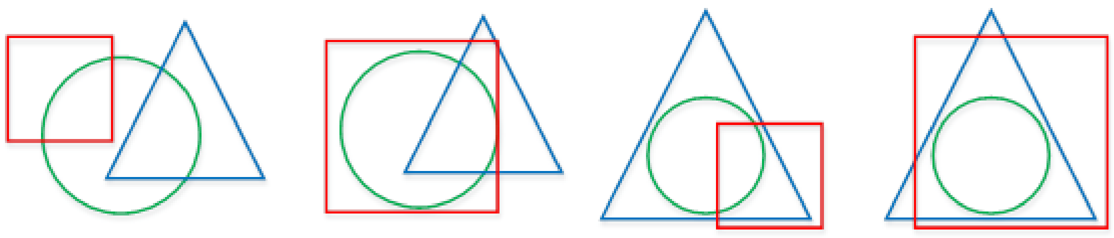

Figure 27 "functional but partial" formal specification that only violate the 3rd criteria

Till now, we *exhaustively* enumerate all the possible branches in Figure 22, which is a *spectrum* of all possible associations[21] between a program and a formal specification for a *particular* engineering problem. There are 3 binary variables, so theoretically there should be $2^3 = 8$ branches. After excluding the impossible and overlapping cases, 5 cases are left in Figure 22. From the perspective of the formal specification, the real-life case would fall into one of the 5 categories on the spectrum of formal specifications:

- Complete but too restrictive;
- Partial and too restrictive;
- Common cause fault;
- Functional but partial;
- Correct;

To capture all the 5 cases on that spectrum, we need three objective parameters (aleatory in the world):

- $isperfect_{SW}$: For the particular software we are verifying, it is objectively perfect (to the particular engineering problem) or not. Same as section 3.1.2, this aleatory uncertainty about the program's perfection is captured by a single indicator parameter. $isperfect_{SW} = 1$ means the software is perfect, otherwise imperfect.
- $iscomplete_{FS}$: Similar, for the particular formal specification we are to use, it is objectively complete or not (to the particular engineering problem by the definition earlier in this section). A single indicator parameter is used to capture this aleatory uncertainty on the completeness. $iscomplete_{FS} = 1$ means the formal specification is complete, otherwise partial.
- $iscompliant_{AC}$: When assembling the program and formal specification into a piece of annotated codes, it is objectively compliant or not. Compliance means, *in principle* (or with a perfect prover), the program should be proved by the formal specification. For example in Figure 26, the "SW-area" is all within the "FS-area", meaning the set of software {input, output} pairs is a sub-set of the formal specification {input, output} pairs set, therefore the piece of annotated codes of them is compliant and would be certainly proved by a perfect prover. This uncertainty of compliance is *objectively* in

[21] Theoretically, there are some cases missed which are believed to be unrealistic in practice, e.g. totally irrelevant relationships among formal specification, program and the engineering problem.

the world once we assembled a program and a formal specification, thus a single indicator parameter is used to capture it. $iscompliant_{AC}$ =1 means the piece of annotated codes is compliant, otherwise incompliant.

It might be useful to explain more about why we are interested in the parameter $iscompliant_{AC}$ and how it works in terms of facilitating the modelling here. To do a Bayesian inference, basically we need to divide several sub-conditions and find out the likelihood function in each sub-condition. The more detailed of the sub-conditions, the harder to elicit the priors of them, but it is easier to write down the likelihood functions. So there seems always the tension between "many parameters used so that easy to write the likelihood functions" and "few parameters used so that easy to elicit priors". In the case of this chapter, using only two parameters $isperfect_{SW}$ and $iscomplete_{FS}$ is impossible to write down the likelihood function. Thus we introduce one more, i.e. $iscompliant_{AC}$, which facilitates the writing of the likelihood function and still makes the elicitation task feasible. However, it would be interesting to explore more other parameters in future to see how various factors affect our claims.

By using the three objective indicator parameters $isperfect_{SW}$, $iscomplete_{FS}$ and $iscompliant_{AC}$, all of the 5 possible cases on that spectrum could be represented. For instance, the event that we are having a "common cause fault" specification is simply a collection of sub-events that $isperfect_{SW}=0$, $iscomplete_{FS}=0$ and $iscompliant_{AC}=1$.

Together with the two objective parameters capturing the uncertainties in the prover, i.e. the false alarm rate R_{t1} and fail to alarm rate R_{t2}, we have 5 objective parameters now to capture the sources of uncertainty in a formal verification.

6.3. A preliminary model

By identifying the 5 parameters capturing uncertainties, some events of interest in a formal verification could be modelled. While in this section, the interest is upon how much confidence can be claimed on the perfection of the particular software when seeing it passed the formal proof using a formal specification, i.e. the posterior subjective probability of perfection of a single program when seeing good formal proof evidence.

6.3.1. The basic set-up

First recap the meaning of defined parameters, and denote their expectations due to the epistemic uncertainties of them.

- For a prover tool (i.e. Frama-C), R_{t1} is an objective parameter represents its false alarm rate. It ranges [0,1] and the assessor cannot know its value for certain, while may have a subjective expectation of its value (e.g. from some benchmark testing of the tools (Shiraishi, Mohan et al. 2015)), say $E(R_{t1}) = t_1$.
- Similarly, R_{t2} is an objective parameter represents the fail to alarm rate of the prover. It ranges [0,1] and the assessor cannot know its value for certain, rather have a subjective expectation of its value, say $E(R_{t2}) = t_2$.
- The parameter $isperfect_{SW} = 1$ represents the event that the software of interest is perfect to the particular engineering problem. The assessor cannot know its perfection for sure, rather have a subjective expectation (i.e. prior belief when no evidence), say $E(isperfect_{SW}) = \theta$.
- For the particular formal specification being used, $iscomplete_{FS} = 1$ represents the event that the formal specification is complete to the particular engineering problem. Again, assessors cannot know it for certain due to the epistemic uncertainty. While he may have subjective beliefs on it, say $E(iscomplete_{FS}) = \lambda$.
- $iscompliant_{AC}$ is about the compliance of a piece of annotated codes. Despite of the practicality of eliciting subjective belief on this parameter (which will be discussed later), assume the assessor had an expectation, say $E(iscompliant_{AC}) = \varphi$.

6.3.2. The events of interest

Our interest is the posterior subjective probability of perfection of the single program when seeing good formal proof evidence, via Bayesian theorem (denoting the event "formal proved evidence" as *e=1* for better illustration.):

$$\begin{aligned} &Pr(isperfect_{SW} = 1 | \text{formal proved evidence}) \\ &= \frac{Pr(isperfect_{SW} = 1, \text{formal proved evidence})}{Pr(\text{formal proved evidence})} \\ &= \frac{Pr(isperfect_{SW} = 1, e = 1)}{Pr(e = 1)} \end{aligned} \tag{6.1}$$

So basically we are interested in two events, i.e. the ones in the denominator and numerator of the right-hand in (6.1) respectively.

The *conditional* event in the denominator, given the random variables (T_1, T_2) and indicator variables $(I_\varphi, I_{SW}, I_{FS})$ to represent the actual value of the 5 objective parameters:

$$Pr\left(e = 1 \middle| R_{t1} = T_1, R_{t2} = T_2, iscompliant_{AC} = I_\varphi, isperfect_{SW} = I_{SW}, iscomplete_{FS} = I_{FS}\right)$$
$$= \begin{cases} 1 - T_1, & when\ I_\varphi = 1 \\ T_2, & when\ I_\varphi = 0 \end{cases} = (1 - T_1)I_\varphi + T_2(1 - I_\varphi) = (1 - T_1 - T_2)I_\varphi + T_2 \tag{6.2}$$

The result means that if the input of the formal verifier (i.e. annotated codes) is objectively compliance (i.e. I_φ=1), then the probability of the prover passed the input is $1 - T_1$ (i.e. the probability that it *did not* make the *false alarm* mistake). Otherwise when the input is not compliance (i.e. I_φ=0), then the probability of the prover passed the input is T_2 (i.e. the probability that it make the *fail to alarm* mistake).

The result (6.2) is based on the fact that given $R_{t1} = T_1, R_{t2} = T_2, iscompliant_{AC} = I_\varphi$, the proof result will be independent with the events of the perfection of the program and completeness of the formal specification:

$$Pr\left(e = 1 \middle| R_{t1} = T_1, R_{t2} = T_2, iscompliant_{AC} = I_\varphi, isperfect_{SW} = I_{SW}, iscomplete_{FS} = I_{FS}\right)$$
$$= Pr\left(e = 1 \middle| R_{t1} = T_1, R_{t2} = T_2, iscompliant_{AC} = I_\varphi\right) = (1 - T_1 - T_2)I_\varphi + T_2$$

Of course, the objective parameters in the real world will not be known with certainty. Ideally, if an assessor would describe his epistemic uncertainty about these unknowns in a joint 3 dimensional distribution, say $f(T_1, T_2, I_\varphi)$, then the *unconditional* event of seeing formal proved evidence is:

$$\begin{aligned} Pr(e = 1) &= \iiint_0^1 Pr\left(e = 1 \middle| T_1, T_2, I_\varphi\right) f\left(T_1, T_2, I_\varphi\right) dT_1 dT_2 dI_\varphi \\ &= E_{T_1, T_2, I_\varphi}\left((1 - T_1 - T_2)I_\varphi + T_2\right) \end{aligned} \tag{6.3}$$

By imposing an assumption that the assessor's subjective belief in the two types of failure rates of the prover should be independent with the compliance of the annotated codes (i.e. the two types of failure rates of the prover have nothing to do with the type of a *particular* input), we get a simplified result (see Appendix D for proof):

$$Pr(e = 1) = (1 - t_1 - t_2) \times \varphi + t_2 \tag{6.4}$$

It is worth to elaborate the two independent assumptions used above.

- At the aleatory level, i.e. (6.2). *Given* the two types of failure rate of the prover (i.e. T_1, T_2) and whether the annotated code is compliance or not (I_φ), the proof result will be independent with the events of the perfection of the program and completeness of the formal specification. For example, the behaviour of an security alarm system will only depends on its own quality (the two failure rates T_1, T_2) and whether the passenger is taking a gun or not (I_φ). For the factors determine the behaviour of the passenger are no longer relevant when *given* the 3 direct factors listed above.
- At the epistemic level, i.e. (6.4). I_φ is the random variable represents a property of a *particular* input. We are not sure about its value, therefore having a subjective expectation of it as φ. Similarly t_1 and t_2 represent the subjective beliefs of the assessors in the two types of failure rates of the prover. My subjective belief about the property of the next input should have nothing to do with my expectation of the quality of the prover. Of course we my say that the more we believe in *this* particular input (i.e. the annotated code) is compliant, the more I believe the prover will give a false alarm on it, which represents the dependence of the event that the input is compliant and the event that there is a false alarm. While the independence assumption used in (6.4) are not about those *two events*, rather about the event of compliance and the unknown false alarm rate (which represents the quality of the prover). Therefore, I believe that T_1 and T_2 are independent with I_φ. By the security alarm system again, my belief in the next passenger is not a terrorist should not be affected by my subjective belief in the quality of the alarm system.

Similarly, the *conditional* event in the numerator, given some random variables to represent the actual value of the 5 objective parameters:

$$Pr\left(isperfect_{SW} = 1, e = 1 \middle| R_{t1} = T_1, R_{t2} = T_2, iscompliant_{AC} = I_\varphi, isperfect_{SW} = I_{SW}, iscomplete_{FS} = I_{FS}\right)$$

$$= \begin{cases} 0, & when\, I_{SW} = 0 \\ 1 - T_1, & when\, I_\varphi = 1, I_{SW} = 1 \\ T_2, & when\, I_\varphi = 0, I_{SW} = 1 \end{cases} = I_{SW}\Big((1 - T_1)I_\varphi + T_2(1 - I_\varphi)\Big) = (1 - T_1 - T_2)I_\varphi I_{SW} + T_2 I_{SW} \tag{6.5}$$

Again, the 5 objective parameters in the real world will not be known with certainty. If an assessor would describe his epistemic uncertainty about these unknowns in a joint 5 dimensional distribution, then the *unconditional* events of seeing formal proved evidence and the software is perfect is:

$$\begin{aligned} Pr\left(isperfect_{SW} = 1, e = 1\right) &= E_{T_1,T_2,I_\varphi,I_{SW},I_{FS}}\left((1 - T_1 - T_2)I_\varphi I_{SW} + T_2 I_{SW}\right) \\ &= (1 - t_1 - t_2) \times \left(\theta \times \varphi + Cov\left(I_{SW}, I_\varphi\right)\right) + t_2 \times \theta \end{aligned} \tag{6.6}$$

See Appendix D for proof (based on the same independent assumption for (6.4)).

6.3.3. Conclusions with the preliminary model

Assemble the numerator (6.6) and the denominator (6.4), we can have our interest – the posterior belief in perfection seeing formal proof evidence:

$$\begin{aligned} &Pr(isperfect_{SW} = 1 | e = 1) \\ &= \frac{(1 - t_1 - t_2) \times \left(\theta \times \varphi + Cov\left(I_{SW}, I_\varphi\right)\right) + t_2 \times \theta}{(1 - t_1 - t_2) \times \varphi + t_2} \end{aligned} \tag{6.7}$$

which is a function of the expectations t_1, t_2, θ, φ and a covariance $Cov\left(I_{SW}, I_\varphi\right)$.

- When $\theta = 0$, then $I_{SW} = 0$ (as $E(I_{SW}) = \theta$), then the covariance is 0. So (6.7) = 0, meaning if I have no prior belief in perfection, then after seeing evidence, I still have 0 confidence in perfection.
- When $\theta = 1$, similar as above reasoning, (6.7) = 1, meaning when the software is certainly perfect, then it is still certainly perfect when passed formal proof.
- When $\varphi = 1$, then $I_\varphi = 1$ (as $E\left(I_\varphi\right) = \varphi$), then the covariance is 0. So (6.7) = θ, meaning if the annotated code is certainly compliance, then the formal proof is totally irrelevant in building confidence of perfection.

Based on the result (6.7), some conclusions could be drawn as follows.

The completeness of the formal specification is not directly relevant

The result (6.7) is not a function of λ, i.e. $E(iscomplete_{FS})$ – the subjective expectation of the completeness of the formal specification, which is because the use of the assumption that given the property of compliance, the completeness of formal specification has nothing to do with the proof result. This conforms to the common argument that verifiers may build confidence via using a *partial* formal specification which is the most likely to be encountered for real-life examples (Prevosto 2013). In other words, the confidence in the completeness of the formal specification should not be the determining factor for the posterior confidence in perfection seeing formal proof evidence.

Complete formal specification seems a "double-edged sword". It can reject imperfect software, but also reject more perfect software at the same time (due to being too restrictive). So for the software of interest, we cannot draw a simple line to state how the completeness of the formal specification affects our confidence in perfection, e.g. the more completeness the better. Rather it works in a more mysterious way that affecting the property of compliance of the annotated input in formal proof, and then *indirectly* affects claiming perfection.

Formal proof will not always build confidence in perfection

To make the good formal proof evidence helpful in *increasing* our confidence in the perfection of software, i.e. $Pr(isperfect_{SW} = 1|e = 1) > \theta$, via trivial exercises, we need satisfy:

$$Cov(I_{SW}, I_{\varphi})(1 - t_1 - t_2) > 0 \tag{6.8}$$

in which $(1 - t_1 - t_2)$ is positive[22]. So we need the covariance $Cov(I_{SW}, I_{\varphi})$ to be positive to make the formal proof helpful in building confidence in perfection. $Cov(I_{SW}, I_{\varphi})$ is essentially about the correlation between the perfection of the program and the compliance of the annotated codes. It seems very hard for the assessors to justify the desirable positive correlation.

$$Cov(I_{SW}, I_{\varphi}) > 0 \tag{6.9}$$

This constraint on the value of the random variables I_{SW} and I_{φ} could be mapped to the cases on the spectrum of the uncertain formal specification defined in section 6.2.3, which is that the assessor has to *believe more in the inexistence* of the events:

- $I_{SW} = 0$ and $I_{\varphi} = 1$: this is the event that we are having a "common cause fault" specification which might raise from a same faulty document or the inherent difficulty of the engineering problem..
- $I_{SW} = 1$ and $I_{\varphi} = 0$: this is an union of the two events:

[22] This is a quite practical assumption. Only reliable provers (i.e. the sum of the two types of error rates is smaller than 1) would be used in real-life projects. The case $1 - t_1 - t_2 < 0$ might of interest in some particular circumstances, but will not be discussed here.

- $I_{SW} = 1, I_{\varphi} = 0$ and $I_{FS} = 1$ means we are having a "complete but too restrictive" specification. For instance, we have considered all the properties which could be covered, but make some of them unnecessarily restrictive.
- $I_{SW} = 1, I_{\varphi} = 0$ and $I_{FS} = 0$ means we are having a "partial and too restrictive" specification. In this case, we not only missed some properties, but also make some of the current ones unnecessarily restrictive.

However, it is still too difficult for the assessor to give a *deterministic* answer (i.e. either yes or no) to questions like "do we have a 'common cause fault' formal specification?" More likely, the assessor may express some confidence in the undesirable events associated with formal specifications. For instance, the assessor may say "I have 20% confidence that we are having a 'common cause fault' formal specification". In next section, a new result will be proposed based on this kind of elicitation from assessors.

The punchline here is that the efficiency of formal verification evidence in claiming perfection of software highly depends on the correlation of the perfection of the software and the compliance of the annotated codes. The confidence in perfection would only increase when they are positively correlated, which essentially means some constraints on the possible associations among the problem, software and formal specification. However, it is really hard for assessors to directly justify the expected positive correlation, so additional elicitations from assessors are needed which will be discussed in next section.

High compliance of annotated codes makes formal verification irrelevant

By somehow, if the assessor justified the $Cov(I_{SW}, I_{\varphi})$ is positive, then (6.7) – the posterior belief in perfection after seeing good formal proof evidence – is

- a decrease function of t_1 and t_2 respectively. This is intuitively right, i.e. the more reliable the prover, gaining more confidence in perfection.
- an increase function of θ, which is obviously right.
- a decrease function of φ, meaning the higher confidence in the compliance of the annotated codes would reduce our posterior confidence in perfection. Informally, this is due to the essence of the formal proof is a *conformance checker* of the software and formal specification, so high confidence in the compliance at beginning will make the formal proof results meaningless, i.e. it is a matter of course to see good formal proof evidence. For better understanding, we could imaging an extreme example that we

build a "universal" formal specification and thus any program will be compliant to it, then our confidence in the compliance of the annotated codes is $\varphi = 1$. In this case, the formal proof result would not have any effect on our confidence in perfection. In reality, we do can find some techniques being used on the annotated codes to increase the verifiers' confidence in its compliance at the very earlier stage of developing a formal specification, e.g. the STADY plug-in of Frama-C (Petiot, Kosmatov et al. 2014). As later argued in (Petiot, Kosmatov et al. 2015), this kind of compliance checking technique is certainly helpful in finding the reason why the formal proof fails. But when the formal proof succeeds, by the monotonicity result here, the use of compliance checking techniques would weaken the posterior confidence in perfection *built from the formal proof evidence*. So based on this observation, it gives some enlightenment on how to properly construct a safety case of perfection from both formal proof evidence and compliance checking evidence, e.g. should they be used as two legs of arguments or in a sequential one-leg way.

6.4. A model with more elicitations from assessors

As argued in earlier section, it is unrealistic for the assessor to justify a positive $Cov(I_{SW}, I_{\varphi})$, therefore more information needs to be elicited. In this section, it is assumed that assessors could say something about spectrum of formal specifications defined in section 6.2.3.

6.4.1. Confidence in possible faulty formal specifications

Since it is unlikely to get a *deterministic* answer of the question "do we have a 'common cause fault' formal specification?" from the assessor, the assessor may express some confidence in the answer. For instance, the assessor may answer "I cannot tell you for certain, but I have 20% confidence that we are using a 'common cause fault' formal specification". In line with that kind of answer, we could define the confidence elicited from assessors on each possible faulty formal specification.

$$Pr(iscompliant_{AC} = 0, isperfect_{SW} = 1, iscomplete_{FS} = 1) = f_1 \tag{6.10}$$

$$Pr(iscompliant_{AC} = 0, isperfect_{SW} = 1, iscomplete_{FS} = 0) = f_2 \tag{6.11}$$

$$Pr(iscompliant_{AC} = 1, isperfect_{SW} = 0, iscomplete_{FS} = 0) = f_3 \tag{6.12}$$

$$\begin{aligned} &Pr\big(iscompliant_{AC} = 0, isperfect_{SW} = 0, iscomplete_{FS} = 0\big) \\ &\quad + Pr\big(iscompliant_{AC} = 1, isperfect_{SW} = 1, iscomplete_{FS} = 0\big) \\ &\quad = f_4 \end{aligned} \tag{6.13}$$

where:

- f_1 is the confidence in a "complete but too restrictive" formal specification
- f_2 is the confidence in a "partial and too restrictive" formal specification
- f_3 is the confidence in a "common cause fault" formal specification
- f_4 is the confidence in the two types of "partial but functional" formal specifications

The only one left on the spectrum of formal specifications is the "correct" formal specification and its associated confidence is $(1 - f_1 - f_2 - f_3 - f_4)$.

6.4.2. Conclusion with the new objective function

With the new elicited information from the assessor, the preliminary result (6.7) could be rewritten into:

$$Pr(isperfect_{SW} = 1|e = 1) = \frac{(1 - t_1 - t_2) \times (\theta - f_1 - f_2) + t_2 \times \theta}{(1 - t_1 - t_2) \times (\theta - f_1 - f_2 + f_3) + t_2} \tag{6.14}$$

See Appendix D for proof. Some new conclusions could be drawn.

When will formal proof certainly increase our confidence in perfection?

To answer this question, we need to solve the inequality $Pr(isperfect_{SW} = 1|e = 1) > \theta$. Via trivial exercise, we get the result:

$$\begin{cases} 1 - t_1 - t_2 > 0 \\ f_3 < \dfrac{(1 - \theta)(\theta - f_1 - f_2)}{\theta} \end{cases} \tag{6.15}$$

The result (6.15) essentially tells us two constraints to guarantee the usefulness of formal proof in claiming perfection. The first one is about the reliability of the prover, which could be easily satisfied in practice. The second constraint seems a confidence upper bound on the possible faulty formal specifications. In other words, assessors should have *limited doubt* in different types of faulty formal specifications when use formal proof evidence to claim perfection of software.

6.5. Chapter summary

Intuitively, formal verification is a strong evidence to support perfection claim of software. However in practice, due to the inevitable uncertainties hidden in a formal verification process, it we cannot naively believe our *proved* program is certainly perfect. In this chapter, probabilistic models are built to check how good formal proof evidence will affect assessor's posterior confidence in the perfection of the software of interest.

Firstly, two sources of aleatory uncertainties (from the prover and formal specification) are identified as non- negligible for claiming the perfection of software. Then by analysing the features of these two uncertainties, we use 5 objective parameters to capture them. It is worth mentioning that a spectrum of formal specifications was proposed, representing all possible associations between a program and a formal specification for a particular engineering problem.

Our interest – the posterior belief in perfection of the program when seeing good formal proof evidence – is then expressed in terms of the marginal expectation of some parameters and a covariance term, i.e. the result (6.7).

Basically we can conclude 3 points from (6.7):

- The confidence of the completeness of formal specification is not directly relevant to our interest. This justifies the fact that verifiers usually do not care too much about the completeness of formal specification in practice and normally will end up with a partial one.
- Rather, the correlation between the perfection of the program and the compliance of the annotated codes is a key factor. This correlation should be explicitly argued in support of the perfection claim if using the formal proof evidence in a safety case. However this seems a very hard task currently, assessors cannot justify the positive or negative correlation without extra information.
- Another observation is from the monotonicity analysis of (6.7), which is the high confidence in the compliance of the annotated codes would weaken the efficiency of formal proof evidence in building confidence in perfection. This justifies the fact that formal proof is nothing more than a *conformance checker* of the software and formal specification, so high confidence in the compliance at beginning will make the formal proof results meaningless, i.e. it is a matter of course to see good formal proof evidence. In recent years, we do can see some new techniques, e.g. the STADY plug-

in of Frama-C (Petiot, Kosmatov et al. 2014), could increase our confidence in the compliance of the annotated codes. Based on this observation, it gives some enlightenment on how to properly construct a safety case of perfection from both formal proof evidence and compliance checking evidence, e.g. should they be used as two legs of arguments or in a sequential one-leg way.

Finally, by assuming we could get more elicitations from the assessor, a refined model (6.14) was proposed based on the beliefs in the different cases on the spectrum of formal specifications. To guarantee that formal proof will certainly build our confidence in perfection, we need constraints on the prover's reliability which is intuitively right and *limited doubts* in different types of faulty formal specifications which should be well argued in a safety case.

Taking a step back, it is still too difficult for an assessor to express beliefs in the cases on the spectrum of formal specifications, i.e. (6.10), (6.11) and (6.12). Future work is needed to help assessors to elicit confidence in them.

All in all, even though this is very *speculative* and *preliminary* work, it shows how this kind of analysis and mathematical modelling can be applied in claiming the perfection of software. Difficulties and questions are discovered as:

- It is hard to write down the likelihood function of the formal proved evidence in terms of simple and straight forward factors such as the perfection of software and the completeness of formal specification. More subtle factors have to be taken into account. But, more factors means more difficult to elicit prior beliefs. There seems a tension between these two sides. How to find a better boundary is of interest in future.
- Another possible way to calculate the confidence in a claim is to simply and conservatively sum all the doubts in the arguments and use it as an overall doubt in the top claim. It might provide more insight to compare this result with the result of the sum of doubt approach.
- This work also provides ideas on conducting some empirical experiments.

7. MODELLING 1OO2 SYSTEM RELIABILITY VIA "QUASI-PERFECTION" CLAIMS

The original LR and LP models (Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a) proposed ways of overcoming the difficulties of lack of independence (at aleatory and epistemic level respectively) in reliability modelling of 1-out-of-2 software-based systems by introducing the notion of probability of perfection (of a population). Chapter 3 in this thesis further extended the original LR and LP models in terms of a subjective probability of perfection of a single program. Here these two paralleled sets of models are generalised. Instead of "perfection", a new notion of "quasi-perfection" is introduced: a small *pfd* is practically equivalent to perfection (e.g. yielding very small chance of failure in the entire life of a fleet of systems). It is believed that the quasi-perfection idea has an analogue of the most "pure" perfection related models in previous chapters, and it might have advantages over the "pure" perfection models in some circumstances. The scope of this chapter is not to exhaustively extend all the models in previous chapters in terms of quasi-perfection, rather to show the possible advantage of it. So only the subjective probability of quasi-perfection of a single program will be investigated, and a conservative end-to-end example arguing system *pfd* via failure-free tests evidence of the software of interest will be presented.

7.1. The introduce of "quasi-perfection"

It has been well argued that the notion of probability of perfection is useful not only on its own (e.g. as a life-time claim), but also plays an important role in the reliability assessment of diverse 1oo2 systems. In the original LR and LP models (Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a), the use of objective probability of perfection (as a population property) overcomes the difficulties of lack of independence (at aleatory and epistemic level respectively) in reliability modelling of 1oo2 software-based systems. However, they require claims on that population property (i.e. the objective probability of perfection) which only population evidence would sufficiently support it. To incorporate the evidence of the single software of interest, Chapter 3 extended the original LR and LP models in terms of the subjective probability of perfection of a *single* program. Since then, argument and evidence for the perfection claim of the *single* channel of interest could be used in the reasoning on the reliability of a diverse 1oo2 system.

Chapter 4 (published as (Zhao, Littlewood et al. 2015)) concentrates on the question what can be claimed about probability of perfection from seeing many failure-free tests. A probability model for this problem was developed, and illustrated with some numerical examples. The approach started from the premise that real assessors can generally only provide *limited* prior belief, rather than a complete distribution; for example, a probability mass at the origin (representing prior confidence in perfection, which is also required in later sections of this chapter), and one percentile for the rest of the distribution. In the face of this difficulty, the approach was *conservative*: from the many (generally an infinite number of) distributions that satisfy the limited prior constraints, the one(s) that gives the most conservative results for the system's posterior *pfd* was chosen. Unfortunately, such results were often *very* conservative – too conservative to be useful. In fact, in the worst case, confidence in perfection did not increase much even after observing an *infinite* number of successful tests. It was observed that this was because, whilst extensive failure-free working may be a result of a program's perfection, it could also be because the program – although not perfect – has a very small *pfd*. So some ways around this problem was proposed, essentially by pruning the large class of allowable prior distributions by excluding ones that seem "unreasonable" in general ways.

This “pruning” solution requires further constraints on the assessor’s prior beliefs. The reasonableness of these further constraints of course has to be discussed and agreed by the regulator and licensee in a real application.

In the work to be described in the remainder of this chapter, a different way around this difficulty is proposed. The idea is to exploit the fact that “perfection” and “extremely small *pfd*” are effectively indistinguishable as explanations for extensive failure-free working, which was firstly observed in (Strigini and Povyakalo 2013) and named as “quasi-fault-freeness”. Similarly here the “quasi-perfection” notion is introduced, and it means the *pfd* of the channel is smaller than some small given number, say ε (which is associated with the operational profile). For example, ε could be chosen so that over the entire lifetime of the system (or fleet of systems) there would be only a small chance of failure, i.e. the lifetime behaviour of the system would be likely to be identical to that of a perfect system.

The value of ε could be chosen from higher level system requirements. For instance, consider the example of a single channel of a nuclear reactor protection system. We might anticipate something like 2 demands on the protection system on average per year, with an anticipated lifetime of 50 years. For 99% confidence of seeing no failures in the expected 100 demands, ε should be about 10^{-4}. More rigorous approach to elicit the ε – the *pfd* bound that defines “quasi perfection” – is an important future work, while in this chapter we shall assume the ε is given.

7.2. A new bound for the reliability of a 1oo2 system based on the possible “quasi-perfection” of one channel

The interest centres on the probability of failure on a random demand of a 1-out-of-2 system with channels A and B. There are two *objective* parameters “in the world”:

- pfd_A for the channel A, and its range is [0,1].
- $isnqp_B$ for the channel B. It could be either 0 or 1. $isnqp_B = 1$ means the channel B is not quasi-perfect (i.e. its *pfd* is bigger than the given ε), otherwise $isnqp_B = 0$ means the channel B is quasi-perfect (i.e. its *pfd* is smaller than the given ε).

Then via the theorem below, we could have a new bound on the system reliability given values for the two objective parameters.

Theorem 0

$$\begin{aligned} &Pr(\text{sys fails [on a random demand]}|pfd_A = P_A, isnqp_B = I_B) \\ &\quad \le \varepsilon(1 - I_B) + P_A \times I_B \end{aligned} \tag{7.1}$$

Proof

$$\begin{aligned} &Pr(\text{sys fails [on a random demand]}|pfd_A = P_A, isnqp_B = I_B) \\ &= Pr(\text{sys fails}|A \text{ fails, B not quasi} - \text{perfect}, pfd_A = P_A, isnqp_B = I_B) \\ &\times Pr(A \text{ fails, B not quasi} - \text{perfect}|pfd_A = P_A, isnqp_B = I_B) \\ &+ Pr(\text{sys fails}|A \text{ fails, B quasi} - \text{perfect}, pfd_A = P_A, isnqp_B = I_B) \\ &\times Pr(A \text{ fails, B quasi} - \text{perfect}|pfd_A = P_A, isnqp_B = I_B) \\ &+ Pr(\text{sys fails}|A \text{ succeeds, B not quasi} - \text{perfect}, pfd_A = P_A, isnqp_B = I_B) \\ &\times Pr(A \text{ succeeds, B not quasi} - \text{perfect}|pfd_A = P_A, isnqp_B = I_B) \\ &+ Pr(\text{sys succeeds}|A \text{ succeeds, B quasi} - \text{perfect}, pfd_A = P_A, isnqp_B = I_B) \\ &\times Pr(A \text{ succeeds, B quasi} - \text{perfect}|pfd_A = P_A, isnqp_B = I_B) \end{aligned} \tag{7.2}$$

The last two terms on the right hand side of the expansion (7.2) are zero trivially, since if A succeeds the 1-out-of-2 system cannot fail.

Now, if B is not quasi-perfect, it is conservative to assume that it fails whenever A does, so the first term on the right hand side of the expansion (7.2) is:

$$\begin{aligned} &Pr(\text{sys fails}|A \text{ fails, B not quasi} - \text{perfect}, pfd_A = P_A, isnqp_B = I_B) \\ &\quad \times Pr(A \text{ fails, B not quasi} - \text{perfect}|pfd_A = P_A, isnqp_B = I_B) \\ &\quad \le 1 \times Pr(A \text{ fails, B not quasi} - \text{perfect}|pfd_A = P_A, isnqp_B = I_B) \\ &\quad = Pr(A \text{ fails}|pfd_A = P_A, isnqp_B = I_B) \\ &\quad \times Pr(\text{B not quasi} - \text{perfect}|pfd_A = P_A, isnqp_B = I_B) = P_A \times I_B \end{aligned} \tag{7.3}$$

Here, a similar assumption as that in the original LR model is needed, i.e. the two events “A fails” and “B is not quasi-perfect” are *conditionally independent*, given the certain value P_A for pfd_A and I_B for $isnqp_B$. The intuitive justification of the assumption is that whether or not B is not quasi-perfect tells us nothing about whether or not A will fail on a random demand.

Note that the events "channel A fails on a random demand" and "program B is not quasi-perfect" are *not unconditionally independent* in general. Informally, seeing A fails on a demand may suggest that the problem being solved by both programs is a "difficult" one, and thus it is less likely that B will be quasi-perfect. That is, learning something about A's probability of failure on demand (e.g. by seeing an A failure), may tell us something about B's probability of being not quasi-perfect: this is an issue of *epistemic* dependence between the model parameters, pfd_A and $isnqp_B$, which will be addressed later.

The second term in the expansion (7.2) is:

$$\begin{aligned}&Pr(\text{sys fails}|A\text{ fails, B quasi}-\text{perfect}, pfd_A = P_A, isnqp_B = I_B)\\ &\times Pr(A\text{ fails, B quasi}-\text{perfect}|pfd_A = P_A, isnqp_B = I_B)\\ &= Pr(\text{A fails, B fails}|A\text{ fails, B quasi}-\text{perfect}, pfd_A = P_A, isnqp_B = I_B)\\ &\times P_A \times (1-I_B)\end{aligned} \tag{7.4}$$

where we relabel the event "System fails" as "A fails and B fails" without change of meaning and reuse the conditional independence assumption of the two events "A fails" and "B is not quasi-perfect" given the certain value P_A for pfd_A and I_B for $isnqp_B$.

Now making explicit the conditioning on event "A fails" of the expression (7.4):

$$= \frac{Pr(\text{A fails, B fails}|\text{B quasi}-\text{perfect}, pfd_A = P_A, isnqp_B = I_B) \times P_A(1-I_B)}{Pr(\text{A fails}|\text{B quasi}-\text{perfect}, pfd_A = P_A, isnqp_B = I_B)} \tag{7.5}$$

and considering in the numerator that "A fails and B fails" is a subset of "B fails":

$$\begin{aligned}&\leq \frac{Pr(\text{ B fails}|\text{B quasi}-\text{perfect}, pfd_A = P_A, isnqp_B = I_B) \times P_A \times (1-I_B)}{Pr(\text{A fails}|\text{B quasi}-\text{perfect}, pfd_A = P_A, isnqp_B = I_B)}\\ &= \frac{Pr(\text{ B fails}|\text{B quasi}-\text{perfect}, pfd_A = P_A, isnqp_B = I_B) \times P_A \times (1-I_B)}{P_A}\\ &\leq \frac{\varepsilon \times P_A \times (1-I_B)}{P_A} = \varepsilon(1-I_B)\end{aligned} \tag{7.6}$$

where reuse the conditional independence assumption of the two events "A fails" and "B is quasi-perfect" given $pfd_A = P_A$ and $isnqp_B = I_B$. And seeing that B is quasi-perfect, its probability of failure is smaller than ε for any (P_A, I_B).

So finally by substituting (7.3) and (7.6) into (7.2), we got the result (7.1).

QED

Note that this result (7.1) is a generalization of the result (3.1) in chapter 3. We could obtain that earlier result by putting $\varepsilon = 0$.

Now we have a counterpart of the modified LR model (i.e. (3.1)) for the quasi-perfection notion, which is about the quasi-perfection of the *single* software of interest. Similar work could be done for the original LR model as well (which has been written up for a publication and is under review). In that case, the objective probability of quasi-perfection as a property of a *population* should be used. It is believed that most of the “pure” perfection models could be correspondingly extended into quasi-perfection models which might have advantages over the “pure” perfection models in some circumstances. The scope of this chapter is not to exhaustively extend all the models in previous chapters in terms of quasi-perfection, rather to show the possible advantage of it. So here only the subjective probability of quasi-perfection of a single program will be investigated, and a conservative end-to-end example arguing system *pfd* via failure-free tests evidence of the software of interest will be presented.

7.3. Conservative reasoning about the epistemic uncertainty

The result above concerns what happens at the aleatory level. In practice, of course, the two objective parameters will not be known with certainty. Ideally, an assessor would describe his epistemic uncertainty about these unknowns – *pfd*$_A$ and *isnqp*$_B$ – in terms of a complete bivariate distribution:

$$F_{pfd_A,isnqp_B}(P_A, I_B) = Pr(pfd_A \le P_A, isnqp_B = I_B) \tag{7.7}$$

The *unconditional* probability of system failure is then:

$$\begin{aligned} & Pr(\text{sys fails [on a randomly selected demand]}) \\ & = E_{pfd_A,isnqp_B}\big(Pr(\text{sys fails } | pfd_A = P_A, isnqp_B = I_B)\big) \\ & \le E_{pfd_A,isnqp_B}(\varepsilon(1 - isnqp_B) + pfd_A \times isnqp_B) \\ & = \iint (\varepsilon(1 - I_B) + P_A \times I_B) dF_{pfd_A,isnqp_B}(P_A, I_B) \end{aligned} \tag{7.8}$$

In reality, it is unlikely that a real-world assessor would be willing or able to offer such a complete bivariate distribution to represent his beliefs about the unknowns of the model. In particular, it is known that people find it hard to express the *dependence* between their beliefs. In this section, some results based on only *partial* and *marginal* beliefs are obtained, and they parallel the earlier results of the modified LP model in section 3.2.2. Similar work has been done for the original LR model (Littlewood and Povyakalo 2013a) as well (which has been written up for a publication and is under review), but out of the scope of this chapter.

Conservative bounds on system *pfd* can be obtained:

Theorem 1

If the assessor could tell us:

$$Pr(pfd_A < P_A) = 1 - \alpha_A \tag{7.9}$$

$$Pr(isnqp_B = 1) = 1 - \omega \tag{7.10}$$

i.e. ω confidence in B is quasi-perfect and α_A doubt in $pfd_A < P_A$, then:

$$Pr(\text{sys fails}) \leq \varepsilon \times \omega + (1 - \omega)P_A + (1 - P_A) \times min\{\alpha_A, 1 - \omega\} \tag{7.11}$$

See Appendix E for proof.

Example 1

The assessor has chosen $\varepsilon = 10^{-7}$ to define quasi-perfection, i.e. if *pfd*$_B$ is smaller than this, he will regard channel *B* to be quasi-perfect. If the assessor is 95% confident that *pfd*$_A$ is smaller than 10^{-5} (i.e. $P_A = 10^{-5}$ and $\alpha_A = 0.05$), and 99% confident that B is quasi-perfect (i.e. $\omega = 0.99$), we have:

$$Pr(\text{sys fails}) \leq \varepsilon \times \omega + (1 - \omega)P_A + (1 - P_A) \times min\{\alpha_A, 1 - \omega\}$$
$$= 0.99 \times 10^{-7} + (1 - 0.99) \times 10^{-5} + (1 - 10^{-5}) \times 0.01 = 0.0100001$$

which is a very conservative result.

It is not hard to see that the result is dominated by the smaller of the two doubts – i.e. the doubt α_A and the doubt $1 - \omega$ on the quasi-perfection of channel B. However, an assessor will not be able to express very small doubts, i.e. ones considerably smaller than desired "reliability claims" (e.g. a 10^{-5} *pfd*). For example, an assessor would not say he has a 10^{-5} doubt in something. So this theorem 1 is not practically helpful in this sense.

Theorem 2

Additionally to (7.9) and (7.10), if the assessor were able to tell a certain upper bound on the *pfd*$_A$:

$$Pr(pfd_A < P_A^U) = 1 \tag{7.12}$$

then by the proof in Appendix E:

$$Pr(\text{sys fails}) \leq \varepsilon \times \omega + (1 - \omega)P_A + (P_A^U - P_A) \times min\{\alpha_A, 1 - \omega\} \tag{7.13}$$

Example 2

Same as the scenario of the Example 1, but additionally with $P_A^U = 10^{-3}$, meaning the assessor is certain that the *pfd* of channel A will be better than 10^{-3}, then:

$$
\begin{aligned}
Pr(\text{sys fails}) &\leq \varepsilon \times \omega + (1-\omega)P_A + (P_A^U - P_A) \times min\{\alpha_A, 1-\omega\} \\
&= 0.99 \times 10^{-7} + (1-0.99) \times 10^{-5} + (10^{-3} - 10^{-5}) \times 0.01 \approx 10^{-5}
\end{aligned}
$$

which is a much better result than the one in Example 1.

This is obviously a more useful result which is dominated by the product of $P_A^U \times \min\{\alpha_A, 1-\omega\}$. Result (7.13) is potentially useful because it is essentially a product of two smaller numbers in which one is a "reliability claim" (i.e. the P_A^U which is essentially small and close to the desired reliability claims), and the other is a doubt (i.e. α_A or $1-\omega$) which could be afterwards learned (i.e. be further decreased) due to the reducible nature of epistemic uncertainty. For instance, Bayesian inference can update the doubt α_A when seeing some V&V evidence of channel A; and similar for the quasi-perfection confidence ω of the channel B when seeing V&V evidence of it.

The objective parameter $isnqp_B$ is either 0 or 1, so its marginal distribution is a 2-point one. It seems the assessor cannot vary his partial beliefs on this 2-point distribution, but simply give a probability, i.e. the (7.10). However, for the marginal distribution of pfd_A, the assessor may express his partial beliefs in various forms. For example, if he knew the first two *moments* of his marginal distributions of pfd_A, i.e. the mean and variance, then:

Theorem 3

$$
\begin{aligned}
Pr(\text{sys fails}) &\leq \varepsilon \times \omega + \sqrt{(1-\omega)\big(E(pfd_A)^2 + Var\ (pfd_A)\big)} \\
&< \varepsilon \times \omega + \sqrt{(1-\omega)}\big(E(pfd_A) + SD\ (pfd_A)\big)
\end{aligned}
\tag{7.14}
$$

See Appendix E for proof

Example 3

If the assessor is 99% confident in the quasi-perfection of channel B (i.e. $\omega = 0.99$), and the expectation of pfd_A is 10^{-4}. Besides, he also knows: $SD\ (pfd_A) \leq 4\ E(pfd_A)$, then via (7.14):

$$
\begin{aligned}
Pr(\text{sys fails}) &< \varepsilon \times \omega + \sqrt{(1-\omega)}\big(E(pfd_A) + SD\ (pfd_A)\big) \\
&\leq 0.99 \times 10^{-7} + \sqrt{(1-0.99)}\big(5 \times E(pfd_A)\big) \approx 0.5 \times 10^{-4}
\end{aligned}
$$

Again, this seems a useful result which is a product of two small numbers. One is associated with the doubt (i.e. $\sqrt{(1-\omega)}$) which could be afterwards updated when seeing more good evidence about channel B. The other one is associated with reliability claim i.e. $E(pfd_A) + SD\ (pfd_A)$. The first term is the expected value of *pfd*$_A$ which could be learned when seeing more evidence of channel A. While the second term is a about how spread out of the subjective distribution of *pfd*$_A$ around the mean, which seems not easy to be elicited from the assessors.

However, if the assessor was able to tell a certain upper bound on *pfd*$_A$ (as he/she did in the example 2), we do not need to elicit the second moment (i.e. the variance) of *pfd*$_A$, as a conservative variance could be calculated from the mean and the certain upper bound. See the theorem below:

Theorem 4

For channel A, if the assessor could tell us a certain upper bound P_A^U of *pfd*$_A$, i.e. (7.12) and a mean, say $E(pfd_A) = M_A$; for channel B, a ω confidence in its quasi-perfection, i.e. (7.10), then:

$$Pr(\text{sys fails}) \leq \varepsilon \times \omega + \sqrt{(1-\omega) \times M_A \times P_A^U} \tag{7.15}$$

Seep Appendix E for proof.

Same as the figures used in example 2 and 3, assume the assessor gives a certain upper bound $P_A^U = 10^{-3}$ and mean of *pfd*$_A$ as $M_A = 10^{-4}$ for channel A; and still 99% confident in the quasi-perfection of channel B (i.e. $\omega = 0.99$), then via (7.15):

$$Pr(\text{sys fails}) \leq \varepsilon \times \omega + \sqrt{(1-\omega) \times M_A \times P_A^U} = 0.99 \times 10^{-7} + \sqrt{10^{-2} \times 10^{-3} \times 10^{-4}}$$
$$\approx 0.3172 \times 10^{-4}$$

This seems an even more useful result than the theorems before in two senses. First, it is relatively easier for the assessor to elicit numbers, comparing to theorem 3 in which requires a variance. Second, the result (7.15) is essentially dominated by the root of a product of 3 small numbers (one doubt and two reliability claims), in which ω and M_A could be individually updated when seeing good evidence from the two channels. And both of the individual afterwards learning of the two channels will make a contribution to the final 1oo2 system reliability.

By now, the 4 theorems above addressed the dependency problem at the epistemic level, at the price of conservatism. Only marginal and partial beliefs expressed in terms of (some of) percentiles (confidence bounds), means or variances of the parameters pfd_A and $isnqp_B$ are required. It is well-known how to do this for pfd_A, for example based on evidence from operationally representative statistical testing: see, e.g. (Littlewood and Wright 1997). In next section, $isnqp_B$ will be considered, i.e. how to reason the confidence ω in the quasi-perfection of channel B.

7.4. Conservative claims about quasi-perfection via failure-free testing evidence

The approach here is an extension of the work the chapter 4, changing the interest from the posterior probability of perfection to the posterior probability of *quasi-perfection*. An analogous informal description will be introduced first, and then the probability model.

The fundamental assumption in this section is that for the channel B, there is a true unknown pfd_B, i.e. an *objective parameter* in the world. The assessor cannot know its value for certain, therefore has a subjective distribution over it, say $f_B(p)$. If the assessor were able to specify a *complete* distribution to represent his prior beliefs about the pfd_B, it is a simple matter to use Bayes' Theorem to obtain his exact posterior distribution after he has seen the evidence. Then he could express his posterior beliefs about quantities of interest such as the expected value of pfd_B (best "point" estimate), percentiles (confidence bounds for pfd_B), and so on. In this case, the interest is the posterior confidence in the quasi-perfection of channel B, i.e. the posterior confidence bound of the pfd_B smaller than the given ε.

However, as argued in other chapters of this thesis, it is impossible for assessors to express a *complete* probability distribution. This observation seems particularly pertinent for software-based systems. Rather, assessors may express something *partially* about the distribution, for example:

$$Pr(pfd_B = 0) = \theta \tag{7.16}$$

$$Pr(0 < pfd_B \leq \varepsilon) = \beta \quad (7.17)$$

$$Pr(pfd_B > y) = \alpha \quad (7.18)$$

i.e. essentially three percentiles on the distribution, see Figure 28. Note the prior belief in the quasi-perfection ω is then $\theta + \beta$.

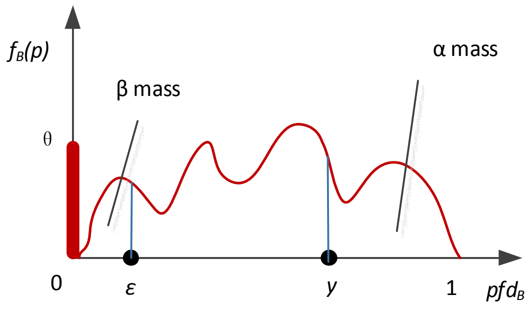

Figure 28 an idealized example of a distribution satisfying the assessor's expressed prior beliefs (7.16) (7.17) and (7.18)

Of course, by making only very restricted assumptions like these we do not completely characterize a distribution for the *pfd* of the channel B: there will be an *infinite* number of distributions that satisfy (7.16), (7.17) and (7.18). The approach in what follows will be to choose the worst case distribution – i.e. the one that gives the most conservative results. In this case it is the posterior probability of quasi-perfection, following the observation of *n* failure-free demands, i.e. the formula below:

$$Pr(0 < pfd_B \leq \varepsilon | \text{n failure free tests}) \\ = \frac{\theta + \int_{0+}^{\varepsilon}(1-p)^n f_B(p)dp}{\theta + \int_{0+}^{\varepsilon}(1-p)^n f_B(p)dp + \int_{\varepsilon+}^{y}(1-p)^n f_B(p)dp + \int_{y+}^{1}(1-p)^n f_B(p)dp} \quad (7.19)$$

The problem now is to find the most conservative *$f_B(p)$*, i.e. the one that minimizes (7.19) subject to the constraints (7.16), (7.17) and (7.18). The value of (7.19) at the minimum called ω^* is the most pessimistic posterior belief in quasi-perfection consistent with the assessor's expressed prior beliefs. It turns out that the most pessimistic *$f_B(p)$* is a 4-point distribution (see Appendix E for proof), as shown in Figure 29[23].

Using this most conservative distribution, we could obtain the most conservative – i.e. a lower bound on – probability of quasi-perfection:

[23] Strictly speaking, there may be other distributions that satisfy the constraints and give the same probability of quasi-perfection; but there are none that give a smaller value.

$$Pr(0 < pfd_B \leq \varepsilon | \text{n failure free tests})$$

$$\geq \frac{\theta + (1-\varepsilon)^n \beta}{\theta + (1-\varepsilon)^n \beta + (1-\varepsilon)^n (1-\theta-\alpha-\beta) + (1-y)^n \alpha} \quad (7.20)$$

$$= \frac{\theta + (1-\varepsilon)^n \beta}{\theta + (1-\varepsilon)^n (1-\theta-\alpha) + (1-y)^n \alpha}$$

$$\geq \frac{\theta}{\theta + (1-\varepsilon)^n (1-\theta-\alpha) + (1-y)^n \alpha} = \omega^* \quad (7.21)$$

where the last line is obtained by conservatively putting $\beta = 0$.

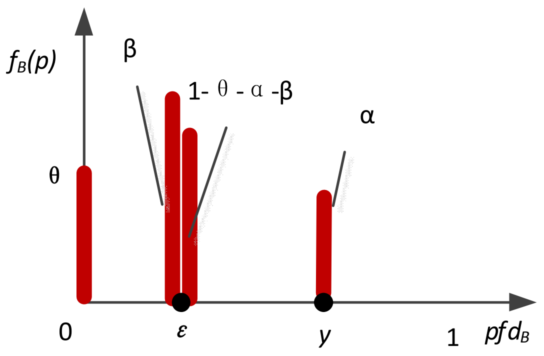

Figure 29 this four-point distribution gives the smallest probability of quasi-perfection, subject to the prior constraints (7.16), (7.17) and (7.18). There are four points of support here, but note that the mass *β* and the mass *1-θ-α-β* are coincident at *ε*.

Putting $\beta = 0$ in (7.20) introduces the opportunity to simplify the assessor's task, albeit at the price of further conservatism: using (7.21) in this way reduces the problem to eliciting just two parameters, i.e. θ and α. Of all the values in (7.20) that β might take (i.e. the assessor might believe), $\beta = 0$ is the most conservative: it gives the smallest posterior probability of quasi-perfection. In what follows, (7.21) will be used rather than (7.20).

The Table XII shows some numerical examples of ω^*after seeing *n* failure-free testing evidence. Note, ε=10^{-7} is used as the definition of quasi-perfection. The last column θ^* is the most conservative posterior belief in "pure" perfection using the result (4.4) in chapter 4. The purpose to illustrate θ^* here is to compare it with the gain of confidence in the quasi-perfection, since each entry in the table is based on the same prior beliefs (i.e. θ, α, y) and amount of evidence (i.e. *n*).

Table XII numerical examples of the posterior belief in quasi-perfection using the 4 points distribution of Figure 29, given ε=10^{-7}

θ	α	y	n	ω^*	θ^*
0.5	0.01	0.001	10^4	0.505300248	0.505050275
0.5	0.01	0.001	10^6	0.530014548	0.505050505
0.5	0.01	0.001	10^8	0.99995551	0.505050505
0.5	0.05	0.001	10^4	0.526563838	0.526314538
0.5	0.05	0.001	10^6	0.551160457	0.526315789
0.5	0.05	0.001	10^8	0.999959142	0.526315789
0.9	0.01	0.001	10^4	0.909173105	0.909090494
0.9	0.01	0.001	10^6	0.917024218	0.909090909
0.9	0.01	0.001	10^8	0.99999546	0.909090909
0.9	0.05	0.001	10^4	0.947416008	0.947366169
0.9	0.05	0.001	10^6	0.952137255	0.947368421
0.9	0.05	0.001	10^8	0.999997478	0.947368421

In each case of the Table XII, there are some increases in the assessor's confidence in both perfection and quasi-perfection when seeing some number of failure-free tests. However, as the observation in section 4.3, the evidence of failure-free testing is generally very weak in supporting claims about probability of perfection. When the number n increases from 10^6 to 10^8, there is no actual gain in confidence in perfection in each case (see section 4.3 for detailed explanation). While for quasi-perfection, there is very significant improvement when n= 10^8. Examining more numerical examples, we can observe that:

- When $n < \frac{1}{\varepsilon}$, there is no actual difference in the confidence gain between quasi-perfection and perfection. For instance, in the first two rows in Table XII seeing 10^4 and 10^6 tests (which are smaller than $\frac{1}{\varepsilon}$, i.e. 10^7), the worst case posterior beliefs in quasi-perfection (ω^*) and perfection (θ^*) are nearly same.
- When $n > \frac{1}{\varepsilon}$, there is obvious advantages of the quasi-perfection notion. For instance the third case in Table XII shows a nearly 99.996% confidence in quasi-perfection comparing to only 50.505% confidence in perfection, after seeing the same amount of testing evidence.

So the first conclusion is that, when $n > \frac{1}{\varepsilon}$, claiming quasi-perfection has its advantage over claiming perfection in terms of the efficiency of the failure-free testing evidence. The theoretical explanation of this advantage is that the limit when n approaches infinity of the result (7.21) is 1, while for perfection claims the limit is not 1 as (4.5) shows.

Readers may ask why we still need the parameter θ, representing prior confidence in perfection, here in this quasi-perfection model. It has been shown that evidence from failure-free operation does not support an increase in confidence of perfection that would be useful for assessing system reliability. However, this evidence, given the same prior beliefs, improves probability of quasi-perfection. Note that it is not hard to prove that just having a prior confidence in quasi-perfection (together with the other confidence bound (7.18)) would also not support this kind of reasoning: the evidence would not help to improve confidence in quasi-perfection.

So the advantage of the quasi-perfection notion is not to avoid the elicitation of a belief in perfection, rather to use the *finite* number of tests more effectively. To recap: a "pruning" idea was used in section 4.4 to solve the problem that many (even an infinite number of) fail-free tests evidence cannot increase our confidence in perfection much. Here the quasi-perfection idea as an alternative solution is proposed (note, this does not mean the quasi-perfection idea is contradict to the "pruning" idea. Actually, assessor can "prune" in this quasi-perfection model which will give even better results). It works because we are claiming something weaker and still with the same priors and evidence, thus having more confidence in that weaker claim.

7.5. End-to-end numerical examples for 1oo2 system *pfd*

So far, a relatively[24] complete argument chain for reasoning about the reliability of a 1oo2 system in which the channel B is quasi-perfection is presented, and each "link" is guaranteed to be conservative. In this section, some end-to-end numerical examples for the *pfd* of the 1oo2 system of interest will be shown, using the theorems in previous section 7.3 and models

[24] Of course you may add more "links" on the argument chain, e.g. models on using other types of evidence in reasoning quasi-perfection of the channel B and models on reasoning the reliability of the channel A.

in 7.4 with some amount of testing evidence from channel B. For the purpose of comparison, the counter-part end-to-end examples using the corresponding "pure" perfection models (in chapter 3 and 4) are also calculated and listed.

In Table XIII, the marginal and partial beliefs (i.e. a certain upper bound and a percentile) for channel A are given. Varying the parameters for channel B, some observations could be summarised:

- Similar as the observation in Table XII, when $n > \frac{1}{\varepsilon}$ the quasi-perfection models can emerge its advantage over "pure" perfection models. Actually when *n* is big, the "pure" perfection models are unbearably bad, due to the fact that the system reliability remains being regardless of the significant increase of *n*.
- There seems a trade-off problem between the choice of ε and the available amount of evidence n. For example, both the expected system *pfd* in the third and fourth row seem good results in terms of the definition of quasi-perfection in each case. In the case $\varepsilon = 10^{-5}$, $\mathrm{n} = 10^{6}$ is sufficient to conservatively claim the 1oo2 system is quasi-perfect. While for the case $\varepsilon = 10^{-7}$, *n* needs to be around 10^{8} to claim the system is quasi-perfect.

Table XIII numerical examples for the 1oo2 system *pfd* using theorem 2. For the channel A, given P_A=10^{-5} α_A=0.05 and P_A^U=10^{-3}.

θ	α	γ	n	ε	ω*	θ*	expected system *pfd* via quasi-perfection (7.13)	expected system *pfd* via perfection (3.10)
0.5	0.01	0.001	0	10^-7	0.5	0.5	5.455E-05	5.45E-05
0.5	0.01	0.001	10^6	10^-7	0.530014548	0.505050505	5.42529E-05	5.44495E-05
0.5	0.01	0.001	10^8	10^-7	0.99995551	0.505050505	1.44485E-07	5.44495E-05
0.5	0.01	0.001	10^6	10^-5	0.999955512	0.505050505	1.0044E-05	5.44495E-05
0.9	0.05	0.001	0	10^-7	0.9	0.9	5.059E-05	5.05E-05
0.9	0.05	0.001	10^6	10^-7	0.952137255	0.947368421	4.7958E-05	5.00263E-05
0.9	0.05	0.001	10^8	10^-7	0.999997478	0.947368421	1.02522E-07	5.00263E-05
0.9	0.05	0.001	10^6	10^-5	0.999997478	0.947368421	1.00025E-05	5.00263E-05

To confirm the observations above, numerical examples of the system reliability using the theorem 4 are also listed as Table XIV. Again, the marginal and partial beliefs required (i.e. a certain upper bound and a mean) for channel A are given. And the conclusions above are still true.

Table XIV numerical examples for the 1oo2 system *pfd* using theorem 4. For the channel A, given $M_A = 10^{-4}$ and $P_A^U = 10^{-3}$.

θ	α	γ	n	ε	ω*	θ*	expected system *pfd* via quasi-perfection (7.15)	expected system *pfd* via perfection (3.12)
0.5	0.01	0.001	0	10^-7	0.5	0.5	0.000223657	0.000223607
0.5	0.01	0.001	10^6	10^-7	0.530014548	0.505050505	0.000216844	0.000222475
0.5	0.01	0.001	10^8	10^-7	0.99995551	0.505050505	2.20926E-06	0.000222475
0.5	0.01	0.001	10^6	10^-5	0.999955512	0.505050505	1.21088E-05	0.000222475
0.9	0.05	0.001	0	10^-7	0.9	0.9	0.000100009	0.0001
0.9	0.05	0.001	10^6	10^-7	0.952137255	0.947368421	6.92781E-05	7.25476E-05
0.9	0.05	0.001	10^8	10^-7	0.999997478	0.947368421	6.02216E-07	7.25476E-05
0.9	0.05	0.001	10^6	10^-5	0.999997478	0.947368421	1.05022E-05	7.25476E-05

It is clear that the quasi-perfection notion is superior to "pure" perfection in reasoning about the reliability of diverse 1oo2 systems when having *large* number of failure-free tests evidence. The required amount of evidence needs to be $n > \frac{1}{\varepsilon}$. This seems still a very large number, so is this quasi-perfection practically useful?

First, it is obviously useful for some systems requiring modest reliability, for which the definition of quasi-perfection is also modest (e.g. $\varepsilon = 10^{-5}$). Second, you can always reduce the required amount of tests by imposing more other types of evidence, or "pruning" the worst case prior distribution used as Figure 29. The quasi-perfection argument chain shows here is a very simple and illustrative 3-steps one (i.e. reasoning the partial and marginal beliefs, then get rid of the epistemic dependency and finally the aleatory dependency), and in real projects, various helpful "links" (e.g. conservative models on combining other types evidence) could be added in and improve the practicality. Finally, it is necessary to emphasise that all results are guaranteed to be conservative.

Someone may still criticise the model here because of the very large amount of failure-free testing required in terms of comparing with other more "traditional and straightforward" models like (Littlewood and Strigini 1993; Miller, Morell et al. 1992; Littlewood and Wright 1997). For instance, it is easy to apply a simple model that $1-(1-\varepsilon)^n$ to calculate the confidence of claiming quasi-perfection of the system when seeing n failure-free testing. Consider the case of the last row in Table XIII, we could claim $1-(1-10^{-5})^{10^6}=0.9999546023$ confidence that the system is quasi-perfect via the simple model. It seems we cannot natively compare the expected system *pfd* of 1.05022E-05 with a confidence bound of 99.99546023% the system *pfd* is better than 10^{-5}, they are two different ways in which the reliability requirement are expressed. There are discussions on the limitations of various ways of expressing reliability requirements (Littlewood and Wright 1997), so naively comparing the two results here may lead to unprecise conclusions. Take a step back, if we really tend to believe the two claims are describing a very similar reliability level of the system, then I would like to argue that the quasi-perfection model is still better in terms of it is a very conservative model and we could always reduce the conservativeness by combining other means (e.g. stop at any stages of the argument chains. See next chapter for detailed discussion). While, it seems that you cannot do much to improve the results in the simple model.

7.6. Chapter summary

The work here firstly extended the models in chapter 3 (which is also an extension of the original LR and LP models (Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a)) in terms of the newly introduced quasi-perfection notion to reason about the reliability of a diverse 1oo2 system. The dependency problems at aleatory and epistemic level are addressed respectively at the price of conservatism. Only marginal and partial beliefs on the two parameters pfd_A and $isnqp_B$ are required. Since there is a large literature about claiming pfd_A, it leaves the question as how to reasonably claim something about $isnqp_B$. Specifically, that is the confidence in the quasi-perfection of the channel B. Then the type of failure-free testing evidence was investigated on claiming quasi-perfection. This again is an extension of the work in chapter 4 and again is guaranteed to be conservative. Finally some end-to-end numerical examples for the reliability of the 1oo2 system of interest are shown.

A complete 3-steps argument chain for the reliability of the system is presented now:

- Step 1: address the aleatory dependency between the objective parameters.
- Step 2: address the epistemic dependency so that only marginal and partial beliefs on the parameters are needed.
- Step 3: Learn the required marginal and partial beliefs from evidence.

An important point is that, over the several steps of the end-to-end assurance argument, *conservatism* is guaranteed, which is an advantage – indeed one could say a necessity – for safety-critical systems.

It has been clearly showed that the quasi-perfection notion is superior to "pure" perfection in that 3-steps argument chain when having *large* (bigger than $\frac{1}{\varepsilon}$) number of failure-free tests evidence. And in principle, the required amount of testing evidence could be reduced by "pruning" on the worst case prior distribution (as shown in section 4.4) and/or combining more other types of evidence.

One important foundation is that the ε here is given. There is more than one feasible approach to select the value of ε, the *pfd* bound that defines quasi-perfection. One may select a value such that the target probability of failure free behaviour over the lifetime of the system is satisfied. Alternatively, considering that there are always trade-offs between confidence *bounds* and confidence *levels*, one can instead, with some additional numerical or algebraic calculations, choose ε such as to get the most favourable claim feasible, within the constraint of required conservatism. The choice of ε is an important future work for the quasi-perfection models.

8. CONCLUSIONS AND FUTURE WORK

In this final chapter, a summary of this thesis is firstly discussed. Then based on a better perceiving of the whole picture of the overall research, a potential framework is proposed to combine the various relating models. Contributions of each chapter are summarised, while some unresolved issues are still apparent, so limitations are summarised as well as contributions, which leads to the discussion of possible future work.

8.1. Summary of the thesis

In chapter 1, we discussed why we are interested in the probability of perfection and why the perfection of software is possible in reality. Examples are illustrated to show that it is not only of interest on its own, but also plays an important role in the assessment of diverse software systems to deal with the inherent problem of dependency.

To better understand what the dependency problem of diverse software is and how probability of perfection solves it, the literature review focuses on the topic of using diversity in software systems. A serial of questions has been looked into: What are software diverse techniques? Is software diversity useful? How to achieve diversity in practice? Finally, how to assess the reliability of a diverse software-based system?

At the end of chapter 2, the original LR and LP models (Littlewood and Rushby 2012; Littlewood and Povyakalo 2013a) was elaborated. As shown at the top left corner in Figure 30, LR and LP models proposed ways of overcoming the difficulties of lack of independence at *aleatory* and *epistemic* level respectively in reliability modelling of 1-out-of-2 software-based systems by introducing the notion of probability of perfection.

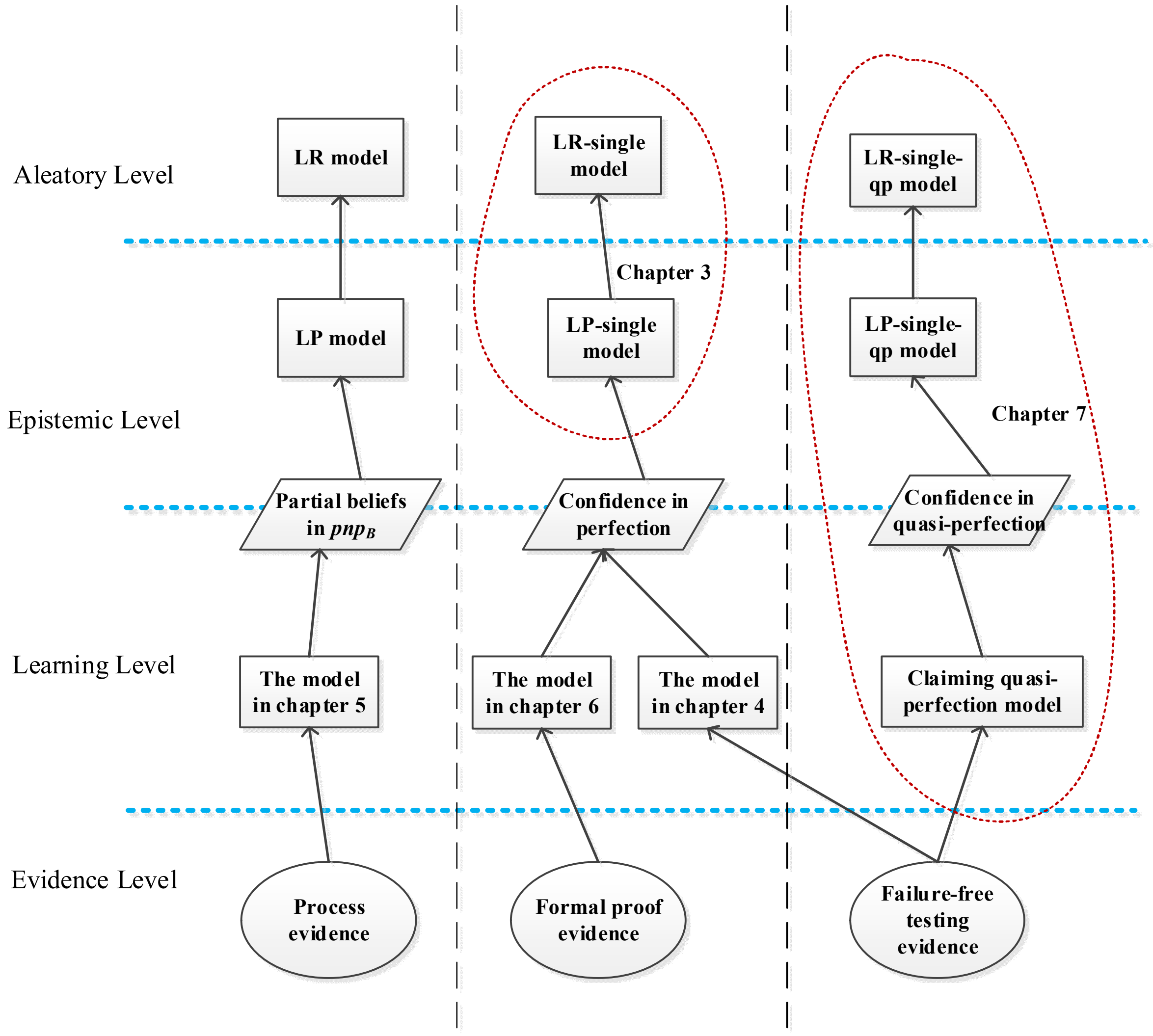

Figure 30 an overview of the thesis – 3 paralleled sets of models. Rectangles represent the various models. Ellipses represent observable evidence. Parallelograms represent subjective beliefs.

The probability of perfection used in original LR and LP models is actually an objective parameter of a *population*, which is a property of the population charactering the development process being used. While for the single software of interest, which is objectively either perfect or not, the probability of the perfection of the single software is a subjective confidence. In chapter 3, the original LR and LP models are modified in terms of subjective probability of perfection, named as LR-single and LP-single models in Figure 30.

Same as the original LR and LP, they work at *aleatory* and *epistemic* level respectively to deal with the dependency problem. Instead of requiring some marginal and partial beliefs on the parameter of a population (i.e. pnp_B), only the confidence in perfection of the *single* software is needed in this case (i.e. confidence in the parameter $isimperfect_B$), assuming that we can know pfd_A from somewhere else.

To help assessors elicit the required confidence in perfection in the LR-single and LP-single models, a conservative Bayesian model was proposed in chapter 4. Via that model, conservative confidence of perfection could be learned when seeing failure-free testing evidence of the single software of interest, therefore being placed at the "learning level" in Figure 30 and demanding the testing evidence from "evidence level".

Intuitively, formal verification result is an important type of evidence for supporting perfection related claims. In chapter 6, a model was built to see how the confidence in perfection of a single program changes when seeing it passes a formal proof.

For the inputs of the original LR and LP models, i.e. marginal and partial beliefs on the objective parameter of a population parameter pnp_B, the model in chapter 5 allows some conservative claims on it via the process evidence. The process evidence is some previous similar products built by the same development process for the same/similar problem have exhibited failure-free working during extensive operational exposure.

Finally in chapter 7, a new notion of "quasi-perfection" is introduced: a small *pfd* is practically equivalent to perfection. Most of the "pure" perfection models could be correspondingly extended into quasi-perfection models which might have advantages over the "pure" perfection models in some circumstances. To show the possible advantage, an end-to-end argument chain was presented in chapter 7, i.e. from new extended LR-single-qp model and LP-single-qp model to claiming quasi-perfection via failure-free runs, and it is guaranteed to be conservative.

To sum up, the work of this thesis describes three parallel sets of models (as Figure 30) which could be used to reason on the reliability of diverse software-based systems. The choice of a lane of models depends on the specific circumstance in practice.

8.2. An overall framework of combining various models

The work shown in Figure 30 is 3 *separated* lanes of models. It would be appealing if the models at the "learning level" could be combined in some way to reason a required belief (which depends on the chosen models at the aleatory and epistemic levels). In other words, can we seek a method to combine various types of evidence to reason either the confidence in perfection, quasi-perfection or beliefs about *pnp*$_B$ (i.e. one of the 3 "ports" between the epistemic level and learning level in Figure 30)?

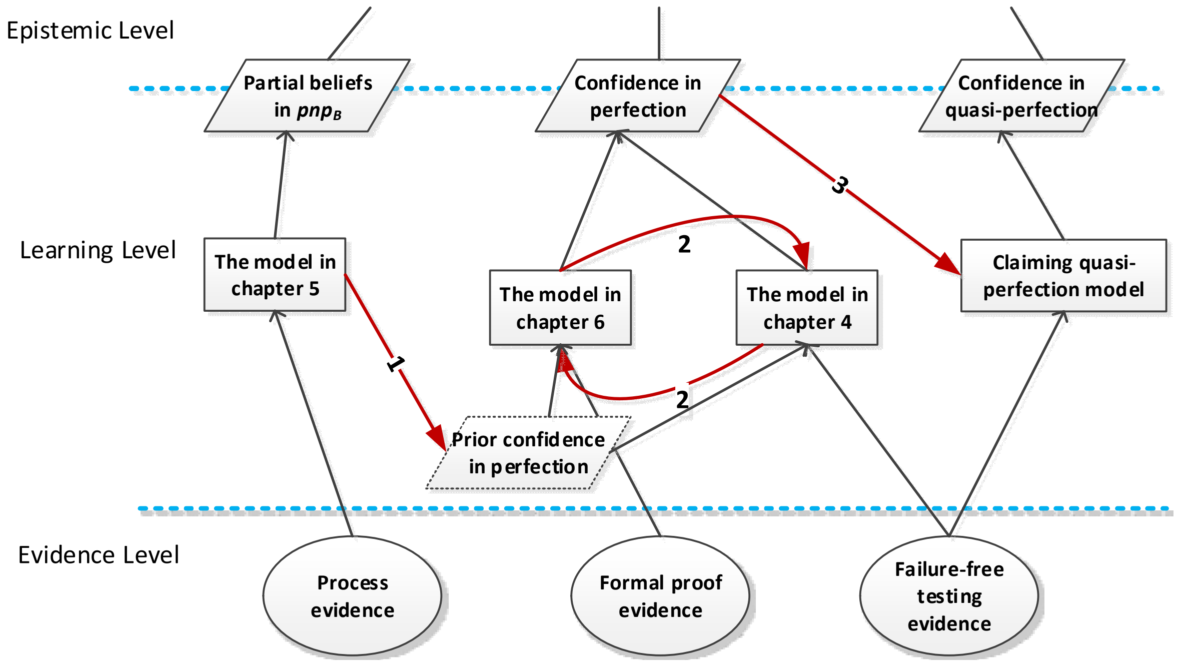

Figure 31 a feasible way to "stitch up" various models at the learning level with 3 type red "strings"

A feasible way to "stitch up" various models at the learning with 3 red "strings" is shown in Figure 31:

- The No.1 red string: the model in chapter 5 is about claiming a posterior confidence bound on the parameter *pnp*$_B$, which could be used as input of the LP model. If we change the objective function of this model to a posterior *mean* value of the *pnp*$_B$ when seeing process evidence, then this posterior mean could be used as an estimate of the *prior* confidence of perfection in the single software of interest which is the input of the models in chapter 4, 6 and 7.
- The No.2 red string: it represents the relation between the two models on claiming perfection of the single software of interest via formal proof evidence and failure-free testing evidence. Essentially the two models are both requiring prior confidence in perfection (which is explicitly plotted in Figure 31) and output a posterior confidence

in perfection. The No.2 red strings means the output of one model could be the input of the other, i.e. the required input of prior confidence in perfection of a model could come from the posterior confidence in perfection from the other.

- The No.3 red string: The perfection related models (used solely or combined) could result in a posterior confidence of perfection which could be used as the required prior confidence in perfection in the quasi-perfection model in chapter 7.

All in all, although there might be other ways to combine the various models, the important insight at the "learning level" is that the priors required by one model could come from the output posteriors of the other models. Then the various types of evidence are combined to reason the reliability of diverse software systems via a chosen set of models at the aleatory and epistemic levels. Depends on the specific way and context of the combining, there is more loose end work to be done, (e.g. shifting the objective function from minimising a confidence bound of pnp_B to minimising the mean of pnp_B). So at this stage, it is a suggestion for feasible future work.

8.3. Evaluation

The contributions and limitations will be discussed chapter by chapter firstly, and then the overall contribution will discussed at the end of this section.

8.3.1. Contributions and limitations of each chapter

- Chapter 3:
    - Contribution 1: explicitly discussed and distinguished two different probabilities of perfection that could be found in current literature, which seems missing in previous work.
    - Contribution 2: rewrite the original LR and LP models in terms of subjective probability of perfection for a single particular system, which is more useful for an assessor working on that specific software.
    - Contribution 3: the discussion on the comparison between the original and modified LR-LP models provides a better understanding of how to use them in different circumstances, which improves the practicality of the two sets of models.

- Limitation 1: The LR-single model seems quite conservative, due to the assumption that if channel B is not perfect then it will fail with certainty whenever channel A fails. In some cases, the current theorems in LP-single model are quite modest too.
- Limitation 2: With some partial prior beliefs, the LR-LP-single models do not show explicit advantages of using the "clear-box" knowledge comparing with the "whole-system-view" approach (section 3.2.3).

- Chapter 4:
    - Contribution 1: The "conservative Bayesian inference for reasoning about system *pfd*" was first proposed in (Bishop, Bloomfield et al. 2011). Section 4.2 applied the similar idea in reasoning the probability of perfection, which confirmed the assertion that the "conservative Bayesian inference" method may have wider applicability.
    - Contribution 2: Together with the "conservative Bayesian inference", the "pruning idea" proposed in section 4.4 seems a new way to elicit priors from the assessor.
    - Limitation 1: If without more information from the assessor, the model is too conservative to be useful. The reason seems to be that testing evidence is weak in supporting perfection (the observation of this limitation could be treated as a contribution in rebutting arguments for perfection with intensive testing evidence).
    - Limitation 2: There is no strong recommendation of how to apply the "pruning idea", rather a pure illustrative example to show the basic principles when eliciting priors in this way.
- Chapter 5:
    - Contribution 1: Informal arguments have long used the process evidence to support system dependability claims. The model here is an attempt to put the argument onto a formal basis via the "conservative Bayesian inference" method again.
    - Contribution 2: to compensate the difficulty of applying the models, a negotiation model was proposed to facilitate the consensus building.
    - Contribution 3: the counter-intuitive result – good news from operational testing of similar products could counter-intuitively decrease the beliefs in

perfection – is providing a warning against the use of unaided informal engineering judgment.

  - Limitation 1: Since we cannot elicit a complete *pfd* distribution, the notion of *R* (an objective parameter of a random selected program is reliable and imperfect) was imposed as a considerable simplification. However, aiding assessors to elicit *R* remains not easy.
  - Limitation 2: The key to the efficacy of the model lies in the number *k* (i.e. the number of versions in the population). In practice, it is unlikely to have a very large *k*.
  - Limitation 3: For brevity, all the results assume that *n* (i.e. the number of testing cases) takes the same value over all *k* systems observed. Clearly, this is unlikely to be true in practice. Ways to overcome this problem are discussed, but they involve ignoring some of the available evidence.
- Chapter 6:
  - Contribution 1: The analysis of the uncertainties (i.e. the two types of faults of provers and the spectrum of formal specifications) lies in a formal proof could be a useful foundation for future modelling work on claiming perfection via formal proof evidence.
  - Contribution 2: the modelling results are justifying some common practice in a formal proof and revealing some pitfalls could be in a safety case.
  - Limitation 1: The current model seems still impractical to give the quantitative posterior confidence in perfection when seeing formal proof evidence (due to requiring some beliefs on the spectrum of formal specification which are not easy to elicit).
- Chapter 7:
  - Contribution 1: it forms a new lane of quasi-perfection related models, which parallels to the perfection related models, see Figure 30. In line with the LR and LP models, the quasi-perfection models enrich the solutions of the dependency issues in the reliability assessment of a 1oo2 diverse system.
  - Contribution 2: an end-to-end argument chain was proposed to show that the quasi-perfection notion is superior to perfection when the amount of testing evidence is large enough.

- Limitation 1: the new LR-single-qp and LP-single-qp models are still giving very conservative results, similar to those in chapter 3.
- Limitation 2: the model on claiming quasi-perfection via failure-free testing evidence is requiring a very large amount of testing evidence when the definition of quasi-perfection is strict (e.g. *pfd* smaller than 10^{-7}).

Even though there are limitations in each chapter and some of them seem hard to overcome,, research directions are discussed in each chapter against the corresponding limitations.

8.3.2. Overall evaluation

From an overall perspective of the thesis, there are firstly two main contributions corresponding to the two research questions in section 1.2:

- The LR-single, LP-single, LR-single-qp and LP-single-qp models in the thesis are broadening the application of the probability of perfection of software-based systems. Each of them has its own advantages over the existing LR and LP models in some circumstances. Depending on the specific context, potential users now have alternative solutions to dependence issue in the reliability assessments of diverse 1oo2 software systems.
- Before this thesis, although much work could be found on *applying* the probability of perfection of software, there seems no literature on how to formally *claim* the probability of perfection via various kinds of available evidence. In this sense, the thesis is a first attempt to reason the probability of perfection of software-based system in rigorous quantitative ways. In addition, the models are compatible in the sense of being able to be combined together in a framework to support (quasi-) perfection claims.

It is important to note that throughout the thesis an important principle of guaranteed conservatism has been applied to deal with the two substantial difficulties lying in the two research questions:

- Tackling the issues of dependence in the reliability assessment of 1-out-of-2 diverse systems at both aleatory and epistemic levels by requiring only *marginal* and *partial* beliefs of each channel.

- Overcoming the well-known difficulty of specifying *complete* Bayesian priors into reasoning with only *partial* prior beliefs.

Both of them are solved at the price of conservatism. Thus the end-to-end results are guaranteed to be conservative, which is obviously desirable and important for safety-critical systems. This "end-to-end conservatism" approach in the presence of all kinds of uncertainty is novel.

This end-to-end conservatism also raises the main limitation that sometimes the results might be too conservative to be useful or quite modest. There are two ways of looking at this problem of possibly excessive conservatism:

- There's a "tension" between being "too partial" in the beliefs expressed, resulting in very conservative results, on the one hand; and requiring too much from the assessor, so that he cannot populate the models with numbers (or the numbers are not believable) on the other hand. The question is how to be "as conservative as necessary, but not more so". So one possible way of looking at the contribution of this thesis is that it is precisely exploring where the acceptable boundary lies between the two sides of the "tension".
- In fact, requiring guaranteed *end-to-end* conservatism is a strong condition – stronger than is usual in some real practical safety cases (where for example "best guess" values for key parameters and/or unjustified assumptions are used). It has been seen as a classical argument that "Our claims for the *pfd*s of the two channels are themselves very conservative: we know that each system is much better than the numbers pfd_A and pfd_B that we are claiming. So, when we use the product $pfd_A \times pfd_B$ for the system *pfd* in our safety case, we can be sure that this is conservative", which is a rather informal and dangerously optimistic reasoning. Actually we do not need a full end-to-end treatment in order to get useful results that are an improvement on that classical $pfd_A \times pfd_B$ approach. Some of the conservatism could be relaxed if we stopped at some stages of a 4 levels end-to-end arguments chain (see Figure 30). Consider, for example, a very crude simplification of the first end-to-end arguments chain in Figure 30. If we have point estimates of pfd_A and pnp_B, we could treat each of these as "true" and use a simple product of them as a bound on system *pfd*. Although ignoring epistemic uncertainty about the parameters in this way is clearly "wrong",

we believe it is nevertheless superior in some ways to the naive product of pfd_A and pfd_B. Specifically, it ignores *only* epistemic uncertainty; the classical approach ignores epistemic uncertainty *and* failure dependence. That is, if we had correct values for pfd_A and pnp_B our result *would* be a bound on system *pfd*. The same is not true of the classical approach: even with true values for pfd_A and pfd_B, their product is not a guaranteed bound on system *pfd*. We would have similar observations for stopping at different stages of different chains of arguments in Figure 30. Curtailing the full end-to-end treatment of uncertainty in ways like this would make the task of the assessor very much easier, but at the price of not ending up with a *guaranteed* conservative claim for the system *pfd* – because some uncertainty will not be accounted for[25]. Nevertheless such reasoning is still superior to the classical $pfd_A \times pfd_B$ approach. Even though we are not advocates for such a curtailing approach, it is available to the user from a practicality point of view.

Take a step back, if you *really* need guaranteed end-to-end conservatism and are only prepared to express rather minimal partial prior beliefs, then the models of this thesis could serve as warnings/support to safety engineers/regulators in the face of claims based on reasoning that is less rigorous than the reasoning in this thesis:

- There is a common conservative view of nuclear regulators for the licensing of safety critical software for nuclear reactors (ONR 2015) that claims of low *pfd* for a computer based system are required to be treated with caution. Generally, the work in this thesis is supportive of this kind of conservative view. It actually provides some formal arguments of this kind of informal scepticism. The results show rigorously what can be claimed in circumstances that are themselves spelled out rigorously.

8.4. Future work

Despite the possible future work of the scattered points in each chapter, four possible directions to expand the work presented in this thesis are as below:

[25] Or, putting it another way, it places on the assessor a requirement to be certain of the values of certain parameters. If with such certainty, there would be guaranteed conservatism in the conclusions.

- **Reduce the conservativeness**: as argued above, the main limitation is the conservatism lies in each model of the thesis. Generally speaking, the conservativeness could be classified into 3 categories:
    - Conservativeness at aleatory level: i.e. the conservative assumptions used in LR, LR-single and LR-single-qp models. The very conservative assumption is if channel B is not perfect then it will fail with certainty whenever channel A fails. One possible solution to this conservatism seems to capture the aleatory uncertainty that B fails when A fails and B is imperfect. Some preliminary work is ongoing in this direction.
    - Conservativeness at epistemic level: More theorems are needed at this level, which might be degraded into pure mathematical problems, e.g. to obtain a tighter bound. Of course, knowing more prior beliefs could improve the usefulness of the results, but it will place harder task on assessors to elicit and learn those beliefs.
    - Conservativeness at learning level: All the models at the learning level are essentially applying the "conservative Bayesian inference" principle. It seems to reduce the inherent conservatism in this approach needs more theorems in the fundamental mathematics, i.e. tighter bound on the objective functions. Otherwise, any solution is essentially switching the difficulties from this level to some other ones.
- **Incorporate more types of evidence**: at the evidence level in Figure 30, there are only 3 types of evidence being considered. However, many other available evidence could be found in a real project seems highly associated with perfection related claims, e.g. symbolic testing evidence. So how to combine more other types of evidence into the perfection related framework of reliability assessment is important.
- **How to elicit conditional or joint priors**: the price of conservatism was paid in the models at learning level to at best facilitate the assessors' task of expressing priors. Assessors may able to express some very partial (e.g. a confidence bound or a mean) and marginal beliefs. However, it is well-known people find it hard to express a conditional probability and/or bivariate uncertainty. In some models of this thesis, such joint priors are needed, e.g. the R in chapter 5 and beliefs of the items on the spectrum of formal specifications in chapter 6. How to rationally elicit them or find ways to around it (maybe at the price of conservatism again) is of future interest.

- **Extend the quasi-perfection idea**: It is believed that the quasi-perfection idea could be furthered extended. When more other types of evidence being incorporated, it might have more advantages over the "pure" perfection models in some cases.

8.5. Final conclusions

This thesis provided 3 parallel sets of (quasi-)perfection related models which could be used individually as a conservative end-to-end argument that from various types of evidence to the reliability of a software-based system. It is not only extending the applications of the probability of perfection of software, but also improving our understanding of it which allows us to claim the probability of perfection of software from various types of evidence.

REFERENCES

Adams, E. (1984) “Optimizing Preventive Service of the Software Products”, IBM Journal of R&D, 28(1):2-14.

Amman, P. E. and J. C. Knight (1987) “Data Diversity: An Approach to Software Fault Tolerance” In Proc. of 17th Intl. Symposium on Fault Tolerant Computing, pages 122-126.

Amman, P. E. and J. C. Knight (1988) “Data Diversity: An Approach to Software Fault Tolerance” IEEE Trans. On Computers, 37 (4), pp. 418-25.

Amjad, H. (2004). “Combining model checking and theorem proving” Doctoral dissertation, University of Cambridge.

Anderson, T and R. Kerr (1976) “Recovery Blocks in Actions: A System Supporting High Reliability.” Newcastle upon Tyne: Computing Laboratory, The University of Newcastle upon Tyne. Computing Laboratory Technical Report Series 93.

Avizienis, A. and Chen, L. (1977). “On the Implementation of N-version Programming for Software Fault Tolerance during Execution”, Proc. the First IEEE-CS International Computer Software and Applications Conference (COMPSAC77), Chicago, Nov 1977.

Avizienis, A., M.R. Lyu and W. Schutz (1988) “In Search of Effective Diversity: A Six Language Study of Fault Tolerant Flight Control Software”, in Eighteenth International Symposium on Fault Tolerant Computing (FTCS 18), Tokyo.

Baudry, B and M. Monperrus (2015) “The Multiple Facets of Software Diversity: Recent Developments in Year 2000 and Beyond” ACM Comput. Surv. 48, 1, Article 16 (September 2015), 26 pages.

Bertolino, A. and Strigini, L. (1998). “Assessing the Risk due to Software Faults: Estimates of Failure Rate versus Evidence of Perfection”. Software Testing, Verification and Reliability, 8(3), 155 - 166.

Bishop, P., R. Bloomfield, B. Littlewood, A. Povyakalo and D. Wright (2011). “Towards formalism for conservative claims about the dependability of software-based systems.” IEEE Transactions on Software Engineering 35(5): 708-717.

Bishop, P.G., D.G. Esp, M. Barnes, P. Humphreys, G. Dahll and J. Lahti (1986). “PODS—A Project on Diverse Software”, IEEE Trans. Software Engineering, Vol SE-12, No 9, pp. 929-940.

Bishop, P.G., D.G. Esp, F.D. Pullen, M. Barnes, P. Humphreys, G. Dahll, B. Bjarland, H. Valisuo (1987). “STEM—Project on Software Test and Evaluation Methods”, Proc. Safety and Reliability Society Symposium (SARSS 87), Manchester, Elsevier Applied Science.

Bishop, P.G. and F.D. Pullen (1991). “Error Masking: A Source of Dependency in Multi-version Programs”, Dependable Computing and Fault Tolerant Systems, Vol. 4. Pp. 53-73.

Bloomfield, R., and Bishop, P. (2010). “Safety and assurance cases: Past, present and possible future - an Adelard perspective”. In Making Systems Safer pp. 51-67. Springer London.

Bloomfield, R. E. and B. Littlewood (2003). “Multi-legged arguments: The impact of diversity upon confidence in dependability arguments” in: Proceedings DSN 2003, IEEE Computer Society, pp. 25–34.

Bloomfield, R., B. Littlewood and D. Wright (2007). “Confidence: Its Role in Dependability Cases for Risk Assessment,” Proc. Int. Conf. Dependable Systems and Networks. Pp. 338 – 346.

Boeing (2015). “Statistical Summary of Commercial Airplane Accidents, Worldwide Operations, 1959-2014.” Seattle, Aviation Safety, Boeing Commercial Airplanes.

Butler, R. W. and G. B. Finelli (1993). “The infeasibility of quantifying the reliability of life-critical real-time software.” IEEE Trans Software Engineering 19(1): 3-12.

CAA (2001). “Regulatory Objective for Software Safety Assurance in Air Traffic Service Equipment” Civil Aviation Authority SW01.

Cotroneo, D., Natella, R., Pietrantuono, R., and Russo, S. (2014) “A survey of software aging and rejuvenation studies”, ACM J. Emerg. Technol. Comput. Syst. 10, 1, Article 8, 34.

Cukic, B., Gunal, E., Singh, H., and Guo, L. (2003), “The Theory of Software Reliability Corroboration” IEICE Trans on Information Systems, Vol E86-D, No 10, pp 2121 -2129.

Cuoq, P., Kirchner, F., Kosmatov, N., Prevosto, V., Signoles, J., and Yakobowski, B. (2012) “Frama-C”. In Software Engineering and Formal Methods, pp. 233-247. Springer Berlin Heidelberg.

Dahll, G. and J. Lahti (1979). “An investigation into the methods of production and verification of highly reliable software”, Proc. SAFECOMP 79.

De Millo, R. A., Lipton, R. J., and A. J. Perlis. (1979) “Social processes and proofs of theorems and programs” Commun. ACM 22, 5, 271-280.

Dunham, J.R. (1986). “Experiments in software reliability: life critical applications”, IEEE Trans. Software Engineering, Vol. SE-12, No. 1, pp. 110 – 123.

Eckhardt, D. E., A. K. Caglayan, J. C. Knight, L. D. Lee, D. F. McAllister, M. A. Vouk, and J. P. J. Kelly, (1991). “An experimental evaluation of software redundancy as a strategy for improving reliability,” IEEE Transactions on Software Engineering, vol. 17, no. 7, pp. 692–702.

Eckhardt, D.E. and L.D. Lee, (1985) “A Theoretical Basis of Multi-version Software Subject to Coincident Errors,” IEEE Trans. on Software Engineering, vol. 11, pp.1511-1517.

FAA (1988). “ Advisory Circular 25.1309-1A: System design and analysis.” Washington DC, Federal Aviation Administration.

Fetzer, J. H. (1988). “Program verification: the very idea”, Commun. ACM 31, 9 (September 1988), 1048-1063.

Gashi, I., Popov, P. T. & Strigini, L. (2004). “Fault diversity among off-the-shelf SQL database servers”. Paper presented at the International Conference on Dependable Systems and Networks, 28 Jun - 1 Jul 2004.

Gmeiner, L. and U. Voges (1980). “Software diversity in reactor protection systems: an experiment” in Safety of computer control systems, R. Lauber, Ed., New York: Pergamon.

Hatton, L. (1997). “N-version design versus one good version,” IEEE Software, vol. 14, no. 6.

Havelund, K., M. Lowry, and J. Penix. (2001) “Formal analysis of a space craft controller using SPIN” IEEE Trans. on Software Engineering, 27(8):749–765.

Horning, J. J., H. C. Lauer, P. M. Melliar-Smith and B. Randell.(1974). “A Program Structure for Error Detection and Recovery”, Lecture Notes in Computer Science, 16:177-193, 1974.

Hommersom, A., Groot, P., Lucas, P. J., Balser, M., and Schmitt, J. (2007) “Verification of medical guidelines using background knowledge in task networks” Knowledge and Data Engineering, IEEE Transactions on, 19(6), 832-846.

HSE (1992). “Safety Assessment Principles for Nuclear Plants,” Health and Safety Executive ISBN 011 882043 5.

HSE (2011). “Step 4 Control and Instrumentation Assessment of the EDF and AREVA UK EPR Reactor.” Bootle, Health and Safety Executive, Office for Nuclear Regulation.

Hunns, D. M. and N. Wainwright (1991). “Software-based protection for Sizewell B: the regulator's perspective” Nuclear Engineering International, vol. September, pp. 38-40.

Huang, Y., Kintala, C., Kolettis, N., & Fulton, N. D. (1995) “Software rejuvenation: Analysis, module and applications” In Fault-Tolerant Computing, 1995. FTCS-25. Digest of Papers, Twenty-Fifth International Symposium on pp. 381-390.

Jalote, P., Y. Huang and C. Kintala (1995) “A Framework for Understanding and Handling Transient Software Failures”, In Proc. 2nd ISSAT Intl. Conf. on Reliability and Quality in Design, Orlando, Florida.

Kelly, J.P.J. and A. Avizienis (1983). “A specification-oriented multi-version software experiment”, in Thirteenth International Symposium on Fault Tolerant Computing(FTCS 13), Milan.

Kelly, J.P.J., D.E. Eckhardt, M.A. Vouk, D.F. McAllister, A. Caglayan (1988) “A Large Scale Second Generation Experiment in Multi-Version Software: Description and Early Results”, in Eighteenth International Symposium on Fault Tolerant Computing (FTCS 18), Tokyo, June 1988

Kemmerer, R. A. (1985). “Testing formal specifications to detect design errors”, Software Engineering, IEEE Transactions on, (1), 32-43.

Knight, J. C. and N. G. Leveson (1986). “An experimental evaluation of the assumption of independence in multi-version programming,” IEEE Transactions on Software Engineering, vol. SE-12, no. 1, pp. 96–109.

Kuball, S., J. May, and G. Hughes (1999) “Structural Software Reliability Estimation” In Proceedings of the 18th International Conference on Computer Safety, Reliability and Security. Toulouse, France: Springer.

Laprie, J. C., Arlat, J., Beounes, C. and Kanoun, K (1990). “Definition and analysis of hardware and-software fault-tolerant architectures.” IEEE Comput. 23, 7, 39–51.

Li, M. and Smidts, C. (2000). “A Ranking of Software Engineering Measures Based on Expert Opinion,” IEEE Transactions on Software Engineering, Vol 29, No 9, pp 811-824, Sept 2003.

Littlewood, B. and D. R. Miller (1989). “Conceptual Modelling of Coincident Failures in Multi-Version Software,” IEEE Trans on Software Engineering, vol. 15, no. 12, pp.1596-1614.

Littlewood, B., Popov, P. T. and Strigini, L. (1999). “A note on reliability estimation of functionally diverse systems”. Reliability Engineering and System Safety. 66, 93–95.

Littlewood, B., Popov, P. T. and Strigini, L. (2000a). “N-version design versus one good version” in International Conference on Dependable Systems and Networks, New York, IEEE Computer Society.

Littlewood, B., Popov, P. T. and Strigini, L. (2000b). “Assessment of the reliability of fault-tolerant software: a Bayesian approach.” 19th International Conference on Computer Safety, Reliability and Security, SAFECOMP’2000, Rotterdam, the Netherlands, Springer.

Littlewood, B., Popov, P. T. and Strigini, L. (2002). “Modelling software design diversity - a review”. ACM Computing Surveys 33(2): 177-208.

Littlewood, B., P. Popov, L. Strigini and N. Shryane (2000c) “Modelling the effects of combining diverse software fault removal techniques,” IEEE Transactions on Software Engineering. SE-26(12): p.1157-1167.

Littlewood, B. and Povyakalo, A. A. (2013a). “Conservative Reasoning about the Probability of Failure on Demand of a 1-out-of-2 Software-Based System in Which One Channel Is ‘Possibly Perfect’”, IEEE Transactions on Software Engineering, vol. 39, no. 11, pp. 1521-1530.

Littlewood, B. and Povyakalo, A. A. (2013b). “Conservative Bounds for the pfd of a 1-out-of-2 Software-Based System Based on an Assessor's Subjective Probability of ‘Not Worse Than Independence’”, IEEE Transactions on Software Engineering, vol. 39, no. 12, pp. 1641 – 1653.

Littlewood, B. and Rushby, J. (2012). “Reasoning about the Reliability of Diverse Two-Channel Systems in which One Channel is ‘Possibly Perfect’.” IEEE Transactions on Software Engineering, vol. 38, no. 5, pp. 1178-1194.

Littlewood, B. and Strigini, L. (1993). “Validation of ultra-high dependability for software-based systems.” CACM 36(11): 69-80.

Littlewood, B. and L. Strigini, (2000). “A Discussion of Practices for Enhancing Diversity in Software Designs”, DISPO project reports, LS_DI_TR-04, City University, London, http://openaccess.city.ac.uk/275/2/DISPO_LS_DI_TR-04.pdf

Littlewood, B. and Wright, D. (1995). “A Bayesian Model that Combines Disparate Evidence for the Quantitative Assessment of System Dependability.” in 14th International Conference on Computer Safety (SafeComp’95), pp. 11 – 13. Belgirate, Italy.

Littlewood, B. and Wright, D. (1997). “Some Conservative Stopping Rules for the Operational Testing of Safety-Critical Software.” IEEE Transactions on Software Engineering, vol. 23, no. 11, pp. 673-683.

Littlewood, B. and D. Wright (2007). “The Use of Multi-Legged Arguments to Increase Confidence in Safety Claims for Software-Based Systems: A Study Based on a BBN of an Idealized Example” IEEE Trans. Software Eng., vol. 33, no. 5, pp. 347-365.

Lyu, M. R. and Y. He (1993). “Improving the N-Version Programming Process through the Evolution of a Design Paradigm”, Proc. IEEE Trans. Reliability, Vol. 42, No. 2, pp. 179 – 189.

Melliar-Smith, P. M., & Schwartz, R. L. (1982). “Formal specification and mechanical verification of SIFT: A fault-tolerant flight control system” Computers, IEEE Transactions on, 100(7), 616-630.

Miller, K. W., Morell, L. J., Noonan, R. E., Park, S. K., Nicol, D. M., Murrill, B. W., & Voas, M. (1992). “Estimating the probability of failure when testing reveals no failures”. IEEE Transactions on Software Engineering, 18(1), 33-43.

MoD (1997). “The Procurement of Safety Critical Software in Defence Equipment” Ministry of Defence Def-Stan 00-55, Issue 2, August, 1997.

Musa, J. D., and A. F. Ackerman (1989). “Quantifying software validation: when to stop testing?” IEEE Software 6.3: 19-27.

Oberkampf, W. L. and J. C. Helton (2004) “Alternative representations of epistemic uncertainty,” Reliability Engineering and System Safety, vol. 85, no. 1–3, pp. 1–10.

ONR (2015), Office for Nuclear Regulation, UK, http://www.onr.org.uk/software.pdf

Petiot, G., Kosmatov, N., Giorgetti, A., and Julliand, J. (2014) “How test generation helps software specification and deductive verification in Frama-C” In Tests and Proofs, pp. 204-211, Springer International Publishing.

Petiot, G., Kosmatov, N., Botella, B., Giorgetti, A., and Julliand, J. (2015). “Your Proof Fails? Testing Helps to Find the Reason” arXiv preprint arXiv:1508.01691.

Popov, P. T. (2002) “Reliability Assessment of Legacy Safety-Critical Systems Upgraded with Off-the-Shelf Components” in Proceedings of the 21st International Conference on Computer Safety, Reliability and Security. Pp. 139-150.

Popov, P. T. (2013). “Reliability Assessment of Legacy Safety-Critical Systems Upgraded with Fault-Tolerant Off-the-Shelf Software” Reliability Engineering and System Safety 117, pp.98–113.

Popov, P., V. Stankovic, and L. Strigini, (2012). “An Empirical Study of the Effectiveness of ‘Forcing Diversity’ Based on a Large Population of Diverse Programs,” Proc. IEEECS 23rd Int’l Symp. Software Reliability Engineering.

Popov, P. and L. Strigini (1998). “Conceptual Models for the Reliability of Diverse Systems - New Results” in 28th International Symposium on Fault Tolerant Computing (FTCS-28), pp. 80-89, Munich, IEEE Computer Society.

Popov, P., L. Strigini and B. Littlewood. (2000) “Choosing between fault tolerance and increased V&V for improving reliability.” in International Conference on Parallel and Distributed Processing Technologies and Applications, Las Vegas, Nevada, CSREA Press.

Popov, P., L. Strigini and A. B. Romanovsky (1999). “Choosing Effective Methods for Design Diversity - How to Progress from Intuition to Science”. In the 18th International Conference on Computer Safety, Reliability and Security. Toulouse, France, pp. 272-285.

Popov, P. and Strigini, L. (2010). “Assessing Asymmetric Fault-Tolerant Software”. Paper presented at the Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on, 1-4 Nov 2010, San Jose, California.

Popov, P., Strigini, L., May, J., and Kuball, S. (2003). “Estimating Bounds on the Reliability of Diverse Systems”, Software Engineering, IEEE Transactions on, 29(4), 345-359.

Randell, B. and Xu, J. (1995) “The Evolution of the Recovery Block Concept.” In: Lyu, M, ed. Software Fault Tolerance. Chichester: John Wiley & Sons, pp.1-22.

Rouquet, J. C. and P. J. Traverse (1986). “Safe and Reliable Computing on Board the Airbus and ATR aircraft.” Safecomp: 5th IFAC Workshop on Safety of Computer Control Systems, Pergamon Press.

Prevosto,V. (2013) “ACSL Mini-Tutorial”. http://frama-c.com/download/acsl-tutorial.pdf

Rushby, J. (2009) “Software Verification and System Assurance,” Proc. Seventh Int’l Conf. Software Eng. and Formal Methods, D.V. Hung and P. Krishnan, eds., pp. 3-10, Nov. 2009.

Rushby, J., Littlewood, B. and Strigini, L. (2014). “Evaluating the Assessment of Software Fault-Freeness” Paper presented at the AESSC 2014, 13-05-2015, Newcastle upon Tyne.

Salako, K. and Strigini, L. (2013). “When Does ‘Diversity’ in Development Reduce Common Failures? Insights from Probabilistic Modelling”. IEEE Transactions on dependable and secure computing, Vol. 11, No. 2. Pp. 193-206.

Shiraishi, S., V. Mohan and H. Marimuthu (2015) “Test suites for benchmarks of static analysis tools” Software Reliability Engineering Workshops (ISSREW), 2015 IEEE International Symposium on, Gaithersburg, MD, 2015, pp. 12-15.

Smidts, C. and Li, M. (2000), “Software Engineering Measures for Predicting Software Reliability in Safety Critical Digital Systems,” NUREG/GR-001 9, Nuclear Regulatory Commission, Washington DC, November 2000.

Smith, I. C., D. N. Wall, and J. A. Baldwin (1991) “DARTS – An experiment into cost of and diversity in safety critical computer systems.” In IFAC/ IFIP/EWICS/SRE Symposium on Safety of Computer Control Systems (SAFECOMP '91). Trondheim, Norway.

Strigini L. (1994) “Engineering judgement in reliability and safety and its limits: what can we learn from research in psychology?” SHIP Project Technical Report T/030.

Strigini, L. and Povyakalo, A. A. (2013). “Software fault-freeness and reliability predictions” Paper presented at the SAFECOMP 2013, 32nd International Conference on Computer Safety, Reliability and Security, 24 - 27 September 2013, Toulouse, France.

Strigini, L. and Wright, D. (2014). “Bounds on Survival Probability Given Mean Probability of Failure per Demand; And the Paradoxical Advantages of Uncertainty”. Reliability Engineering and System Safety, 128, pp. 66-83.

Trivedi, Kishor S., and Kalyanaraman Vaidyanathan. (2002) “Software reliability and rejuvenation: Modeling and analysis.” IFIP International Symposium on Computer Performance Modeling, Measurement and Evaluation. pp. 318-345. Springer Berlin Heidelberg.

Van der Meulen, M.J.P. and M.A. Revilla (2008). “The Effectiveness of Software Diversity in a Large Population of Programs”. IEEE Transactions on Software Engineering, 34(6). pp. 753-764.

Voges, U (1994). “Software Diversity”, Reliability Engineering and System Safety, vol. 43, Issue 2, pp. 103–110.

Zhao, X., Littlewood, B., Povyakalo, A., and Wright, D. (2015) “Conservative Claims about the Probability of Perfection of Software-based Systems”, IEEE 26th International Symposium on Software Reliability Engineering (ISSRE2015), pp. 130-140.

APPENDIX A

Proof for the Theorem 1 in section 3.2.2:

Denote the probability mass (red bar) at the corresponding area as M_1, M_2, M_3 and M_4 (and also use them as marks of the areas) in the Figure 32, then $Pr(pfd_A \geq P_A, isimperfect_B = 1) = M_4$. By (3.6) and (3.7), $M_4 + M_3 = 1 - \theta$ and $M_4 + M_2 = \alpha_A$. Therefore $0 \leq M_4 \leq \min\{\alpha_A, 1-\theta\}$. Then:

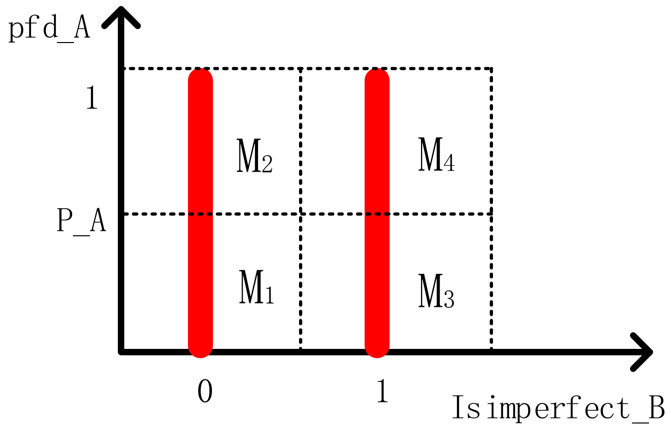

Figure 32 the joint distribution over two objective parameters *pfd$_A$* and *isimperfect$_B$*

$$\begin{aligned} &Pr(\text{sys fails}) \leq E(pfd_A \times isimperfect_B) \\ &= \int_0^{P_A} pfd_A \times 0 dpfd_A + \int_{P_A+}^{1} pfd_A \times 0 dpfd_A + \int_0^{P_A} pfd_A \times 1 dpfd_A \\ &\qquad + \int_{P_A+}^{1} pfd_A \times 1 dpfd_A \end{aligned} \tag{A.1}$$

$$\begin{aligned}
&\le P_A \times 0 \times M_1 + 1 \times 0 \times M_2 + P_A \times 1 \times M_3 + 1 \times 1 \times M_4 \\
&= P_A \times (1 - \theta - M_4) + M_4 \\
&= P_A \times (1 - \theta) + (1 - P_A) M_4 \\
&\le (1 - \theta) P_A + (1 - P_A) min\{\alpha_A, 1 - \theta\} \qquad \text{(A.2)}
\end{aligned}$$

The result (A.2) can be seen as follows. The first line of (A.1) is an expression of the full probability over the 4 cells in Figure 32. Then the inequality is derived by replacing the 4 probabilities with upper bounds. Or an informal explanation could be that the product $pfd_A \times isimperfect_B$ is 0 within the areas M$_1$ and M$_2$. Thus the contribution to $E(pfd_A \times isimperfect_B)$ associated with the two areas is 0. Hence the first two terms in (A.1). In the area M$_3$, the product $pfd_A \times isimperfect_B$ is a random variable which is everywhere smaller than $P_A \times 1$ and the probability mass is M$_3$. Thus the contribution to $E(pfd_A \times isimperfect_B)$ associated with that area is bounded by $P_A \times 1 \times M_3$. Hence the third term in (A.1). Similarly, the area M$_4$ gives the fourth term in (A.1).

QED

Proof for the Theorem 2 in section 3.2.2:

With the newly added certain upper bound, the distribution in Figure 32 changed to Figure 33:

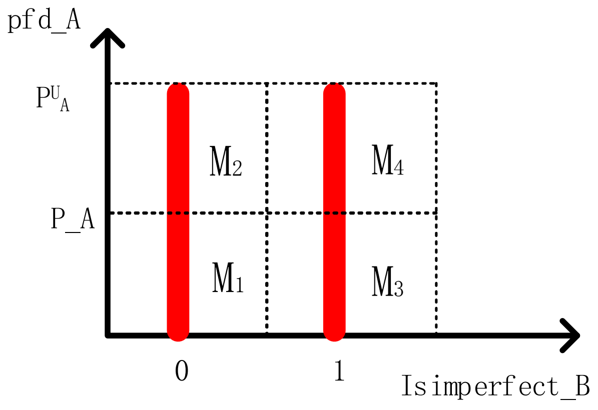

Figure 33 the joint distribution over two objective parameters *pfd_A* and *$isimperfect_B$* with a certain upper bound on *pfd_A*

Again, denote the probability mass (red bar) at the corresponding area as M$_1$, M$_2$, M$_3$ and M$_4$ (and also use them as the marks of the areas) in the Figure 33, then $P(P_A^U > pfd_A \ge P_A, isimperfect_B = 1) = M_4$. Also by (3.6) and (3.7), $M_4 + M_3 = 1 - \theta$ and $M_4 + M_2 = \alpha_A$, therefore $0 \le M_4 \le \min\{\alpha_A, 1 - \theta\}$. Then, similar as the proof for theorem 1.

$$Pr(\text{sys fails}) \le E(pfd_A \times isimperfect_B)$$

$$\begin{aligned}
&\leq P_A \times 0 \times M_1 + P_A^U \times 0 \times M_2 + P_A \times 1 \times M_3 + P_A^U \times 1 \times M_4 \\
&= P_A \times (1 - \theta - M_4) + P_A^U \times M_4 \\
&= P_A \times (1 - \theta) + (P_A^U - P_A)M_4 \\
&\leq (1 - \theta)P_A + (P_A^U - P_A)min\{\alpha_A, 1 - \theta\} \qquad \text{(A.3)}
\end{aligned}$$

QED

Proof for the Theorem 3 in section 3.2.2:

By the Cauchy-Schwarz inequality,

$$\left(E(pfd_A \times isimperfect_B)\right)^2 \leq E\left(pfd_A{}^2\right) \times E\left(isimperfect_B{}^2\right) \qquad \text{(A.4)}$$

As $E\left(isimperfect_B{}^2\right) = E(isimperfect_B) = 1 - \theta$, then:

$$\begin{aligned}
&Pr(\text{sys fails}) \leq E(pfd_A \times isimperfect_B) \\
&\leq \sqrt{E\left(pfd_A{}^2\right) \times E\left(isimperfect_B{}^2\right)} = \sqrt{E\left(pfd_A{}^2\right) \times (1 - \theta)} \qquad \text{(A.5)} \\
&= \sqrt{(1 - \theta)\left(E(pfd_A)^2 + Var\ (pfd_A)\right)}
\end{aligned}$$

Also because: $E(pfd_A)^2 + Var\ (pfd_A) < \left(E(pfd_A) + \ SD\ (pfd_A)\right)^2$, then (A.5) turns into:

$$\begin{aligned}
&Pr(\text{sys fails}) \leq E(pfd_A \times isimperfect_B) \\
&\leq \sqrt{(1 - \theta)\left(E(pfd_A)^2 + Var\ (pfd_A)\right)} \qquad \text{(A.6)} \\
&< \sqrt{(1 - \theta)}\left(E(pfd_A) + \ SD\ (pfd_A)\right)
\end{aligned}$$

QED

Proof for the Theorem 4 in section 3.2.2

It is trivial to know the most conservative distribution of *pfd_A*, satisfying the mean (i.e. $E(pfd_A) = M_A$) and a certain upper bound constraints (i.e. (3.9)), is a two point one as Figure 34. Here the conservatism means to maximize the variance of pfd_A, i.e. makes the distribution as spread out as possible. Then the variance is

$$Var(pfd_A) = (M_A - P_A^U)^2 \frac{M_A}{P_A^U} + (M_A - 0)^2 \left(1 - \frac{M_A}{P_A^U}\right) = M_A P_A^U - M_A{}^2 \qquad \text{(A.7)}$$

And by the result in Theorem 3:

$$Pr(\text{sys fails}) \leq E(pfd_A \times isimperfect_B)$$
$$\leq \sqrt{(1-\theta)\big(E(pfd_A)^2 + Var\,(pfd_A)\big)} = \sqrt{(1-\theta)(M_A P_A^U)} \qquad \text{(A.8)}$$

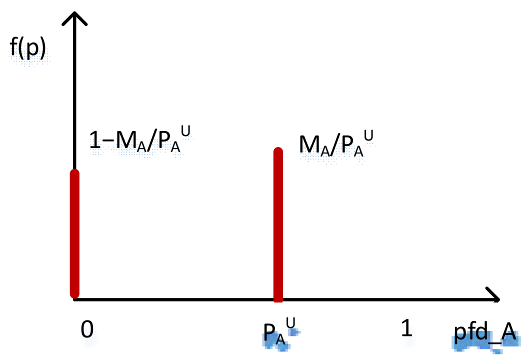

Figure 34 the most conservative (i.e. to maximise the variance) 2-point distribution of *pfd_A* satisfying a mean and a certain upper bound constraints.

QED

APPENDIX B

The problem is to find the most conservative prior distribution *f(p)*, still satisfying (4.1) and (4.2), which minimizes (4.3), then calculate the corresponding posterior probability mass at the origin, i.e. θ^*.

By the mean value theorem for integrals, we could find two values, say P_1 and P_2, satisfying the equations below:

$$(1-P_1)^n \int_{0+}^{y} f(p)dp = \int_{0+}^{y} (1-p)^n f(p)dp \tag{B.1}$$

$$(1-P_2)^n \int_{y+}^{1} f(p)dp = \int_{y+}^{1} (1-p)^n f(p)dp \tag{B.2}$$

, where $0 < P_1 \leq y, y < P_2 \leq 1$.

From the prior constraints (4.1) and (4.2), we know:

$$\int_{0+}^{y} f(p)dp = 1 - x - \theta \tag{B.3}$$

$$\int_{y+}^{1} f(p)dp = x \tag{B.4}$$

So put (B.3), (B.4) into (B.1), (B.2):

$$(1-P_1)^n(1-x-\theta)=\int_{0+}^{y}(1-p)^n f(p)dp \tag{B.5}$$

$$(1-P_2)^n\,x=\int_{y+}^{1}(1-p)^n f(p)dp \tag{B.6}$$

Then put (B.5) and (B.6) into (4.3), we can get:

$$\theta^* =\frac{\theta}{\theta+\int_{0+}^{1}(1-p)^n f(p)dp}=\frac{\theta}{\theta+(1-P_1)^n(1-x-\theta)+(1-P_2)^n\,x} \tag{B.7}$$

To minimise (B.7), we can easily see that when both P_1 and P_2 reach their lower bound, θ* reaches its lower bound. That is, when P_1 =0 and P_2 =y.

$$\theta^* =\frac{\theta}{\theta+(1-P_1)^n(1-x-\theta)+(1-P_2)^n\,x}\geq\frac{\theta}{1-x+(1-y)^n x} \tag{B.8}$$

Here, strictly, P_1 cannot reach the 0 point but infinitely close to it, so this most conservative prior *f(p)* is a “special” 3 points distribution, as shown in Figure 5, i.e. the 0 point with mass θ, the P_1 point which is infinitely close to the original point with mass $1-x-\theta$, and the P_2 point at *y* with mass *x*. There is no such distribution in reality, please see footnote 8 for detailed clarification.

APPENDIX C

Proof for the most conservative prior distribution in Figure 11:

By the mean value theorem for integrals, we could find two values, say P_1 and P_2, satisfying the equations below:

$$L(P_1)\int_0^{y-} g(\theta_{PP})d\theta_{PP} = \int_0^{y-} L(\theta_{PP})g(\theta_{PP})d\theta_{PP}$$

$$L(P_2)\int_y^1 g(\theta_{PP})d\theta_{PP} = \int_y^1 L(\theta_{PP})g(\theta_{PP})d\theta_{PP}$$

where the L is the likelihood function as the expression (5.2) and $0 \le P_1 < y, y \le P_2 \le 1$.

As the one percentile prior knowledge $Pr(\theta_{PP} < y) = \alpha_\theta$, we know:

$$\int_0^{y-} g(\theta_{PP})d\theta_{PP} = \alpha_\theta$$

$$\int_y^1 g(\theta_{PP})d\theta_{PP} = 1 - \alpha_\theta$$

Therefore the objective function:

$$\alpha_\theta{}^* = \frac{\int_0^{y-} L(\theta_{PP})g(\theta_{PP})d\theta_{PP}}{\int_0^1 L(\theta_{PP})g(\theta_{PP})\,d\theta_{PP}} = \frac{L(P_1) * \alpha_\theta}{L(P_1) * \alpha_\theta + L(P_2) * (1 - \alpha_\theta)} = \frac{1}{1 + \frac{L(P_2) * (1 - \alpha_\theta)}{L(P_1) * \alpha_\theta}}$$

As the likelihood function L is an increase function, so:

$$\alpha_\theta{}^* = \frac{1}{1 + \frac{L(P_2) * (1 - \alpha_\theta)}{L(P_1) * \alpha_\theta}} \le \frac{1}{1 + \frac{L(y) * (1 - \alpha_\theta)}{L(y) * \alpha_\theta}} = \alpha_\theta$$

The inequality step is showing the most conservative case is the one when both P_1 and P_2 are at the *y* point (actually P_1 is *infinitely* close to *y* point, please see the very similar point in footnote 8 for clarification), i.e. the most conservative prior $g(\theta_{PP})$ showed in Figure 11.

Then, with the most conservative prior $g(\theta_{PP})$ showed in Figure 11, we cannot learn anything about the probability of perfection, i.e. θ_{PP} from the process evidence.

QED

Proof for the most conservative prior distribution in Figure 13:

By the mean value theorem for integrals, we could find 3 values, say P_1, P_2 and P_3 satisfying the equations below:

$$L(P_1)\int_0^{y_1-} g(\theta_{PP})d\theta_{PP} = \int_0^{y_1-} L(\theta_{PP})g(\theta_{PP})d\theta_{PP}$$

$$L(P_2)\int_{y_1}^{y_2-} g(\theta_{PP})d\theta_{PP} = \int_{y_1}^{y_2-} L(\theta_{PP})g(\theta_{PP})d\theta_{PP}$$

$$L(P_3)\int_{y_2}^{1} g(\theta_{PP})d\theta_{PP} = \int_{y_2}^{1} L(\theta_{PP})g(\theta_{PP})d\theta_{PP}$$

where the L is the likelihood function as the formula (5.2). And

$$0 \le P_1 < y_1, y_1 \le P_2 < y_2, y_2 \le P_3 \le 1$$

As the two percentile prior knowledge – $Pr(\theta_{PP} < y_1) = \alpha_{\theta 1}$ and $Pr(y_1 \le \theta_{PP} < y_2) = \alpha_{\theta 2}$. we know:

$$\int_0^{y_1-} g(\theta_{PP})d\theta_{PP} = \alpha_{\theta 1}$$

$$\int_{y_1}^{y_2-} g(\theta_{PP})d\theta_{PP} = \alpha_{\theta 2}$$

$$\int_{y_2}^{1} g(\theta_{PP})d\theta_{PP} = 1 - \alpha_{\theta 1} - \alpha_{\theta 2}$$

Therefore our two objective functions:

$$\alpha_{\theta 1}^{*} = \frac{\int_0^{y_1-} L(\theta_{PP})g(\theta_{PP})d\theta_{PP}}{\int_0^1 L(\theta_{PP})g(\theta_{PP})\,d\theta_{PP}} = \frac{L(P_1) * \alpha_{\theta 1}}{L(P_1) * \alpha_{\theta 1} + L(P_2) * \alpha_{\theta 2} + L(P_3) * (1 - \alpha_{\theta 1} - \alpha_{\theta 2})}$$

$$= \frac{1}{1 + \frac{L(P_2) * \alpha_{\theta 2} + L(P_3) * (1 - \alpha_{\theta 1} - \alpha_{\theta 2})}{L(P_1) * \alpha_{\theta 1}}}$$

$$\alpha^*_{\theta 1\cap\theta 2} = \frac{\int_0^{y_2-} L(\theta_{PP})g(\theta_{PP})d\theta_{PP}}{\int_0^1 L(\theta_{PP})g(\theta_{PP})\,d\theta_{PP}} = \frac{L(P_1)*\alpha_{\theta 1} + L(P_2)*\alpha_{\theta 2}}{L(P_1)*\alpha_{\theta 1} + L(P_2)*\alpha_{\theta 2} + L(P_3)*(1-\alpha_{\theta 1}-\alpha_{\theta 2})}$$

$$= \frac{1}{1+\frac{L(P_3)*(1-\alpha_{\theta 1}-\alpha_{\theta 2})}{L(P_1)*\alpha_{\theta 1}+L(P_2)*\alpha_{\theta 2}}}$$

As the likelihood function L is an increase function, for $\alpha^*_{\theta 1}$:

$$\alpha^*_{\theta 1} = \frac{1}{1+\frac{L(P_2)*\alpha_{\theta 2}+L(P_3)*(1-\alpha_{\theta 1}-\alpha_{\theta 2})}{L(P_1)*\alpha_{\theta 1}}}$$

$$\leq \frac{1}{1+\frac{L(y_1)*\alpha_{\theta 2}+L(y_2)*(1-\alpha_{\theta 1}-\alpha_{\theta 2})}{L(y_1)*\alpha_{\theta 1}}}$$

It reaches the upper bound when

$$P_1 = y_1^-, P_2 = y_1, P_3 = y_2$$

So we got the most conservative prior distribution in the Figure 13 (left-hand side) and the corresponding $\alpha^*_{\theta 1}$ as (5.7).

Similarly for $\alpha^*_{\theta 1\cap\theta 2}$:

$$\alpha^*_{\theta 1\cap\theta 2} = \frac{1}{1+\frac{L(P_3)*(1-\alpha_{\theta 1}-\alpha_{\theta 2})}{L(P_1)*\alpha_{\theta 1}+L(P_2)*\alpha_{\theta 2}}} \leq \frac{1}{1+\frac{L(y_2)*(1-\alpha_{\theta 1}-\alpha_{\theta 2})}{L(y_1)*\alpha_{\theta 1}+L(y_2)*\alpha_{\theta 2}}}$$

It reaches the upper bound when

$$P_1 = y_1^-, P_2 = y_2^-, P_3 = y_2$$

So we got the most conservative prior distribution in the figure 6 (right-hand side) and the corresponding $\alpha^*_{\theta 1\cap\theta 2}$ as (5.8).

QED

Proof for the most conservative prior distribution in Figure 15:

If we introduce 8 variables with their ranges,

$$\begin{cases} 0 \leq \theta_{PP1} < y, 0 \leq R_1 < r \\ y \leq \theta_{PP2} \leq 1, 0 \leq R_2 < r \\ 0 \leq \theta_{PP3} < y, r \leq R_3 \leq 1 \\ y \leq \theta_{PP4} \leq 1, r \leq R_4 \leq 1 \end{cases}$$

by the mean value theorem for integrals, we would have (as the Figure 14, the probability masses associated with the four regions are label as *M₁*, *M₂*, *M₃*, *M₄*):

$$Pr(\theta_{PP} < y|\, process\ evidence)$$
$$= \frac{\int_0^1\int_0^1 (I_{\theta_{PP}<y}(\theta_{PP}) \times [\theta_{PP}+R]^k) g_{<\theta_{PP},R>}(\theta_{PP},R)d\theta_{PP}dR}{\int_0^1\int_0^1 [\theta_{PP}+R]^k g_{<\theta_{PP},R>}(\theta_{PP},R)d\theta_{PP}dR}$$
$$= \frac{[\theta_{PP1}+R_1]^k M_1 + [\theta_{PP3}+R_3]^k M_3}{[\theta_{PP1}+R_1]^k M_1 + [\theta_{PP3}+R_3]^k M_3 + [\theta_{PP2}+R_2]^k M_2 + [\theta_{PP4}+R_4]^k M_4}$$
$$= \frac{1}{1+\frac{[\theta_{PP2}+R_2]^k M_2 + [\theta_{PP4}+R_4]^k M_4}{[\theta_{PP1}+R_1]^k M_1 + [\theta_{PP3}+R_3]^k M_3}} \le \frac{1}{1+\frac{y^k M_2 + [y+r]^k M_4}{[y+r]^k M_1 + M_3}}$$
$$= \frac{1}{1+\frac{y^k(\gamma_r - M_1) + [y+r]^k(1-\gamma_\theta-\gamma_r+M_1)}{[y+r]^k M_1 + (\gamma_\theta - M_1)}}$$

The inequality step above indicates $\theta_{PP1} = y^-, R_1 = r^-, \theta_{PP2} = y, R_2 = 0, \theta_{PP4} = y, R_4 = r$ and $\theta_{PP3} + R_3 = 1$, which is the Figure 15**.**

And it is not hard to see the last formula is a decreasing function of M_1. And the range of M_1 is $0 \le M_1 \le min(\gamma_\theta, \gamma_r)$, so, when $M_1 = 0$, we reach the upper bound γ_θ^*.

$$Pr(\theta_{PP} < y|\ \text{process evidence}) \le \frac{1}{1+\frac{y^k\gamma_r + [y+r]^k(1-\gamma_\theta-\gamma_r)}{\gamma_\theta}} = \gamma_\theta^*$$

QED

Proof for the most conservative prior distributions in Figure 17 and Figure 18:

First set the $Pr(\theta_{PP} < y_1|\ \text{process evidence})$ as the objective function.

Introduce 12 variables with their ranges,

$$\begin{cases} 0 \le \theta_{PP1} < y_1, 0 \le R_1 < r \\ y_1 \le \theta_{PP2} < y_2, 0 \le R_2 < r \\ y_2 \le \theta_{PP5} \le 1, 0 \le R_5 < r \\ 0 \le \theta_{PP3} < y_1, r \le R_3 < r_U \\ y_1 \le \theta_{PP4} < y_2, r \le R_4 < r_U \\ y_2 \le \theta_{PP6} \le 1, r \le R_6 < r_U \end{cases}$$

By the mean value theorem for integrals and similar reasoning above (as in Figure 16, the probability masses associated with the 6 regions are tagged as M_i):

$$Pr(\theta_{PP} < y_1 | \text{ process evidence}) = \frac{\int_0^1 \int_0^1 (I_{\theta_{PP}<y1}(\theta_{PP}) \times [\theta_{PP} + R]^k) g_{<\theta_{PP},R>}(\theta_{PP}, R) d\theta_{PP} dR}{\int_0^1 \int_0^1 [\theta_{PP} + R]^k g_{<\theta_{PP},R>}(\theta_{PP}, R) d\theta_{PP} dR}$$

$$= \frac{[\theta_{PP1} + R_1]^k M_1 + [\theta_{PP3} + R_3]^k M_3}{[\theta_{PP1} + R_1]^k M_1 + [\theta_{PP3} + R_3]^k M_3 + [\theta_{PP2} + R_2]^k M_2 + [\theta_{PP4} + R_4]^k M_4 + [\theta_{PP5} + R_5]^k M_5 + [\theta_{PP6} + R_6]^k M_6}$$

$$\leq \frac{[y_1 + r]^k M_1 + [y_1 + r_U]^k M_3}{[y_1 + r]^k M_1 + [y_1 + r_U]^k M_3 + {y_1}^k M_2 + [y_1 + r]^k M_4 + {y_2}^k M_5 + [y_2 + r]^k M_6}$$

$$= \frac{[y_1 + r]^k M_1 + [y_1 + r_U]^k (\gamma_{\theta 1} - M_1)}{[y_1 + r]^k M_1 + [y_1 + r_U]^k (\gamma_{\theta 1} - M_1) + {y_1}^k M_2 + [y_1 + r]^k (\gamma_{\theta 2} - M_2) + {y_2}^k (\gamma_r - M_1 - M_2) + [y_2 + r]^k (1 - \gamma_r - \gamma_{\theta 2} - \gamma_{\theta 1} + M_1 + M_2)}$$

The inequality step above indicates the location of a probability mass point within each area.

The last formula above is a decreasing function in terms of M_1 and M_2 respectively. So, when $M_1 = M_2 = 0$, we reached the upper bound $\gamma_{\theta 1}^*$:

$$\gamma_{\theta 1}^* = \frac{[y_1 + r_U]^k \gamma_{\theta 1}}{[y_1 + r_U]^k \gamma_{\theta 1} + [y_1 + r]^k \gamma_{\theta 2} + {y_2}^k \gamma_r + [y_2 + r]^k (1 - \gamma_r - \gamma_{\theta 2} - \gamma_{\theta 1})}$$

The corresponding most conservative prior distribution is shown in Figure 17 and Figure 18, depends on the specific values assigned on the parameters.

Note that, when $1 - y_1 \leq r_U < 1$, by the reasoning above, we got the most conservative prior distribution as the figure below:

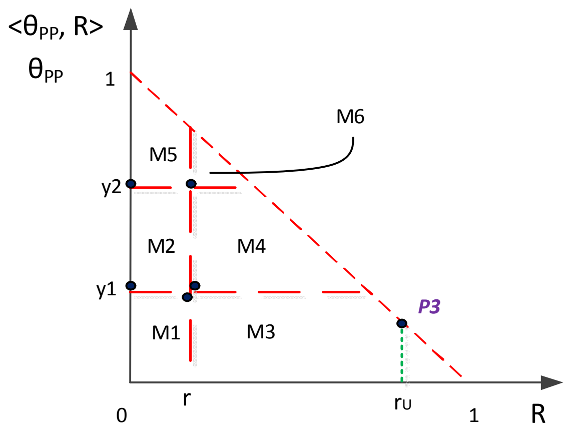

Figure 35 a most conservative prior distribution giving the counter-intuitive results

In this case, the P_3 will always win out. As the P_3 point is below y_1, we will always have $Pr(\theta_{PP} < y_1 | \, process\ evidence) > Pr(\theta_{PP} < y_1)$, which is the "counter-intuitive" result again.

Similarly as the reasoning above, it is not hard to get the most conservative prior distribution for the objective function $Pr(\theta_{PP} < y_2 | \text{ process evidence})$ as below:

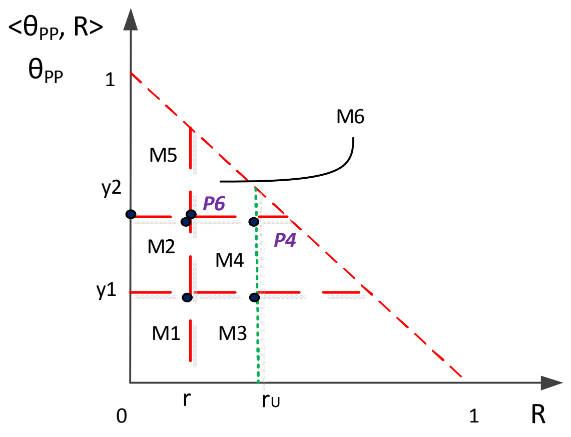

Figure 36 a most conservative prior distribution giving the counter-intuitive results

When seeing good process evidence, the P_4 point will always win out of the other points on the joint distribution, no matter how the r_u varies in its range. In other words, the mass at other points will always move to the P_4 point. As the P_4 point is below y_2, we will have $Pr(\theta_{PP} < y_2 | \, process\ evidence) > Pr(\theta_{PP} < y_2)$, which is the "counter-intuitive" result again.

QED

APPENDIX D

The simplification for (6.4):

$P(e=1)=E_{T_1,T_2,I_\varphi}\left((1-T_1-T_2)I_\varphi+T_2\right)$

$=E_{T_1,T_2,I_\varphi}\left((1-T_1-T_2)I_\varphi\right)+E_{T_1,T_2,I_\varphi}(T_2)$

$=E_{I_\varphi}\left(E_{T_1,T_2|I_\varphi}\left((1-T_1-T_2)I_\varphi\middle|I_\varphi\right)\right)+E_{I_\varphi}\left(E_{T_1,T_2|I_\varphi}\left(T_2\middle|I_\varphi\right)\right)$

$=E_{I_\varphi}\left(I_\varphi\ E_{T_1,T_2|I_\varphi}\left((1-T_1-T_2)\middle|I_\varphi\right)\right)+E_{I_\varphi}\left(E_{T_1,T_2|I_\varphi}\left(T_2\middle|I_\varphi\right)\right)$

$=(1-t_1-t_2)\times\varphi+t_2$

Here the independent assumption is that T_1 and T_2 are independent with I_φ.

QED

The simplification for (6.6):

$\mathrm{P}(e=1, isperfect_{SW}=1)=E_{T_1,T_2,I_\varphi,I_{SW},I_{FS}}\left((1-T_1-T_2)I_\varphi I_{SW}+T_2 I_{SW}\right)$

$=E_{T_1,T_2,I_\varphi,I_{SW}}\left((1-T_1-T_2)I_\varphi I_{SW}\right)+E_{T_2,I_{SW}}(I_{SW}T_2)$

$=E_{T_1,T_2}(1-T_1-T_2)\times E_{I_\varphi,I_{SW}}\left(I_\varphi I_{SW}\right)$

$+E_{T_2}(T_2)\times E_{I_{SW}}(I_{SW})$

$=(1-t_1-t_2)\times\left(\theta\times\varphi+Cov\left(I_{SW},I_\varphi\right)\right)+t_2\times\theta$

The result is based on the same independent assumption for the simplification of (6.4).

QED

The simplification for (6.14):

By (6.7),

$$Pr(isperfect_{SW}=1|e=1)=\frac{(1-t_1-t_2)\times\left(\theta\times\varphi+Cov\left(I_{SW},I_{\varphi}\right)\right)+t_2\times\theta}{(1-t_1-t_2)\times\varphi+t_2}$$

In the numerator,

$$\begin{aligned}\theta\times\varphi+Cov\left(I_{SW},I_{\varphi}\right)=E\left(I_{\varphi}I_{SW}\right)&=\Pr(iscompliant_{AC}=1,isperfect_{SW}=1)\\&=\Pr(iscompliant_{AC}=1,isperfect_{SW}=1,iscomplete_{FS}=1)\\&+\Pr(iscompliant_{AC}=1,isperfect_{SW}=1,iscomplete_{FS}=0)\\&=\Pr(isperfect_{SW}=1,iscomplete_{FS}=1)-f_1\\&+\Pr(isperfect_{SW}=1,iscomplete_{FS}=0)-f_2=\theta-f_1-f_2\end{aligned}$$

In the denominator,

$$\begin{aligned}\varphi=E\left(I_{\varphi}\right)=\Pr(iscompliant_{AC}=1)&\\&=\Pr(iscompliant_{AC}=1,isperfect_{SW}=1,iscomplete_{FS}=1)\\&+\Pr(iscompliant_{AC}=1,isperfect_{SW}=1,iscomplete_{FS}=0)\\&+\Pr(iscompliant_{AC}=1,isperfect_{SW}=0,iscomplete_{FS}=1)\\&+\Pr(iscompliant_{AC}=1,isperfect_{SW}=0,iscomplete_{FS}=0)\\&=\Pr\left(isperfect_{SW}=1,iscomplete_{FS}=1\right)-f_1\\&+\Pr(isperfect_{SW}=1,iscomplete_{FS}=0)-f_2+0+f_3\\&=\theta-f_1-f_2+f_3\end{aligned}$$

Replace and assemble the new denominator and numerator, we get result (6.14)

QED

APPENDIX E

Proof for the Theorem 1 in section 7.3:

Via (7.8):

$$\begin{aligned} Pr(\text{sys fails}) &\leq E_{pfd_A,isnqp_B}(\varepsilon(1-isnqp_B)+pfd_A\times isnqp_B) \\ &= E_{pfd_A,isnqp_B}\big(\varepsilon(1-isnqp_B)\big) \\ &+ E_{pfd_A,isnqp_B}(pfd_A\times isnqp_B) \\ &= \varepsilon\times\omega+E_{pfd_A,isnqp_B}(pfd_A\times isnqp_B) \end{aligned} \tag{E.1}$$

The second term in the right-hand expression (E.1) could be expanded using the proof for the theorem 1 in section 3.2.2 (see Appendix A, and simply replace $isimperfect_B$ by $isnqp_B$)

$$E_{pfd_A,isnqp_B}(pfd_A\times isnqp_B)\leq(1-\omega)P_A+(1-P_A)min\{\alpha_A,1-\omega\} \tag{E.2}$$

Assemble (E.1) and (E.2) gives the result (7.11).

QED

Proof for the Theorem 2 in section 7.3:

Now, the second term in the right-hand expression of (E.1) could be expanded using the proof for the theorem 2 in section 3.2.2 (see Appendix A, and simply replace $isimperfect_B$ by $isnqp_B$):

$$E_{pfd_A,isnqp_B}(pfd_A\times isnqp_B)\leq(1-\omega)P_A+(P_A^U-P_A)min\{\alpha_A,1-\omega\} \tag{E.3}$$

Assemble (E.1) and (E.3) gives the result (7.13).

QED

Proof for the Theorem 3 in section 7.3

Same as the theorem 3 in section 3.2.2 (proved in Appendix A), simply replace $isimperfect_B$ by $isnqp_B$:

$$\begin{aligned} E_{pfd_A,isnqp_B}(pfd_A \times isnqp_B) &\le \sqrt{(1-\omega)\big(E(pfd_A)^2 + Var\,(pfd_A)\big)} \\ &< \sqrt{(1-\omega)}\big(E(pfd_A) + SD\,(pfd_A)\big) \end{aligned} \tag{E.4}$$

Assemble (E.1) and (E.4) gives the result (7.14).

QED

Proof for the Theorem 4 in section 7.3

Same as the theorem 4 in section 3.2.2 (proved in Appendix A), simply replace $isimperfect_B$ by $isnqp_B$:

$$\begin{aligned} E(pfd_A \times isnqp_B) &\le \sqrt{(1-\omega)\big(E(pfd_A)^2 + Var\,(pfd_A)\big)} \\ &= \sqrt{(1-\omega)(M_A \times P_A^U)} \end{aligned} \tag{E.5}$$

Assemble (E.1) and (E.5) gives the result (7.15).

QED

Proof for the most conservative distribution of Figure 29:

The problem now is to find the most conservative $f_B(p)$, i.e. the one that minimizes (7.19) subject to the constraints (7.16), (7.17) and (7.18).

By the mean value theorem for integrals, we could find three values, say P_1, P_2 and P_3 satisfying the equations below:

$$(1-P_1)^n \int_{0+}^{\varepsilon} f_B(p)dp = \int_{0+}^{\varepsilon} (1-p)^n f_B(p)dp \tag{E.6}$$

$$(1-P_2)^n \int_{\varepsilon+}^{y} f_B(p)dp = \int_{\varepsilon+}^{y} (1-p)^n f_B(p)dp \tag{E.7}$$

$$(1-P_3)^n \int_{y+}^{1} f_B(p)dp = \int_{y+}^{1} (1-p)^n f_B(p)dp \tag{E.8}$$

, where $0 < P_1 \leq \varepsilon, \varepsilon < P_2 \leq y$ and $y < P_3 \leq 1$.

From the prior constraints (7.16), (7.17) and (7.18), we know:

$$\int_{0+}^{\varepsilon} f_B(p)dp = \beta \tag{E.9}$$

$$\int_{\varepsilon+}^{y} f_B(p)dp = 1-\theta-\beta-\alpha \tag{E.10}$$

$$\int_{y+}^{1} f_B(p)dp = \alpha \tag{E.11}$$

Then our objective function (7.19) turns to:

$$\begin{aligned} &Pr(0 < pfd_B \leq \varepsilon | \text{n failure free tests}) \\ &= \frac{\theta + \int_{0+}^{\varepsilon}(1-p)^n f_B(p)dp}{\theta + \int_{0+}^{\varepsilon}(1-p)^n f_B(p)dp + \int_{\varepsilon+}^{y}(1-p)^n f_B(p)dp + \int_{y+}^{1}(1-p)^n f_B(p)dp} \\ &= \frac{\theta + (1-P_1)^n \beta}{\theta + (1-P_1)^n\beta + (1-P_2)^n(1-\theta-\alpha-\beta) + (1-P_3)^n\alpha} \\ &= \frac{1}{1+\frac{(1-P_2)^n(1-\theta-\alpha-\beta)+(1-P_3)^n\alpha}{\theta+(1-P_1)^n\beta}} \end{aligned} \tag{E.12}$$

To minimize (E.12) is to maximize P_1 and minimize P_2 and P_3. That is let $P_1=\varepsilon$, $P_2=\varepsilon^+$ and $P_3=y^+$, which is in response to the prior distribution in Figure 29.

QED