# **E**xtensions to **M**odels of **C**oincident **F**ailure in **M**ultiversion **S**oftware

*by*

Kizito Oluwaseun Salako

kizito@csr.city.ac.uk

*The Centre for Software Reliability*

*City University*

*London EC1V 0HB*

*United Kingdom*

June 15, 2012

# Contents

# List of Figures

# List of Symbols

| | | |
|---|---|---|
| EL | Eckhardt and Lee Model | 17 |
| LM | Littlewood and Miller Model | 17 |
| ISA | The Independent Sampling Assumption | 18 |
| *pfd* | Probability of failure on demand | 24 |
| $(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X(\cdot))$ | Probabilistic model of demand occurrence | 25 |
| $\mathcal{X}$ | Demand space | 25 |
| $\Sigma_{\mathcal{X}}$ | Sigma–algebra of events in model of demand occurrence | 25 |
| $X$ | Random variable defined over the demand space | 26 |
| $\mathrm{P}_X(\cdot)$ | Probability measure that defines the distribution of $X$ | 26 |
| *pmf* | Probability mass function | 26 |
| $\omega(\pi, x)$ | Score function, for version $\pi$, that is defined on demand space | 29 |
| $\underset{Y}{\mathbb{E}}(\cdot)$ | Mathematical expectation with respect to the distribution of $Y$ | 30 |
| $\underset{Y}{\mathrm{Cov}}(\cdot, \cdot)$ | Covariance of a pair of random variables that are both functions of the random variable Y | 30 |
| $d_i$ | The $i$th software development process activity | 31 |
| di | A realisation of the random variable $Di$ | 31 |
| $\Omega_{d_i}$ | Sample space for the $i$th software development process activity | 31 |
| $\Sigma_{d_i}$ | Event space for the $i$th software development process activity | 31 |
| $Di$ | Random variable defined over the sample space, $\Omega_{d_i}$ | 31 |
| $\mathrm{P}_{Di}(\cdot \vert d(i-1), \ldots, d1)$ | Probability measure that defines conditional $Di$ distribution | 31 |
| $\big(\Omega_{d_i}, \Sigma_{d_i}, \mathrm{P}_{Di}(\cdot \vert d(i-1), \ldots, d1)\big)$ | Probabilistic model of development process activity $d_i$ | 31 |
| $\mathcal{P}$ | Set of all programs of interest | 33 |
| $\Sigma_{\mathcal{P}}$ | Event space for single–version development process | 33 |
| $\Pi$ | Random variable defined over the set of all programs, $\mathcal{P}$ | 33 |
| $(\mathcal{P}, \Sigma_{\mathcal{P}}, \mathrm{P}_{\Pi}(\cdot \vert di, \ldots, d1))$ | Conditional probabilistic model of final software deployment | 33 |
| $\mathrm{P}_{\Pi}(\cdot)$ | Probability measure that defines distribution of $\Pi$ | 33 |

| | | |
|---|---|---|
| $\mathrm{P}_{\Pi}\left(\cdot\|di,\ldots,d1\right)$ | Probability measure that defines conditional distribution of $\Pi$ | 33 |
| $\Omega_{d_1}\times\ldots\times\Omega_{d_{n-1}}\times\mathcal{P}$ | Sample space for single–version development process model | 34 |
| $\Sigma_{\Omega_{d_1},\ldots,\Omega_{d_{n-1}},\mathcal{P}}$ | Event space for single–version development process model | 34 |
| $\underset{D1,\ldots,D(n-1),\Pi}{\mathrm{P}}$ | Probability measure that defines the joint distribution of $\{D1,\ldots,\Pi\}$ | 34 |
| $\theta(x)$ | Difficulty function defined on demand space | 36 |
| $\Pi_A$ | The random variable modelling the development of a version – either according to methodology A or with respect to channel A | 38 |
| $\theta_A(x)$ | Difficulty function – either induced by use of methodology A or related to channel A's development | 42 |
| $\theta_1(x)$ | Difficulty function related to channel 1's development | 43 |
| $q_{AA}$ | The expected system *pfd* of a 1–out–of–2 system with channels built using the same methodology | 47 |
| $q_{AB}$ | The expected system *pfd* of a 1–out–of–2 system with one channel built using methodology $A$ and the other channel built using methodology $B$ | 47 |
| $\theta_{A;e_1,\ldots,e_N}(x)$ | The Difficulty function, related to channel A, that is defined over a set of N activities and the demand space | 70 |
| $\theta_{A;e_1,\ldots,e_N}$ | $\theta_{A;e_1,\ldots,e_N}(x)$ | 70 |
| $\theta^*_A(x_i)$ or $\theta^*_A i$ | $i$th component of the vector denoted by $\theta_A$ (or $\theta^*_A$) in the basis $\left\{\bar{P}_1,\ldots,\bar{P}_n\right\}$ | 89 |
| $\left\{\bar{P}_1,\ldots,\bar{P}_n\right\}$ | The basis in which the components of vectors of interest are difficulty functions | 91 |
| $\left\{\hat{P}_1,\ldots,\hat{P}_n\right\}$ | The basis of unit vectors in the direction of the vectors $\left\{\bar{P}_1,\ldots,\bar{P}_n\right\}$ | 92 |
| $P_i$ | Shorthand for the symbol $\mathrm{P}_X(x_i)$ | 95 |
| $\bar{P}$ | Vector that models the difficulty function which fails on all demands | 99 |
| $\theta_A$ *or* $\theta^*_A$ | The vector whose $i$th component in basis $\left\{\bar{P}_1,\ldots,\bar{P}_n\right\}$ is $\theta^*_A i$ | 101 |
| $\theta_A(x_i)$ or $\theta_A i$ | $i$th component of the vector denoted by $\theta_A$ (or $\theta^*_A$) in the basis $\left\{\hat{P}_1,\ldots,\hat{P}_n\right\}$ | 101 |
| $\mathbf{V}$ | A vector space | 190 |
| $\mathbb{R}$ | The set of real numbers | 190 |

# Acknowledgments

I would like to thank and acknowledge my supervisor, Lorenzo Strigini, for his helpful guidance, deep insight, attention to detail and collaboration since my joining CSR (*The* C*entre for* S*oftware* R*eliability*).

In particular, I would like to acknowledge the following contributions he has made. An initial statement and proof of Eq. (2.19), on page 48. This equation shows, under a form of indifference between two development process methodologies, forcing diversity results in an expected probability–of–failure–on–demand (*pfd*) that is no worse (and may be better) than if diversity were allowed to occur naturally. Also, in Chapter 3, the idea of depicting the *generalised models of coincident failure* as directed acyclic graphs (that is, depicting these models as BBNs), and thereby being a vehicle for proofs of relationships between alternative ways of organising development processes. Also, an initial statement and proof of Lemma 3.2.5 on page 75. The creation of Fig.'s 2.1 and 2.3, on pages 23 and 29 respectively. An initial proof of Eq. (5.25) on page 147. Finally, parts of Chapters 3 and 6 are based on prose collaboratively written by Lorenzo and me as part of the following projects which funded some of the research presented here:

- the U.K. E*ngineering and* P*hysical* S*ciences* R*esearch* C*ouncil* (EPSRC) under the project DIRC (I*nterdisciplinary* R*esearch* C*ollaboration in* D*ependability of Computer-Based Systems*);
- the DISPO (DI*verse* S*oftware* P*r*O*ject*) projects, funded by British Energy Generation Ltd and BNFL Magnox Generation under the Nuclear Research Programme via contracts PP/40030532 and PP/96523/MB.

I thank both Bev Littlewood and David Wright for their patient reading of the material I present here, and offering useful suggestions and helpful critique.

I thank Dave Eckhardt, Larry Lee, Bev Littlewood and Doug Miller for developing the conceptual models that stimulated the work presented here.

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

I deeply appreciate my family (my parents, sisters and brothers) who have been a constant source of encouragement and support; they are my biggest fans!

Finally, I thank all of my friends whose friendship over the years has been stronger than my inability to keep in touch properly due to PhD work; I'll be "letting my hair down" a bit more now via Facebook, Skype and live Jazz shows!

# Abstract

Fault–tolerant architectures for software–based systems have been used in various practical applications, including flight control systems for commercial airliners (e.g. AIRBUS A340, A310) as part of an aircraft's so–called *fly–by–wire* flight control system [1], the control systems for autonomous spacecrafts (e.g. Cassini–Huygens Saturn orbiter and probe) [2], rail interlocking systems [3] and nuclear reactor safety systems [4, 5]. The use of diverse, independently developed, functionally equivalent software modules in a fault–tolerant configuration has been advocated as a means of achieving highly reliable systems from relatively less reliable system components [6, 7, 8, 9]. In this regard it had been postulated that [6]

> "*The independence of programming efforts will greatly reduce the probability of identical software faults occurring in two or more versions of the program.*"

Experimental evaluation demonstrated that despite the independent creation of such versions positive failure correlation between the versions can be expected in practice [10, 11]. The conceptual models of Eckhardt et al [12] and Littlewood et al [13], referred to as the EL model and LM model respectively, were instrumental in pointing out sources of uncertainty that determine both the size and sign of such failure correlation. In particular, there are two important sources of uncertainty:

- *The process of developing software*: given sufficiently complex system requirements, the particular software version that will be produced from such a process is not known with certainty. Consequently, complete knowledge of what the failure behaviour of the software will be is also unknown;
- *The occurrence of demands during system operation*: during system operation it may not be certain which demand a system will receive next from the environment.

To explain failure correlation between multiple software versions the EL model introduced the notion of ***difficulty***: that is, given a demand that could occur during system operation there is a chance that a given software development team will develop a software component that fails when handling such a demand as part of the system. A demand with an associated high probability of developed software failing to handle it correctly is considered to be a "difficult" demand for a development team; a low probability of failure would suggest an "easy" demand. In the EL model different development teams, even when isolated from each other, are identical in how likely they are to make mistakes while developing their respective software versions. Consequently, despite the teams possibly creating software versions that fail on different demands, in developing their respective versions the teams find the same demands easy, and the same demands difficult. The implication of this is the versions developed by the teams do not fail independently; if one observes the failure of one team's version this could indicate that the version failed on a difficult demand, thus increasing one's expectation that the second team's version will also fail on that demand. Succinctly put, due to correlated "difficulties" between the teams across the demands, "independently developed software cannot be expected to fail independently". The LM model takes this idea a step further by illustrating, under rather general practical conditions, that negative failure correlation is also possible; possible, because the teams may be sufficiently diverse in which demands they find "difficult". This in turn implies better reliability than would be expected under naive assumptions of failure independence between software modules built by the respective teams.

Although these models provide such insight they also pose questions yet to be answered. Firstly, the thesis scrutinizes the related assumptions of ***independence*** and ***perfect isolation***, both of which lie at the heart of the models. In the models multiple software versions are assumed to be developed by individual, perfectly isolated development teams resulting in the (probabilistic) conditionally independent development of the software. Both the implications of these assumptions and the consequences of their relaxation are considered. Certainly, the possibility of achieving "independence during software development" and the effects thereof are important practical considerations. Indeed, the independent development of the channels in a fault-tolerant, software–based system by ***perfectly isolated*** development teams has been advocated as an ideal to strive for. Justification for this point of view is that if the teams are perfectly isolated then they are independent in the decisions that they make during development. Surely, this should have a positive effect on the joint failure behaviour of the software modules that are ultimately developed, since there is no apparent reason for the teams to necessarily make similar mistakes? In this sense isolation would appear to be an ideal to strive for. However, the models point out a flaw in this reasoning by demonstrating that the software modules could still exhibit significant positive failure correlation. Nevertheless, despite the apparent inevitability of such failure correlation, is perfect isolation still an ideal in some sense that can be formalised? We show that there are situations where this is indeed the case (see Chapter 3). Also, if perfect isolation is not achievable in practice can some meaningful approximation of it be attained? We present generalisations of the model that achieve such an approximation by using an interplay of isolation and interaction between the development processes of the software versions (see Chapters 3 and 6).

This thesis explores the generality of the results from the conceptual models, extending the models where appropriate. Upon first inspection it might appear that the assumptions of "perfect isolation" and "independence" are fairly strong and, therefore, restrict the applicability of the models' results. To what extent is this true? By considering practical scenarios that appear to violate some of the models' assumptions the thesis demonstrates how the models are, nevertheless, relevant for these scenarios in at least two ways: in some cases these apparent violations of the model assumptions are not violations at all and, in some other cases where these are actual violations, the models may be modified to take them into account (see Chapters 3 and 6). We demonstrate both of these cases in the thesis. In addition, for practical scenarios that escape the range of applicability of the models, the thesis considers model extensions to cater for such scenarios (see Chapter 6). The result of this is further understanding, under more general conditions, of the implications of activities during the development of a fault–tolerant, software–based system on the expected system reliability.

"Independence" is important for another, computationally relevant reason. Perfect team isolation is used to justify ***probabilistic independence*** in the models[1]. However, justifying "perfect isolation", and therefore probabilistic independence, may be difficult to accomplish in practice. Additionally, there are situations where team interaction is desirable. The thesis explores extensions to the models that relax the "perfect isolation" assumption and yet still maintain computational convenience as much as possible. This is particularly useful for expressions involving the expected pfd (see Chapter 3).

Measures of reliability such as pfd, and other probabilities of interest used in the models presented here, can be difficult to estimate in practice. Not only can they require significant effort and resource investment to estimate, there is a need for the estimates obtained to be "consistent". Consistent in the sense that there are defined mathematical relationships between the probabilities which must hold in practice. There are 2 ways in which the models, and their generalisations, can assist with these estimation issues: the models provide both conservative estimates for unknown reliability measures and consistency checks for reliability estimates.

- Conservative estimates are useful since only some estimates of the pfds may be readily available in practice. Using available estimates the models allow for the specification of attainable bounds on other probabilities of interest. As a result extreme values for these probabilities can be used to inform conservative analysis, motivated by questions of the kind "What is the worst pfd value I can expect for a fault–tolerant system, given that I have estimates for the pfds of the system's component software?". This thesis, by analysing extensions of the models, specifies a number of attainable extreme values for expected pfds under various practical scenarios (see Chapters 4 and 5).

- Consistency checks are useful when estimates for all of the probabilities of interest are available. The estimates may have been obtained from different sources or via diverse means. Therefore, it is imperative to check whether these estimates are consistent with each other. If it is theoretically impossible for one measure to be larger than another then estimates of these probabilities should exhibit the same relationship. Myriad consistency checks in the form of inequalities may be defined from the models. This thesis elaborates on this theme using generalisations of the models to state useful relationships between expected pfds (see Chapter 5).

Decisions about how best to organise a development process may be informed by knowledge of how activities in the process ultimately affect the expected system pfd. In this regard the models developed in this thesis are again useful. They state mathematical relationships between pfds under specified practical conditions. Consequently, identifying when these conditions hold in practice is sufficient to guarantee some relevant ordering of the expected pfds. This is the case despite not knowing the precise values of the probabilities involved in practice; *the mathematical results describe relationships between the probabilities, whatever their actual values may be.* So, by simply using the results of the generalised models, justification can be given to prefer some action, or decision, concerning software development over another (see Chapters 3, 4 and 5).

[1]Probabilistic independence is a mathematically convenient property: it justifies substituting joint probabilities with the product of marginal probabilities. Consequently, this has practical implications. For a fault–tolerant system may be constructed from constituent *Commercial off–the shelf* (COTS) Software. Probabilistic independence would then justify the expected ***Probability of Failure on Demand*** (*pfd*) for the system being the product of the expected pfds for the constituent software

While most of the extensions in the thesis address the aforementioned concerns directly some of the extensions concern the application of alternative, largely visual descriptions applied to the probabilistic models of diversity (see Chapters 2, 3, 4 and 5). This recasting of the models, from probabilistic to both graph-based and geometric terms, brings with it increased insight into the relationships between the modelled entities. These alternative representations also act as formal "vehicles" for the mathematical proofs that justify the practically relevant theorems stated in this thesis. While other representations could be used for the proofs, the advantages of using the descriptions we have chosen are: the proofs become "natural" consequences of well known mathematical methods, manipulations and results; better understanding of the relationships between "measures of interest" in the problem domain (such as expected system pfd and "variation of difficulty over the demands"); scenario representations which are both suggestive, and indicative of how to approach proofs; and various, equivalent scenario representations which may be used in proofs.

# Chapter 1

# The Introduction

## 1.1 Reliability via Fault-Tolerance

A defence against design faults in all kinds of systems is redundancy with diversity. In its simplest form this means that a system is built out of a set of subsystems (known as *versions*, *channels*, *lanes*[1]) which perform the same or equivalent functions in possibly different ways and are connected in a "parallel redundant" (1-out-of-N) or a voted scheme[2]. The rationale for such designs is as follows. A fault-tolerant design that uses multiple, identical copies of a subsystem will contain identical design faults in each of the copies: any circumstances in which one of them were to fail (ultimately due to these design faults) would tend to cause the other copies to fail as well possibly with results that, despite being incorrect, may be plausible, consistent and thus cannot be recognised as failures. Diversity eliminates the certainty of design faults being reproduced identically in all channels of the redundant system. One can hope that any faults (rare, given good quality development) will be unlikely to be similar between channels, causing them not to fail identically in exactly the same situations. A low probability of common faults can be sought by seeking "independence" between the developments of the multiple versions. For instance,

- for custom–developed components development teams work separately within the constraints of the specifications and general project management directives, making their separate design choices (and possible mistakes). These directives may also be specifically geared at "forcing" more diversity, e.g. mandating different architectures or different development methods [8, 14, 15];
- when re-using pre-existing components for the diverse channels one can seek assurance that the developments were indeed separate and, for instance, did not rely on common

[1]Special attention will be paid to a scenario of diversity between *software* versions since most previous literature in computing refers to this scenario; however, the method in this thesis can be applied more generally.

[2]A parallel redundant (1-out-of-N) system is one in which correct system functioning is assured, provided at least one channel functions correctly. For a majority voted system correct system functioning is assured if a majority of channels function correctly. Many other architectures are possible, however these are the simplest, practical scenarios where evaluation problems arise.

component libraries or designs.

How effective is diversity as a function of how it is obtained? Answering this question will help in deciding when to use diverse designs and how to manage their development. Controlled experiments have been conducted that investigate the impact of naturally occurring diversity in multiversion software on system reliability [10]. However, experimental results can be hard to generalise, especially to high-reliability systems, and such questions have generated lively debates in which positions have been mostly supported by appealing to experience and individual judgment. In this thesis a rigorous, probabilistic description of the issues involved is presented. The aim is to clarify the assumptions used in this debate, separating questions that require an empirical answer from those that can be answered by deduction, while providing useful insight.

There is little one can say, a priori, about the probability of common failure for a *specific* pair of versions. Some pairs may have no faults that lead to failures on the same demand. In some other pairs every time one version fails the other one will fail as well. But can we at least predict something about the *average* results of applying diversity in a certain system? One of the early questions was thus: will the average pair of versions behave like a pair of two average versions failing independently[3]? The famous experiment by Knight and Leveson [10] refuted this conjecture: this "independence on average" property did not apply to the specific population of versions that their subject programmers developed, hence cannot be assumed to hold in general. On average, a pair of versions failed together with far higher probability than the square of the average *probability of failure on demand* (or *pfd*, for short. See 2.2) of individual versions (though far less frequently than the average individual version). This leads to the conjecture that the general law in diverse systems may be, unfortunately, one of positive correlation – on average – between version failures. Experimental evidence was not enough to support or refute such general claims. Probabilistic modeling offered a way of understanding what may be going on in diverse development. The breakthrough was due to Eckhardt and Lee [12]. The bases of their approach were:

- from the viewpoint of reliability, a program can be completely described by its behaviour – success or failure – on every possible demand. Only "on–demand" or "demand–based" systems are considered, as opposed to continuous time systems such as control systems. An on–demand system receives a discrete demand from the environment and the result of processing it is either a success (a correct response) or a failure (an incorrect response) by the system;
- the process of developing a program is itself subject to variation. So, one cannot know in advance (or even, in practice, after delivery) exactly which program will result from it and, crucially, which faults it contains. This development process can be modelled as a process of *random sampling* which selects one program from the population of all

[3]This is often seen as a computationally ideal condition. It would allow us to gain assurance of very high reliability of the redundant system at relatively low cost. For instance, we could trust that a 3-version parallel system has a probability–of–failure–on–demand (*pfd*) of no more than $10^{-6}$ at the rather affordable cost of demonstrating that each version has a *pfd* of no more than $10^{-2}$. An even better scenario would be one in which common failures never happen, of course.

possible programs. The visible properties of the development (system specifications, methods used, choice of developers) do not determine *exactly which program* is created, but they determine the *probabilities* of it being any specific one;

- the *operational environment* within which a program is expected to function is such that there may be uncertainty concerning which *input* – or *demand* – will be processed by the program next. This uncertainty is adequately modelled as a probability distribution over the space of possible inputs/demands that may occur during operation;
- some demands are more difficult for the developers to treat correctly than others. One can formally model this "***difficulty***" (see 2.2) of each particular demand via the probability of a program, "randomly" chosen by the development process, failing on that specific demand.

In this modeling framework the reputedly ideal condition of *complete isolation* between the developments of the various versions is represented by the assumption that each program version is selected (sampled) *independently* of the selection of any other. This is made more precise in chapter 2. Eckhardt and Lee [12] then showed that if all versions are produced independently by identical development processes then "positive correlation on average" is inevitable unless (implausibly) all the demands have identical difficulty. Later, Littlewood and Miller [13] pointed out that each version may be developed by a *different* process: this is indeed the purpose of "***forcing diversity***". With this less restrictive assumption the "correlation on average" between failures of the versions could even be negative resulting in a probability of failure for the system that is better than if the channels were expected to fail independently. There is even the extreme possibility of a zero system failure probability, despite the development processes of the system's constituent software channels being such that they have a non-negligible probability of producing programs that fail on some demands[4]. These two models (called *EL model* and *LM model* in what follows) both bring important insights:

- *perfect isolation* –and thus independence – between the developments of the versions does not guarantee independence between their failures. Independent developments guarantee that, *given a specific demand*, two – independently "sampled" – versions will fail independently on that demand, and yet this in turn implies non-independence for a randomly chosen demand.
- *Diverse redundancy is always beneficial.* For a 1–out–of–N system built out of independently developed COTS software the expected *pfd* of the system is guaranteed never to be worse than the expected *pfd*s of its constituent software channels. Such a result, which is not always possible in the context of the generalised models of diversity presented in this thesis, is useful in deciding whether fault–tolerance can be expected

[4]There will always be some difference between the development processes of different versions, so the Littlewood-Miller assumptions always apply when the development processes of the individual versions are independent. On the other hand, it is difficult (actually, there is no obvious method) to quantify how far the true conditions of development are from the Eckhardt-Lee, worst-case assumptions. So, if an assessor wishes to err on the side of conservatism, the Eckhardt-Lee assumptions are more appropriate.

to bring reliability improvements over single–version systems. In this regard *the use of a 1–out–of–N system architecture does not guarantee improved system reliability, in general.* This will be demonstrated for the 1–out–of–2 case in Chapter 2.

- *Various bounds under different scenarios can be stated* between expected system *pfd*s resulting from forcing diversity, and expected system *pfd*s resulting from letting diversity occur naturally (that is, as a consequence of the development teams being isolated from one another). More generally, there are a number of reliability related measures used in the modeling for which myriad bounds may be stated under different conditions.

- *A clear, formal description of conditions that increase failure dependence*, and thus of which goals we should pursue when we try to "force" diversity.

These implications have been explored in many other applications of the same modeling approach, e.g. to the choice of fault removal methods [16, 17], to security [18] and to human–machine systems [19].

What are the consequences of dependence between software development processes? In the EL and LM models this issue does not arise: perfectly isolated development teams develop the programs for a multiple-version system. Consequently, the teams are *probabilistically independent* in how they develop their respective versions. For brevity, call this the "***independent sampling assumption***" (ISA) . The ISA has two useful properties: it is mathematically simple enough to allow elegant theorems like the EL model's implication of "positive correlation on average", and it models perfect separation between the developments of the versions[5]. But there are many reasons for doubting that it will normally be realised in practice. For instance, doubters point out that:

- some communication will tend to occur between the version development teams, at least indirectly;

- developers often share common education background or use the same reference books, etc.;

- the management of a multiple-version development will exert common influences on the development teams, e.g. by distributing clarifications and amendments to the specifications.

These scenarios prompt several questions: do they violate the ISA? If they do, do they invalidate the message from the Eckhardt and Lee breakthrough: that is, is failure independence more likely than the EL model suggests, and is it prudent to assume positive correlation? And do they invalidate any other practical guidance drawn from these models? In conclusion, *what are the practical implications of possible statistical dependencies* between the development (sampling) of versions? Answering these questions forms a major part of the current

[5]Actually, the ISA is a necessary consequence of perfect team isolation, but not a sufficient condition since it can be used in cases where certain information is shared between the development processes (see Chapter 6 : Section 6.1 for details).

work. This requires clarifying how relevant aspects of real-world processes are mapped into modeling assumptions.

In the process of this analysis, a further important question arises: *"is 'perfectly isolated development teams' really an 'extreme optimistic' assumption for the EL special case?"*, which is what gives the EL result its value as a warning: "even under the most optimistic assumptions – perfect isolation in development – still you should expect identical processes to produce positive correlation of failures". So far, authors who recommend separation of version developments have plausibly argued that this would prevent the propagation of mistakes between the teams developing the different versions (so called "*fault leaks*" phenomena) [6, 7, 20, ]. It is plausible that this propagation may occur, via either the direct imitation of erroneous solutions or the sharing of similar viewpoints and strategies (e.g. high-level architectural decisions) which frame the development problems similarly for the different teams, creating similar "blind spots" or error-prone subtasks. This could lead to a significant increase in the probability of the teams making sufficiently similar mistakes and thus increased failure correlation between the teams' software. In effect the teams' respective softwares are expected to have increased failure diversity and reduced individual reliabilities. However, doubters of this line of reasoning point out that there are also benefits from teams being allowed to interact. For example, the opportunity for erroneous viewpoints to be corrected and the bringing together of various problem–solving ideas could lead to improvements in the software being developed. So the teams' respective software are expected to have increased individual reliabilities and reduced failure diversity. Both viewpoints appear plausible. In order to answer the question of which of these is to be preferred extensions to the LM and EL models that relax "perfect isolation" are explored. Additionally, what if "perfect isolation" turns out not to be a best case assumption? What changes when one relaxes it? In particular, does the pessimistic warning from the EL model become invalid?

Forcing diversity can lead to substantial reliability gains: the LM model predicts the possibility of negative failure correlation. In general, however, forcing diversity does not guarantee that the probability the resulting system fails in operation will be no larger than what would be the case if diversity were allowed to occur naturally. Nevertheless, there are cases where such an ordering is guaranteed and some of these are pointed out by the LM model. Indeed, *the model provides the possibility for stating several "tight" bounds on the expected* pfd *for a 1–out–of–2 system* under various conditions, including requirements on expected single–version *pfd*s and the degree to which relevant probability distributions are known. Following in this spirit the current work explores such bounds more generally. These bounds are useful both to justify using certain expected *pfd* values in conservative analysis and to provide a set of consistency checks for the reliability measures.

## 1.2 Layout of the Thesis

The thesis presents work with two main focal points. Chapters 2 and 3 focus on developing and generalising the LM model to cater for dependence during the development, and use, of multiversion software. Chapters 4 and 5 focus on developing and using a geometric model of software diversity to define attainable bounds on system reliability. In total the thesis contains 6 more chapters, which we detail as follows:

Chapter 2 presents a detailed development of the EL and LM models that significantly expands their original formulation. The extra detail is useful and necessary in modeling more general scenarios considered later in the thesis. Also, consequences suggested by the models are discussed, with some new results that have not been stated previously.

Chapter 3 develops and explores generalisations of the EL/LM models that relax the assumption of independently developed software versions.

Chapter 4 is a treatise on a geometric formulation of the LM model. This geometric model increases the insight into various relationships between the probabilities used in the EL/LM models. In addition, the geometry provides a natural framework for solving the reliability optimisation problems of Chapter 5.

Chapter 6 discusses and summarizes the main contributions of the work presented in this thesis, with future work suggested in Chapter 7.

In addition to these chapters, the thesis also contains Appendices. Appendix A is a review of "finite–dimensional inner–product spaces" – the mathematical formalism used in Chapter 5 to obtain extremal values for the expected system *pfd* resulting from forcing diversity. Appendix B discusses the well–known Knight and Leveson diversity experiment [10], providing a conceptual model to explain why the only negatively correlated pairs of versions developed are those that exhibit no coincident failure. And finally, Appendix C is a discussion of ways in which the LM model may be applied in practice.

# Chapter 2

# Models of Coincident Failure in Multiversion Software

In this chapter we develop the models of "Eckhardt and Lee" (EL, for short), and "Littlewood and Miller" (LM, for short) in more detail than they have been developed historically [12, 13]. The approach presented here in developing the models draws liberally from the modern approach to probability theory which, itself, uses the framework of *measure theory*[1]. As much as possible an appeal to intuition, as well as some rigour, is made. The intention here is not to give a rigorous treatise on probability theory but, instead, to give a rigorous development of the EL and LM models. Many excellent references, on both probability and measure theory, exist [21, 22, 23, 24, 25, 26, 27]. The historical lack of this extra level of mathematical detail in the development of the EL/LM models is related to the assumption of "perfectly isolated" development teams. To appreciate this first consider that the respective software development processes of isolated development teams each contain activities that do not have predetermined outcomes (e.g. software testing). For our purposes a software development process is a set of activities, methods, practices, and transformations used by people to develop software [28]. Now, as a consequence of the isolation these development processes can be argued to be independent. Consequently, any uncertainty in the activities thereof may be "averaged over" in a way that preserves independence and results in the outcomes of individual activities being "hidden" in the models' probability distributions: only the final versions that are developed by the teams are explicitly modelled (see Section 2.3 and Eq. (2.4) for further detail). Despite this simplification the activities still have a visible effect in the models as they affect the probability of which software is ultimately produced by the teams. So, perfect isolation simplifies the models in ways that keep the model relevant and make the extra rigour unnecessary. There are, however, two reasons why more detailed modelling is required in an attempt to generalise the models:

1. The relaxation of the "perfectly isolated" teams assumption requires the models to ac-

[1]Measure theory is an important part of the field of mathematical analysis and is the basis for defining a theory of integration that can cater for discontinuous functions. Henri Lebesgue, in his seminal work in 1902 entitled "*Intègrale, longueur, aire*", was a significant driving force in the development of the theory.

count for teams interacting in ways that ultimately affect the reliabilities and diversity of the software they develop. Naturally, such interaction could take place before and during software development. Hence, activities before and during software development should be explicitly modelled;

2. The relaxation of model assumptions affects aspects of the probabilistic models in ways that require extra detail to appreciate. For instance, due to the "perfectly isolated" teams assumption, *conditional independence* lies at the heart of EL/LM models: conditional independence is essential for the covariation in the models that captures notions of diversity. In Chapter 3 we argue that only an interplay of isolation and interaction between the development teams can justify using conditional independence in a generalised model of diversity.

In the course of the development of the EL/LM models presented here there will be a number of concepts that we shall generalise and make more precise. We also state a number of results and viewpoints not previously stated in the original development of these models. These include

- demonstrating that forcing diversity does not, in general, guarantee better reliability than if diversity were allowed to occur naturally (Section 2.5);
- showing that there are at least two senses in which the use of fault–tolerance results in system reliability that is no worse than the reliability of a single version system (Section 2.5). This result, however, is not necessarily true in more general settings than those of the EL/LM models and we show this in Chapters 3 and 6;
- demonstrating that the assumption of "perfectly isolated" development teams justifies the use of a *product probability space* as the appropriate *probability space* that models the development of a 1–out–of–2 system by perfectly isolated development teams (These concepts are defined in Sections 2.2 and 2.4). This is the basis for why the assumption justifies probabilistic independence in how the teams develop their respective versions;
- developing visual representations of the models that facilitate the analyses of the models (Section 2.6). These are a Graph–based representation that depicts the consequences of conditional independence relationships in the models, and a Geometric representation that facilitates maximizing and minimizing reliability measures such as expected probability of failure on demand.

We develop the models as follows. First of all a reference scenario will be described to set the scene for the modelling. Then two models are developed and combined: a model of the occurrence of demands during system operation and a model of the development of a 1–out–of–2 system under the assumption of perfectly isolated development teams. The combination of these is the LM model, which contains the EL model as a special case.

## 2.1 Reference Scenario and Terminology

The reference scenario is a system that may be implemented either as a single version or as a diverse 2–channel, 1–out–of–2 system such as a nuclear power plant safety protection system which is conceptually depicted in Fig. 2.1. Despite being very simple this scenario has practical applications (in safety systems) and presents the essential difficulties of evaluating a probability of common failure. The term “versions” (or “program versions”) is used in

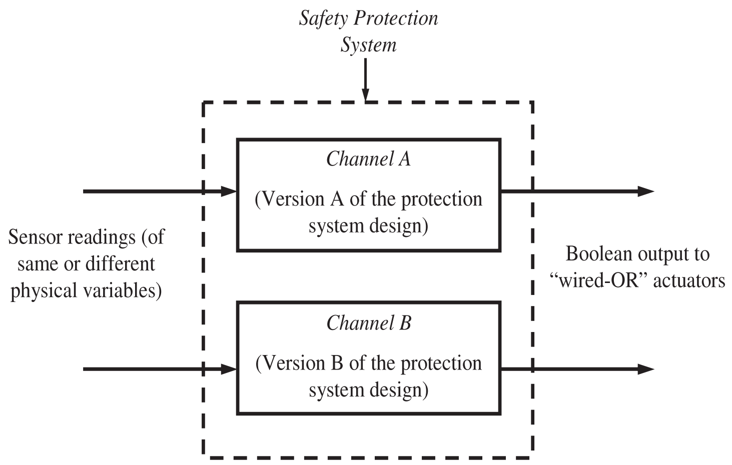

Safety Protection System
Channel A
(Version A of the protection system design)
Channel B
(Version B of the protection system design)
Sensor readings (of same or different physical variables)
Boolean output to “wired-OR” actuators

**Figure 2.1:** Our reference system is an abstraction of a plant safety protection system (e.g. for a nuclear power plant) with two redundant, diverse channels: a simple 1-out-of-2 system. Given that a *demand* (a potentially hazardous state of the plant) is submitted to the system during its operation the system has to recognise the demand and respond correctly. The successful response to a demand is for the system to initiate a plant shut-down procedure. The demands of interest here are the ones for which the correct response is a plant shutdown. In general, a protection system needs to decide whether a system input is hazardous or not, and then issue the correct response. So, both *false–positives* and *false–negatives* are potential failure modes. However, in a situation where incorrectly handling demands that require a system shut down can have catastrophic consequences, compared with benign consequences related to incorrectly handling other types of demands, a probability of interest might be the probability of a false–negative as we investigate here. Thus, the output of each channel and of the whole system is logically a boolean variable.

the sense of diverse, equivalent implementations of the system functions. Following common usage, when there is no risk of ambiguity, “version” will also be called a “channel” of the two–version system. The other common meaning of the term “versions” to designate the results of successive changes to a program, or “releases” will be avoided.

Reference is made to the following simple picture of multiple–version development: separate “version development teams”, each producing one version (and possibly further divided into sub-teams for design, coding, inspection, testing, etc). One “project management team” or “manager” defines the requirement specifications that the development teams must implement and the constraints under which they have to work, handles specification updates and decides on final acceptance of the developed versions. Each version is developed in a *version development process*: a sequence of activities with uncertain outcomes that culminate in the development of a version. Similarly, a *system* or *joint development process* is the combination of the development processes for the two or more versions in a system, plus the way they are co-ordinated. When the development of a single version system is under consideration the terms may be used interchangeably.

There are aspects of version development processes – such as the individuals that make up a development team, or the technologies, resources and algorithms used in developing the versions, or the various activities that or indeed the activities/stages in the development process – that are potentially controllable and may differ from one development process to another. For our purposes, given a collection of such aspects, a ***methodology*** is an instantiation of such a collection. For example, the programming language used may differ from one process to another, so the use of the C++ programming language (as opposed to Java, C#, Pascal, Visual Basic, e.t.c.) in a given development process forms part of the methodology employed for that development process. Other examples of parts of a methodology include a choice of integrated development environment to use (Visual Studio, Netbeans, Eclipse, e.t.c) and a choice of various combinations of Verification and Validation approaches (e.g. software testing, proof of correctness or requirements tracing [29]). Note that a methodology in this context not only requires a choice of a software development framework/approach (such as enhanced waterfall models, the Spiral model, or Agile methods [30, 31, 32]), but also a determination of how these approaches will be carried out, who will be involved in the development and what resources, technologies, solutions and methods will be employed. In this sense we may speak of the development of a program according to some methodology.

Attention is given solely to an "on demand" system. It receives a demand from the environment and the result of processing it is either a success (a correct response) or a failure (an incorrect response) by the system[2]. The notions of system *demands* and *inputs* will be used interchangeably[3]. For while the models capture the behaviour of systems on demands the results are also applicable in terms of system inputs.

## 2.2 Modelling System Failure

The dependability measure of interest is the ***probability of failure on demand*** (*pfd*). This is the probability that a given version or system fails on a random demand. In order to evaluate this it is important to know what the behaviour of the system on each demand is. For current purposes the nature of the required response to a demand – e.g., whether it is turning on an alarm signal or controlling complex mechanical actuators – is irrelevant. We need only distinguish between two types of response to a demand – success, i.e. correct

[2]We will deal exclusively with software that can be analysed in terms of discrete demands. A demand can be as simple as a single invocation of a re-entrant procedure, or as complex as the complete sequence of inputs to a flight control system from the moment it is turned on before take-off to when it is turned off again after landing. So, both discrete event systems and continuously operating systems have been adequately modelled using discrete notions of demand. While it is possible to describe probability of failure as a function of a continuous notion of demand the added model complexity is not worth introducing for the purpose of this discussion [33, 34].

[3]The set of system demands may be viewed either as a subset of the set of system inputs (e.g. for the protection system of a nuclear power plant particular ranges of the system input – temperature – that should cause a protection system to shut down a plant) or a set of subsets of the system inputs (e.g. for an airplane any sequence of system inputs – pilot actions or environmental events – from the time a plane takes off that can potentially lead to a plane crash). The models cater for both of these viewpoints.

behaviour, and failure. Also, consideration is given exclusively to failures due to design faults, i.e. failures not covered by the usual analysis methods for "systematic" failures.

When executing the software there is uncertainty about which demand it will next receive from its environment. This suggests that the occurrence of the "next demand" may be adequately modelled as a probability distribution. Defining a probabilistic model is equivalent to defining a model of an experiment whose outcome is unknown beforehand. The convention is to define 3 related, relevant constructs:

1. The set of all of the possible outcomes of conducting the experiment, called a ***sample space***. For example, the numbers 1,2,...,6 will make up the six possible outcomes of the experiment "a fair dice is rolled and only one side of the dice, the "top" side with its respective number, is noted afterwards".

2. The set of unique "answers" to all relevant questions that can be asked of the experiment outcomes, called an ***event space***. The "answers" are realised as subsets of the *sample space* called events. This *event space* is required to be non-empty and closed under both complementation (the complement of an event is also an event) and countable unions (the union of a countable number of events is an event). Formally, the *event space* is referred to as a *sigma-algebra* of events and it is usual, but not necessary, for this to be the so–called *Power set of the sample space*: the set of all subsets of the *sample space*. For example, in the "dice rolling" experiment given previously, we may want to know whether the visible number seen on the "top" side is an even number. Three outcomes – the numbers 2,4 and 6 – constitute the event "The number observed is an even number". The complement of this subset of outcomes – the numbers 1,3 and 5 – is also an event, and both of these events are defined by the question "Is an even number observed?". These are just some of the possible events in the space of relevant events related to this experiment.

3. A bounded, non–negative, countably additive, real–valued function defined on the *event space*. This so–called ***probability measure*** captures the notion of uncertainty concerning the occurrence of events. The value of the probability measure associated with a given event is the probability of the occurrence of that event. For instance, for the "dice rolling" experiment the probability of the event "the visible number is even" is $\frac{1}{2}$. This is the value, associated with the event, of some relevant probability measure.

So, in order to define an adequate probabilistic model of how demands occur during system operation, we only need to define 3 constructs, say $(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_{\mathcal{X}}(\cdot))$. This triple, called a *Probability Space*, is comprised of a *sample space*, an *event space* and a *probability measure* respectively. For our purposes the *sample space* of demands or ***demand space***, $\mathcal{X}$, is the discrete, countable set of all of the demands that are possible in a given practical scenario. The *event space*, $\Sigma_{\mathcal{X}}$, is the set of all of the answers to the relevant questions that may be asked about demands that occur in practice. We are interested in questions like "will the next demand that occurs in practice cause a particular software version to fail?". The affirmative answer to this question, the set of all of the demands that cause a particular version to fail, is a subset of $\mathcal{X}$ and is a member of $\Sigma_{\mathcal{X}}$. It will be convenient to assume that each demand is an atomic event: that is, each demand is contained in $\Sigma_{\mathcal{X}}$. Henceforth, we shall assume a similar requirement for all of the *sample space* and *event space* pairs we shall use throughout the thesis. Finally, the probability of various events related to the random occurrence of demands is given by the *probability measure* $\mathrm{P}_X(\cdot)$.[4] The $X$ in the subscript is intentional: it is a ***random variable***[5] defined over $\mathcal{X}$. By "restricting" $\mathrm{P}_X(\cdot)$ to the demands we define the ***demand profile***, i.e. an assignment of probabilities to every possible demand, $x \in \mathcal{X}$. The demand profile summarises and depends on the circumstances in which the system is used (and will generally be known with some degree of imprecision[6]). That is, the demand profile is a model of the environmental conditions under which the system operates. Technically, $\mathrm{P}_X(x)$ is a *probability mass function* (*pmf*) defined over $\mathcal{X}$ and with respect to $X$. The phrase "a randomly chosen demand" means the occurrence of a demand according to this probability rule. We follow the convention of using uppercase letters (e.g. $X$) for random variables and lowercase letters (e.g. $x$) for the values (numbers or vectors or names) which they can take. For practical reasons we shall assume that only those demands that can occur in the practical scenario of interest are modelled; there may exist demands in this application domain that may be possible in other scenarios but not the one currently modelled. To this end, we require that *every demand has a non-zero probability of occurrence*[7]; that is, $\mathrm{P}_X(x) > 0$ for all $x \in \mathcal{X}$. This requirement will be useful in Chapter 4 when defining an invertible transformation between a pair of bases in a finite–dimensional vector space.

Each version may contain faults determined by the uncertain and variable process of software development. The consequence of such faults is that they result in the version

[4]This measure induces a discrete probability distribution function. Only discrete probability distributions are used in the current work. There are applications for which continuous distributions are more suitable. However, many of the fundamental notions developed using discrete distributions carry over almost seamlessly to the continuous case. Also, currently it is not clear that added insight comes with the added complexity of the continuous case.

[5]Random variables are functions between measurable spaces (e.g. probability spaces) such that the pre–image of events in one space are events in another space. Consequently, random variables are useful for picking out events from the event space whose probabilities are of interest. This is most apparent when the random variables are *indicator/score functions*: given a sample space $\Omega$, an event space $\Sigma$ and an event $E \in \Sigma$ an indicator/score function, $\omega_E$, defined with respect to $E$ is a real–valued function $\omega_E : \Omega \to \mathbb{R}$ such that

$$\omega_E(w) := \begin{cases} 0, & \text{if } w \notin E \\ 1, & \text{if } w \in E \end{cases}$$

So, the indicator function maps the event $\{1\}$ (related to the real line) to the event $E = \omega_E^{-1}(1) = \{w \in \Omega : \omega_E(w) = 1\} \in \Sigma$. Use will be made of such random variables in this thesis to describe the failure behaviour of software.

[6]Admittedly, the degree of imprecision alluded to here might be significant, so that one might question using such a model of demand occurrence. However, the problem of having to use partial knowledge of a probabilistic law which governs some process is not unique to our current task. Indeed, various fields in Science and Engineering use observations of past process realisations to "infer" the probability of the process evolving in a given way (e.g. weather patterns, the rise and fall of stock-market options, and the dynamics of disease spreading). In any case, because many of the results presented in this work are invariant (that is, still hold true) with respect to different demand profiles, knowledge of actual demand profiles is not required to use the results in practice.

[7]This is the requirement that the set $\mathcal{X}$ contains no ***null*** demands. This mathematically convenient requirement is justified for our needs since we are considering only practical scenarios for which the demand space is discrete, not continuous.

failing, with certainty, when certain demands are submitted to the version in operation. In practice, it is possible for a software version to not exhibit this determinism, and sometimes fail or succeed when handling the same input from its operational environment. However, such lack of determinism may be explained by changes in the internal state of the version: different internal states result in the version responding differently to the same environmental input[8]. So, by defining the demands to include both stimulus from the environment and the internal states of all the software versions that may be deployed in the system, it is possible to recover the determinism we require. Please see Fig. 2.2 for an example of a demand space definition that accomplishes this. This is not the only appealing definition of demands.

There is an alternative definition that also reconciles the possibly different internal states of the programs with our desire for the programs to fail or succeed deterministically on each demand. To appreciate this consider the following. Given initial states for component software, the internal states of the software after a period of operation is a deterministic consequence of the software handling a sequence of stimuli from the environment. For 1–out–of–N systems, each environmental stimulus is required to be handled by all of the components in parallel. We may choose to define a demand as a possible sequence of such environmental stimuli. This means that the same demand – a given a sequence of environmental stimuli – is submitted to all the software components in the fault–tolerant configuration, resulting in the components having possibly different internal states due to how each software carries out its response to the stimuli. Consequently, given a demand, each software component is in some "knowable"[9] state and either fails or succeeds with certainty. Again, we have recovered determinism given the possible non–determinism that may arise from programs being in different states.

However, without appealing to these alternative definitions for demands, the programs modeled in this thesis have deterministic failure behaviour by the following requirement: each program is set to some agreed initial state whenever the program needs to handle stimulus from its operational environment. Therefore, in summary, we have discussed multiple ways to ensure the following postulate holds.

[8]In practice, it is common for a program input to sometimes cause the program to fail and at other times succeed, depending on the internal state of the program. This behaviour is typified by so–called *Heisenbugs*: software failures that are not always reproducible (in particular, when attempts are made to study the failure). However, in a number of such cases, uncertainty about whether a given Heisenbug causes failure or not has been shown to have both *epistemic* (uncertainty due to a lack of knowledge) and *aleatory* (uncertainty in the real world that does not decrease with knowledge) qualities.

1. *Epistemic Uncertainty*: This uncertainty reduces when knowledge is gained about the conditions under which the internal state of a program and its operational environment interact to result in failure. Consequently, by defining the demands (as we have suggested) to include both the possible internal states of programs and external (environmental) conditions under which the programs operate each program will always give the same response to the same demand;

2. *Aleatory Uncertainty*: Here, the environmental conditions under which a Heisenbug results in failure may be known, but the occurrence of these conditions is governed by uncertainty from the environment. So, unless there was some way in which the distribution for this "real world" uncertainty visibly approached certainty (e.g. a Bernoulli distribution with relatively very small mass for one point, in which case a score function is arguably an adequate approximation) there is no obvious method, or reason for using such a program in practice.

[9]knowable, in the sense that it is a deterministic consequence of the algorithm implemented in the software, the initial state of the software, and the sequence of environmental stimuli handled by the software.

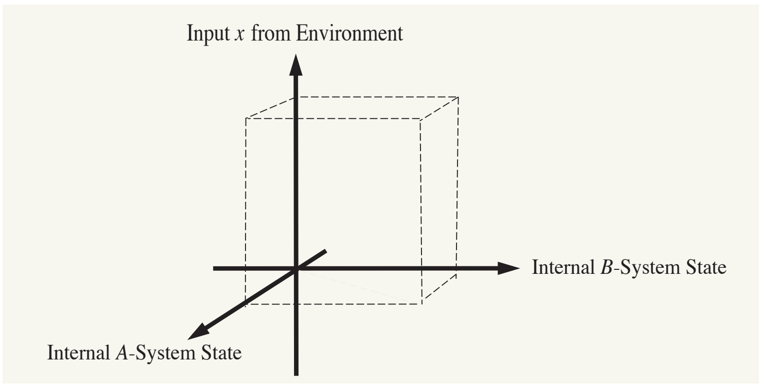

Input $x$ from Environment
Internal $B$-System State
Internal $A$-System State

**Figure 2.2:** A depiction of a demand space where each demand is a triplet: an input from the environment, and a pair of internal states for software versions A and B. Such a demand space definition would be adequate if only versions A or B could make up the system. In general, the internal states of all versions that could potentially make up the system should form part of such a definition of demands. Consequently, if $n-1$ possible versions could form part of the system, then an adequate demand space would contain demands that are *n–tuples*: an input from the environment, and $n-1$ possible states (one potential state per potential version). An example of a demand resulting from such a definition is the following. Let the $n-1$ software versions $\pi_1, \ldots, \pi_{n-1}$ have associated possible internal states $s_1, \ldots, s_{n-1}$. Then, together with a stimulus $x$ from the operational environment of the system, the *n–tuple* $(x, \pi_1, \ldots, \pi_{n-1})$ is a system demand. Alternatively, many systems have the desirable property (desirable, from a modeling standpoint) that they assume some given initial state when handling input from the environment, in which case there is no need to cater for possibly different system states in our modeling.

**Postulate 2.2.1.** *Each program/version behaves deterministically – it either always fails or always succeeds – on each demand submitted to it in operation.*

The claim here is that contained in each demand, therefore, is complete information about the conditions (both internal and external to the program) under which a given set of programs have to respond to some input. Consequently, due to faults, a version fails deterministically on certain demands. This defines the versions *failure set*: the set of all demands upon which the version fails. The sum of the probabilities of all these demands is the *pfd* of that version. That is, the demand profile associates the failure set of a version with a specific value of *pfd*. Referring to Fig. 2.3, a version fails when subjected to a demand that is part of its "failure set", a failure set that itself is determined by mistakes made during the versions development. Independence between the failures of two versions would mean that the *pfd* associated with the intersection of the two versions' failure sets is exactly equal to the product of the probabilities associated to each of the two failures sets. There is no obvious reason why this should be so. Furthermore, the same pair of versions could be employed under different demand profiles. It would seem extraordinary that all possible demand profiles maintained an invariant of failure independence. As an extreme case for a pair of versions the two failure sets might be disjoint, giving zero common *pfd*, or they might be identical, or one contained in the other.

We can formalise these concepts further. Given a program that behaves deterministically, i.e. for each demand it either deterministically processes it correctly or deterministically fails

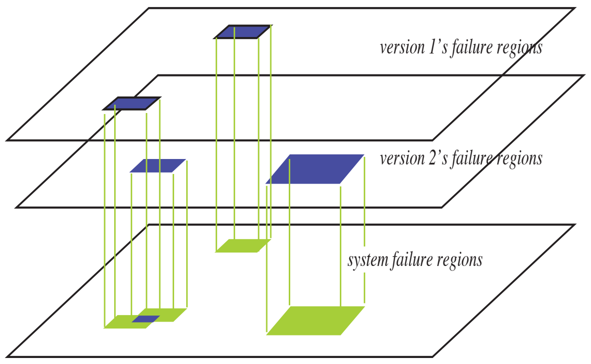


**Figure 2.3:** Example of overlaps between the failure regions of two diverse software versions. The horizontal, rectangular surfaces each represent the complete set of system demands. The projection on the highest surface depicts those demands on which version 1 fails. The projection on the second highest surface depicts those demands on which version 2 fails. The overlaps of these projections, depicted in the lowest surface, shows those set of demands for which a 1-out-of-2 system, built from versions 1 and 2, will fail in operation.

to process it correctly, we can define a boolean ***score function***, $\omega(\pi, x)$. This is defined for each pair of demand, $x$, and program/system, $\pi$, as:

$$\omega(\pi, x) := \begin{cases} 0, \text{ if } \pi \text{ processes } x \text{ correctly} \\ 1, \text{ if } \pi \text{ fails on } x \end{cases}$$

This function models complete knowledge about the failure behaviour of every program of interest on any demand of interest. In practice, given a program and demand pair, it is possible to determine whether the program responds correctly, or not, to the demand[10]. Therefore, the value of the score function on a given program–demand pair can be defined. However, due to limited resource, it may be impractical to obtain the value of the score function for all program–demand pairs of interest. For example, the demand space might be too large and the complexity of making exhaustive testing infeasible. As a result, the complete score function of a program or system is usually unknown. Nevertheless, the score function is a useful device for reliability modelling. Successful correction of faults can be modelled as changes in the score function for some demands from 1 to 0. If we choose a demand $X$ at random (according to the given demand profile) and look at the score function of a particular program, $\pi$, on this demand, $\omega(\pi, X)$, then $\omega(\pi, X)$ is itself a random variable. This random variable, $\omega(\pi, X) : \mathcal{X} \to \mathbb{R}$, allows us to define the *failure region* of the version $\pi$, which is the event "the set of all demands which cause $\pi$ to fail". That is, $\{x \in \mathcal{X} : \omega(\pi, x) = 1\} \in \Sigma_{\mathcal{X}}$. Consequently, the *pfd* for $\pi$ is the probability of this

[10]This determination may occur "by reason of use" in the sense that an erroneous response to a demand by a program is determined to be such only after the program has been deployed and is in operation.

event, given as[11]

$$\mathrm{P}_X\left(\{x \in \mathcal{X} : \omega\left(\pi, x\right) = 1\}\right) = \sum_{x \in \mathcal{X}:\omega(\pi,x)=1} \mathrm{P}_X(x) = \sum_{x \in \mathcal{X}} \omega(\pi, x)\,\mathrm{P}_X(x).$$

But this is merely the *expected value of the random variable* $\omega(\pi, X)$. This property – being able to write probability statements as expectations – is not an accident, and shall be used frequently in the current work. The "trick" to evaluating the probability of an event is to define an appropriate random variable from a score function, and then take its expectation. Therefore,

$$pfd := P(\pi \text{ fails on } X) = \mathbb{E}_X(\omega(\pi, X)) = \sum_{x \in \mathcal{X}} \omega(\pi, x)\,\mathrm{P}_X(x), \tag{2.1}$$

where the notation $\underset{X}{\mathbb{E}}(\omega(\pi, X))$ designates the expected value, or mean, of the random variable $\omega(\pi, X)$, with respect to the distribution of the random variable $X$.

Furthermore, we may consider a two-version 1–out–of–2 system, its score function given by the product of the score functions of the two program versions that make up the system. Indeed, the system score is 1 (failure) if and only if both versions' scores are also 1 (both fail). Let two specific program versions in a system be $\pi_1$ and $\pi_2$. Then, the *pfd* of the system they form is:

$$\begin{aligned} P(\pi_1 \textit{ and } \pi_2 \textit{ both fail on } X) &= \underset{X}{\mathbb{E}}(\omega(\pi_1, X)\omega(\pi_2, X)) = \sum_{x \in \mathcal{X}} \omega(\pi_1, x)\omega(\pi_2, x)\,\mathrm{P}_X(x) \\ &= pfd_1 pfd_2 + \underset{X}{\mathrm{Cov}}(\omega(\pi_1, X), \omega(\pi_2, X)), \end{aligned} \tag{2.2}$$

where the sign of the covariance term captures the nature (positive or negative) of the failure correlation between the program versions $\pi_1$ and $\pi_2$. A negative covariance term implies that whenever one version fails the other version is less likely to fail than it would be if the versions failed independently. Equation (2.2) demonstrates that there are 3 main contributions to the system *pfd* – the *pfd*s of the single versions and their failure covariation. Clearly, a very reliable system may be obtained with single versions that are very reliable or are significantly diverse. There is a trade–off possible here. For processes that produce highly reliable versions the benefits of diversity may be undermined by the versions not being significantly diverse. We are more precise about this trade–off later on, in the context of diversity experiments, where versions produced that fail together exhibit positive correlation (see Appendix B on page 207).

## 2.3 Modelling the Development of a Program

In this section several probabilistic models will be derived all of which model the development process but with varying levels of uncertainty about what is known of both the circumstances

[11]Because it is a probability measure the first equality here is justified by a certain additivity property of $\mathrm{P}_X$ over disjoint events. The event $\{x \in \mathcal{X} : \omega\left(\pi, x\right) = 1\}$ is the union of disjoint events, each of which is a unique demand that causes $\pi$ to fail.

surrounding the process and the outcomes of activities within the process. It is because of these varying levels of uncertainty that conditional probability distributions, conditional mass functions or conditional measures will be used frequently. The aim is to build the LM model using a "bottom up" approach: we define probabilistic models of atomic stages in the development of a single program and build these up into a model for the entire process. In the next section we use this to define a model for the development of a 1–out–of–2 system built by perfectly isolated teams. This approach both elaborates on the original description of the models in [12, 13], and introduces entities that are necessary for discussing subtleties involved in generalising the models.

For a sufficiently complex program its development process is variable and uncertain. To see this appreciate that a typical development process has an associated methodology and, therefore, consists of a sequence of ***activities***, possibly beginning with activities such as "specification development, system design and team selection" and ultimately ending with the final activity: the deployment of a software program. Some of these activities may have deterministic outcomes (e.g. the choice of which development platform to use may already be known from the start) and others may not (e.g there may be a number of possible and plausible system designs to choose from). Each activity with an uncertain outcome in the process may be modelled probabilistically, in much the same way as the occurrence of demands in the previous section. For the sake of illustration suppose a development process implements an iterative waterfall approach. Activities in the process – such as requirements definition, program design, implementation, and validation – do not have predetermined outcomes. The so–called *deliverables* or *milestones* associated with each stage/activity in the process will constitute the activity outcomes. More generally, outcomes of activities may also include the observable actions and interactions of team members. In fact, ***any observable that results from the performance of an activity is an activity outcome***. Suppose there are $n$, ordered, development process activities labelled $d_1, \ldots, d_n$, each of which has uncertain/variable outcomes. With each activity, say $d_i$, we associate a random variable, $Di$. A realisation of $Di$ will be written as $di$, not to be confused with the activity label $d_i$. For instance, the testing and debugging of some software is a development process activity – labelled $d_i$ say – the result of which is some software – a realisation $di$ of some random variable $Di$ – with failure behaviour that is not completely known, in general. Note that the initial software version upon which the debugging is carried out is itself the random outcome of activities in the development process prior to testing, such as coding, system design and specification writing. The uncertainty in knowing which software version results from debugging is dependent on the initial software to be debugged. This is an example of a more general principle; ***the outcomes of activities in the development process determine the uncertainty in the outcomes of subsequent activities***. Formally, for the activity $d_i$ and associated random variable $Di$ we define a collection/family of probabilistic models, where each probabilistic model is indexed by the outcomes of prior activities in the development process. A member of the collection would be $\big(\Omega_{d_i}, \Sigma_{d_i}, \mathrm{P}_{Di}\left(\cdot|d(i-1), \ldots, d1\right)\big)$, where $d(i-1), \ldots, d1$ are the respective outcomes/instances of the random variables $D(i-1), \ldots, D1$ associated with the respective activities $d_{i-1}, \ldots, d_1$. The terms are similar to the terms in the "demand occurrence" model outlined earlier (see page 25): $\Omega_{d_i}$ and $\Sigma_{d_i}$ are, respectively, a discrete sample space and event space for the $d_i$ activity. The random variable, $Di$, models the outcome of the activity and, consequently, must take on values from the sample space $\Omega_{d_i}$. To this end, we write $di \in \Omega_{d_i}$ as a realisation of $Di$. In this thesis, as a first–pass approximation, we model activities as discrete probability spaces with possibly very large, but finite sample spaces.

The last member of the model is $\mathrm{P}_{Di}\left(\cdot|d(i-1),\ldots,d1\right)$, a conditional probability measure conditional on the realisations of the random variables $D(i-1),\ldots,D1$. This measure assigns probabilities to events that occur in the $d_i$th activity, *given the outcomes of all of the previous activities in the development process.* This is the notion that what has already transpired during the development process should impact later stages in the process. Figure 2.4 is an example of a typical development process with related activities. This graphical depiction of a development process with nodes representing the random outcomes of the activities is intended to be suggestive; the development process may be represented by a ***Bayesian Belief Network*** with a topology as depicted.

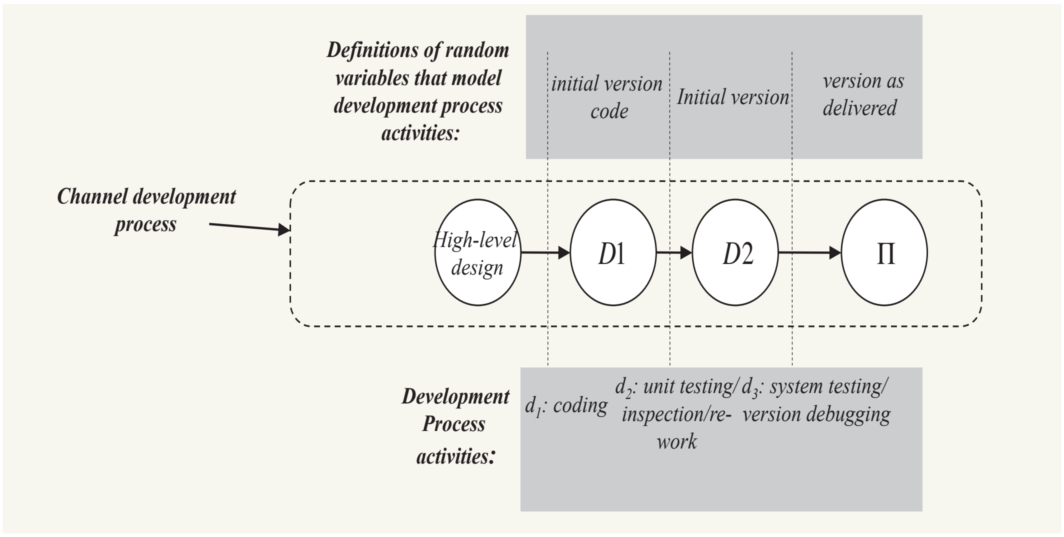

Definitions of random variables that model development process activities:
initial version code
Initial version
version as delivered
Channel development process
High-level design
D1
D2
Π
Development Process activities:
d1: coding
d2: unit testing/ inspection/re-work
d3: system testing/ version debugging

**Figure 2.4:** An example of a simple model of the development process for a version. The process consists of three activities, $d_1, d_2$ and $d_3$. The results of the activities are random and are adequately modelled by associated random variables (i.e. D1, D2 and Π) and probabilistic models – $\left(\Omega_{d_1}, \Sigma_{d_1}, \mathrm{P}_{D1}(\cdot)\right)$, $\left(\Omega_{d_2}, \Sigma_{d_2}, \mathrm{P}_{D2}\left(\cdot|d1\right)\right)$ and $\left(\mathcal{P}, \Sigma_{\mathcal{P}}, \mathrm{P}_{\Pi}\left(\cdot|d2,d1\right)\right)$. Special attention is given to activity $d_3$ whose outcome is the final version that is deployed. The probabilistic model for $d_3$ is $\left(\mathcal{P}, \Sigma_{\mathcal{P}}, \mathrm{P}_{\Pi}\left(\cdot|d2,d1\right)\right)$ where the random variable Π can take on any value from the space of programs $\mathcal{P}$, e.g. $\pi$, with probability determined by $\mathrm{P}_{\Pi}\left(\cdot|d2,d1\right)$, e.g. $\mathrm{P}_{\Pi}\left(\pi|d2,d1\right)$.

Special attention is given to the final activity in the development process. This activity determines the actual program that is deployed for operation; this is the ultimate result of the development process. Conceptually, we model this activity as the random selection of a program from the population of all programs that (at least hypothetically) can be written to the same requirement[12]. Here, by requirement we are appealing to a notion of correct

[12]It is possible to define the space of programs in different ways depending on how these conceptual models

functioning for the program to be built. This notion is not to be confused with the existence of a formal, functional specification document which might have errors in it. Instead, the fact that errors in a formal specification can be detected, even if it is after the software has been developed or deployed or observed to have failed, is evidence that a notion of correct functioning can exist and can be appealed to. Therefore, we need to define a probabilistic model, $\big(\mathcal{P}, \Sigma_{\mathcal{P}}, \mathrm{P}_{\Pi}\left(\cdot|d(n-1),\ldots,d1\right)\big)$, with a *sample space*, $\mathcal{P}$, of all possible program versions that could be developed to the same requirements; an *event space* $\Sigma_{\mathcal{P}}$; a random variable $\Pi$ defined over $\mathcal{P}$ and a conditional probability mass function (*pmf*), $\mathrm{P}_{\Pi}(\pi|d(n-1),\ldots,d1)$, where $\pi \in \mathcal{P}$. We can reasonably assume that $\mathcal{P}$ is finite: any program must fit in some form of computer memory whose size is always finite irrespective of advances in computer engineering. So, given a maximum feasible memory size, $L$, that is to be used in the development process we can assume as the set of all possible programs the set of all possible series of $L$ zeros and ones. Of course, many of these "programs" will have zero probability of being produced. The population of programs, $\mathcal{P}$, will contain programs with all sorts of failure behaviour, ranging from succeeding on all demands to failing on all demands. A realisation of the random variable, $\Pi$, is a single version. The *pmf*, $\mathrm{P}_{\Pi}(\pi|d(n-1),\ldots,d1)$ for $\pi \in \mathcal{P}$, defines the relevant probability distribution for the random variable $\Pi$. This distribution gives the probability of events of the kind $\{\Pi = \pi\}$, i.e. the probability that the final program that is deployed is $\pi$. We call a distribution such as $\mathrm{P}_{\Pi}\left(\pi|d(n-1),\ldots,d1\right)$ a ***version–sampling distribution***. It is dependent on the circumstances surrounding the development process and the outcomes of the activities thereof – e.g. the software specification, the members of the development team, the methods used in developing the software (including the verification and validation policy), the schedule and budget constraints, etc. An illustration of this is given by the example in Fig. 2.4 where $\mathrm{P}_{\Pi}\left(\pi|d2,d1\right)$ is dependent on the realisations of the random variables $D1$ and $D2$. Furthermore, $\mathrm{P}_{\Pi}\left(\pi|D(n-1),\ldots,\ D1\right)$ is itself a random variable since it is a measurable function of the random variables $D(n-1),\ldots,D1$ and, consequently, its expectation may be taken to give another probability measure. That is, $\mathrm{P}_{\Pi}\left(\pi\right) = \underset{D(n-1),\ldots,\ D1}{\mathbb{E}}\left[\mathrm{P}_{\Pi}\left(\pi|D(n-1),\ldots,D1\right)\right]$ is the probability that for a random set of development process circumstances/activities the final version deployed is $\pi$. This allows us to define yet another probabilistic model $\big(\mathcal{P}, \Sigma_{\mathcal{P}}, \mathrm{P}_{\Pi}\left(\right)\big)$. This is the probabilistic model to use when all of the activities in the development process have as yet undetermined outcomes. For instance, before the development process has begun.

So far we have defined models for the individual activities in the process. In order to define a model for the process as a whole we require a *joint probability distribution* that retains all of the information contained in the distributions defined so far. So, we require a model such as

$$\left(\Omega_{d_1} \times \ldots \times \Omega_{d_{n-1}} \times \mathcal{P},\ \Sigma_{\Omega_{d_1} \times \ldots \times \Omega_{d_{n-1}} \times \mathcal{P}}\ ,\ \underset{D1,\ldots,D(n-1),\Pi}{\mathrm{P}}\right). \tag{2.3}$$

are used. Indeed, the space of programs can be defined as the space of all possible programs or, given a fixed amount of memory to be used in the development process, the space of all programs with binaries that can fit in such memory. However, there are implications for the definitions of other entities in the model (such as the score function and the demand space) whenever a particular definition of the program space is used. This is discussed in more detail in Chapter 6.

Each term has significance:

- $\Omega_{d_1} \times \ldots \times \Omega_{d_{n-1}} \times \mathcal{P}$, a cartesian product of the activities' sample spaces, is the sample space for the model of the development process. Each member of this set is an n–dimensional vector (or *n–tuple*) whose components are ordered outcomes of the activities in the development process. For example, if the outcomes of the four activities "specification writing", "coding", "debugging", and "program deployment" are "a particular specification", "a specific initial program version", "a specific subsequent program version" and "a specific deployed program" then this quadruplet of outcomes will form a member of the sample space. So, each member of the sample space represents a unique set of actions, decisions and artefacts that result from activities in the development of a program. That is, *this is the set of all of the ways in which the process can evolve.*

- $\Sigma_{\Omega_{d_1} \times \ldots \times \Omega_{d_{n-1}} \times \mathcal{P}}$ is the set of all of the development process events. For instance, the event "the development process results in the development of program $\pi$" is contained in this set and will be the collection of all of the elements of $\Omega_{d_1} \times \ldots \times \Omega_{d_{n-1}} \times \mathcal{P}$ that contain $\pi$ in their nth component.

- $\underset{D1,\ldots,D(n-1),\Pi}{\mathrm{P}}$ is the probability measure that assigns probabilities to events. This defines the joint distribution of the random outcomes of each activity in the process. This distribution contains all of the information contained in each marginal distribution associated with an activity: it is possible to write the joint distribution as a product of the marginal distributions associated with the activities. Because of this relationship we have

$$
\begin{aligned}
\mathrm{P}_{\Pi}(\pi) &= \underset{D(n-1),\ldots,D1}{\mathbb{E}} \left[\mathrm{P}_{\Pi}\left(\pi | D(n-1), \ldots, D1\right)\right] \\
&= \sum_{\Omega_{d_1} \times \ldots \times \Omega_{d_{n-1}}} \mathrm{P}_{\Pi}\big(\pi | d_1, \ldots, d_{n-1}\big) \underset{D1,\ldots,D(n-1)}{\mathrm{P}} \big(d_1, \ldots, d_{n-1}\big) \\
&= \sum_{\Omega_{d_1} \times \ldots \times \Omega_{d_{n-1}}} \underset{D1,\ldots,D(n-1),\Pi}{\mathrm{P}} \big(d_1, \ldots, d_{n-1}, \pi\big) \qquad (2.4)
\end{aligned}
$$

  a *version–sampling distribution* is obtained from the probability measure by summing over all of the activities' outcomes, except the final activity. Using any of the models of single–version development this "trick" of obtaining *version–sampling distributions* from probability measures can always be achieved by taking analogous expectations to the one in Eq. (2.4). This is a useful place to point out that, in the original formulation of both the EL and LM models, each channel development process is modeled by a probability space such as $(\mathcal{P}, \Sigma_{\mathcal{P}}, \mathrm{P}_{\Pi}(\cdot))$, where $\mathrm{P}_{\Pi}(\cdot)$ is a probability measure defined by the *version–sampling distribution* $\mathrm{P}_{\Pi}(\pi)$. This probability space does not explicitly model the activities of the channel development process. However, the effect of the activities are taken into account since they are used in Eq. (2.4) to obtain $\mathrm{P}_{\Pi}(\pi)$. So, the activities' outcomes are "hidden" in the probability distribution. This is adequate as long as such activities do not create dependence between the channels' development processes. In Chapter 3 we consider scenarios where activities create dependence and, consequently, need to be explicitly modelled.

This simplified[13] probabilistic model matches what we know about the variability of the outcome of a software development process. Although the process is closely controlled we know that its result – the software produced including the faults it contains – is not strictly determined by it. For instance, an assessor who is given all the documentation about the software development (usually showing, among other things, no evidence of residual faults in the delivered software) still does not know on which demands, if any, the software may fail due to unknown faults although he/she may have an approximate idea of the quality to be expected from the software. We can expect different "values" of the circumstances of development to induce different joint distributions of the random variables, $\{D1, \ldots, D(n-1), \Pi\}$. Given a particular development scenario (with its particular circumstances) and thus a joint distribution of $\{D1, \ldots, D(n-1), \Pi\}$, $\omega\left(\pi, x\right)$ then represents the score function of a specific program version, $\pi$, on demand $x$. So, developing a program under given circumstances can indeed be described as running this stochastic production process once, or equivalently "extracting, at random, a sequence of actions and decisions resulting in a program" from a population of many different ways in which the development process could have evolved, with their associated probabilities.

Various related *version–sampling distributions*, such as $\mathrm{P}_{\Pi}(\pi)$ and each $\mathrm{P}_{\Pi}\big(\pi|d_1, \ldots, d_{n-1}\big)$ in Eq. (2.4), can be defined. These distributions are conditional, and they may differ on the random variables being conditioned on. The relevant *version–sampling distribution*, indeed which probabilistic model to use, is determined by how much is certain, or assumed to be known, about the outcomes of development process activities. In fact, it is possible to describe a whole range of *version–sampling distributions* – from those conditional on "complete uncertainty" about the outcomes of all the modelled activities in the development process (i.e. $\mathrm{P}_{\Pi}\left(\pi\right)$) to those conditional on "complete certainty" about all of the activity outcomes except the last one $\big($i.e. $\mathrm{P}_{\Pi}(\pi|d(n-1), \ldots, d1)\big)$. All of these distributions describe the "act" of selecting a program at random from $\mathcal{P}$. In any of these contexts I will use phrases like "a randomly selected program [version]" to mean: the program may have been delivered but it is still unknown in that its score function is unknown.

From the foregoing discussion if the development process is conducted multiple times it may yield different versions with different failure regions. As a consequence given a particular demand the process may sometimes result in a version that fails on that demand and at other times the resulting version will succeed. If a process is more likely to result in a version that fails on the demand as opposed to one that succeeds on it then this could be indicative of some unwanted situation during development. For example, it could mean there is an ambiguity in the initial software specification with the effect that the development team has a tendency to make certain kinds of mistakes and insert certain faults into the version, or it could be a

[13]The sense in which this model is a simplification is discussed in Chapter 6.

problem with the software development tools used making certain mistakes more likely, or it could be a debugging phase that is grossly inadequate and has limited coverage, and so on. Each of these situations may affect the probabilities of individual failure regions being present in the software being developed. Ultimately, this affects the probabilities of individual demands being failure points during system operation. To formalise this each demand has an associated probability that a randomly selected program fails on the demand. We call ***difficulty function*** (a term introduced in [13] to name a concept initially formulated by Eckhardt and Lee [12]) the function defined on the demand space, $\mathcal{X}$, whose value on each demand is the associated probability of failure on the respective demand. This is a function $\theta : \mathcal{X} \to [0,1]$. Recall that each possible version, $\pi$, has an associated score function, $\omega(\pi, x)$, defined on the space of demands, $\mathcal{X}$. So, given "complete uncertainty" about the outcomes of activities in the development process (and thus the probabilistic model $\big(\mathcal{P}, \Sigma_{\mathcal{P}}, \mathrm{P}_{\Pi}(\cdot)\big)$), the *difficulty function* on a particular demand, $x$, is then:

$$\theta\left(x\right) := \mathop{\mathbb{E}}_{\Pi}\left(\omega(\Pi, x)\right) = \sum_{\pi \in \mathcal{P}} \omega\left(\pi, x\right) \mathrm{P}_{\Pi}\left(\pi\right). \tag{2.5}$$

This can be generalised further. If there is certainty concerning some of the activities but not all, for instance some of the activities in the development process have already occurred and their outcomes are known, then the relevant probabilistic model will have a conditional *version–sampling distribution*, conditional on these known outcomes. Suppose the conditional values are $(d1, \ldots, di) \in \Omega_{d_1} \times \ldots \times \Omega_{d_i}$. Then, the relevant probabilistic model is

$$\Big(\Omega_{d_{i+1}} \times \ldots \times \Omega_{d_{n-1}} \times \mathcal{P}, \quad \Sigma_{\Omega_{d_{i+1}} \times \ldots \times \Omega_{d_{n-1}} \times \mathcal{P}}, \quad \mathop{\mathrm{P}}_{D(i+1),\ldots,D(n-1),\Pi}\left(\cdot | d1, \ldots, di\right)\Big)$$

and the probability that the development process results in a version that fails on the demand $x$ is

$$\begin{aligned}
&\theta\left(x | d1, \ldots, di\right) \\
:=\ & \mathop{\mathbb{E}}_{D(i+1),\ldots,D(n-1),\Pi} \big(\omega\left(\Pi, x\right) | d1, \ldots, di\big) \\
=\ & \sum_{\Omega_{d_{i+1}} \times \ldots \times \Omega_{d_{n-1}} \times \mathcal{P}} \omega\left(\pi, x\right) \mathop{\mathrm{P}}_{D(i+1),\ldots,D(n-1),\Pi}\left(d(i+1), \ldots, d(n-1), \pi | d1, \ldots, di\right) \\
=\ & \sum_{\mathcal{P}} \omega\left(\pi, x\right) \mathrm{P}_{\Pi}\left(\pi | d1, \ldots, di\right) \\
=\ & \mathop{\mathbb{E}}_{\Pi} \big(\omega\left(\Pi, x\right) | d1, \ldots, di\big)
\end{aligned} \tag{2.6}$$

which is also a difficulty function. Further still, any average of this difficulty function with respect to any marginal distribution obtained from the joint distribution of the random variables $D1, \ldots, Di$ is also a difficulty function.

The difficulty function – a function yielding difficulties (probability of failure on each demand) – is a characterisation of how some demands may be more "difficult" to develop for than others. The difficulties arise for various reasons related to the uncertainty in the development process. Examples include an inherent intellectual/conceptual difficulty of the problem being solved, the experience of the development team members, the complexity of an algorithm being implemented, the availability of certain tools and resources for the development process and the conditions under which the development process occurs. The result of any of these examples is to affect the probabilities of the programs that can be created and, consequently, for each demand how likely it is to develop a program that fails on the demand. For some demands the likelihood of the development process resulting in a version that fails on the demand may be high. For other demands it may be very likely that the version produced will succeed in appropriately handling the demand. So, there may be *variation in difficulties across the demands.* Later on this chapter when discussing the case for the development of a 1–out–of–2 system we shall demonstrate that this variation in difficulty is integral in reasoning about the independent failure of the channels (see Section 2.5).

We round up this section by demonstrating how to evaluate the *expected single–version pfd.* This is the probability that the development process results in a program that fails in operation. To illustrate this a combination of two models is required: a model of demand occurrence in operation, $\big(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X(\cdot)\big)$, and a model of single–version development[14], such as $\big(\mathcal{P}, \Sigma_{\mathcal{P}}, \mathrm{P}_{\Pi}(\cdot)\big)$. Equation (2.1) shows how to evaluate single–version *pfd*, given a version, $\pi$. So, the expected *pfd* is given by taking the averaging over all single–version *pfd*s. That is, the probability that a randomly selected version, $\Pi$, fails on a random demand, $X$, in operation is

$$P\big(\Pi \text{ fails on a random } X\big) = \mathop{\mathbb{E}}_{\Pi}\Big[\mathop{\mathbb{E}}_{X}[\omega(\Pi, X)]\Big] = \mathop{\mathbb{E}}_{X}\Big[\mathop{\mathbb{E}}_{\Pi}[\omega(\Pi, X)]\Big] = \mathop{\mathbb{E}}_{X}\big[\theta(X)\big].$$

So, the *expected pfd is the expectation of the difficulty function.* Generalising this only requires using a different single–version development model to perform analogous expectations.

## 2.4 Modelling the Development of a 1–out–of–2 System

We turn our attention to model a 1–out–of–2 system development process. We assume that each channel of the system is developed by a unique development team. Furthermore, we assume that the teams are ***perfectly isolated*** from each other: no form of communication between the teams – neither direct nor indirect communication – can occur. This effectively means that during the development process the teams cannot affect each other[15]. This is an important point which we formalise in the following observability criterion.

**Observability Criterion 2.4.1.** *An observer embedded in the development process of a*

[14] Any model of single–version development like the ones discussed so far can be used

[15] Ideally, perfect isolation would mean each team would think they are the only team producing software. Nevertheless, were the teams to be aware of the existence of other teams without communicating with them the effect on the form of the model will still be the same.

*perfectly isolated development team should not be able to confirm, or refute, the existence of any other development process by observing activity outcomes in the process she is embedded in.*

As a consequence of this criterion the probabilistic model of the 1–out–of–2 system is precisely the *product probability space* obtained from the pair of probabilistic models that model each channels development process. Suppose one channel is labelled "$A$" and the other "$B$". Let $\big(\Omega_A, \Sigma_A, \mathrm{P}_A\big)$ model the development of channel $A$. Similarly, let $\big(\Omega_B, \Sigma_B, \mathrm{P}_B\big)$ model the development of channel $B$. For simplicity we shall assume that the development processes have corresponding activities: for each activity related to channel A development there is a corresponding activity related to the development of channel B, and vice–versa, and these corresponding activities have identical, related sample spaces[16] Also, we assume that the outcome of each development process activity is not known beforehand: there is uncertainty in each development process activity. Then, consequently, the processes necessarily have the same number of modelled activities and the same form of probabilistic model. This simplification does not come at a price concerning the main results of the model. The form of the probabilistic model for each process is given in Eq. (2.3) because each of the channels are developed in isolation. So, for instance, we could have written the sample space $\Omega_A$ more verbosely as $\Omega^A_{d_1} \times \ldots \times \Omega^A_{d_{n-1}} \times \mathcal{P}$ where we have indicated that this is relevant for channel $A$ by adding superscript $A$s. Note that there is no superscript on $\mathcal{P}$ as it is the same for the development of both channels $A$ and $B$: both development teams are randomly choosing their respective versions from the same population of versions. Also, we could have written the event space, $\Sigma_A$, as $\Sigma_{\Omega^A_{d_1} \times \ldots \times \Omega^A_{d_{n-1}} \times \mathcal{P}}$ and the probability measure, $\mathrm{P}_A$, as $\underset{D1_A,\ldots,D(n-1)_A,\Pi_A}{\mathrm{P}}$. Consequently, we may define the probabilistic model of the development of the 1–out–of–2 system as

$$\big(\Omega_A \times \Omega_B,\, \Sigma_A \times \Sigma_B,\, \mathrm{P}_A \times \mathrm{P}_B\big). \tag{2.7}$$

Technically, this is the *product probability space* obtained from the models

$$\big(\Omega_A, \Sigma_A, \mathrm{P}_A\big) \text{ and } \big(\Omega_B, \Sigma_B, \mathrm{P}_B\big).$$

While this space can always be defined as a mathematical construct it is not the model of joint system development in general. Its use here is justified purely by perfect isolation. Keeping in mind that we are dealing with finite, discrete probability distributions there are some noteworthy points to make:

- Perfect isolation justifies the sample space of the model being the cartesian product of the marginal processes, $\Omega_A \times \Omega_B$. This is because, under perfect isolation, if one were to place an observer in one of the development processes for a channel it would be impossible for the observer to prove that another process either exists or not. As far

[16]This assumption is made purely to reduce the need for extra notation; other than extra notation there is not a significant amount of added model complexity in considering a more general case. Indeed, developing models for the case where the channels' respective development processes have different types of activities between them follows an analogous treatment to the simpler case presented here.

as an observer is concerned his/her development process *exhibits no difference between developing a single-version system and developing a channel for a 1–out–of–2 system.* This is an important point that will be useful later when comparisons need to be made between the reliability of single–version systems and the reliability of related Fault–tolerant systems.

- the event space, $\Sigma_A \times \Sigma_B$, is the smallest event space that can be constructed containing sets of the kind $S_A \times S_B$, where $S_A \in \Sigma_A$ and $S_B \in \Sigma_B$.[17] This standard notation is *not the cartesian product* of the event spaces for the marginal processes.

- The product of the probability measures, $\mathrm{P}_A \times \mathrm{P}_B$, is also justified by isolation. In this situation, given the state of the world (such as the educational backgrounds of the team members, or the hardware on which the teams carry out their software development), there is no reason for the teams to be correlated in how they develop their respective versions: knowledge of one team's choices during software development tells you nothing about the other team's choices[18]. In effect, the teams are independent in their selection of versions from $\mathcal{P}$. Consequently, the probability measure for events of the combined process factors into a product of the marginal probability measures. Note that $\mathrm{P}_A \times \mathrm{P}_B$ is not a cartesian product but is instead a product of functions. That is,

$$\begin{aligned} & (\mathrm{P}_A \times \mathrm{P}_B)\big(d1_A, \ldots, d(n-1)_A, \pi_A, d1_B, \ldots, d(n-1)_B, \pi_B\big) \\ = \quad & \mathrm{P}_A\big(d1_A, \ldots, d(n-1)_A, \pi_A\big)\mathrm{P}_B\big(d1_B, \ldots, d(n-1)_B, \pi_B\big), \end{aligned} \tag{2.8}$$

where $(d1_A, \ldots, d(n-1)_A, \pi_A) \in \Omega_A$ and $(d1_B, \ldots, d(n-1)_B, \pi_B) \in \Omega_B$.

In summary, ***isolation justifies probabilistic independence of program selection***: that is, because the isolated teams make "independent" decisions, their joint distribution of developing a pair of versions factors into the product of their respective distributions of developing single versions. So, isolation is a very convenient property to check when trying to decide whether a given practical scenario is adequately modeled as exhibiting (probabilistically) independently developed versions. This prompts the following question. Are there other forms of justification for probabilistic independence of program selection that may be used in practice? To prove that other forms of justification exist, one need only construct a suitable example scenario: a scenario for which team isolation does not hold in a practically meaningful way, and yet the scenario is adequately modelled by probabilistic independence in a non–trivial way[19]. However, constructing such a scenario is problematic, in general. One problem lies in trying to define the sample space as the cartesian product of

[17]Technically, $\Sigma_A \times \Sigma_B$ is the *sigma–algebra* generated by sets of the form $S_A \times S_B$, where $S_A \in \Sigma_A$ and $S_B \in \Sigma_B$

[18]In Chapter 3 we shall demonstrate that certain sources of randomness, such as a shared educational background of the team members, induces dependence between the teams in how they develop their versions.

[19]Here we are excluding the following trivial case, and variations thereof: $X$ and $Y$ are conditionally independent random variables, conditional on the random vector $X, Y$. Trivial cases of this kind are not useful since the conditional probabilities are deterministic functions of the conditioning random variables. Consequently, this amounts to multiplying joint probabilities by 1, an operation that brings no benefit in

a pair of sample spaces: that is, defining a sample space for which members are ordered pairs of marginal sample spaces, where these ordered pairs make sense in the context of a joint development process with communicating/dependent teams. By their very nature cartesian products, such as $\Omega_A \times \Omega_B$, encapsulate notions of "independence" or "perpendicularity" so that

*any $a \in \Omega_A$ may be paired with any $b \in \Omega_B$ to form a valid ordered pair $(a, b) \in \Omega_A \times \Omega_B$.*

However, for software development with interacting development teams, only certain ordered pairs will make sense: that is, ordered pairs that agree on the results of team interaction (for instance, the discussions that occurred between the teams or the test suites used by both teams if they share test suites). This inability of being able to ensure all members of the cartesian product make sense suggests that such a cartesian product does not make a well defined state space for an experiment with dependence. Indeed, only when the teams are isolated does the experiment sample space always make sense as a cartesian product. Given a cartesian product as a sample space for interacting teams, one might attempt to make nonsensical ordered pairs irrelevant by assigning zero probability to their occurrence. Certainly, doing so requires assigning zero probability to the occurrence of at least one of the members of each nonsensical ordered pair. However, note that a nonsensical ordered pair of outcomes is comprised of perfectly legitimate marginal outcomes, where these marginal outcomes might necessarily have non–zero marginal probabilities of occurrence (e.g. note that $a \in \Omega_A$, from the ordered pair $(a, b)$, is a member of the sample space associated with the probabilistic model $\big(\Omega_A, \Sigma_A, \mathrm{P}_A\big)$). Consequently, in general, it may not be possible to assign zero probability events for the product probability space without contradicting the probability of occurrence some legitimate outcome in at least one of the marginal probability spaces. So, given these difficulties in constructing a suitable example, it is unclear what other forms of justification (if any) exist for a scenario to be modeled by independent *version–sampling distributions*[20].

If team isolation was shown to be "necessary and sufficient" in the less than rigorous sense we have just outlined, then any of our models exhibiting probabilistic independence can immediately be asserted to describe a process of isolated software development. This has further implications for the scope and applicability of the EL and LM models. In any case, some generalisations of the models presented in Chapter 3 use an interplay of both *activities common to the teams' development processes*, and *team isolation*, to achieve models that relax the assumption of "perfectly isolated" teams.

The model stated in (2.7) is the relevant model when there is "complete uncertainty" concerning the development process. However, similar to the single–version development process models, it is possible and sometimes necessary to instantiate models at varying levels of uncertainty about the "combined" development process. So, given the values

understanding the possible dependence between $X$ and $Y$. That is,

$$P(X, Y) = P(X, Y)\, P(X|X, Y)\, P(Y|X, Y) = P(X, Y) \cdot 1 \cdot 1,$$

since $P(X|X, Y) = P(Y|X, Y) = 1$.

[20]A more detailed discussion of the nature of this independence is given in Chapter 3.

$(d1_A, \ldots, di_A) \in \Omega^A_{d_1} \times \ldots \times \Omega^A_{d_i}$ and $(d1_B, \ldots, dj_B) \in \Omega^B_{d_1} \times \ldots \times \Omega^B_{d_j}$, the relevant probabilistic model is

$$\left( \begin{array}{c} \Omega^A_{d_{i+1}} \times \ldots \times \Omega^A_{d_{n-1}} \times \mathcal{P} \times \\ \Omega^B_{d_{j+1}} \times \ldots \times \Omega^B_{d_{n-1}} \times \mathcal{P} \end{array}, \Sigma_{\substack{\Omega^A_{d_{i+1}} \times \ldots \times \Omega^A_{d_{n-1}} \times \mathcal{P} \times \\ \Omega^B_{d_{j+1}} \times \ldots \times \Omega^B_{d_{n-1}} \times \mathcal{P}}}, \left( \begin{array}{c} \underset{D(i+1)_A, \ldots, \Pi_A}{\mathrm{P}} \times \\ \underset{D(j+1)_B, \ldots, \Pi_B}{\mathrm{P}} \end{array} \right) \left( \cdot \,\middle|\, \begin{array}{c} d1_A, \ldots, di_A, \\ d1_B, \ldots, dj_B \end{array} \right) \right). \tag{2.9}$$

In particular, note that like Eq. (2.8) the model still exhibits probabilistic independence. For we have

$$\begin{aligned} & \left( \underset{D(i+1)_A, \ldots, D(n-1)_A, \pi_A}{\mathrm{P}} \times \underset{D(j+1)_B, \ldots, D(n-1)_B, \pi_B}{\mathrm{P}} \right) \left( \begin{array}{c} d(i+1)_A, \ldots, \Pi_A, \\ d(j+1)_B, \ldots, \Pi_B \end{array} \,\middle|\, \begin{array}{c} d1_A, \ldots, di_A, \\ d1_B, \ldots, dj_B \end{array} \right) \\ = & \underset{D(i+1)_A, \ldots, \Pi_A}{\mathrm{P}} \left( d(i+1)_A, \ldots, \Pi_A \,|\, d1_A, \ldots, di_A \right) \underset{D(j+1)_B, \ldots, \Pi_B}{\mathrm{P}} \left( d(j+1)_B, \ldots, \pi_B \,|\, d1_B, \ldots, dj_B \right), \end{aligned} \tag{2.10}$$

where $(d(i+1)_A, \ldots, \pi_A) \in \Omega^A_{d_{i+1}} \times \ldots \times \mathcal{P}$, and $(d(j+1)_B, \ldots, \pi_B) \in \Omega^B_{d_{i+1}} \times \ldots \times \mathcal{P}$. In fact, no matter how much uncertainty is eliminated from the models *isolation will always imply independence of the* version–sampling distribution*s*. It is this property that we refer to as the ***Independent Sampling Assumption*** (ISA). We explore this consequence of "perfectly isolated" development teams further in Chapter 3.

The notion of *difficulty function* extends to the case of a 1–out–of–2 system development. Consider the case when there is "complete uncertainty" concerning the development process activities. Then the relevant probabilistic model is given by (2.7) and the probability that the development process results in a pair of versions that fail on a given demand, $x$, is

$$\begin{aligned} & \theta_{AB}(x) \\ := & \underset{\substack{D1_A, \ldots, D(n-1)_A, \Pi_A, \\ D1_B, \ldots, D(n-1)_B, \Pi_B}}{\mathbb{E}} \left( \omega\left(\Pi_A, x\right) \omega\left(\Pi_B, x\right) \right) \\ = & \sum_{\Omega_A \times \Omega_B} \left( \omega\left(\pi_A, x\right) \omega\left(\pi_B, x\right) \right) \mathrm{P}_A\left(d1_A, \ldots, d(n-1)_A, \pi_A\right) \mathrm{P}_B\left(d1_B, \ldots, d(n-1)_B, \pi_B\right) \\ = & \sum_{\mathcal{P} \times \mathcal{P}} \left( \omega\left(\pi_A, x\right) \omega\left(\pi_B, x\right) \right) \mathrm{P}_{\Pi_A}(\pi_A) \mathrm{P}_{\Pi_B}(\pi_B) \\ = & \left( \sum_{\mathcal{P}} \omega(\pi_A, x) \mathrm{P}_{\Pi_A}(\pi_A) \right) \left( \sum_{\mathcal{P}} \omega(\pi_B, x) \mathrm{P}_{\Pi_B}(\pi_B) \right) \end{aligned}$$

$$= \underset{\Pi_A}{\mathbb{E}}\left[\omega\left(\Pi_A, x\right)\right]\underset{\Pi_B}{\mathbb{E}}\left[\omega\left(\Pi_B, x\right)\right]$$

$$= \theta_A(x)\theta_B(x) \tag{2.11}$$

The third equality above is justified by using the relationship in Eq. (2.4).[21] We can now see yet another dramatic consequence of the "perfect isolation" assumption; the probability that the development produces a pair of versions that fail on the demand is equal to the product of the probabilities of each marginal process producing a version that fails on the demand. That is, *"perfect isolation" implies conditional failure independence, conditional on a demand.*

It is unsurprising that this can be generalised further since the precise meaning of the difficulty function depends on what the underlying probabilistic model being used is. If, for instance, the model in Eq. (2.9) is used then, given the values $d1_A, \ldots, di_A \in \Omega^A_{d_1} \times \ldots \times \Omega^A_{d_i}$ and $d1_B, \ldots, dj_B \in \Omega^B_{d_1} \times \ldots \times \Omega^B_{d_j}$ the probability that the development process results in a pair of versions that fails on a demand, $x$, is

$$\theta_{AB}\left(x \,\middle|\, \begin{array}{c} d1_A, \ldots, di_A, \\ d1_B, \ldots, dj_B \end{array}\right)$$

$$:= \underset{\substack{D(i+1)_A, \ldots, D(n-1)_A, \Pi_A \\ D(j+1)_A, \ldots, D(n-1)_B, \Pi_B}}{\mathbb{E}} \left(\omega\left(\Pi_A, x\right)\omega\left(\Pi_B, x\right) \,\middle|\, \begin{array}{c} d1_A, \ldots, di_A, \\ d1_B, \ldots, dj_B \end{array}\right)$$

$$= \underset{\Pi_A}{\mathbb{E}}\Big(\omega\left(\Pi_A, x\right)|d1_A, \ldots, di_A\Big)\underset{\Pi_B}{\mathbb{E}}\Big(\omega\left(\Pi_B, x\right)|d1_B, \ldots, dj_B\Big)$$

$$= \theta_A\big(x|d1_A, \ldots, di_A\big)\theta_B\big(x|d1_B, \ldots, dj_B\big). \tag{2.12}$$

So, conditional failure independence is preserved for the form of the difficulty function.

## 2.5 EL and LM Model Results

Suppose a model of a 1–out–of–2 system development process with perfectly isolated development teams, say $\big(\Omega_1 \times \Omega_2, \Sigma_1 \times \Sigma_2, \mathrm{P}_1 \times \mathrm{P}_2\big)$, has been defined from the models, $\big(\Omega_1, \Sigma_1, \mathrm{P}_1\big)$ and $\big(\Omega_2, \Sigma_2, \mathrm{P}_2\big)$, of the development of constituent channels. Further, suppose that a model

[21] Also, *Fubini's theorem* was used to justify the factoring of the sums.

of "demand occurrence", say $(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X)$, is independently defined. The *LM model* is the combined pair of probabilistic models,

$$\left\{\big(\Omega_1 \times \Omega_2, \Sigma_1 \times \Sigma_2, \mathrm{P}_1 \times \mathrm{P}_2\big), (\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X)\right\}.$$

If, in particular, $\big(\Omega_1, \Sigma_1, \mathrm{P}_1\big)$ and $\big(\Omega_2, \Sigma_2, \mathrm{P}_2\big)$ have identical distributions then this is the *EL model*. Identical distributions would mean both channels are developed using identical methodologies: in essence, the teams develop their respective channels under similar circumstances and using similar resources. An immediate consequence of these definitions is as follows. Recall, as was demonstrated in Eq. (2.4), that models of single–version development such as $\big(\Omega_1, \Sigma_1, \mathrm{P}_1\big)$ have related *version–sampling distributions*. Because difficulty functions are expectations of score functions with respect to the *version–sampling distributions* ( for instance, see Eq.s (2.5),(2.6) ) the difficulty functions for the channels are necessarily identical in the EL model but not in the LM model. So, given a demand the teams have identical probabilities of producing versions that fail on that demand in the EL model; this need not be the case for the LM model.

We can now derive the first result of the EL and LM models: for two software versions, independently developed by two teams, it is inappropriate to estimate their joint *pfd* by multiplying their individual *pfd* estimates. To illustrate this suppose we have a practical scenario with a level of uncertainty so that it is adequately modelled by Eq. (2.7); however, note that any EL and LM model can be used to illustrate the result. The EL model assumes that despite the teams being separated the same constraints (circumstances of development imposed on the development teams) cause the teams to develop their versions roughly "in the same way"; that is, the teams are equally likely to make the same mistakes during the development of their respective software. In effect the teams have identical *version–sampling distributions* – that is, $\mathrm{P}_{\Pi_1}(\pi) = \mathrm{P}_{\Pi_2}(\pi)$ for $\pi \in \mathcal{P}$ – and, consequently, identical difficulty functions – that is, $\theta_1(x) = \theta_2(x) = \theta(x)$ for each demand $x$. What is the expected system *pfd* under these conditions? This is the probability that the development process results in a pair of versions that fail together in operation. It is given by taking the expectation of the system pfd ( given by Eq. (2.2) ) with respect to the distribution of version pairs, given by Eq. (2.7). Intuitively, we are summing the probabilities of all the unique ways in which a pair of versions is developed, deployed and subsequently fails in operation. Therefore, upon submitting a randomly chosen demand to a randomly chosen pair of versions the probability of both failing is:

$$\begin{aligned}
& P(\Pi_1, \Pi_2 \text{ both fail on X}) \\
= & \underset{\Pi_1,\Pi_2}{\mathbb{E}}\left[\underset{X}{\mathbb{E}}\left[\omega(\Pi_1, X)\omega(\Pi_2, X)\right]\right] \\
= & \underset{X}{\mathbb{E}}\left[\theta_{12}(X)\right] \\
= & \underset{X}{\mathbb{E}}\left[\theta_1(X)\theta_2(X)\right] \\
= & \underset{X}{\mathbb{E}}\left[\theta_1(X)\right]\underset{X}{\mathbb{E}}\left[\theta_2(X)\right] + \underset{X}{\mathrm{Cov}}\big(\theta_1(X), \theta_2(X)\big)
\end{aligned}$$

$$\begin{aligned}
&= \underset{X}{\mathbb{E}}\left[\underset{\Pi_1}{\mathbb{E}}\left[\omega(\Pi_1, X)\right]\right]\underset{X}{\mathbb{E}}\left[\underset{\Pi_2}{\mathbb{E}}\left[\omega(\Pi_2, X)\right]\right] + \underset{X}{\mathrm{Var}}\left(\theta(X)\right)\\
&= \underset{\Pi_1}{\mathbb{E}}\left[\underset{X}{\mathbb{E}}\left[\omega(\Pi_1, X)\right]\right]\underset{\Pi_2}{\mathbb{E}}\left[\underset{X}{\mathbb{E}}\left[\omega(\Pi_2, X)\right]\right] + \underset{X}{\mathrm{Var}}\left(\theta(X)\right)\\
&= P\left(\Pi_1 \text{ fails on } X\right)P\left(\Pi_2 \text{ fails on } X\right) + \underset{X}{\mathrm{Var}}\left(\theta(X)\right)\\
&= \left(P\left(\Pi_1 \text{ fails on } X\right)\right)^2 + \underset{X}{\mathrm{Var}}\left(\theta(X)\right) \qquad (2.13)
\end{aligned}$$

where $\underset{X}{\mathrm{Var}}\left(\theta(X)\right)$ designates the variance of the difficulty function, $\theta(X)$. Since the variance of any random variable is non-negative (2.13) shows that the average system *pfd* will generally be worse than the value of the product term, $\left(P\left(\Pi_1 \text{ fails on } X\right)\right)^2$. This product is the value of the expected system *pfd* if the versions were expected to fail independently. That is, the independently produced versions cannot be expected to fail independently. Intuitively, the teams are similar in how they produce their respective versions, despite being isolated from one another. Consequently, the teams have identical difficulty functions. This means that demands that are difficult for one team are also difficult for the other team. So, if one team produces a version that fails in operation then the probability that the other team produces a version that fails on the same demand that caused the first failure is more likely than it would be under a naive assumption of failure independence between the system channels. That is,

$$\begin{aligned}
P\left(\Pi_1 \text{ fails on } X|\Pi_2 \text{ fails on } X\right) &= \frac{P\left(\Pi_1, \Pi_2 \text{ both fail on X}\right)}{P\left(\Pi_2 \text{ fails on } X\right)}\\
&= \frac{\left(P\left(\Pi_1 \text{ fails on } X\right)\right)^2 + \underset{X}{\mathrm{Var}}\left(\theta(X)\right)}{P\left(\Pi_1 \text{ fails on } X\right)}\\
&= P\left(\Pi_1 \text{ fails on } X\right) + \frac{\underset{X}{\mathrm{Var}}\left(\theta(X)\right)}{P\left(\Pi_1 \text{ fails on } X\right)}\\
&\geq P\left(\Pi_1 \text{ fails on } X\right), \qquad (2.14)
\end{aligned}$$

since $P\left(\Pi_1 \text{ fails on } X\right) = P\left(\Pi_2 \text{ fails on } X\right)$, $\underset{X}{\mathrm{Var}}\left(\theta(X)\right) \geq 0$, and it is reasonable to assume $P\left(\Pi_1 \text{ fails on } X\right) > 0$. Note, since any version pair the teams produce must handle the same demand in operation there is an inevitable, non-negative failure correlation between the channels described by $\underset{X}{\mathrm{Var}}\left(\theta(X)\right)$.

This inevitability of failure correlation also holds in the more general context of the LM model. Like the EL case, *independently produced versions cannot be expected to fail independently.* However, unlike the EL model, the LM model recognises that constraints on the teams and the circumstances surrounding the versions development processes are unlikely to be identical. This is equivalent to requiring that $\left(\Omega_1, \Sigma_1, \mathrm{P}_1\right)$ and $\left(\Omega_2, \Sigma_2, \mathrm{P}_2\right)$ need not have the same distributions. Therefore, the teams are not likely to have identical difficulty functions: in general, $\theta_1(x) \neq \theta_2(x)$ for some $x$. So, unlike the EL situation,

$$\begin{aligned}
&P(\Pi_1, \Pi_2 \text{ both fail on X})\\
&= P(\Pi_1 \text{ fails on } X)P(\Pi_2 \text{ fails on } X) + \underset{X}{\mathrm{Cov}}\left(\theta_1(X), \theta_2(X)\right), \qquad (2.15)
\end{aligned}$$

where we have a covariance instead of a variance, and different expected single–version *pfd*s. Possibly, the covariance could be negative: the average system *pfd* could be better than

$$P(\Pi_A \text{ fails on } X)P(\Pi_B \text{ fails on } X).$$

So, the lower bound for the mean system *pfd* is no longer this product term. Negative covariation would mean that if one team produces a version that fails in operation then the other team is less likely to also produce a version that fails than it would be if the channels were expected to fail independently. That is, using an argument similar to the one used in Eq. (2.14), we can show that

$$P\left(\Pi_1 \text{ fails on } X | \Pi_2 \text{ fails on } X\right) \le P(\Pi_1 \text{ fails on } X),$$

whenever $\underset{X}{\mathrm{Cov}}\left(\theta_1(X), \theta_2(X)\right) \le 0$. Such covariation could be the consequence of "***forced diversity***" – that is, the imposition of different constraints, on the channels' developments and their respective developers, by the management team. The point is where failure diversity occurs "*naturally*" in the EL model (that is, purely as a consequence of the random decisions made by the teams) under "forced diversity" the teams are "encouraged" to come to their decisions from different viewpoints, under different circumstances and in different ways. In effect the teams are diverse in how they develop their versions (i.e. they sample from $\mathcal{P}$ with different *version–sampling distributions*). The aim is for this diversity to ultimately translate into failure diversity, on average.

Given two particular versions, $\pi_1$ and $\pi_2$, it is always true that the probability of the versions failing together in operation is always less than, or equal to, each versions probability of failing in operation. This is because for each demand, $x$, it is always the case that $\omega\left(\pi_1, x\right)\omega\left(\pi_2, x\right) \le \omega\left(\pi_1, x\right)$ or $\omega\left(\pi_2, x\right)$ and, consequently,

$$\underset{X}{\mathbb{E}}\left[\omega\left(\pi_1, X\right)\omega\left(\pi_2, X\right)\right] \le \underset{X}{\mathbb{E}}\left[\omega\left(\pi_1, X\right)\right] \;\; or \;\; \underset{X}{\mathbb{E}}\left[\omega\left(\pi_2, X\right)\right]. \tag{2.16}$$

This result is valid for a pair of versions and it indicates that in this context "*fault–tolerance is always good*". However, when there is uncertainty about which versions make up the channels of the system this result does not always hold. This is particularly important in the situation where the system has not been built yet. Is the expected reliability of a single version system worse, in general, than that of a 1–out–of–2 system? The LM and EL models allow such comparisons of expected single–version *pfd* with expected system *pfd*. This is because

*the "perfectly isolated teams" assumption*[22] *implies that the probability a channel (of a 1–out–of–2 system) that fails in operation is developed* ***is identical to*** *the probability a single–version system that fails in operation is developed.*

These probabilities are, in general, different as they are related to different experiments.

[22]see Criterion 2.4.1, page 38.

The former probability is related to the experiment where a pair of versions are created by isolated development teams and the latter probability is related to the experiment where only a single version is created. Consequently, the LM model shows that in this context *fault–tolerance is always good* since

$$P\big(\Pi_1 \text{ fails on } X\big) = \mathop{\mathbb{E}}_{X}\left[\theta_1\left(X\right)\right] \geq \mathop{\mathbb{E}}_{X}\left[\theta_1\left(X\right)\theta_2\left(X\right)\right] = P\big(\Pi_1, \Pi_2 \text{ both fail on } X\big). \quad (2.17)$$

The point is that the probability on the far left of Eq. (2.17) is the probability that a single–version system that fails in operation is developed while on the far right is the probability that a 1–out–of–2 system that fails in operation is developed. While such comparisons and conclusions are possible under LM this is not always feasible in more general settings explored later. In general, fault–tolerance is not guaranteed to improve expected reliability. It is worth pointing out that a comparison of marginal and joint probabilities is always possible so that Eq. (2.17) is always true: the probabilities of marginal events are, in general, larger than the probabilities of related joint events. In this sense the veracity of Eq. (2.17) is not a consequence of the LM model. However, this is not the sense in which Eq. (2.17) is being used.

Forcing diversity does not always result in expected system *pfd* that is smaller than it would be if diversity were allowed to occur naturally. As an example consider two ways in which the development of a version may be organised, labelled "$A$" and "$B$" say. We have defined these as development process ***methodologies***. Recall that methodologies will involve the specification of stages in the development process, the activities to be carried out, the technologies used and the definition of tasks and responsibilities within the development team, among other things. The effect of a choice of methodology for a development process is to affect the way the software is developed and, consequently, to determine the probabilistic law which describes how the respective team develops a program from the population of programs, $\mathcal{P}$. So, suppose that use of methodology "$A$" results in the applicable probabilistic model being $\big(\Omega_A, \Sigma_A, \mathrm{P}_A\big)$, and similarly for "$B$". In Chapter 3 we shall generalise this effect by showing that in addition to methodologies there are also other sources of randomness that ultimately influence the *version–sampling distribution*s. Note that the notation for the probabilistic models used here has a different meaning from its use previously; here the labels indicate which methodology is used in the development process and not which channel is being developed. Given a choice between two development methodologies, "$A$" and "$B$", there are three ways in which a joint system development process may be organised: the channels are both developed using methodology "$A$", the channels are both developed using methodology "$B$", or the channels' development processes employ different methodologies. So, there are two EL models and one LM model possible. Finally, suppose the methodologies are such that for each demand, $x$, use of methodology "$A$" ensures a smaller probability of producing a version that fails on $x$ compared to the related probability when "$B$" is used. That is, the difficulty functions are ordered so that $\theta_A\big(x\big) \leq \theta_B\big(x\big)$, for each $x \in \mathcal{X}$. If the related expected system *pfd*s for the three scenarios are $q_{AA}, q_{BB}$ and $q_{AB}$ then

$$q_{AA} \leq q_{AB} \leq q_{BB}, \text{ that is}$$
$$\mathop{\mathbb{E}}_{X}\left[\theta_A^2(X)\right] \leq \mathop{\mathbb{E}}_{X}\left[\theta_A(X)\theta_B(X)\right] \leq \mathop{\mathbb{E}}_{X}\left[\theta_B^2(X)\right]$$

So, *in general, forcing diversity does not guarantee better expected reliability than if diversity were allowed to occur naturally.* There are, however, cases for which forcing diversity produces the smallest reliability estimate, compared with if diversity were allowed to occur naturally. For instance, as was shown in [13],

$$\text{if } q_{AA} = q_{BB} \text{ then } q_{AB} \leq q_{AA}, q_{BB}. \tag{2.18}$$

$q_{AB}$ is a lower bound for the three expected reliabilities. This is a consequence of the geometric properties of difficulty functions; a proof of this using the *Cauchy-Schwarz* inequality is given later on (see Chapter 5). This inequality is not only useful for aiding intuition concerning relationships between *pfds*; it suggests a practical preference. For suppose that an assessor of a 1–out–of–2 systems development has no actual estimates of the expected *pfd*s $q_{AA}, q_{BB}$ and $q_{AB}$. However, suppose further that the assessor has evidence to justify that using methodology "*A*" to develop both channels will result, on average, in the same reliability that would result if methodology "*B*" alone were used instead. Then, if the assessor had estimates for $q_{AA}$ and $q_{BB}$ she would have to be indifferent between these expected *pfd*s. Consequently, inequality (2.18) must hold, and a clear preference on this basis is presented; diversity should be forced[23].

Here we sound a note of caution. These results, like most of the results presented in this work, involve comparisons between averages. Therefore, care must be taken when interpreting and using these results in practice. For there is uncertainty about the actual pairs of versions that will be developed and deployed in the system, and it is in the face of such uncertainty that these results are most useful. However, if questions are asked about the *pfd* of a system made up of a specified pair of versions, the results presented here should not be applied directly. Indeed, it is quite possible that the system, when built, possesses a *pfd* significantly different from the expected system *pfd*. One can appreciate the context in which the results presented here apply, and when they do not, by treating the system *pfd* as a random variable. This random variable has realisations given by Eq. (2.2), and an expectation – the expected system *pfd* – given by either Eq. (2.13) when diversity is allowed to occur naturally, or Eq.(2.15) when diversity is forced. Now, the results presented here are comparisons between the means of such random variables. Unsurprisingly, therefore, a pair of such distributions may exhibit the property that one of the distributions has a smaller mean, and yet realisations from these distributions are not restricted to emulate this ordering. In summary, when the results indicate forcing diversity to be potentially beneficial, this is a reasonable course of action given uncertainty about the actual *pfd* of the system when

[23]Of course, a more complete decision process will take into account other factors, such as the economic costs of implementing the various alternatives. The preference results presented here should be viewed as forming part of a wider strategy for decision making.

built.

With this caveat in mind, one more condition under which forcing diversity is a preference can be stated. Again, we consider the 3 scenarios that lead to the expected system *pfd*s $q_{AA}, q_{BB}$ and $q_{AB}$. However, suppose a hypothetical assessor is indifferent between using the "$A$" methodology to develop both channels and using the "$B$"methodology to develop both channels. There are two situations under which this would be the case:

1. The assessor has an equal balance of "pros and cons". For example, while the use of the "$A$" methodology is expected to result in better reliability the "$B$" methodology is more affordable and it is debatable whether the reliability gains of using "$A$" are worth the extra cost. In this case, despite $q_{AA} \leq q_{BB}$ the assessor is unwilling to make a clear choice between the two: any one will do;
2. The assessor does not have sufficient evidence to indicate an ordering between the expected pfds $q_{AA}$ and $q_{BB}$, despite the assessor believing that it is unlikely these pfds are identical and, consequently, some ordering must exist.

In either of these cases the assessor's indifference between the two methodologies can be modelled as the assessor being equally likely to pick one over the other. In addition, suppose the cost of forcing diversity is not significantly different from the cost of using a single methodology for developing both channels of the system. This would mean a decision about whether or not to force diversity will depend on the sizes of the expected pfds. Should the assessor advocate forcing diversity, or should the system be built by using the same methodology to develop both channels? Even without having actual estimates for the expected *pfd*s the assessor can claim

$$q_{AB} \leq \frac{q_{AA} + q_{BB}}{2}, \tag{2.19}$$

whatever the (possibly unknowable) expected system *pfd* values may be.

*Proof.* This follows naturally because[24]

$$0 \leq \mathop{\mathbb{E}}_{X}\left[\theta_A(X) - \theta_B(X)\right]^2 = \mathop{\mathbb{E}}_{X}\left[\theta_A^2(X) + \theta_B^2(X) - 2\theta_A(X)\theta_B(X)\right] = q_{AA} + q_{BB} - 2q_{AB}. \blacksquare$$

In Chapter 5 we present a generalisation of this result for 1–out–of–N systems.

## 2.6 Visual Representations of the EL and LM models

Modelling coincident failure can be a task filled with subtlety and nuance. To facilitate proofs as well as aid understanding this section outlines two forms of representing the EL and LM models. These visual representations of the models will be used extensively in subsequent chapters. One representation uses Graphical models and the other representation uses vector space geometry. The graphical models will be useful in discussions about relaxing the "perfect

[24]There is a more general proof using *Holder's inequality*, and the relationship between the *arithmetic–mean* and the *geometric–mean* of a collection of non-negative real numbers. This generalisation is presented later on in Chapter 5, Section 5.2.

isolation" assumption in the models while the geometrical models will be useful for proving bounds that are a consequence of the models.

### 2.6.1 The EL/LM Model as a BBN

The conclusions of the EL and LM models are consequences of the conditional independence relations between the random events in these models. Therefore, we may describe the EL and LM models via ***Bayesian belief networks*** (BBNs), as in Fig. 2.5. BBNs are useful because:

- they can be used as visual representations of conditional independence relations (and therefore, joint distributions) between Random events;
- they can be analyzed, by using transformations on the topology of the BBN, to elicit the consequences of conditional independence relations between sets of Random events. These transformations, justified by the property of *d–separation* which we discuss shortly, represent marginalisations of joint probability distributions. The result is that quite complex, and seemingly different, graph topologies are simplified into the same canonical form. We demonstrate these canonical topologies in Chapter 3.

For a discussion on Graphical Models (of which our BBNs are a particular case), marginalisation, d–separation and conditional independence models please see [35, 36, 37].

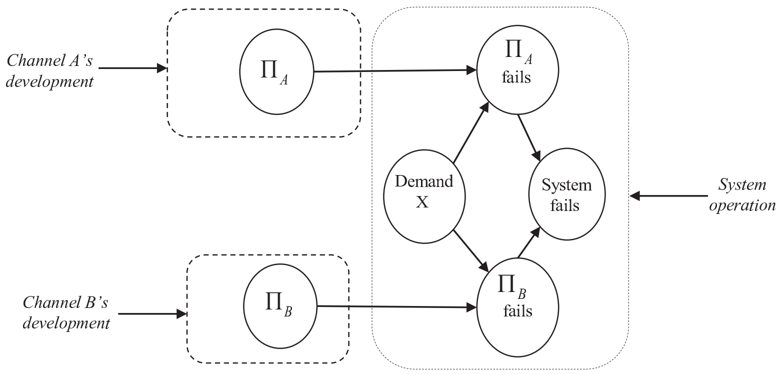


**Figure 2.5:** A Bayesian network for the EL and LM models. The two versions are chosen independently (ISA) which is represented by the absence of common parent nodes for $\Pi_A$ and $\Pi_B$. Also, they fail independently *conditionally* on the randomly chosen demand $X$, as shown by the presence of the single common ancestor $X$ for the two nodes "$\Pi_A$ fails", "$\Pi_B$ fails".

For the BBN in Fig. 2.5 each node is a random variable. The nodes without common parents are mutually independent random variables. The nodes with common parents are conditionally independent, *conditional* on the values of all of the *common*, parent nodes. For each node a conditional probability distribution is defined: the distribution of the random variable associated with that node, conditional on all the values of the random variables associated with the parents of the node. More generally, 2 random variables in a BBN, say $X$ and $Y$, are conditionally independent, conditional on some set of random variables $Z$, if every path (consecutive sequence of edges and nodes) between $X$ and $Y$ is *blocked* by $Z$. In this case we say that $Z$ *d–separates* $X$ and $Y$. A path, $p$, is said to be blocked by $Z$ if and only if

- $p$ contains a chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ such that $m \in Z$, or;
- $p$ contains an inverted fork (or collider) $i \to m \leftarrow j$ such that $m \notin Z$ and no descendant of $m$ is in $Z$.

If $Z$ does not *d–separate* $X$ and $Y$ and all the nodes/elements of $Z$ are ancestors of $Y$, say, then it is possible to marginalise (calculate an average of) the joint distribution of $Y$ and $Z$ to obtain the distribution of $Y$. This is an important transformation. It allows many, seemingly different, BBN topologies of development processes to be transformed into a form that focuses on sources of dependence in the development and operation of fault-tolerant software. In Fig. 2.5 the node "Demand $X$" *d–separates* "$\Pi_A$ fails" and "$\Pi_B$ fails", thus implying that "$\Pi_A$ fails" and "$\Pi_B$ fails" are conditionally independent, conditional on "Demand $X$". This is the same conditional failure independence that is demonstrated in Eq. (2.11) as a consequence of the "perfect isolation" assumption. In terms of difficulty functions this is the result that for each demand, $x$, we have

$$P\left(\Pi_A,\, \Pi_B \text{ both fail on } x\right) = \theta_{AB}(x) = \theta_A(x)\theta_B(x) = P\left(\Pi_A \text{ fails on } x\right) P\left(\Pi_B \text{ fails on } x\right).$$

In summary, the conditional independence relationships in the EL/LM models may be represented by the BBN in Fig. 2.5. Furthermore, conditional independence relationships between random variables can be determined by using the property of *d–separation* between the nodes of some suitable BBN topology. This will be useful in Chapter 3, as we demonstrate in the following section.

### 2.6.2 Conditional Independence and BBN Transformations

In Chapter 3, by using conditional independence to model dependent software development, we present a variety of probabilistic models. In many respects these models represent different possible configurations of the software development process. For instance, the models may describe the actions of different teams using different tools to develop systems that perform different actions. These differences, in turn, may result in BBNs with different topologies. Nevertheless, despite these differences, there is one important way in which these models may be "similar". It turns out that *the BBNs may satisfy analogous conditional independence relationships between analogous sets of random variables.* Consequently, by averaging over random variables that do not have analogues in this sense, we transform seemingly different BBN topologies into a topology that captures all of the conditional independence relationships "shared" by all of the BBNs. That is, different BBNs may possess marginal distributions which have "similar" conditional independence relationships, despite being obtained from models of seemingly different development scenarios. We may demonstrate this similarity by averaging joint distributions over unimportant random variables. This gives us a recipe for proving some results about how best to organise the development of a system.

If a result is a consequence of certain conditional independence relationships, believed to hold in a particular representative BBN, then we can argue that this result also holds for any scenario with an associated BBN that, itself, contains a marginal distribution with "the same" conditional independence relationships as the representative BBN. Further still, as an alternative to proving this "similarity" between BBNs by averaging over unimportant random variables, the previously defined property of *d–separation* can be used to prove that "similar" conditional independence relationships hold across the BBNs.

We now show, by way of example, what is meant by a collection of BBNs having "similar" or "the same" conditional independence relationships. Suppose an extension of the LM model results from modeling a development process, and in this extended model the conditional independence relationships depicted in Fig. 2.6 hold. If we are also presented with the BBN depicted in Fig. 2.7 we may convince ourselves that each conditional independence relationship in Fig. 2.7 also holds in Fig. 2.6 by averaging over random variables. For

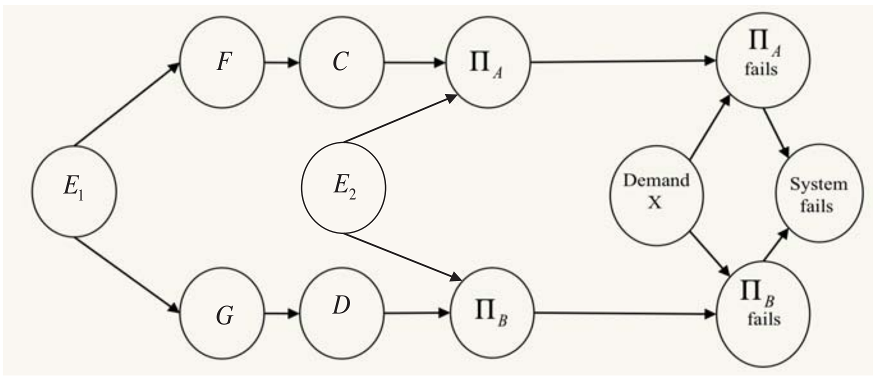


**Figure 2.6:** The model depicted is representative of BBN topologies resulting from extending the LM model, extensions which we discuss in more detail in Chapter 3. Here, note the similarity between this model and the LM model (depicted in Fig. 2.5) in terms of the network of random variables $\Pi_A$, $\Pi_B$, Demand $X$, $\Pi_A$ fails, $\Pi_B$ fails and System fails.

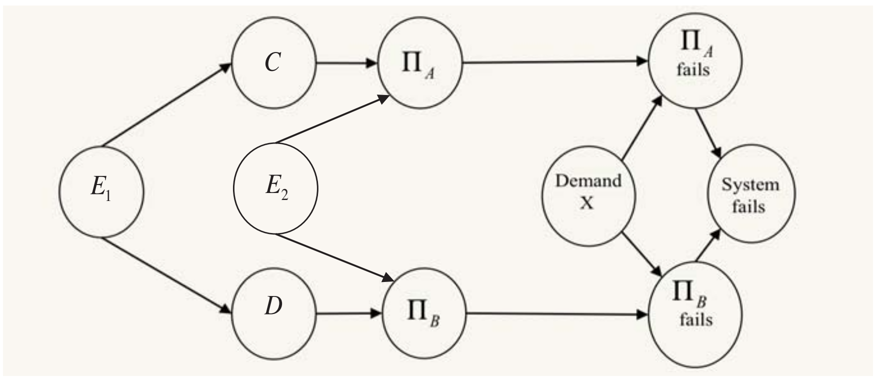


**Figure 2.7:** This BBN topology results from taking the expectation – with respect to the random variables $F$ and $G$ – of the joint distribution depicted in Fig. 2.6. Hence, this is a marginal distribution of the joint distribution in Fig. 2.6. All of the conditional independence relationships indicated in this BBN also hold for Fig. 2.6.

instance, to show that in either BBN the random variables $C$ and $D$ are conditionally independent – conditional on the pair of random variables $E_1$, and $E_2$ – we can use mathematical expectation to prove the following factorisation:

$$P(C, D|E_1, E_2) = P(C|E_1, E_2)\, P(D|E_1, E_2)\,. \tag{2.20}$$

*Proof.* Let's begin with the BBN in Fig. 2.6. Assume that all the random variables depicted are discrete and finite. Also, for the sake of simplifying notation, we define the random vector $Z$ as follows.

$$Z := (\Pi_A,\, \Pi_B,\, \text{Demand } x,\, \Pi_A \text{ fails},\, \Pi_B \text{ fails},\, \text{System fails})\,.$$

Then, upon taking an expectation of the joint probability distribution indicated by Fig. 2.6, using both "the law of total probability" and conditional independence relationships suggested by Fig. 2.6, we see that:

$$\begin{aligned}
P(C,\, D\,|E_1,\, E_2) &= \sum_{z,\, f,\, g} P(z,\, f,\, g,\, C,\, D\,|E_1,\, E_2)\\
&= \sum_{z,\, f,\, g} P(z\,|\, f,\, g,\, C,\, D,\, E_1,\, E_2)\, P(f,\, g,\, C,\, D\,|\, E_1,\, E_2)\\
&= \sum_{f,\, g} \sum_{z} P(z\,|\, f,\, g,\, C,\, D,\, E_1,\, E_2)\, P(f,\, g,\, C,\, D\,|\, E_1,\, E_2)\\
&= \sum_{f,\, g} P(f,\, g,\, C,\, D,\,|\, E_1,\, E_2)\\
&= \sum_{f,\, g} P(C,\, D,\,|\, f,\, g,\, E_1,\, E_2)\, P(f,\, g,\,|\, E_1,\, E_2)\\
&= \sum_{f,\, g} P(C\,|\, f,\, g,\, E_1,\, E_2)\, P(D\,|\, f,\, g,\, E_1,\, E_2)\, P(f,\, g,\,|\, E_1,\, E_2)\\
&= \sum_{f,\, g} P(C\,|\, f,\, g,\, E_1,\, E_2)\, P(D\,|\, f,\, g,\, E_1,\, E_2)\, P(f\,|\, E_1,\, E_2)\, P(g\,|\, E_1,\, E_2)\\
&= \sum_{f,\, g} P(C\,|\, f,\, E_1,\, E_2)\, P(D\,|\, g,\, E_1,\, E_2)\, P(f\,|\, E_1,\, E_2)\, P(g\,|\, E_1,\, E_2)\\
&= \sum_{f,\, g} P(C,\, f\,|\, E_1,\, E_2)\, P(D,\, g\,|\, E_1,\, E_2)\\
&= \sum_{f} P(C,\, f\,|\, E_1,\, E_2) \sum_{g} P(D,\, g\,|\, E_1,\, E_2)\\
&= P(C\,|\, E_1,\, E_2)\, P(D\,|\, E_1,\, E_2)\,,
\end{aligned}$$

where we have used $\sum\limits_{z} P(z\,|\, f,\, g,\, C,\, D,\, E_1,\, E_2) = 1$. Thus, Eq. (2.20) holds for Fig. 2.6. "Similarly", an analogous development based on the BBN in Fig. 2.7 gives:

$$P(C,\, D\,|E_1,\, E_2) = \sum_{z} P(z,\, C,\, D\,|E_1,\, E_2)$$

$$
\begin{aligned}
&= \sum_{z} P\left(z \,|\, C,\, D,\, E_1,\, E_2\right) P\left(C,\, D \,|\, E_1,\, E_2\right) \\
&= P\left(C,\, D, \,|\, E_1,\, E_2\right) \\
&= P\left(C, \,|\, E_1,\, E_2\right) P\left(D, \,|\, E_1,\, E_2\right),
\end{aligned}
$$

so that Eq. (2.20) holds again, but this time for Fig. 2.7. ■

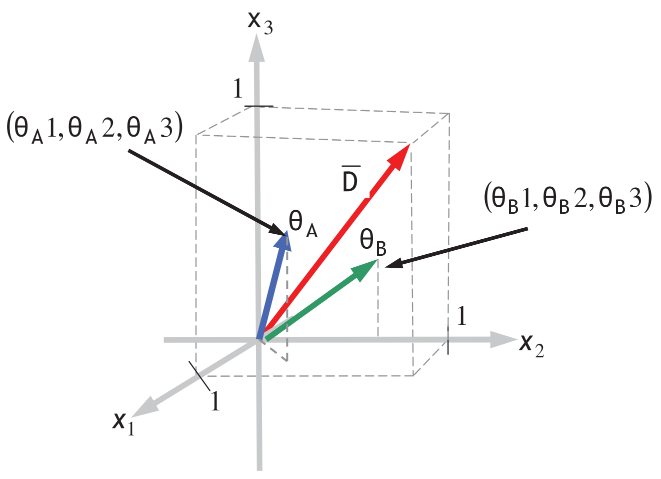
Alternatively, however, we could have proved that both BBNs satisfy Eq. (2.20) by using *d–separation*: that is, by noting that both BBNs have the property that each path between the pair of random variables $C$ and $D$ is *blocked* by the pair of random variables $E_1$ and $E_2$. We demonstrate this for the BBN in Fig. 2.6 alone, since an almost identical treatment shows how the other BBN satisfies the property. In Fig. 2.6 there are 4 paths between the random variables $C$ and $D$, with each path blocked by the set consisting of the random variables $E_1$ and $E_2$ as follows:

1. The path, $C \leftarrow F \leftarrow E_1 \rightarrow G \rightarrow D$, is blocked, since it includes the chain $F \leftarrow E_1 \rightarrow G$ containing $E_1$
2. The path, $C \rightarrow \Pi_A \leftarrow E_2 \rightarrow \Pi_B \leftarrow D$, is blocked, since it includes the chain $\Pi_A \leftarrow E_2 \rightarrow \Pi_B$ (containing $E_2$)
3. The path, $C \rightarrow \Pi_A \rightarrow \Pi_A$ fails $\leftarrow$ Demand $X \rightarrow \Pi_B$ fails $\leftarrow \Pi_B \leftarrow D$, is blocked since it contains the chain $\Pi_A \rightarrow \Pi_A$ fails $\leftarrow$ Demand $X$ (where neither $E_1$ nor $E_2$ is a descendant of "$\Pi_A$ fails")
4. The path, $C \rightarrow \Pi_A \rightarrow \Pi_A$ fails $\rightarrow$ System failure $\leftarrow \Pi_B$ fails $\leftarrow \Pi_B \leftarrow D$, is blocked since it contains the chain $\Pi_A$ fails $\rightarrow$ System failure $\leftarrow \Pi_B$ fails (where neither $E_1$ nor $E_2$ is a descendant of "System fails").

In summary, this pair of proofs allow us to assert the same result: *taking the expectation of the joint distribution in Fig. 2.6 – with respect to the random variables F and G – gives the marginal distribution in Fig.2.7, and this is a transformation of a BBN that preserves the conditional independence relationship stated as Eq.* (2.20). In fact, all of the conditional independence relationships in Fig.2.7 are preserved from Fig.2.6. In this way, we use marginalisations to justify why different BBNs share a common canonical form, in Chapter 3.

### 2.6.3 Difficulty Functions as Vectors

As a precursor of a more complete and general treatment given in Chapter (4) we shall briefly discuss the depiction of difficulty functions as vectors in some *n-dimensional vector space*. Consider an LM model, $\big\{\big(\Omega_A \times \Omega_B, \Sigma_A \times \Sigma_B, \mathrm{P}_A \times \mathrm{P}_B\big), (\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X)\big\}$, where $\mathcal{X}$ and $\Omega_A \times \Omega_B$ are both finite. Recall that the difficulty functions for the developments of the channels are bounded, real–valued functions of the demands. So, for a finite number of demands we can specify each of these difficulty functions as a vector whose components are the values of the difficulty function on each of the demands. For instance, suppose there are $n$ demands. Then, given some ordering/numbering of the demands the difficulty function, $\theta_A(x)$, can be written as $\big(\theta_A 1, \ldots, \theta_A n\big)$ where $\theta_A i$ is the difficulty associated with the $i$th demand. In lower dimensions such n–dimensional vectors or n–tuples can be depicted, as in Fig. 2.8. Here, difficulty functions defined on a demand space with 3 demands are depicted



$x_3$
1
$(\theta_A 1, \theta_A 2, \theta_A 3)$
$\bar{D}$
$(\theta_B 1, \theta_B 2, \theta_B 3)$
$\theta_A$
$\theta_B$
1
$x_2$
1
$x_1$

**Figure 2.8:** The difficulty functions, $\theta_A = \big(\theta_A 1, \theta_A 2, \theta_A 3\big)$ and $\theta_B = \big(\theta_B 1, \theta_B 2, \theta_B 3\big)$, are drawn as 3–dimensional vectors. The demand space is defined as $\mathcal{X} := \{x_1, x_2, x_3\}$. The arrows representing the difficulty functions must lie within the closed region defined by the *unit hypercube* which is the hyperrectangle that has the vector $\bar{D} = \overline{(1, 1, 1)}$ as its diagonal. This is because each component of the difficulty vector is a probability and, consequently, must lie in the interval $[0, 1]$.

as lying on, or within, the *unit cube*. Such a 3–dimensional diagram may also be used to depict vectors in higher dimensions if the diagram is viewed as an *embedding diagram*: that is, the axes represent multi–dimensional *sub–spaces*[25] that are *orthogonal*[26] to each other. Admittedly, it is difficult to visualise vectors in more than three dimensions. However, diagrams of this kind are useful in illustrating concepts, relationships, properties and results that are true for any finite dimensional vector space and, therefore, true for difficulties defined on any finite demand space. Also, simple diagrams like this can be suggestive and aid intuition in proofs.

## 2.7 Summary

Historically, the EL/LM models have been used to clarify the implications of achieving high–reliability systems by employing diversity. At the heart of these models is the assumption that "perfectly isolated" development teams each develop a software version. This assumption eliminates a number of possible dependencies, that would have existed otherwise, between the teams. For instance, under the LM model, teams do not collaborate in a number of activities such as coding, debugging and testing. Consequently, activities that could be possible sources of dependence are not explicitly modelled in the original formulation of the EL/LM models. Instead, each software development process is modelled as the random selection of a version from a population of possible versions that could be written. However, it is precisely these kind of activities that should be modelled when considering possible sources of dependence

[25] A vector space contained within a vector space.

[26] A generalisation of the notion of *perpendicularity*. A definition is given in Chapter 4.

between the development of the versions. Therefore, in order to model situations where the teams are not isolated from each other, it is necessary to develop the EL/LM models in more detail than they have been developed previously. To this end, this chapter has presented a detailed re–development of the LM model. This allowed us to re–derive previously known results, point out new results, and offer new viewpoints. Some re–derived results include:

- defining the LM model as a combination of two probabilistic models: a model of the occurrence of demands, and a model of the development of a pair of software versions where each version is developed by an isolated development team. The activities in the development process, which have not been explicitly modelled in the original formulation of the LM model, are modelled in detail (see Section 2.5);
- defining the notions of difficulty, difficulty function and variation of difficulty. For each demand this function gives the probability that a randomly chosen program fails on the demand. This allows for formalizing notions of failure diversity between multiple software versions (see Section 2.3);
- demonstrating that independently developed versions may not fail independently (see Section 2.5, Eq. (2.13) and Eq. (2.15));
- showing that if a system assessor were indifferent between expected system pfds resulting from not forcing diversity, then forcing diversity results in expected system pfd that is no worse, and may be better (see Section 2.5 and Eq. (2.18)).

Some new results and viewpoints include:

- stating observability criterion 2.4.1, which makes more explicit the consequence of "perfectly isolated development teams" assumption. The criterion stipulates that an observer embedded in the development process of a perfectly isolated development team should not be able to confirm, or refute, the existence of any other development process by observing activity outcomes in the process she is embedded in;
- demonstrating that forcing diversity does not, in general, guarantee better expected reliability than if diversity were allowed to occur naturally (see Section 2.5). In Chapter 5, Theorem 5.2.1, we give a necessary and sufficient condition on the sizes of a pair of difficulty functions for forced diversity to not worsen expected reliability;
- showing that under the LM model there are at least two senses in which fault–tolerance is guaranteed to result in reliability that is not worse than the reliability of a single version system (see Section 2.5). A fault–tolerant configuration of any pair of versions is guaranteed to have a pfd that is not larger than the pfds of each of the versions. Also, the expected pfd of a 1–out–of–2 system, built by a pair of isolated development teams, is no worse than the related expected pfds of single–version systems built in isolation. In more general settings (see Chapters 3 and 6) than those of the EL/LM models we show the second result is not necessarily true: *in general, using fault–tolerance does not guarantee the resulting expected system pfd can be no worse than otherwise*;
- developing visual representations of the models that facilitate the analyses of the models (Section 2.6). These are a Graph–based representation that depicts the consequences

of conditional independence relationships in the models, and a Geometric representation that facilitates maximizing and minimizing reliability measures such as expected probability of failure on demand.

In the following chapter we shall generalise the LM model by considering a model that uses *conditional independence* to relax the assumption of perfect isolation between development teams.

# Chapter 3

# Generalised Models of Coincident Failure

The LM model does not appropriately model all forms of the development of a 1–out–of–N system. The Independent Sampling Assumption (ISA) of the EL and LM models is a plausible modelling consequence of the ideal of complete separation between the developments of the two versions: with "perfect" separation there is no way that the development of one version may influence the development of another one. As a consequence of the ISA there is *conditional independence*, given a specific demand $x$, between the failures of the two versions. That is, for any given demand the probability of that demand being a failure point for one version does not depend on whether it is a failure point for the other version. This in turn implies $\theta_{AB}(x) = \theta_A(x)\theta_B(x)$: the probability of building a two-version system that fails on $x$ is the product of the probabilities of each version development team building a version that fails on $x$. Thus, the ISA allows one to derive Eq.'s (2.13) and (2.15). There are, however, several reasons for studying scenarios in which the ISA is false:

- complete separation may be impossible for various practical reasons. So, we ought to study the effects of the inevitable, though possibly small, departures from it;
- communication between the teams may in some cases be desirable either because:
  - it causes positive correlation between failures on each demand but improves the reliabilities of the individual versions so much that the net effect is improved system dependability, OR;
  - perhaps it can be engineered to cause negative correlation in such a way as to improve system dependability.
- even without communication between the teams the management may wish to improve the expected *pfd* of the diverse system by enforcing methods that plausibly violate conditional independence. Examples are:
  - the choice of algorithms to be implemented, allowing the teams to choose freely but with the constraint that they use different algorithms for the same subset of

  the demand space. The hope is to produce negative correlation between the team's mistakes on the same demand;
  - regarding quality assurance measures, mandating some common procedure which may cause *positive* correlation. For instance, testing the two versions on the same test cases may be a cost-effective way of improving the reliability of both versions created. This, in turn, improves the reliability of the fault-tolerant system.

- more subtly, we will argue that the ISA is equivalent to assuming that the version sampling distributions incorporate complete knowledge of dependencies, if any, during system development. But to answer some important questions, we may need to model scenarios in which the precise form of some dependencies are not known with certainty, i.e. they are random variables. This turns out to violate the ISA. Examples of such uncertain dependencies in development could be unforeseen deviations of the time and funds available for specific tasks from the pre–set project calendar and budget.

So, in this chapter we explore generalisations of the LM model; presenting ways in which dependent development processes may be modelled. Some models keep useful properties of the LM model, such as being able to make meaningful comparisons between the development of a single–version system and the development of a multi–version system. Other models are so general that it becomes difficult to make such comparisons. Importantly, we shall see that by using an interplay of *activities that are common to both channels'* and *team isolation* we obtain probabilistic models with dependence "built out of" atomic LM-*like* models. LM-*like* in the sense that these models exhibit the ISA even if the channel development processes are not perfectly isolated[1]. A consequence of being able to construct models out of LM-*like* models in this way is that the LM model can be shown to have more general applicability than just the case of complete isolation between the development teams. Also, the sense in which the EL model is optimistic – that is, the claim that the independent development of the channels optimizes system reliability – can be demonstrated. Finally, preference criteria concerning how best to organise a development process given rather general conditions can be stated and proven.

## 3.1 Modelling Dependence between Version Developments

A very general model of system development is given by

$$\left\{ \left(\Omega_{\mathscr{D}}, \Sigma_{\mathscr{D}}, \mathrm{P}_{\mathscr{D}}\right), \left(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_{X}\right) \right\},$$

[1]In a given system development process there may be activities, common to both processes, that have already been realized so that the outcomes of these activities are already and remain fixed for the duration of the development process. These outcomes result in certain information being shared between the processes, and is a manifestation of the fact that the ISA is a necessary consequence of perfect team isolation, but not a sufficient condition for it. That is, there exist situations where the ISA does not hold, and yet creation of the pair of versions is modelled as the conditionally independently selecting a pair of versions from a space of versions $\mathcal{P}$.

where $\Omega_{\mathscr{D}}$ is a sample space of all of the modelled outcomes of activities in the development process, $\Sigma_{\mathscr{D}}$ a set of related events, $\mathrm{P}_{\mathscr{D}}$ a probability measure over $\Sigma_{\mathscr{D}}$, and $\left(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X\right)$ is an independent, probabilistic model of demand occurrence in operation. Here, the sample space $\Omega_{\mathscr{D}}$ is not necessarily the cartesian product of the sample spaces for each development process activity's probabilistic model. Additionally, the probability measure $\mathrm{P}_{\mathscr{D}}$ is not necessarily the product measure of the measures related to the probabilistic models of the channels' development processes. While this model is general enough to model development processes that have various forms of dependence its generality is also its shortcoming; it is too general to be of practical use. In the LM model because the system model – a *product probability space* – is completely determined by the models of each channel's *isolated* development, comparisons can be made between expected version *pfd*s (where the relevant probabilistic experiment here is one in which a single–version system is being developed in isolation) and the expected system *pfd* (where the relevant probabilistic experiment here is one in which a multi–version system is being developed by isolated teams). This is useful in discussing how much of a benefit, if any, fault–tolerance brings over single version systems. In addition, equations related to the LM model, such as Eq.'s (2.13), (2.15), have covariance terms that manifestly capture the notion of dependent failure between the channels. These are intuitive and appealing constructs that aid the understanding of diversity and its effects.

As an alternative to the general model it would be desirable to have a model that relaxes the "complete isolation" assumption while retaining desirable properties of the LM model. One possibility is a model that uses an interplay of "dependence inducing" activities with isolated activities in the joint development process. The idea is to have some activities with outcomes that are used by each of the channel development processes. However, apart from these activities, no other activities are common to all of the channel development processes. Practical examples of such a protocol include: a manager of a joint development process who intermittently issues specification clarification updates to all of the, otherwise isolated, development teams; or the versions produced by isolated development teams being subjected to tests with the same test suite, the latter example being modelled in [38]. The result of this is a process that is adequately described by a class of models that model dependence via conditional independence. Let us construct an example member of this class of models. Consider the example where a manager may issue a single specification clarification to all of the development teams. The precise nature of the update is unknown before hand as this will depend on the nature of errors/ambiguities found in the initial specification, if any. Consequently, the act of issuing an update can be modelled probabilistically as a probability distribution over all of the possible updates that may be issued. Let $\left(\Omega_E, \Sigma_E, \mathrm{P}_E\left(\cdot\right)\right)$ be the model of this activity where, following usual convention, $\Omega_E$ is the set of all possible updates that can be issued, $\Sigma_E$ the set of all update classes of interest and $\mathrm{P}_E\left(\cdot\right)$ a probability measure defined over $\Sigma_E$. Apart from these updates the teams are kept isolated from each other. This means that after an update is issued the teams continue to develop their respective versions oblivious to what the decisions of the other teams are. Therefore, given that an update is issued, the joint development process is adequately modelled as the *product probability space* of the marginal probabilistic models describing the development of each channel, just like in the LM model. Suppose an update $e \in \Omega_E$ is issued to both teams in the development of a 1–out–of–2 system. We have the pair of conditional probabilistic models $\left(\Omega_1, \Sigma_1, \mathrm{P}_1\left(\cdot|e\right)\right)$ and $\left(\Omega_2, \Sigma_2, \mathrm{P}_2\left(\cdot|e\right)\right)$, each conditional on the update $e$ and each modelling the development of a unique channel. $\Omega_1$ is the set of the outcomes of all of the modelled actvities in the development of channel 1 except specification clarifications issued, $\Sigma_1$ is the set of all events of interest and $\mathrm{P}_1\left(\cdot|e\right)$ is a probability measure that is conditional on the update $e$. So, for $s_1 \in \Omega_1$ the probability that the activities in the development of channel 1 have the outcome $s_1$, given that clarification $e$ has been issued to the teams, is $\mathrm{P}_1\left(s_1|e\right)$. Similarly, for the respective terms in channel 2's model. Given any particular "$e$", each of these models is conceptually identical to the models of channel development in the LM model (see Section 2.5). So, for each clarification $e$ we may define the LM model $\left\{\left(\Omega_1 \times \Omega_2, \Sigma_1 \times \Sigma_2, \mathrm{P}_1\left(\cdot|e\right) \times \mathrm{P}_2\left(\cdot|e\right)\right), \left(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X\right)\right\}$, where the probabilistic models for channels 1 and 2's development are conditionally independent, conditional on the specification updates. In fact, by doing this we are defining an entire family or class of LM models "indexed" by the possible specification clarifications. So, for a randomly issued update, this class of LM models and the probabilistic model of updates, $\left(\Omega_E, \Sigma_E, \mathrm{P}_E\left(\cdot\right)\right)$, together define an appropriate model of our practical scenario. Models based primarily on this kind of formulation can be said to model dependence via conditional independence. This is captured by the following postulate .

**Postulate 3.1.1.** *Via an interplay of non isolated and isolated development we can define a probabilistic model which uses conditional independence to model dependent software development between the teams.*

Furthermore, such an interplay seems to be the only reasonable justification for such a model to be valid in practice. This is explained in more detail later in this chapter (see Section 3.2.1). The rest of the chapter is concerned with analysing the sort of model illustrated above. The example of this class of model given above – that is, the model of a joint development process consisting of otherwise isolated channel developments that are dependent on a possible clarification being issued – is one of the simplest examples of dependent development processes; more exotic cases are certainly possible. The more exotic cases result in probabilistic models that appear quite complex. Consequently, in order to ease the analysis of such models *Graphical models/BBN*s, introduced in Section 2.6.1, will be used to describe and analyse the probabilistic models.

## 3.2 Modelling Dependent Software Development Processes

The possible sources of dependence between the teams in a development project, like many other activities/circumstances in this process, can be modelled as random variables. We shall refer to these dependencies as [random] *influences*. That is, they are activities/environmental circumstances which affect the developments of both system channels and whose outcomes may not be known beforehand. The simpler interpretation of this "randomness" is as a source of "uncertainty in the world": we are trying to predict the effects of a process that has yet to happen and is affected by random factors. However, it can just as naturally represent "uncertainty in knowledge": the development has taken place, but what we know about it is limited; we still do not know the values of all these variables.

At this abstract level it does not matter whether an influence represents an event external to the software development process (e.g. a change in the requirements on which both version developments depend, flu epidemic affecting the health of team members, adverse weather conditions that effect the availability of necessary resources) or generated internally (e.g. an activity such as the selection of a single test suite to be used in testing all of the system channels), or even interactions between the teams (e.g. the specific information exchanged between the teams, represented as a random variable or set of random variables). The point is any of these sources of uncertainty affect the way in which the software is developed. Consequently, any version sampling distribution that models the development of the final version deployed by a development team is dependent on these influences. If we accept that some of these influences may be controllable (e.g. choosing team members or choosing programming language to develop in) and some may not be (e.g. adverse weather conditions) then this notion of influence generalizes the notion of methodology given in Chapter 2 (see Section 2.1). *A* ***methodology*** *is a collection of potentially controllable influences.* For example, the development process for a channel 'A' may contain influences (which are actually activities) such as "specification definition", "program design", and "implementation". Additionally, there could be influences such as "mutual code inspection" (where each team is required to inspect the other team's code) that affect both teams, or adverse weather conditions that affect each team's availability. Such influences are common to both team's development processes. To a large extent we will be interested in the effects of *common* influences: those that affect the developments of both channels. We will show that such common influences may indeed increase correlation between version failures, *or* they may reduce it.

Similar to Fig. 2.5 on page 49, which we reproduce in Fig. 3.1 for the readers convenience, we can describe scenarios involving influences via Bayesian networks (or "Bayesian belief networks", BBNs), such as in Fig. 3.2. Such Bayesian networks depict system development processes in which there may be multiple influences common to the two version development processes, as well as some separate influences. We note that nodes that are ancestors of only the $\Pi_A$ or only the $\Pi_B$ nodes do not represent *common* influences. Also, note that the BBNs representing the EL and LM models (see Fig. 3.1) appear to contain only the part of the BBNs in either Fig. 3.2 or Fig. 3.3 to the right of the final versions $\Pi_A$ and $\Pi_B$. This is a consequence of two facts:

1. the channels' development processes being perfectly isolated implies that the influences in a given process may be averaged over, independently of influences in the other process;
2. any common influences that exist between the channels' development processes have their values instantiated (e.g. the test suite to be used to test both versions has already been determined) and, consequently, do not contribute as common sources of uncertainty on which each channel's development process is dependent. Since these are not

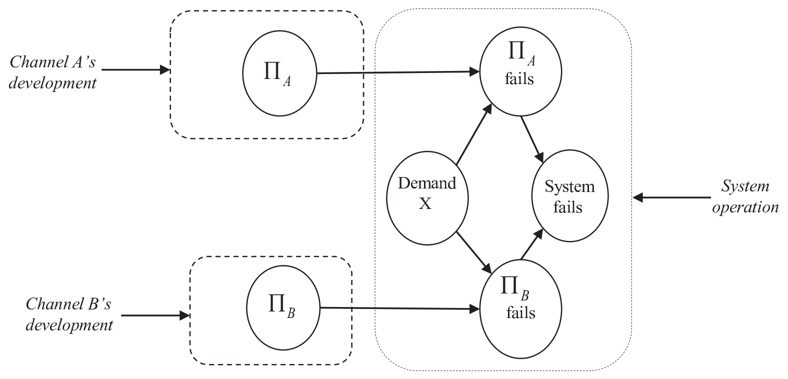


**Figure 3.1:** This Bayesian network for the EL and LM models, reproduced here for the readers convenience, was first introduced on page 49. The two versions are chosen independently (ISA) which is represented by the absence of common parent nodes for $\Pi_A$ and $\Pi_B$. Also, they fail independently *conditionally* on the randomly chosen demand $X$, as shown by the presence of the single common ancestor $X$ for the two nodes "$\Pi_A$ fails", "$\Pi_B$ fails".

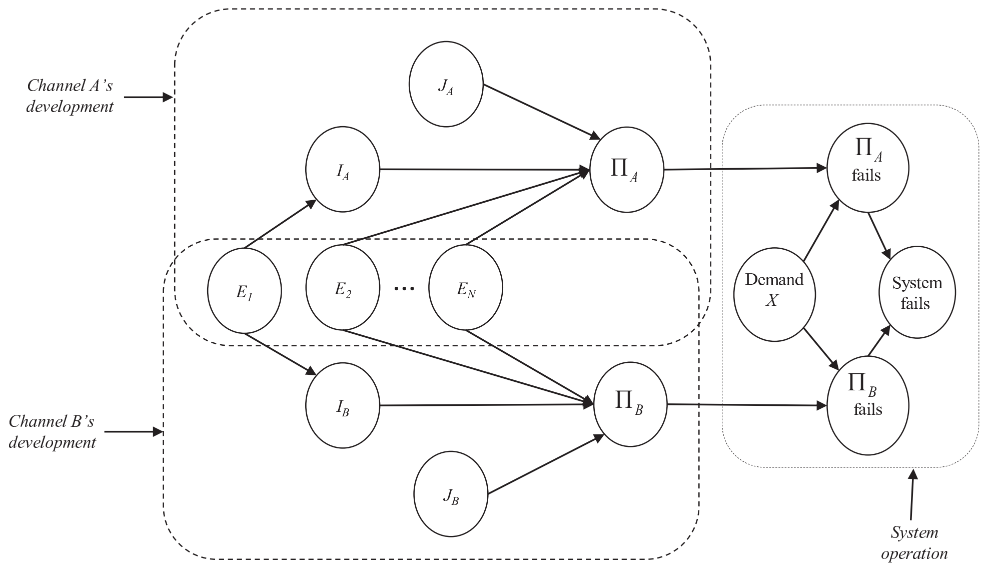


**Figure 3.2:** This graph is a Bayesian network depicting a (two-version) system development process affected by multiple influences, some of them common to the two versions. The nodes to the left of $\Pi_A$ and $\Pi_B$ might represent, for instance, specific design artefacts, test techniques and test cases selected, and influences on these various aspects of development like communication between the teams and the project management. The influences may interact in complex ways, e.g. mistakes in a specification document may affect choices of test cases and both affect which version is delivered. Some influences affect only one process and not the other, such as nodes $J_A$ or $I_B$. Other influences affect both development processes, such as nodes $E_1, \ldots, E_N$. We have added the large, rounded rectangles to identify the three main subsets of the BBN corresponding to the two processes of developing the two versions and of operating the system.

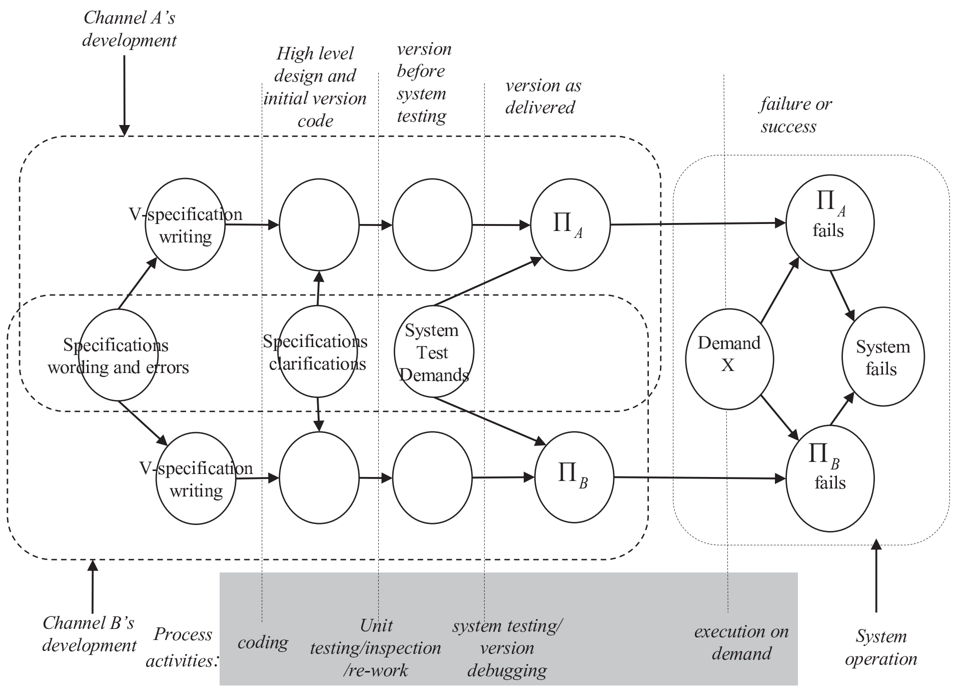


**Figure 3.3:** Here we have represented some possible examples of common influences between the developments of two versions. These influences affect various stages of the development processes and invalidate the Independent Sampling Assumption. The nodes in the top and bottom row represent artefacts at successive stages of production of the two versions. To avoid cluttering the diagram, the names of the artefacts are listed above it instead of writing the node's name inside the circle. Under the diagrams we have named the activities that transform an artefact into the next one. Each one is subject to some degree of randomness in its results, justifying the representation of each artefact as a random variable, whose distribution is determined by the exact values of its parent nodes: the artefact upstream/"to the left" of it, and in most stages a common influence as well, represented by a node in the middle row.

sources of randomness they are not depicted in the BBN; this is a common feature of all of the BBNs. ***In any given BBN any influences with predetermined values are not depicted and the distributions of the random variables that are depicted are conditional, conditioned on the known values of these influences.***

Like the EL and LM models there are conditions under which these BBNs are seen to exhibit conditionally independent version sampling distributions. We characterise this in the following theorem.

**Theorem 3.2.1.** *If all common influences are given specific values rather than chosen randomly OR if the version development processes have no common influences (they are isolated from each other), then the two final versions are chosen independently.*

Going to an extreme viewpoint, we could assume as known every detail of the two developments, down to the full behaviour of the two specific program versions created, say $\pi_A$ and $\pi_B$, on every point of the input space. Then, the difficulty functions $\theta_A(x)$ and $\theta_B(x)$ collapse to the "score functions" $\omega(\pi_A, x)$ and $\omega(\pi_B, x)$, and independence conditional on each demand is guaranteed by the fact that the score functions can only take the values 0 or 1. Therefore, as a consequence of Theorem 3.2.1 the ISA applies as well as its consequence: "with identical version development processes, your expectation of the probability of joint failures should be greater than the product of your expected *pfd*s for the two versions."

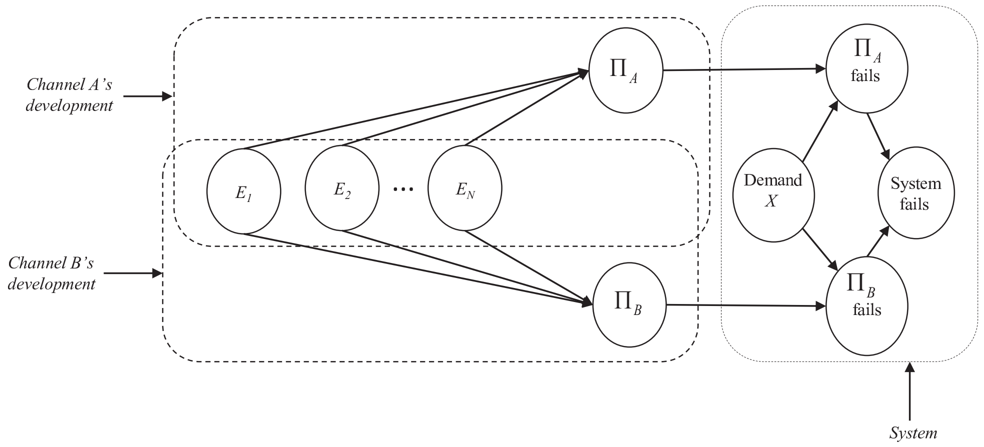


**Figure 3.4:** The Bayesian network in Fig. 3.2, and all those we have examined that represent interesting scenarios of development, can be transformed into a shape like this one in which the only influences are direct "parent" nodes of $\Pi_A$ and $\Pi_B$ and are common to both version development processes. The intermediate nodes through which the influences affect $\Pi_A$ and $\Pi_B$ have been removed by marginalising the distributions in the BBN of Fig. 3.2 with respect to the non-common influences. This is justified by applying *d–separation* (see page 49 for a definition) to Fig. 3.2 to first determine conditionally independent sets and, subsequently, marginalising where appropriate (see text, and pages 49,50). Thus, all these scenarios can be described by the same form of equation that applies to this figure. We study this form of equation in this chapter.

We can transform Fig. 3.2 or Fig. 3.3 into a form "without" activities in the development process that do not represent common influences, such as Fig. 3.4. As already stated in sections 2.6, 2.6.1, and 2.6.2 (see pages 49 and 50) this transformation is accomplished by averaging (marginalising) over those nodes (random variables) in Fig. 3.2 that do not appear in Fig. 3.4. This transformation preserves the meaning of the original BBN – Fig. 3.2 or Fig. 3.3 – in an important sense: it implies no conditional independence assumptions that were not already implied by the original BBN; and all the joint distributions between the nodes in Fig. 3.4 are unchanged from those between the homologous nodes in the original BBNs [35]. However, Fig. 3.4 omits the details of how the non-common influences affect the development process (e.g. which phase of development they affected).

So, scenarios with quite different common influences, even if these influences affect different phases of development (e.g., errors in specification *vs* choices of the same system test cases for both version development processes), can be reduced to a common mathematical form from the viewpoint of dependence relations. We can therefore formulate a set of theorems that depend only on the presence of common influences. All scenarios of dependence in system development that uses an interplay of isolation and interaction, some resulting in more complex BBNs than Fig. 3.2, can be transformed in this way. The result of this is the same theorems are applicable to seemingly different scenarios. In the following sections we discuss further justification for, and the implications of, these models.

### 3.2.1 Justifying Conditional Independence in Practical Situations

In this chapter we are interested in practical situations which may be modelled by BBNs that, via marginalisation (see Section 2.6.1), simplify to the canonical form Fig. 3.4. Therefore, in order to use the results of this Chapter in a given practical scenario, one must seek justification for why the conditional independence relationships in a representative BBN hold in the practical scenario. This is a non–trivial task. What characteristics of practical situations can be used to justify a given BBN as adequately capturing all of the conditional independence relationships required? It seems reasonable to give the following as an example of a scenario that can be argued to satisfy a BBN topology similar to Fig. 3.4.

Consider a practical scenario as shown in Fig. 3.3. This depicts a system development process in which periods of isolated development are interspaced/alternated with mutually independent periods of dependent development. Each channel's development process has two types of activities that are conducted by the respective teams: those with outcomes that affect the development of all of the channels and those with outcomes that affect only the respective development processes that the activities are part of. The effect of these two kinds of activities is that the teams make "independent" decisions about their respective development processes, given the outcomes of any dependence creating activities that have occurred. In order to use this argument, the pair of activities that need to be identified in practice are:

1. *Dependence Creating Activities*: These have outcomes that are shared amongst the channel development processes. These uncertain outcomes are random variables (indicated as nodes located in the middle of the BBN) such as "Specification wording and

errors" which would be an outcome of an activity of specification writing for the system, or "System test Demands" which would be the suite of demands that are to be used in system testing. In the present example scenario these activities have the property that their outcomes are relevant for all of the channels. Also, these activities are mutually independent so that the outcome of any common activity does not affect any other common activity. For instance, in the example of Fig. 3.3 the generation of a test suite does not require knowledge of any specification clarifications that may have been issued to the teams. This mutual independence is not necessary and certainly does not hold in general: for instance, there are cases where specification clarifications may inform the decision of which demands are representative, and therefore should form part of a test suite.

2. *Independent Activities*: Contrastingly, there are also activities whose outcomes are relevant only for the respective process in which the activity is found. Again, the uncertain outcomes of the activities are random variables, depicted as labels of nodes. So, for each development process there is a relevant activity of designing the software with the outcome "a High–level design is produced for the channel" (represented by the node labelled "High–level design"), or the activity of an initial period of coding with the outcome "some initial version code is produced" (represented by the node labelled "initial version code"). In this example such activities are conducted by the teams in isolation. That is, each team's actions while conducting these activities, and the outcomes thereof, do not influence any other development team, and the effect of events outside of the teams' development processes that could affect all of the teams (such as civil unrest, accidental fires or flu epidemics) are mitigated as much as possible (such as remote access to work files, contingency plans and redundancy in team member expertise).

Are there other types of practical scenarios that may be identified as satisfying conditional independence relationships similar to the canonical form? Such a practical scenario ***does not use the aforementioned interplay of isolation and dependence***, and yet is justifiably described by a non–trivial model that ***exhibits conditional probabilistic independence, conditional on the outcomes of common activities***. Put another way, the task is to construct an example scenario in which the teams are not isolated from each other, and yet observable characteristics of the development of the channels justify modeling this as the conditionally independent selection of a pair of programs, conditional on the outcomes of all common influences. However, constructing such an example can be problematic, as we show by way of example.

Let us try to construct a simple example based on a system development process consisting of a single common influence. First we will write out the necessary form of conditional independence, and then think about what observable characteristics of the development process (other than perfect team isolation) may justify such a form as holding in practice. For simplicity, let the set of possible programs that may be developed for the system channels be $\mathcal{P} := \{\pi_1, \pi_2\}$. Perhaps the space of possible versions is so small because the common influence excludes the possibility of all other potential pairs of versions (e.g. there are only two simple algorithms that the teams may choose to implement). In any case, this simplification is not a hindrance since we expect useful observable characteristics to be independent of the sizes, as well as the number, of common influences. Now, for any given value of the common influence (e.g. the value $e$), the three conditional probabilities $\{P(\pi_1,\, \pi_1|e)\,,\, P(\pi_2,\, \pi_2|e)\,,\, P(\pi_1,\, \pi_2|e)\}$ define the probability of producing a pair of versions. Then, if conditional independence holds, the marginal probabilities $P(\pi_1|e)\,,\, P(\pi_2|e)$ should be such that:

$$\begin{aligned} P(\pi_1,\, \pi_1|e) &= P(\pi_1|e)\, P(\pi_1|e) \\ P(\pi_2,\, \pi_2|e) &= P(\pi_2|e)\, P(\pi_2|e) \\ P(\pi_1,\, \pi_2|e) &= P(\pi_1|e)\, P(\pi_2|e) \end{aligned}$$

Viewing this as a system of three equations in the two unknown marginal probabilities, solutions are not always guaranteed. In fact, a solution exists if and only if the probabilities $\{P(\pi_1,\, \pi_1|e)\,,\, P(\pi_1,\, \pi_2|e)\,,\, P(\pi_2,\, \pi_2|e)\}$ are in geometric progression, a property that is not expected to hold generally in practice. Hence, we seek observable characteristics of the development process that justify the joint probabilities of program pairs being in geometric progression. It is not clear what such observable characteristics might be, other than isolated development processes. In general, there are other hurdles to overcome in justifying conditional independence in practice[2]. This problem is common to any engineering process that may be modeled using conditionally independent random variables. To summarize, we suggest the following postulate.

**Postulate 3.2.2.** *Extensions to the LM model which model dependence via conditional independence can be used to represent development processes that utilise an interplay of periods of team isolation with periods of dependence creating activity.*

[2]In general, one might attempt to construct an example scenario in which a characteristic of the development process justifies conditional independence. Consider a scenario with multiple common influences. For each possible set of outcomes for the common influences, say $e_1, \ldots, e_n$, an observer looking at the development of a channel (say channel $i$, where $i = 1, 2$) in such a scenario can model this development process as $\left(\Omega_{i|e_1,\ldots,e_n}, \Sigma_{i|e_1,\ldots,e_n}, \mathrm{P}_{i|e_1,\ldots,e_n}\right)$. Given the common activities' outcomes $e_1, \ldots, e_n$ the sample space $\Omega_{i|e_1,\ldots,e_n}$ will contain all of the possible ways in which the team developing channel $i$ conducts the development while interacting with the other team. From the perspective of an observer studying development process $i$ every possible sequence of observable actions, discussions, decisions, or exchange of ideas that ultimately results in the development of a software version for channel $i$ will be represented as elements of $\Omega_{i|e_1,\ldots,e_n}$. Given such models for the marginal development processes the model of the 1–out–of–2 system's development process should be the *product probability space* $\left(\Omega_{1|e_1,\ldots,e_n} \times \Omega_{2|e_1,\ldots,e_n}, \Sigma_{1|e_1,\ldots,e_n} \times \Sigma_{2|e_1,\ldots,e_n}, \mathrm{P}_{1|e_1,\ldots,e_n} \times \mathrm{P}_{2|e_1,\ldots,e_n}\right)$, as was pointed out in Chapter 2. However, it was also pointed out in Chapter 2 (Section 2.4, page 40) that it does not make sense for a model of dependent version development processes to exhibit probabilistic independence. For, $\Omega_{1|e_1,\ldots,e_n} \times \Omega_{2|e_1,\ldots,e_n}$ does not form a sensible sample space for dependent system development since

*any member of* $\Omega_{1|e_1,\ldots,e_n}$ *may be paired with any member of* $\Omega_{2|e_1,\ldots,e_n}$.

This implies that the sample space contains outcomes that do not reflect agreement on the interactions that occur between the teams: potentially there could be disagreement, between an arbitrary ordered pair of marginal outcomes, concerning what the value of a realized dependence is. One might attempt to overcome this by defining such nonsensical outcomes as zero probability events. However, to do so requires defining legitimate non–zero probability marginal events as zero probability events. In summary, difficulties arise when attempting to construct an example of dependent software development which is modeled using conditional independence, where such conditional independence is not justified by development team isolation.

### 3.2.2 Effect of a single common influence

For the sake of simplicity consider two version development processes sharing a single common influence, $E$, as in Fig. 3.5. If we first consider a specific instantiation of the two processes (including the specific value – say $e$ – taken by $E$) then, like the LM model, the two version sampling distributions (represented by the nodes $\Pi_A$ and $\Pi_B$) are conditionally independent, conditional on the value $e$. Alternatively, consider if only the distribution of $E$ is known rather than its actual value. That is, we do not know how the factor $E$ manifests itself during the specific development project considered but only the *a priori* constraints on the development process. This violates the ISA as can be seen from Fig. 3.5 where the nodes that represent the single version difficulty functions, $\Pi_A$ fails and $\Pi_B$ fails, have the common ancestor node $E$. That is, $E$ induces covariation between the difficulty functions of the version development processes.

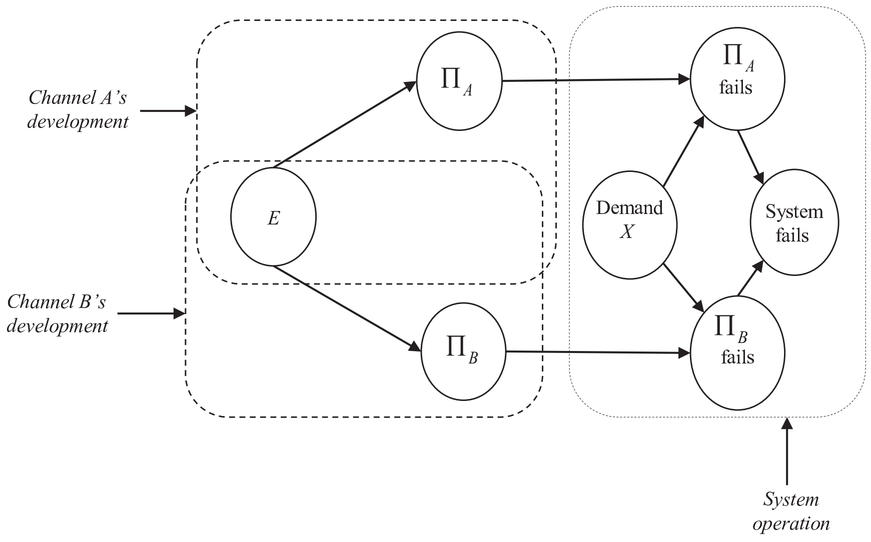


**Figure 3.5:** Bayesian network for a system with one common influence between the development processes. For example, the same test suite is used to test both versions.

So, there are different possible viewpoints on the same process depending on which common influences, among the potentially variable common influences during development, we assume as fixed (that is, either as fully known or as fully determined by events that have "already happened"). Therefore, the fact that some viewpoints imply the ISA and some do not is just an interesting mathematical curiosity, with an aspect of mathematical convenience when the ISA does apply. If we refer back to the "fault leak links" between version developments discussed in [8] we can now see that some of them could usefully be represented in the difficulty function without violating the ISA, and others could not, but this depends to some extent on the observation point that we choose. Upon considering the effects of a common influence with uncertain value we will usually need to assume the ISA is violated.

If, on the other hand, we are considering the effects of a particular value of the influence then the ISA is not violated. For instance, consider the concern that two development teams are likely to have had similar technical education and, thus, share some preferred solutions for typical problems, leading to similar "typical" errors which may tend to cause failures on the same demands. If we are considering two specific, existing, isolated teams their educational backgrounds are determined, their typical errors are described in their respective difficulty functions, and thus the common influence "Educational background" is instantiated: it is not a source of uncertainty. Consequently, the similarity of educational backgrounds does not violate the ISA. The likelihood that it causes common failures is fully described by the two difficulty functions through a contribution to their covariance over the space of demands (see Eq. (2.15)). On the other hand suppose that the teams are yet to be selected. Each possible team has its associated difficulty function. Upon choosing the teams, the way they will develop their respective versions is certainly affected by a common, random factor – "Educational background" – and, consequently, the ISA is violated and the teams' difficulty functions are dependent. This dependence could take the form of positive, zero or negative covariation between the teams' version sampling distributions (and, consequently, the teams' difficulty functions). For a given pair of versions, $\pi_A$ and $\pi_B$ say, positive covariation would mean that the probability of one team producing version $\pi_A$, given that the other team has produced version $\pi_B$, is more likely than it would be if the teams independently produced their respective versions. The increased conditional probability is due to the teams developing their versions in similar ways due to their shared background: the teams develop their respective versions, $\pi_A$ and $\pi_B$, using similar technical approaches, techniques and solutions. This is analogous to the teams finding the same demands difficult to program for, which is described by positive correlation between the teams' difficulty functions. This is no surprise since the common influence "demand" is conceptually the same as the common influence "Educational background". In Chapter 6 : Section 6.2 this conceptual equality will be used to generalise some optimisation results (applicable to the LM model and discussed in Chapters 4 and 5) so that they apply to the generalised models given here.

### 3.2.3 The general case: multiple common influences

Consider multiple, mutually independent influences such as $E_1, ...E_N$ in Fig. 3.2. In addition, there are also the influences $I_A, J_A$ and $I_B, J_B$, affecting development processes related to teams $A$ and $B$ respectively. Since the nodes $I_A, J_A$ and $I_B, J_B$ are ancestors of only $\Pi_A$ and $\Pi_B$ respectively then we may transform the BBN as previously discussed in section 2.6 by averaging over these nodes. The only ancestral nodes of the $\Pi_A$ and $\Pi_B$ nodes will be the nodes $E_1, ...E_N$ that model common influences between how the teams develop their versions. Clearly, the teams are not independent and the ISA does not hold. As a consequence the teams difficulties, represented by the nodes "$\Pi_A$ fails" and "$\Pi_B$ fails", are dependent conditionally on the values of the common influences $E_1, ...E_N$ and the demand $X$. So, for the system difficulty associated with the demand $x$, we have

$$\theta_{AB}(x)$$

$$\begin{aligned}
&= \mathbb{E}[\omega(\Pi_A, x)\omega(\Pi_B, x)] \\
&= \mathbb{E}[\theta_{AB;E_1,\ldots,E_N}(x)] \\
&= \mathbb{E}[\theta_{A;E_1,\ldots,E_N}(x)\theta_{B;E_1,\ldots,E_N}(x)] \\
&= \mathbb{E}[\theta_{A;E_1,\ldots,E_N}]\,\mathbb{E}[\theta_{B;E_1,\ldots,E_N}] + \underset{E_1,\ldots,E_N}{\mathrm{Cov}}(\theta_{A;E_1,\ldots,E_N}, \theta_{B;E_1,\ldots,E_N}) \\
&= \theta_A(x)\theta_B(x) + \underset{E_1,\ldots,E_N}{\mathrm{Cov}}(\theta_{A;E_1,\ldots,E_N}, \theta_{B;E_1,\ldots,E_N}), \qquad (3.1)
\end{aligned}$$

where $\theta_{A;e_1,\ldots,e_N} \equiv \theta_{A;e_1,\ldots,e_N}(x)$ (and similarly for $\theta_{B;e_1,\ldots,e_N}(x)$) is the difficulty for team A given the values $e_1, \ldots, e_N$ and $x$ for the common influences and demand respectively[3]. Also, we used the fact that $\theta_{AB;e_1,\ldots,e_N}(x) = \theta_{A;e_1,\ldots,e_N}(x)\theta_{B;e_1,\ldots,e_N}(x)$, where $\theta_{AB;e_1,\ldots,e_N}(x)$ is the difficulty for the system given a demand, $x$, and values for all of the common influences. So, the equality in the preceding sentence states that $\theta_{AB;e_1,\ldots,e_N}(x)$ exhibits conditional independence, conditional on $e_1, \ldots, e_N$ and $x$. This property can be seen graphically since the canonical form Fig. 3.4 reduces to a BBN with only the three rightmost nodes present when the values $e_1, \ldots, e_N$ and $x$ are given. However, in contrast, equation (3.1) suggests that the system difficulty function, $\theta_{AB}(x) = \mathbb{E}[\theta_{AB;E_1,\ldots,E_N}(x)]$, does not exhibit conditional independence, conditional on $x$. That is, $\theta_{AB}(x) \neq \theta_A(x)\theta_B(x)$ in general. This lack of dependence is a direct consequence of the violation of the ISA (similar to the single common influence case) and is due to uncertainty arising from the common influences. This uncertainty induces covariation between the teams' corresponding difficulties. This covariation is captured by the $\underset{E_1,\ldots,E_N}{\mathrm{Cov}}(\theta_{A;E_1,\ldots,E_N}, \theta_{A;E_1,\ldots,E_N})$ term in Eq. (3.1). Again, from Fig. 3.4, we can appreciate this graphically since even if $x$ is given the three rightmost nodes in the BBN still have the nodes $E_1, \ldots, E_N$ as common parent nodes; the BBN cannot be further simplified in a useful way. To illustrate the intuitive meaning of this covariation consider the case when the covariance is positive. Then, upon observing that a version produced by one team fails on a given demand $x$, we suspect that the values of the common influences $E_1, \ldots, E_N$ were such that under these conditions the team found demand $x$ difficult to program/develop/cater for. Therefore, since both teams are exposed to these conditions, we might expect that the other team would also find demand $x$ difficult to program for.

### 3.2.4 Implications and interesting special cases

In this section we will focus on specific scenarios in which the equations introduced above imply a clear preference between alternative policies for the development of a diverse system. That is, we single out sets of *sufficient conditions* under which a policy should be preferred to an alternative one. Our focus is on selecting cases in which the equations would actually help in decision making: conditions that one may recognise as approximately satisfied in the

[3]The transformation of our BBNs into the canonical form represented by Fig. 3.4 ultimately implies that as far as difficulty functions (the nodes "$\Pi_A$ fails" and "$\Pi_B$ fails" in the BBNs) are concerned the demands are not conceptually different from the outcomes of common influences during the development process: all are points with respect to which failure correlation can occur. So, an alternative notation for the difficulty functions could be $\theta_A(e_1, \ldots, e_N, x)$. However, we will not use this notation as we wish to remind ourselves of the special property of the demands being "outside" of the development process in the notation. Also, relatedly, we want to remind ourselves of the special place the demands have in the LM model as being the only common point of failure correlation.

real world scenario in which one is called to make a decision.

### Mirrored version development processes

An interesting special case is that of two version development processes being (stochastically) *mirrored*, by which we mean the subnetworks that describe the developments of channel A and channel B (see in Fig. 3.3 and Fig. 3.4) are identically distributed. In other words the two processes are probabilistically isomorphic, as in the EL model. Additionally, the processes may have some common influence. In such a case it may be possible to "*decouple*" the processes with respect to the common influence.

> ***Decoupling*** *involves replacing a common influence with a pair of probabilistically independent influences, one for each channel development process. The common influence and the related pair of influences are required to be identically distributed.*

An example of this is a generation procedure for randomly generating test cases to be used in testing the versions developed. As a common influence such a procedure is used to generate test cases which are then used for testing in each channel development process. As a result the same test cases are used to test all of the versions under development. Alternatively, if within each channel development process the procedure is used to generate a possibly unique set of test cases then this would be the decoupled equivalent. Note, the common influence and its associated pair of influences have identical probability models: that is, they have probabilistic models with identical sample spaces and identical probability values assigned to the same events. In particular, the need for identical sample spaces provides an essential characteristic of practical scenarios for which decoupling is possible.

**Observability Criterion 3.2.3.** *For a practical scenario in which decoupling is possible an observer, embedded in a channel's development process, cannot confirm or refute whether an activity is a common influence by observing the outcomes of activities in the development process she is embedded in.*

For example, a scenario in which the development teams necessarily confer with each other before implementing software solutions is one which violates this criterion; members of the teams (i.e. observers embedded in this experiment) know that their period of interaction is a common influence. The observability property here is similar, but not the same as the observability property of the LM model, highlighted in Criterion 2.4.1 on page 38. There a lack of information prevented an observer from determining whether a development process involves the creation of either a single version by a single team, or a pair of versions by a pair of (perfectly isolated) development teams. However, in the current context a lack of information prevents an observer, who might be aware of the existence of another development team, from determining whether an influence is common to both teams or not. Logically, if an observer cannot determine whether another team exists or not then the observer cannot determine whether an influence is common or not. This means that a practical scenario in which Criterion 2.4.1 holds (itself a consequence of the “perfectly isolated teams” assumption of the LM model) is a scenario in which Criterion 3.2.3 above holds.

Graphically, decoupling transforms a graph, such as in Fig. 3.5, into another graph, such as in Fig. 3.6 where $E, E_A$ and $E_B$ are identically distributed. A special case of decoupling was discussed in [38]: the example of replacing a single test suite generation procedure, used to generate test suites applied to the testing of both software versions, with a pair of independent test suite generation procedures, one for each development process. Does decoupling make the system better or worse from the point of view of the expected system *pfd*? From equation (3.1) we may obtain the expected system *pfd* by averaging $\theta_{AB}(x)$ over all possible system demands. However, this is simply an iterated sum over the difficulties $\theta_{AB;e_1,\dots,e_N}$. So, given some ordering in (3.1) of the common influences from innermost sum to outermost sum (say, $E_1$ to $E_N$), it is instructive to see what happens when the innermost sum is evaluated while leaving all of the other sums the same. We obtain

$$\theta_{AB}(x) = \sum_{e_N,\dots,e_1} \theta_{AB;e_1,\dots,e_N}(x)\mathrm{P}_{E_1}(e_1)\dots\mathrm{P}_{E_N}(e_N)$$

$$= \sum_{e_N,\dots,e_2} \mathbb{E}[\theta_{AB;E_1,e_2,\dots,e_N}]\mathrm{P}_{E_2}(e_2)\dots\mathrm{P}_{E_N}(e_N). \tag{3.2}$$

We have chosen this form of the equation to demonstrate the effect of decoupling with respect to $E_1$. This effect can be seen upon scrutiny of the $\mathbb{E}[\theta_{AB;E_1,e_2,\dots,e_N}]$ term. Note that Eq. (3.2) is invariant under a permutation of the common influences due to their mutual independence. So, the choice of the innermost sum being with respect to $E_1$ is without loss of generality. Also, for mirrored version development processes, $\theta_{A;e_1,e_2,\dots,e_N}(x) = \theta_{B;e_1,e_2,\dots,e_N}(x)$, for all $e_1,\dots,e_N,x$ . That is, the difficulty functions are equal. Therefore,

$$\mathbb{E}[\theta_{AB;E_1,e_2,\dots,e_N}] = \mathbb{E}[\theta^2_{A;E_1,e_2,\dots,e_N}] = \big(\mathbb{E}[\theta_{A;E_1,e_2,\dots,e_N}]\big)^2 + \underset{E_1}{\mathrm{Var}}\left[\theta_{A;E_1,e_2\dots e_N}(x)\right], \tag{3.3}$$

where $\mathrm{Var}_{E_1}\left[\theta_{A;E_1,e_2\dots e_N}(x)\right] \geq 0$ always, since it is a variance. So, $E_1$ induces a variance term which would not exist if the development processes were decoupled. So, the expected system *pfd*, viewed as an expectation of the expression “$\big(\mathbb{E}[\theta_{A;E_1,e_2,\dots,e_N}]\big)^2 + \underset{E_1}{\mathrm{Var}}\left[\theta_{A;E_1,e_2\dots e_N}(x)\right]$” in Eq. (3.3), is smaller when the system is decoupled with respect to $E_1$. We can state the following general rule.

**Preference Criterion 3.2.4. : *Decoupling of mirrored version development processes*.** *Given a finite number of independent, common influences between two mirrored development processes substituting a common influence with two independent influences (one for each process) with the same distribution as the removed influence yields either an unchanging or better expected system* pfd.

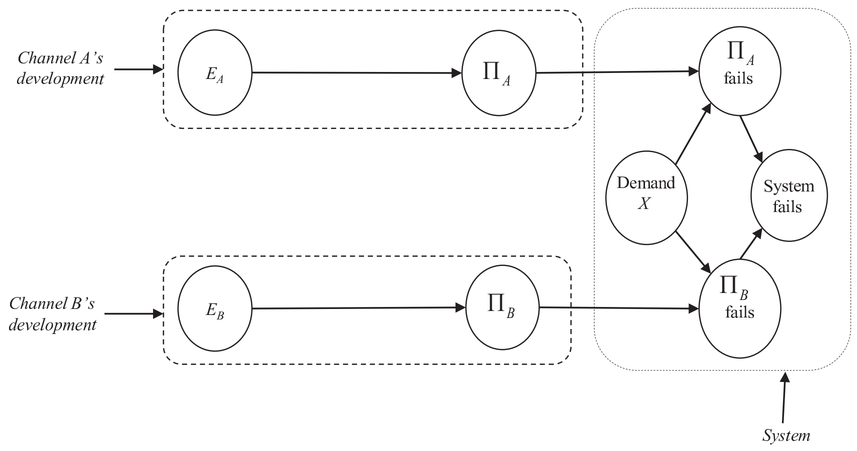


**Figure 3.6:** Here, the system development process is as in Fig. 3.5, except for the fact that the two teams choose their test suites independently. The version sampling distributions and difficulty functions for the two channels are the same as in Fig. 3.5, but the process in this figure produces better system *pfd* (equal *pfd* as a limiting case). The random selection of demands is the only source of uncertainty in common between the channels.

Notice that this result suggests that the system development process with the smallest reliability value is one for which there are no common influences, or all of the common influences are predetermined. In other words, the best case scenario is a model that exhibits the ISA where the version sampling distributions are identical, just like in the EL model. Now, advocates of team isolation have argued that such a policy is beneficial in minimizing the occurrence of so–called "*fault leaks*" between the teams, all in a bid to achieve failure independence between the channels [6, 7, 20, ]. By isolating the teams such a policy removes (or predetermines) all of the common influences from the system development process, resulting in the best case reliability given by an EL-*like* model. It is now clear that, in the context of decoupling a system development process consisting of mirrored version development processes, the EL model is a best case scenario. However, isolation does not take care of the common influence "*a demand is submitted to the system in operation*", which lies "outside" of the the system development process and cannot be removed by isolation. Consequently, as was pointed out in the original EL formulation [12], independent channel failures cannot be achieved, in general. Additionally, note that:

- after applying this "decoupling" the resulting processes are still mirrored and, therefore, this criterion still applies: "decoupling" with respect to *any finite number* of common influences is an improvement;
- In [38] similar results were obtained for a more restricted class of common influence (common test suite generation for both channels' development processes) than the common influence classes presented here. Furthermore, while the version sampling distributions used in [38] were conditionally independent with respect to common influences the scenarios modelled all satisfied observability criterion 3.2.3 and decoupling was always

possible. However, observability criterion 3.2.3 is not necessary for preference criterion 3.2.8 below to hold;

- simply removing a common influence does not guarantee improvement. For instance, one way of removing the common influence in the example of common test suite generation is to eliminate testing altogether. This would change the two version sampling distributions, presumably for the worse and, therefore, may worsen the expected system *pfd*. Instead, "decoupling" as defined above is beneficial because it does not change the two version sampling distributions but only (and for the better) their joint distribution.

**Non-mirrored version development processes**

For the case of development processes that are *not* mirrored what are the implications of decoupling, if any? Non-mirrored, version development processes are not identically distributed. So, there exist $e_1, \ldots, e_N, x$ for which $\theta_{A;e_1,e_2,\ldots,e_N}(x) \neq \theta_{B;e_1,e_2,\ldots,e_N}(x)$. Thus, instead of (3.3), we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{E_1}[\theta_{AB;E_1,e_2,\ldots,e_N}] &= \mathop{\mathbb{E}}_{E_1}[\theta_{A;E_1,e_2,\ldots,e_N}\theta_{B;E_1,e_2,\ldots,e_N}] \\
&= \mathop{\mathbb{E}}_{E_1}[\theta_{A;E_1,e_2,\ldots,e_N}]\mathop{\mathbb{E}}_{E_1}[\theta_{B;E_1,e_2,\ldots,e_N}] \\
&\quad + \mathop{\mathrm{Cov}}_{E_1}[\theta_{A;E_1,e_2,\ldots,e_N}, \theta_{B;E_1,e_2,\ldots,e_N}].
\end{aligned} \tag{3.4}$$

Comparing this with (3.3) we see that $E_1$ induces covariation, instead of variation, between the development teams' corresponding difficulties (that is, $\mathop{\mathrm{Cov}}_{E_1}[\theta_{A;E_1,e_2,\ldots,e_N}, \theta_{B;E_1,e_2,\ldots,e_N}]$). In general, covariances may be negative, positive or zero so there appears to be the potential for decoupling to result in increasing, reducing or not changing the expected system *pfd*. Decoupling would be beneficial if the covariance term in (3.4) was positive for all possible outcomes $e_2, \ldots, e_N, x$. For, in this case, the expected system *pfd* for systems built without decoupling is obtained by taking the expectation of Eq. (3.4) with respect to $E_2, \ldots, E_N, X$ as:

$$\begin{aligned}
pfd_{\text{no decoupling}} &= \mathbb{E}[\theta_{AB}(X)] \\
&= \mathop{\mathbb{E}}_{E_2,\ldots,E_N,X}\left[\mathop{\mathbb{E}}_{E_1}\left[\theta_{AB;E_1,E_2,\ldots,E_N}(X)\right]\right] \\
&= \mathop{\mathbb{E}}_{E_2,\ldots,E_N,X}\left[\mathop{\mathbb{E}}_{E_1}\left[\theta_{A;E_1,E_2,\ldots,E_N}(X)\,\theta_{B;E_1,E_2,\ldots,E_N}(X)\right]\right] \\
&= \mathop{\mathbb{E}}_{E_2,\ldots,E_N,X}\Bigg[\mathop{\mathbb{E}}_{E_1}\left[\theta_{A;E_1,E_2,\ldots,E_N}\right]\mathop{\mathbb{E}}_{E_1}\left[\theta_{B;E_1,E_2,\ldots,E_N}\right] \\
&\quad + \mathop{\mathrm{Cov}}_{E_1}\left[\theta_{A;E_1,E_2,\ldots,E_N}, \theta_{B;E_1,E_2,\ldots,E_N}\right]\Bigg] \\
&= \mathop{\mathbb{E}}_{E_2,\ldots,E_N,X}\left[\mathop{\mathbb{E}}_{E_1}\left[\theta_{A;E_1,E_2,\ldots,E_N}\right]\right]\mathop{\mathbb{E}}_{E_2,\ldots,E_N,X}\left[\mathop{\mathbb{E}}_{E_1}\left[\theta_{B;E_1,E_2,\ldots,E_N}\right]\right] \\
&\quad + \mathop{\mathrm{Cov}}_{E_2,\ldots,E_N,X}\left[\mathop{\mathbb{E}}_{E_1}\left[\theta_{A;E_1,E_2,\ldots,E_N}\right], \mathop{\mathbb{E}}_{E_1}\left[\theta_{B;E_1,E_2,\ldots,E_N}\right]\right]
\end{aligned}$$

$$+ \underset{E_2,\ldots,E_N,X}{\mathbb{E}}\left[\underset{E_1}{\mathrm{Cov}}\left[\theta_{A;E_1,E_2,\ldots,E_N},\theta_{B;E_1,E_2,\ldots,E_N}\right]\right], \tag{3.5}$$

while the expected system *pfd* for systems built by decoupled channel development processes is:

$$\begin{aligned} pfd_{\text{decoupling}} &= \mathbb{E}\left[\theta_{AB}(X)\right] \\ &= \underset{E_2,\ldots,E_N,X}{\mathbb{E}}\left[\underset{E_1}{\mathbb{E}}\left[\theta_{A;E_1,E_2,\ldots,E_N}\right]\underset{E_1}{\mathbb{E}}\left[\theta_{B;E_1,E_2,\ldots,E_N}\right]\right] \\ &= \underset{E_2,\ldots,E_N,X}{\mathbb{E}}\left[\underset{E_1}{\mathbb{E}}\left[\theta_{A;E_1,E_2,\ldots,E_N}\right]\right]\underset{E_2,\ldots,E_N,X}{\mathbb{E}}\left[\underset{E_1}{\mathbb{E}}\left[\theta_{B;E_1,E_2,\ldots,E_N}\right]\right] \\ &\quad + \underset{E_2,\ldots,E_N,X}{\mathrm{Cov}}\left[\underset{E_1}{\mathbb{E}}\left[\theta_{A;E_1,E_2,\ldots,E_N}\right],\underset{E_1}{\mathbb{E}}\left[\theta_{B;E_1,E_2,\ldots,E_N}\right]\right]. \end{aligned} \tag{3.6}$$

The difference between Eq. (3.5) and Eq. (3.6) is the term

$$\underset{E_2,\ldots,E_N,X}{\mathbb{E}}\left[\underset{E_1}{\mathrm{Cov}}\left[\theta_{A;E_1,E_2,\ldots,E_N},\theta_{B;E_1,E_2,\ldots,E_N}\right]\right].$$

So, under conditions that ensure this term is non–negative,

$$pfd_{\text{decoupling}} \le pfd_{\text{no decoupling}}\,. \tag{3.7}$$

Indeed, we now proceed to show that non-negative covariation is guaranteed in the following special case: when corresponding difficulty functions,

$$\text{for instance } \theta_{A;e_1,e_2,\ldots,e_N}(x) \text{ and } \theta_{B;e_1,e_2,\ldots,e_N}(x)\ ,$$

are monotonic functions (both non-increasing or both non-decreasing) of $e_1$, given any set of values $e_2,\ldots,e_N,x$. We prove this by a corollary to the following lemma.

**Lemma 3.2.5.** *Let $f$ and $g$ be functions of the Random Variable $Y$. Let there exist a point $y=\bar{y}$, such that*

$$\begin{cases} f(y) \ge \mathbb{E}_Y\left(f(Y)\right) & : \quad y \le \bar{y}, \\ f(y) \le \mathbb{E}_Y\left(f(Y)\right) & : \quad y > \bar{y} \end{cases}$$

*so that $\int_{y\le\bar{y}}\left(f(y)-\mathbb{E}_Y\left(f(Y)\right)\right)dF_Y(y) \ge 0$ (The integral here is the measure-theoretic Lebesgue integral taken with respect to $F_Y(y)$, the cumulative distribution function of $Y$, viewed as a probability measure) and let $g$ be a monotonically decreasing, real-valued function of $y$. That is, $g$ is a real-valued function such that $g(x_1) \le g(x_2)$ whenever $x_1 \ge x_2$. Then*

$$\underset{Y}{\mathrm{Cov}}\left(f(Y),g(Y)\right) \ge 0$$

*Proof.* Consider that

$$\begin{aligned} 0 &= \int_y \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, dF_Y(y) \\ &= \int_{y>\bar{y}} \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, dF_Y(y) + \int_{y\le\bar{y}} \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, dF_Y(y) \\ \Rightarrow -\int_{y\le\bar{y}} \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, dF_Y(y) &= \int_{y>\bar{y}} \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, dF_Y(y) \end{aligned}$$

The first equality was deduced from the definition of expectation of Random variable while the second equality is a consequence of the additive property of Lebesgue integrals. Using this result, the requirements $\begin{cases} f(y) \ge \mathbb{E}_Y(f(Y)) & : \quad y \le \bar{y}, \\ f(y) \le \mathbb{E}_Y(f(Y)) & : \quad y > \bar{y} \end{cases}$ and $g$, a monotonically decreasing function of $y$, we see that

$$\begin{aligned} \int_{y>\bar{y}} \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, g(y) dF_Y(y) &\ge g(\bar{y}) \int_{y>\bar{y}} \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, dF_Y(y) \\ = -g(\bar{y}) \int_{y\le\bar{y}} \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, dF_Y(y) &\ge -\int_{y\le\bar{y}} \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, g(y) dF_Y(y) \\ \Rightarrow \int_y \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, g(y) dF_Y(y) &\ge 0 \end{aligned}$$

Recall: We may always write $\underset{Y}{\mathrm{Cov}}(f(Y), g(Y)) = \int_y \Big(f(y) - \underset{Y}{\mathbb{E}}(f(Y))\Big)\, g(y) dF_Y(y)$. Consequently, using this above gives the required result,

$$\underset{Y}{\mathrm{Cov}}(f(Y), g(Y)) \ge 0\,. \qquad \blacksquare$$

**Corollary 3.2.6.**

$$\underset{E_2,\ldots,E_N,X}{\mathbb{E}} \left[ \underset{E_1}{\mathrm{Cov}}[\theta_{A;E_1,E_2,\ldots,E_N}(X), \theta_{B;E_1,E_2,\ldots,E_N}(X)] \right] \ge 0$$

*whenever the difficulty functions, $\theta_{A;e_1,e_2,\ldots,e_N}(x)$, and $\theta_{B;e_1,e_2,\ldots,e_N}(x)$, are monotonically decreasing functions of the influence $E_1$, for all realisations $e_2, \ldots, e_N, x$ of the random variables $E_2, \ldots, E_N, X$ .*

*Proof.* For every demand $x$ and any common influences $e_2, \ldots, e_N$, suppose $\theta_{A;e_1,e_2,\ldots,e_N}(x)$, $\theta_{B;e_1,e_2,\ldots,e_N}(x)$ and the random variable $E_1$ have the properties of $f$, $g$ and $Y$ respectively, from the last Lemma. Then, for each demand $x$, and influences $e_2, \ldots, e_N$, we have

$$\underset{E_1}{\mathrm{Cov}}[\theta_{A;E_1,e_2,\ldots,e_N}(x), \theta_{B;E_1,e_2,\ldots,e_N}(x)] \ge 0\ .$$

Therefore,

$$\mathop{\mathbb{E}}_{E_2,\ldots,E_N,X}\left[\mathop{\mathrm{Cov}}_{E_1}[\theta_{A;E_1,E_2,\ldots,E_N}(X),\theta_{B;E_1,E_2,\ldots,E_N}(X)]\right]\geq 0\,. \qquad \blacksquare$$

We have just proved the following preference criterion.

**Preference Criterion 3.2.7. :** ***Decoupling of diverse version development processes****. If two version development processes with a finite number of independent, common influences possess difficulty functions that are positively correlated with respect to some common influence E (for any values of the other common influences and the demand), then substituting E with two independent influences (one for each process) with the same distribution as the removed influence yields either an unchanging or better expected system* pfd.

As an example consider this scenario: Our two-version system is to control a new model of some kind of equipment. Part of its planned V&V process will be system testing in the equipment prototype. A certain time budget has been allocated for this system testing phase but the actual time available may vary depending on when the prototype is actually ready. Both teams are thus exposed to the same random variation: "time available for testing on the prototype" is a common influence[4]. It is reasonable to expect an increase in the time available for testing to improve the "difficulty" (i.e. reduce the numerical value of the difficulty function) for every demand given fixed values for all other influences, and a decrease in time would make the difficulties worse. Therefore, the teams' difficulty functions can be viewed as monotonically decreasing functions of the common influence "time". This is sufficient for the covariance term in (3.4) to be positive. So, uncertainty about the actual time available for testing adds to the probability of common failure of the average pair of versions developed, as compared to what would be calculated by just assuming the average effect of the time influence. Although one may never be able to calculate these expected *pfd*s from the equations in practice this is not a purely theoretical result. It tells us that if we had a choice between purchasing versions that had been originally affected by such a common influence (but without knowing its value: how much system testing time was actually available) and others that were developed independently – the development processes being otherwise *statistically* equal in the two scenarios – we should expect better system *pfd* from the latter.

Of course, a dual theorem applies for negative covariance. Substituting two non–common, identically distributed influences with an identically distributed common one such that the covariance with respect to it is negative will improve the expected system *pfd*. Creating forms of negative covariance this way – that is, introducing random influences that whenever they happen to hamper one version development team on some demands, help the other team on the same demands – is useful in software development. By doing this we achieve increased

[4]Actually, the concept of time used here is not necessarily the number of hours spent on testing, say. For instance, it may be more useful to consider "time" as the number of test cases considered.

levels of reliability at the cost of producing a single random influence. This generalizes a "more reliability at no extra economic cost" result that was first discussed in [38]. There it was noted that in the case of testing it is hard to imagine strategies giving such negative covariance.

It is possible, given a joint process with mirrored development processes and common influences, to mandate that some activity be independently carried out in different ways by the individual teams resulting in diverse channel development processes. A peculiarity of this is a form of irreversibility. Once we alter two mirrored version development processes by "diversifying" some part of them, and thus turning variance terms in our equations into covariance terms, adding a common influence will not undo the diversification, in the sense that it will not reintroduce variance terms into the expression for the average system *pfd*. That is, adding this additional "coupling" may make the system less reliable, on average, but will not produce a variance term in the equations; we have shown such variance terms are a sufficient condition for decreased reliability. Mathematically, once two version development processes are stochastically "diverse" (i.e., non-mirrored), adding common influences cannot make them "non-diverse" again.

## Mirrored versus non–mirrored version development processes

Finally, again we turn to the case of mirrored, version development processes with a common influence, $E$, and consider how this compares with a related case of non–mirrored version development processes. In [13] it was shown that, under LM model assumptions, if the expected system *pfd*s of two 1–out–of–2 systems built with different methodologies are identical, then forcing diversity in the development of a 1–out–of–2 system cannot result in worse reliability. Indeed, forcing diversity can result in better reliability. We discussed this result in Section 2.5 (see Eq. (2.18)), and prove the result in Section 5.2. In this section we show that this result still holds upon relaxing the LM model assumption of perfectly isolated teams?

A plausible way of "increasing diversity" is to require one of the two development teams to change its process in some aspect (change the probability distributions associated to some node in the BBN for the channel development process) so as to obtain a process that is intuitively different but promises to be no worse than the previous process. The difference between the two processes might be in the technology used, or the testing methods, or in the algorithms implemented. Let us indicate the two processes as $\alpha$ and $\beta$. Given that the development of the channels of a 1–out–of–2 system are conditionally independent, conditional on some common influence, is it better to build the channels using the same process for each channel's development (in effect having homogeneous development processes), or is forcing diversity preferable? We proceed to show that under indifference we expect forcing diversity to result in a system that is no worse than if homogeneous development processes were used.

First, let us label the channels of the system as $A$ and $B$. Each channel may be developed using either process $\alpha$, or process $\beta$. The development of a channel, as before, is modelled as the random selection of a software version from a population of software. If a channel, $A$ say, is developed using a process, $\alpha$ say, then the random variable that models the channel's development is written as $\Pi_{A\alpha}$. So, all 4 combinations $\Pi_{A\alpha}$, $\Pi_{A\beta}$, $\Pi_{B\alpha}$, $\Pi_{B\beta}$ are meaningful. Further, recall from earlier in the chapter (see the discussion surrounding Eq. (3.1) on page 69) that the channels' development processes may share a common influence. We model this influence as the random variable $E$ and, given an outcome of $E$, the channels are assumed to be developed independently. Suppose a single process, say $\alpha$, is used to develop each channel. Then, given a realisation of $E$, the development process is modelled as the independent, identically distributed, random selection of a pair of versions, $\Pi_{A\alpha}$ and $\Pi_{B\alpha}$. If, however, a unique process is used in developing each channel, $\alpha$ for channel $A$ and $\beta$ for channel $B$ say, we model this using the diverse pair of conditionally independent random variables $\Pi_{A\alpha}$ and $\Pi_{B\beta}$, conditional on $E$. The implication of this conditional independence is that given a system demand, $x$, and common influence realisation, $e$, the channels are expected to fail independently. In terms of expectations, this is the equation

$$\mathbb{E}\left[\ \omega(\Pi_{A\alpha},x)\ \omega(\Pi_{B\beta},x)\ |x,e\right] = \mathbb{E}\left[\omega(\Pi_{A\alpha},x)|x,e\right]\ \mathbb{E}\left[\omega(\Pi_{B\beta},x)|x,e\right]. \tag{3.8}$$

Suppose that $\alpha$ and $\beta$ are equivalent in that they satisfy the "indifference" condition introduced in [13]: a system made of two versions, $\Pi_{A\alpha}$ and $\Pi_{B\alpha}$, produced by $\alpha$ has the same expected *pfd* as one of two versions, $\Pi_{A\beta}$ and $\Pi_{B\beta}$, produced by $\beta$:

$$\mathbb{E}_{X,\Pi_{A\alpha},\Pi_{B\alpha},E}\left[\omega(\Pi_{A\alpha},X)\omega(\Pi_{B\alpha},X)\right] = \mathbb{E}_{X,\Pi_{A\beta},\Pi_{B\beta},E}\left[\omega(\Pi_{A\beta},X)\omega(\Pi_{B\beta},X)\right],$$

where $\mathbb{E}_{X,\Pi_{A\alpha},\Pi_{B\alpha},E}\left[\omega(\Pi_{A\alpha},X)\omega(\Pi_{B\alpha},X)\right]$ and $\mathbb{E}_{X,\Pi_{A\beta},\Pi_{B\beta},E}\left[\omega(\Pi_{A\beta},X)\omega(\Pi_{B\beta},X)\right]$ are the average *pfd*s for the $\alpha$ and $\beta$ based systems respectively. In summary, using

- each pair of conditionally independent random variables, $(\Pi_{A\alpha},\Pi_{B\beta})$, $(\Pi_{A\alpha},\Pi_{B\alpha})$ and $(\Pi_{A\beta},\Pi_{B\beta})$;
- the ***Cauchy–Schwarz*** inequality[5];
- each pair of identically distributed random variables, $(\Pi_{A\alpha},\Pi_{B\alpha})$ and $(\Pi_{A\beta},\Pi_{B\beta})$;
- and indifference;

as follows:

$$\mathbb{E}_{X,\Pi_{A\alpha},\Pi_{B\beta},E}\left[\ \omega(\Pi_{A\alpha},X)\ \omega(\Pi_{B\beta},X)\ \right]$$

$$= \mathbb{E}\left[\mathbb{E}\left[\ \omega(\Pi_{A\alpha},X)\ \omega(\Pi_{B\beta},X)\ |X,E\right]\right]$$

$$= \mathbb{E}\left[\mathbb{E}\left[\omega(\Pi_{A\alpha},X)|X,E\right]\ \mathbb{E}\left[\omega(\Pi_{B\beta},X)|X,E\right]\right]$$

$$\leq \sqrt{\mathbb{E}\left[\mathbb{E}\left[\omega(\Pi_{A\alpha},X)|X,E\right]\ \mathbb{E}\left[\omega(\Pi_{A\alpha},X)|X,E\right]\right]} \times \sqrt{\mathbb{E}\left[\mathbb{E}\left[\omega(\Pi_{B\beta},X)|X,E\right]\ \mathbb{E}\left[\omega(\Pi_{B\beta},X)|X,E\right]\right]}$$

$$\leq \sqrt{\mathbb{E}\left[\mathbb{E}\left[\omega(\Pi_{A\alpha},X)|X,E\right]\ \mathbb{E}\left[\omega(\Pi_{B\alpha},X)|X,E\right]\right]} \times \sqrt{\mathbb{E}\left[\mathbb{E}\left[\omega(\Pi_{A\beta},X)|X,E\right]\ \mathbb{E}\left[\omega(\Pi_{B\beta},X)|X,E\right]\right]}$$

[5]Let $f(x)$ and $g(x)$ by any two real, integrable functions and $X$ a random variable. The Cauchy-Schwarz inequality holds: $\mathbb{E}[f(X)g(X)] \leq (\mathbb{E}[f(X)^2]\,\mathbb{E}[g(X)^2])^{1/2}$. The inequality is a necessary consequence of the properties of ***inner–products*** which are explored in greater detail in Chapter 4

$$= \sqrt{\mathbb{E}_{X,\Pi_{A\alpha},\Pi_{B\alpha},E}\left[\ \omega(\Pi_{A\alpha},X)\ \omega(\Pi_{B\alpha},X)\ \right]} \times \sqrt{\mathbb{E}_{X,\Pi_{A\beta},\Pi_{B\beta},E}\left[\ \omega(\Pi_{A\beta},X)\ \omega(\Pi_{B\beta},X)\ \right]}$$

$$= \mathbb{E}_{X,\Pi_{A\beta},\Pi_{B\beta},E}\left[\ \omega(\Pi_{A\beta},X)\ \omega(\Pi_{B\beta},X)\ \right], \tag{3.9}$$

we conclude that

$$\mathbb{E}_{X,\Pi_{A\alpha},\Pi_{B\beta},E}\left[\ \omega(\Pi_{A\alpha},X)\ \omega(\Pi_{B\beta},X)\ \right] \le \mathbb{E}_{X,\Pi_{A\beta},\Pi_{B\beta},E}\left[\ \omega(\Pi_{A\beta},X)\ \omega(\Pi_{B\beta},X)\ \right]. \tag{3.10}$$

In a natural way this result may be extended to version development processes with a finite number of common influences: the result is still valid if the common influence $E$ is replaced by a set of common influences $E_1, \ldots, E_N$ that, collectively, exhibit the properties of $E$. This result justifies the following preference criterion.

**Preference Criterion 3.2.8. :** ***Diversification between version developments****. If using either a process $\alpha$ exclusively, or a process $\beta$ exclusively to develop a pair of versions for a 1–out–of–2 system results in the same expected system* pfd *value, then using process $\alpha$ for one version and process $\beta$ for the other yields expected system* pfd *which is at least as good or better.*

In particular, this generalizes a similar result in [13] obtained under the ISA. The difference in the present case is that the teams are no longer necessarily isolated; ISA may not hold and the system development processes are ones for which version sampling distributions are conditionally independent, conditional on common influences.

## 3.3 Summary

In this chapter we have explored models that generalise the LM model. These models, to various degrees, relax the assumption of “perfectly isolated” development teams. We have focused on modelling practical scenarios that preserve essential properties of the LM model while still allowing the teams to be dependent. We have been particularly interested in practical situations in which dependent software development can be adequately modelled by version sampling distributions that exhibit conditional independence, conditional on outcomes of activities common to both channels’ development processes (see Sections 3.1 and 3.2). For example, those practical scenarios in which there is an alternating of dependence creating influences with isolated channel development (see Section 3.2.1). In such scenarios it may be possible to transform the system development process into one that obeys the Independent Sampling Assumption (ISA). This is accomplished by either predetermining the outcomes of influences common to both channels’ development processes, or/and *decoupling* the system development process. *Decoupling* is the act of replacing a given common influence with a pair of independent, identically distributed influences (one for each channel development process) that share the same distribution as the common influence. In particular, a consequence of being able to *decouple* a system development process is that preferences, in terms of system reliability, can be stated between a system development process without decoupling and one with decoupling. We demonstrated that

- for a system built by employing the same methodology for each channels' development process (we term this a *mirrored development process*), introducing dependence between the development processes in such a way that the processes are, otherwise, isolated from each other cannot improve the resulting system reliability. This preference demonstrates the sense in which the EL model is optimistic. A common influence added to the homogeneous system development process of the EL model in such a way that the teams are, otherwise, isolated from each other cannot improve system reliability. The best situation for a homogeneous system development process is one in which the development teams are isolated;
- for a *non-mirrored development process* such that the channels have positive failure correlation with respect to some common influence (for all values of the other influences), decoupling with respect to this influence cannot worsen the resulting system reliability and might improve it.

Also, we generalised a result first stated in [13] and rederived in Chapter 2, Section 2.5. This result stated an indifference condition under which *forcing diversity*[6] is expected to result in system reliability that cannot be worse (and might be better) than if diversity were not forced. There, the teams were assumed to be "perfectly isolated". This has been weakened to the case where the teams have dependence creating influences between them and, otherwise, remain isolated. The generalised result is as follows. Given two development process methodologies, "A" and "B", an assessor who has reason to believe that the expected system *pfd* for an "AA" system is the same as the expected system *pfd* for a "BB" system (that is, the assessor is indifferent between these *pfd*s) should also believe that the expected system *pfd* for an "AB" system cannot be worse than either of the other two expected *pfd*s (see Preference Criterion 3.2.8). In fact, using the geometric arguments of Chapter 4 we shall show that the expected system *pfd* for an "AB" system is likely to be strictly better, since the practical conditions under which these expected system *pfd*s are equal are unlikely; *under this sense of indifference only pairs of identical difficulty functions can achieve such an equality.* In practice we expect differences, even if they are subtle ones, between the difficulty functions associated with the channels' development processes.

Decoupling is possible only if observability Criterion 3.2.3 is satisfied. This criterion requires that *an observer embedded in the development process for a channel should not be able to confirm, or refute, whether an activity is common to the channels' development processes by observing the outcomes of activities in the development process in which she is embedded.* This is similar to, but not the same as, the observability Criterion 2.4.1 in Chapter 2. That criterion stipulates that *an observer embedded in a channel's development process should not be able to confirm, or refute, the existence of another channel's development process by observing the outcomes of activities within the development process in which she is embedded.* However, it is the case that a system development process in which Criterion 2.4.1 holds is necessarily one for which Criterion 3.2.3 also holds (see Section 3.2); Criterion 2.4.1

[6]Recall, forcing diversity is the requirement that the channels be developed using different methodologies

could still hold even if there were activities with outcomes that impact on each channel's development process, but cannot hold if such outcomes were informative about whether common influences exist or not (since this would imply the existence of multiple development teams).

# Chapter 4

# Geometric Models of Coincident Failure

In Chapter 3 we stated criteria that indicate preferable ways of organising the development of multi–version software. The criteria are appealing for the following reason: knowledge of expected system *pfd*s are not required for using the criteria in practice. Instead, a manager of a development process only needs to justify that her development process satisfies the characteristics for the criteria to be applied. This is useful since, in practice, expected system *pfd*s may be either unknowable or difficult to estimate. However, a shortcoming of this is that the criteria give little indication about the limits placed on the expected system *pfd* by process diversity. That is, the preference criteria specify an ordering between expected system *pfd*s, but without giving information about "how big" such an ordering might be. For instance, the criteria may indicate that forcing diversity should be pursued (since the resulting expected system *pfd* can be no worse, and may be better, than if diversity was allowed to occur naturally). But, is the expected system *pfd* strictly better and, if so, by how much? What are the limits on expected reliability when forcing diversity?

In answering these and other related questions, we propose the use of *attainable bounds* on expected system *pfd*s. Such bounds specify the limits on expected system reliability, given estimates for other probabilities that characterise the practical scenario. For instance, a typical bound would be the maximum expected *pfd* of a 1–out–of–2 system built by forcing diversity, given estimates of the expected *pfd*s for the system channels. In obtaining such bounds the following terminology will be used. We refer to the measure whose value is to be bounded — in this case the expected system *pfd* – as an ***objective function***. The available knowledge and conditions under which a bound should be attainable are called the ***constraints***. And, the problem of deriving an attainable bound for an objective function, subject to constraints, is referred to as an ***extremisation problem***.

So, given these bounds, how much benefit can forcing diversity bring? That depends on if one is interested in potential gains to expected system *pfd*, or potential shortcomings. Optimistically, comparing the attainable lower bound on expected system *pfd* with expected channel *pfd* gives the best expected reliability improvement possible due to diversity. However, while this may be nice to know, it is not prudent. A more cautious consideration would be to see how bad things can get. So, a similar comparison – but using the attainable upper bound instead – indicates the least benefit from forcing diversity. And, from such comparisons, we may even state cases when forcing diversity is preferred: these are cases where the largest possible expected system *pfd*, resulting from forcing diversity, is significantly lower than the related expected system *pfd*s when diversity is not forced.

## 4.1 Requirements for an Approach to Extremising *PFD*s

Our aim, therefore, is to perform the constrained extremisation of the expected system *pfd*. This can be a non-trivial task because:

- some of the *pfd*s are quadratic in form, which can make their use as constraints or objective functions result in difficult extremisation problems.
- the nature of the extremisation problems themselves can make the use of conventional methods – such as multivariable calculus – unhelpful. In particular, calculus can have limited use in those extremisation problems where the shape of the region of potential solutions implies that stationary points do not determine the extremes of interest.
- purely algebraic approaches[1] to extremisation can fall short in adequately describing the relationship between linear constraints and quadratic objective functions. The problem here is intuitive relationships can be difficult to state, making the extremisation problem unnecessarily difficult.
- a purely probabilistic approach can have the following shortcoming: functions of probabilities may not have any obvious probabilistic interpretation, where such functions arise naturally from the problem being solved. An example of such a function would be a ratio of expected *pfd* and the square–root of a related expected system *pfd*[2].

So, upon taking these concerns into account, we have the following 3 sets of requirements for a suitable approach to our extremisation problems.

1. The models being analysed should:
    - explicitly take into account how diversity may be forced during the system development processes.
    - allow comparisons between expected system *pfd*s resulting from different development processes.
2. The analysis approach should be sufficiently flexible to handle the following types of constraints, preferably in a consistent and intuitive manner.
    - Expected *pfd*s (for either the channels or the system)
    - Demand profiles

[1] Purely algebraic, in the sense that these approaches do not appeal to geometric relationships and notions like angles, projections, rotations and lengths.

[2] In Section 4.6, page 102, we show that such a ratio completely describes the angle between a difficulty function and the demand profile, and is a very useful notion in maximising the variation of difficulty.

  - Activities during the system development process
  - Difficulty functions

3. The analysis should derive:
  - *attainable* bounds on the expected system *pfd* resulting from forcing diversity.
  - orderings between expected system *pfd*s resulting from different development processes.

All of these requirements are satisfied by approaching the extremisation problems using geometry. Our proposed geometric approach takes into account all of the aforementioned constraint types, and describes intuitive relationships between objective functions and constraints. As a bonus, a number of relationships between difficulty functions, or between difficulty functions and the demand profile, can be stated. This expands both the understanding and lexicon of the models of diversity. To this end, this chapter casts the probabilistic models defined so far in the framework of vector–spaces. In doing so, insight into relationships between various constructs, such as difficulty functions and expected *pfd*s, is gained. Also, extremisation problems concerning such entities benefit from the mathematical methods in a geometric context. For, while such extremisation problems may be approached using other methods, there are several cases in which geometric characterisations of these problems reduce them to relatively simple corollaries of well–known vector manipulations in appropriate finite–dimensional vector–spaces.

To aid with the proofs of various extreme *pfd* values, the development of vector–space methods presented here follows so–called "*coordinate–free*" or "*coordinate– independent*" formulations. The central idea is that many geometric properties/rules can be stated without appealing to particular coordinate systems for representing the vectors[3]. In fact, a number of the theorems in the model we develop will not use particular coordinates in their statement. Instead, these theorems are stated using the *inner–product*; a notion of vector lengths and angles between vectors, defined in a manner independent of a choice of coordinate system. While these theorems can be expressed in terms of coordinate systems, doing so would obscure the ubiquity of the theorems. This coordinate system invariance is particularly useful when solving optimisation problems, since a convenient coordinate system – one in which the problem being solved is made relatively simple – can be used to prove a result that is guaranteed to be true irrespective of the coordinate system used in its proof. If we chose not to use coordinate–independent formulations, then we would have to justify why a theorem proved in one coordinate system is also true in another; an unnecessary complication. In summary, **there are geometric theorems that can be stated without appealing to particular coordinate systems, and theorems involving expected system *pfd*s can be recast as such coordinate–independent geometric theorems**. Such a recasting is achieved by making two observations. Firstly, by noting that a number of theorems in the LM model concern the expected (system) *pfd* which, being the probability of an event, is a real number. Consequently, the expected system *pfd* can be regarded as a so–called

[3]In Appendix A, from page 190, we show that choosing a coordinate system is equivalent to choosing a *basis*: that is, equivalent to choosing a minimal set of vectors that completely describe the vector–space of interest

*scalar*[4]. Secondly, note that the invariant geometric theorems of interest are precisely those that state relationships between scalars (such as lengths or angles). So, by constructing a geometric model such that theorems about expected system *pfd*s are equivalent to theorems about lengths and angles, these theorems are coordinate–system invariant and we may prove them using any coordinates of our choosing. This application of coordinate–free geometry to modelling coincident software failure is novel, and inspired by similar approaches used in Mathematical physics[5].

This chapter focuses on presenting a geometric formulation of the LM model, and thereby applying vector–space methods to extremisation problems involving difficulty funtions and expected *pfd*s. The work presented here relies heavily on vector–space methods. Appendix A is a review of vector–spaces and linear algebra. The reader who is unfamiliar with vector–space methods may refer to the appendix. Some texts that provide an excellent introduction and comprehensive treatment of some of the material presented here include [40, 41, 42, 43, 44]. A brief outline of the sections that make up this chapter is as follows:

4.2 *A Geometric Approach to Extremising PFDs*: A description of the steps involved in extremising *pfd*s by appealing to geometry;

4.3 *The Geometry of the LM Model*: Develops geometric forms of EL/LM model concepts including representing difficulty functions as vectors, failure sets as vectors, expectations as inner–products and expected single–version *pfd*s as the magnitudes of projections of difficulty functions onto the demand profile;

4.4 *Transformation between Bases*: Discusses a transformation that casts the inner–product suggested by the EL/LM model into its canonical form. The canonical form of the inner–product is the preferable form to perform extremisations in. This is because in canonical form calculations to determine angles and vector lengths simplify, almost becoming as simple as similar calculations performed in a high–school level course on (Euclidean) geometry. For instance, consider the generalized Pythagoras theorem for the lengths of the sides of a right–angled triangle. The canonical form simplifies the general form of the theorem into a relationship between sums of squares of numbers, just like the simple 2–dimensional version of the theorem. Consequently, intuition gained in reasoning about 2, or 3–dimensional geometry can be drawn upon in higher dimensional settings.

4.5 *A Brief Clarification on Notation*: A brief discussion of the notation used in this thesis when discussing the geometric formulation of the LM model;

4.6 *Extremisation via Angles, Magnitudes and Planes*: Develops and proves theorems that relate difficulty functions, their variation, their size, the angles they make with the demand profile, and their expectations. Also, we demonstrate that software versions

[4]Here we make a comment about some of those entities that are modelled as scalars, and those modelled as vectors within this thesis. On the one hand a *pfd* has a geometric correspondence as a scalar and not a vector: it has magnitude, but no preferred way in which it can be said to have a direction. On the other hand, if one considered a random variable whose realisations are *pfd*s, then vector representations of such a random variable are possible. In summary, for our purposes in this thesis, the probability of an event is modelled as a scalar, and random variables as vectors.

[5]For example, in modern formulations of Albert Einstein's "General Theory of Relativity", vector–space methods are used in such a way that they conform to Einstein's *principle of general covariance*: that is, "the laws of physics must be of such a nature that they apply to systems of reference in any kind of motion" [39]. In particular, the geometric formulation of physical laws are coordinate–invariant.

or failure regions (i.e. difficulty functions with only 1's and 0's as values) have special properties. For instance, these are the only difficulty functions such that the cosines of the angles they make with the demand profile are their magnitudes. This means that it is impossible for any difficulty function that is not a version to have this property and, as a consequence, this property will be useful in defining attainable upper and lower bounds for such difficulty functions;

4.7 *Preliminary Results of Geometry–based Extremisation*: Various extremisation problems and their solutions that will be useful in extremisation problems explored in Chapter 5. These include determining the largest and smallest angles a difficulty function can make with the demand profile. Also, determining the largest difficulty function in the set of all difficulty functions with the same predefined mean.

## 4.2 A Geometric Approach to Extremising *PFD*s

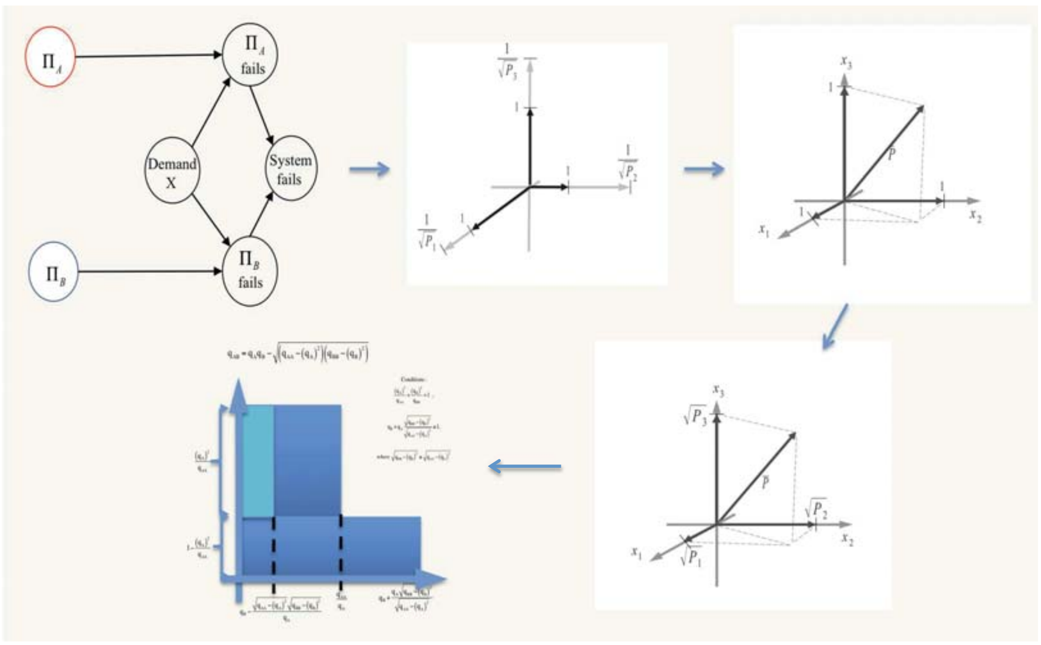


**Figure 4.1:** A depiction of the 5 steps in extremising expected system *pfd*: 1) A practical scenario is modeled using the LM model, or any suitable extension thereof (see Chapter 3 for more detail); 2) This, in turn, defines an equivalent vector–space representation; 3) Regions of possible solutions within this vector–space are determined by the constraints of the problem; 4) Computations to derive the bound are simplified by a change in how the vectors are represented (a so–called *change of basis vectors*); 5) The bound is obtained and the solution is recast from being in terms of the vector–space, into an equivalent form under the LM model (extension).

Given constraints and an objective function, our geometric approach involves the following 5 steps, illustrated in Fig. 4.1:

1: *Extensions to the LM model*: We start with a model of coincident failure in multi–version software: either the LM model, or any of its extensions that have the canonical form discussed in Chapter 3. Which of these models we use will depend on whether the system channels are developed in an independent, or dependent manner.

2: *Geometric representation of the LM model extensions*: The aforementioned model of coincident failure is used to define an equivalent vector–space representation: a *finite–dimensional, real, inner–product space* in a given *basis*. The number of dimensions for this vector–space is either the number of demands (in the case where one starts with the LM model), or the number of possible ways of realising the random vector consisting of all common–influences and a randomly chosen demand (in the case where an LM model extension is used instead). Also, in this vector–space, Difficulty functions define vectors with non–negative components and a length less than 1. Demand profiles also define certain vectors, and regions in the vector–space. And finally, expected *pfd*s define an inner–product. In–so–doing, angles and lengths in this vector–space are defined.

3: *Constraints and Change–of–Basis*: The problem constraints define the region of the vector–space with all of the potential solutions. Also, to make the problem easier to solve, by a change of basis we can simplify the formulae for calculating angles and lengths.

4: *Transformation of vectors*: The extremisation is accomplished by transforming the vectors contained in the region of potential solutions, in a bid to to obtain a vector (or a collection of vectors) which achieve the extreme. Typical transformations include projections, rotations and scalings. The transformations which solve a given problem will, in general, be different from the transformations which solve some other problem.

5: *Solutions in the LM model*: We reinterpret the geometric solutions obtained, casting them in terms of the LM model extensions we began with. This usually means that from the geometric solution we specify a pair of difficulty functions and a demand profile that achieve a given bound.

For the sake of simplicity we shall consider only those practical situations which are adequately modeled by the LM model. However, those situations modeled by LM model extensions follow an analogous development. The primary difference is that a random vector (consisting of all the common–influences and a randomly chosen demand) is considered, instead of just a randomly chosen demand as we do in what follows. So in what follows, by essentially "relabeling" the demands as vectors of both demands and common–influences, we also cover those scenarios modeled by any of the LM model extensions discussed in Chapter 3.

## 4.3 The Geometry of the LM Model

In this section we use the LM model to define an equivalent geometric model. This continues the discussion started in Section 2.6.3 on page 53, where difficulty functions were viewed as vectors in a finite–dimensional vector–space. We proceed by making educated guesses about what geometric interpretations to give to such concepts as difficulty functions, and expected *pfd*s. These interpretations will guide us in formally defining the LM model geometrically.

To this end, consider an LM model,

$$\left\{\left(\Omega_A \times \Omega_B, \Sigma_A \times \Sigma_B, \mathrm{P}_A \times \mathrm{P}_B\right), \left(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X\right)\right\},$$

where $\mathcal{X}$ is finite and assumed to contain $n$ demands, and $\Omega_A \times \Omega_B$ is finite. We are free to choose some ordering of the demands from 1 to $n$, so that $\mathcal{X} := \{x_1, \ldots, x_n\}$. A related difficulty function, say $\theta^*_A(x)$ for each $x \in \mathcal{X}$, is a bounded, real–valued function of the demands[6]; that is, $\theta^*_A : \mathcal{X} \to [0,1]$. By considering the value of the difficulty function on each demand, we may write the difficulty function as an $n$–tuple $(\theta^*_A 1, \theta^*_A 2, \ldots, \theta^*_A (n-1), \theta^*_A n)$, where for notational convenience we have written

$$\theta^*_A\left(x_i\right) \quad \text{as} \quad \theta^*_A i$$

for $i = 1, \ldots, n$. Clearly, such an $n$–tuple can be viewed as the components of an n–dimensional vector with respect to some basis. We denote this vector as $\theta^*_A$ to remind ourselves that the components of the vector in a given basis is equivalent to the difficulty function $\theta^*_A\left(x\right)$. The particular basis relevant here will be defined later in the chapter (see page 91). Observe that there are $n$ unique demands in the definition of the difficulty function, and $n$ unique basis vectors in the definition of the components, so we expect some correspondence between the demands and the basis. Given the basis, we can depict a difficulty function as an arrow in $\mathbb{R}^n$, where the so–called *usual basis* in $\mathbb{R}^n$ represents our yet to be defined basis[7]. The difficulty function's "arrow–head" lies at the point specified by the components of the difficulty function, and the "arrow–tail" is hinged at the origin. Now, by definition, difficulties are probabilities. In particular, they take on values in the *closed unit interval* – the inclusive interval from 0 to 1 on the real number line. This means the arrows representing difficulty functions must lie in the positive, n–dimensional unit hyperrectangle. This is the hyperrectangle formed by the related usual basis in $\mathbb{R}^n$ and, consequently, has a diagonal vector with all of its components equal to 1. This restriction means that the sum or difference of two difficulty functions is not necessarily a difficulty function, since such operations could result in vectors that lie outside the hyperrectangle. So, difficulty functions do not form a vector–space. Restrictions on how to approach certain extremisation problems involving difficulty functions occur as a result of this. In particular, *constrained extremisation* such as the kind involving a fixed demand space model $(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X)$ cannot rely simply on the bounds determined by rather general inequalities such as the Cauchy–

[6]A note on notation. In the LM model the symbol $\theta_A(x)$ has referred to a difficulty function related to either a channel $A$ or a methodology $A$. However, when performing vector calculations later in the thesis there will be a need to use this symbol for another purpose (see Section 4.5). Therefore, in the geometric model, the traditional LM difficulty function associated with $A$ is denoted instead by $\theta^*_A(x)$.

[7]It is always possible to relate a basis in a finite–dimensional, inner–product space with the *usual basis* in $\mathbb{R}^n$ (see Appendix A, pages 195 and 201). ***However, care must be taken in using diagrams resulting from such a correspondence since this correspondence says nothing about what the lengths of, and angles between, vectors are.*** Lengths and angles are defined by an inner–product, and unless the basis is *orthonormal* – where such orthogonality is defined according to a given inner–product – the representation of basis vectors as the *usual basis* in some $\mathbb{R}^n$ can be misleading. For instance, a basis vector of an $n$–dimensional inner–product space may have a length "5". However, when depicted as one of the *usual basis* vectors in $\mathbb{R}^n$, the length of the vector is "5" despite the *usual basis* vectors having a length "1" according to the *usual inner–product* in $\mathbb{R}^n$.

Schwarz inequality. Later on, beginning with Section 4.6 on page 102, we show how such problems may be approached.

So far we have suggested that difficulty functions may be depicted as vectors in a suitable $\mathbb{R}^n$. However, we have said nothing about notions of vector lengths, or angles between vectors. To properly define such notions we need to define a relevant ***inner–product*** (see Appendix A, page 201, for a definition of inner–product). The reader is encouraged not to assume that because we are depicting vectors in $\mathbb{R}^n$ the relevant inner–product is the *usual inner–product*[8] in $\mathbb{R}^n$. Instead, the relevant inner–product is the one suggested by the LM model. That is, suggested by the expected system *pfd*, which is an expectation of a product of difficulty functions. This is because such an expectation can be viewed as a bilinear, real–valued function of vectors that satisfies the properties of inner–products. We prove this as follows:

*Proof.* We have shown that difficulty functions, such as $\theta^*_A(X), \theta^*_B(X)$ and $\theta^*(X)$, are both random variables and vectors. This is true of any bounded, real–valued function of the demands: these form a vector–space. Furthermore, the measure–theoretic mathematical expectation can be viewed as a real–valued function defined on this vector–space. So, for real numbers $a, b$ we have

*Linearity*:

$$\begin{aligned}\underset{X}{\mathbb{E}}\left[\left(a\theta^*_A(X) + b\theta^*_A(X)\right)\theta^*(X)\right] &= \sum_{i=1}^{n}\left(a\theta^*_A(x_i) + b\theta^*_A(x_i)\right)\theta^*(x_i)\,\mathrm{P}_X(x_i) \\ &= a\sum_{i=1}^{n}\theta^*_A(x_i)\,\theta^*(x_i)\,\mathrm{P}_X(x_i) + b\sum_{i=1}^{n}\theta^*_B(x_i)\,\theta^*(x_i)\,\mathrm{P}_X(x_i) \\ &= a\,\underset{X}{\mathbb{E}}\left[\theta^*_A(X)\,\theta^*(X)\right] + b\,\underset{X}{\mathbb{E}}\left[\theta^*_B(X)\,\theta^*(X)\right].\end{aligned}$$

*Symmetry*:

$$\begin{aligned}\underset{X}{\mathbb{E}}\left[\theta^*_A(X)\,\theta^*_B(X)\right] &= \sum_{i=1}^{n}\theta^*_A(x_i)\,\theta^*_B(x_i)\,\mathrm{P}_X(x_i) \\ &= \sum_{i=1}^{n}\theta^*_B(x_i)\,\theta^*_A(x_i)\,\mathrm{P}_X(x_i) \\ &= \underset{X}{\mathbb{E}}\left[\theta^*_B(X)\,\theta^*_A(X)\right].\end{aligned}$$

*Positive–definiteness*:

$$\underset{X}{\mathbb{E}}\left[\theta^*_A(X)\,\theta^*_A(X)\right] = \underset{X}{\mathbb{E}}\left[\left(\theta^*_A(X)\right)^2\right] = \sum_{i=1}^{n}\left(\theta^*_A(x_i)\right)^2\mathrm{P}_X(x_i),$$

which as a finite sum of nonnegative numbers is non-negative. In particular, the sum will

[8]Given an arbitrary pair of vectors $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ in $\mathbb{R}^n$, where $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{R}$, the *usual inner–product* has the computational form $\sum_{i=1}^{n} a_i b_i$.

equal zero if, and only if, $\theta_A^*(x) = 0$ whenever $\mathrm{P}_X(x) \neq 0$. This is the requirement that the difficulty function has the value zero ***almost everywhere***: the set of demands on which the difficulty takes on non-zero values has probability zero. Recall, for our purposes $\mathrm{P}_X(x) > 0$ for all $x$ (see Section 2.2). Therefore, the expectation is equal to zero if, and only if, the difficulty function has a zero value everywhere. ■

So, based on this observation, we can write out the computational form of the relevant inner–product. Before we do this we introduce the angular brackets notation, $\langle\ ,\ \rangle$, for the inner–product. Given the vectors $\theta_A^*$ and $\theta_B^*$, we write the inner–product of the vectors as $\langle\theta_A^*, \theta_B^*\rangle$. So, the computational form of $\langle\theta_A^*, \theta_B^*\rangle$, in the basis where the components of $\theta_A^*$ and $\theta_B^*$ are difficulties, is

$$\langle\theta_A^*, \theta_B^*\rangle = \underset{X}{\mathbb{E}}\left[\theta_A^*(X)\,\theta_B^*(X)\right] = \sum_{i=1}^{n} \theta_A^*(x_i)\,\theta_B^*(x_i)\,\mathrm{P}_X(x_i) = \sum_{i=1}^{n} \theta_A^* i \theta_B^* i \mathrm{P}_X(x_i). \quad (4.1)$$

It is interesting to scrutinize the form of the sum $\sum_{i=1}^{n} \theta_A^*(x_i)\,\theta_B^*(x_i)\,\mathrm{P}_X(x_i)$ in Eq. (4.1) above, question how such a form may arise and, thereby, define the basis in which vectors of interest have difficulties as their components. If we denote the basis we wish to define as

$$\mathcal{S} := \left\{\bar{P}_1, \ldots, \bar{P}_n\right\}\ , \quad (4.2)$$

we require that each vector in the $n$–dimensional vector–space can be written as a unique linear combination of the $n$, linearly independent vectors $\bar{P}_1, \ldots, \bar{P}_n$. That is, for arbitrary vector $\theta_A^*$ we may write

$$\theta_A^* = \sum_{i=1}^{n} \theta_A^* i \bar{P}_i$$

Assume that $\mathcal{S}$ is an *orthogonal basis* so that, in particular, $\left\langle\bar{P}_i, \bar{P}_j\right\rangle = 0$ for $i \neq j$. Then, from the properties of an inner–product, we require that

$$\langle\theta_A^*, \theta_B^*\rangle = \left\langle \sum_{i=1}^{n} \theta_A^* i \bar{P}_i, \sum_{i=1}^{n} \theta_B^* i \bar{P}_i \right\rangle = \sum_{i=1}^{n} \theta_A^* i \theta_B^* i \left\langle\bar{P}_i, \bar{P}_i\right\rangle \quad (4.3)$$

which is the same form as Eq. (4.1). This suggests that we identify each $\mathrm{P}_X(x_i)$ term with $\left\langle\bar{P}_i, \bar{P}_i\right\rangle$. At this point it is useful to note that the inner–product can be used to define the magnitude of a vector (See Appendix Section A.2).

**Definition 4.3.1.** *Given a vector, $\bar{V}$, the magnitude of the vector, $\|\bar{V}\|$, is defined as*

$$\|\bar{V}\| := \sqrt{\langle\bar{V}, \bar{V}\rangle}.$$

So, for each $\bar{P}_i$, we have the relationship

$$\left\|\bar{P}_i\right\| = \sqrt{\left\langle \bar{P}_i, \bar{P}_i \right\rangle} = \sqrt{\mathrm{P}_X\left(x_i\right)} \tag{4.4}$$

From the foregoing, we define the action of the inner–product on the basis vectors as

$$\left\langle \bar{P}_i, \bar{P}_j \right\rangle = \begin{cases} \mathrm{P}_X\left(x_i\right), & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

for $i, j = 1, \ldots, n$ and $i \neq j$. In this sense the magnitudes of the vectors are induced by the demand profile: that is, specifying the demand profile is a specification of the sizes of these basis vectors. Closely related to this basis is an *orthonormal* basis which we define as follows. Via the inner–product each $\bar{P}_i$ has an associated unit vector[9], $\hat{P}_i$, such that $\bar{P}_i = \left\|\bar{P}_i\right\| \hat{P}_i$. Therefore, we define the orthonormal basis

$$\hat{\mathbb{S}} := \left\{\hat{P}_1, \ldots, \hat{P}_n\right\}. \tag{4.6}$$

We are now in a position to define the family of vector–spaces, with associated inner–products, that will be suitable to model relationships between difficulty functions and the demand profile. This is captured in the following postulate.

**Postulate 4.3.2.** *There exists a finite–dimensional vector–space, an inner–product defined with respect to the vector–space and a basis for the vector–space such that*

- *each difficulty function defines the components, in the basis, of a unique vector in a subset of the space;*
- *for a pair of vectors, labelled $\theta_A^*$ and $\theta_B^*$, in the subset of the vector–space the following definition holds.*

$$\left\langle \theta_A^*, \theta_B^* \right\rangle = P\begin{pmatrix} \textit{A randomly chosen program pair fails} \\ \textit{on a randomly occurring demand} \end{pmatrix},$$

*where the probability is computed using the corresponding difficulty functions and the demand profile.*

Any finite-dimensional vector–space with an associated inner–product that has the properties of this postulate is sufficient for our modelling needs. Such a space contains vectors that correspond, uniquely, to difficulty functions. That is, each difficulty function defines the components of a unique vector. ***Accordingly, we refer to these vectors as difficulty functions***. The lengths of these vectors, which are scalars, are probabilities (expected system *pfd*s). In fact, these vectors and the geometric relationships between them (that is, all the properties defined by the inner–product including angles and lengths) interact non-trivially to give probabilities (expected system *pfd*s). It turns out that, because of the properties of the inner–product, the postulate implies that there is a unique inner–product

[9]Given any vector, $\bar{V}$ say, we shall denote the unit vector in the same direction of $\bar{V}$ as $\hat{V}$.

defined on the vector–space that is suitable for our purposes. This is because once the behaviour of the inner–product is defined for the subset of vectors that are "picked out" by the difficulty functions, and the basis in the postulate is used, then the behaviour of the inner–product on any vector in the vector–space is defined. If in addition one recalls that for any given positive integer, $n$, all $n$–dimensional vector–spaces are equivalent (see Section A.1), then we may take the view point that we are working with only one vector–space and one associated inner–product. Thus, Postulate 4.3.2 defines a unique geometric model of coincident failure in multi–version software.

Given a vector–space, such as one from the family of vector–spaces postulated in Postulate 4.3.2, we are free to choose different bases to represent vectors in the space as n–tuples. However, each of these bases lie in the same vector–space. Consequently, while the $n$–tuple representation of each vector in the space will change from basis to basis, the vectors themselves are not changing: that is, there is an underlying vector–space that is fixed. In this underlying vector–space, vectors of a given length remain the same length, and any pair of vectors with a given angle between them should not be re–oriented as a consequence of a change in basis. We expect, therefore, that any statement in terms of the inner–product should be basis invariant. What does this mean in actual computations? One way in which a basis is useful is it allows computations to be carried out with vectors (see Sections A.1 and A.2). In fact, a choice of basis determines the computational form of the inner–product. For instance, formulae for calculating expected system *pfd*s in the LM model are the computational form of an inner–product in a given basis (see Eq. (4.1)). Other computational forms are possible given a different choice of basis vectors. To obtain a particular computational form simply substitute the basis representations for an arbitrary pair of vectors into the inner–product. For example, for an $n$–dimensional vector–space $\mathbf{V}$ with defined inner–product $\langle\ ,\ \rangle$, basis choice $\{\bar{b}_1, \ldots, \bar{b}_n\}$, and vectors $U = (U1)\bar{b}_1 + \ldots + (Un)\bar{b}_n \in \mathbf{V}$ and $W = (W1)\bar{b}_1 + \ldots + (Wn)\bar{b}_n \in \mathbf{V}$, the computational form of the inner–product is given as

$$\langle U, W\rangle = \langle (U1)\bar{b}_1 + \ldots + (Un)\bar{b}_n\ ,\ (W1)\bar{b}_1 + \ldots + (Wn)\bar{b}_n\rangle = \sum_{i=1,j=1}^{n,n} (Ui)(Wj)\langle \bar{b}_i, \bar{b}_j\rangle. \tag{4.7}$$

However, note that *the inner–product does not change under a change of basis*: its value on a given pair of vectors (that is, the far left of Eq. (4.7)) is the same, irrespective of what basis is used to describe the vectors in the space (and, consequently, change the computational form on the far right of Eq. (4.7)). This makes sense since one of our requirements for the inner–product is that it defines absolute notions of length and angles for a given vector–space[10].

Here are some examples of how reliability measures, defined in the LM model, are related to their equivalent geometric expressions. Consider the vector $\bar{P}$ which is defined as follows:

$$\bar{P} := \bar{P}_1 + \ldots + \bar{P}_n$$

[10]For a more detailed discussion of the relationship between inner–products, lengths and angles please see Appendix Section A.2.

$$= \|\bar{P}_1\| \hat{P}_1 + \ldots + \|\bar{P}_n\| \hat{P}_n$$
$$= \sqrt{\mathrm{P}_X(x_1)}\hat{P}_1 + \ldots + \sqrt{\mathrm{P}_X(x_n)}\hat{P}_n,$$

where we have used the relationship between the bases $\mathcal{S}$ and $\hat{\mathcal{S}}$, and we also used Eq. (4.4). This demonstrates that the vector $\bar{P}$, with respect to the basis $\hat{\mathcal{S}}$, has components $\left(\sqrt{\mathrm{P}_X(x_1)}, \ldots, \sqrt{\mathrm{P}_X(x_n)}\right)$. But, this is an $n$–tuple that is completely defined by the demand profile $(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X)$. Consequently, ***we refer to the vector $\bar{P}$ as the demand profile***, to emphasize the point that ***in the basis $\hat{\mathcal{S}}$ this vector has components that are completely defined by the demand profile.*** Then, the expected single–version *pfd*, $q_A$, is given by

$$\langle \theta_A^*, \bar{P} \rangle = \left\langle \sum_{i=1}^{n} \theta_A^* i \bar{P}_i, \sum_{j=1}^{n} \bar{P}_j \right\rangle = \sum_{i=1}^{n} \theta_A^* i \langle \bar{P}_i, \bar{P}_i \rangle = \mathbb{E}\left[\theta_A^*(X)\right] = q_A. \tag{4.8}$$

Observe that $\bar{P}$ is a unit vector, since the magnitude of the vector is 1. That is,

$$\|\bar{P}\|^2 = \langle \bar{P}, \bar{P} \rangle = \sum_{i=1}^{n} \mathrm{P}_X(x_i) = 1. \tag{4.9}$$

Consequently, the following theorem holds.

**Theorem 4.3.3.** *The expected single–version* pfd *is the magnitude of the projection of the relevant difficulty function onto the demand profile.*

We shall explore this relationship further when discussing planes. Similarly, the expected system *pfd*, $q_{AA}$, is given by

$$\langle \theta_A^*, \theta_A^* \rangle = \left\langle \sum_{i=1}^{n} \theta_A^* i \bar{P}_i, \sum_{j=1}^{n} \theta_A^* j \bar{P}_j \right\rangle = \sum_{i=1}^{n} (\theta_A^* i)^2 \langle \bar{P}_i, \bar{P}_i \rangle = \mathbb{E}\left[\theta_A^*(X)^2\right] = q_{AA}. \tag{4.10}$$

In a natural way, from this result we may define the "size of a given difficulty function", say $\theta_A^*$, to be $\|\theta_A^*\| = \sqrt{\langle \theta_A^*, \theta_A^* \rangle} = \sqrt{q_{AA}}$.

Let us briefly recap what has been done so far. In this section we began making a connection between vector–space mathematics and the LM model, by defining difficulty functions as n–dimensional vectors. These vectors lie in an appropriately bounded region of the non-negative region in $\mathbb{R}^n$. In the same breadth we also defined the inner–product, induced by the demand profile, that is relevant for the difficulty functions. These are the first steps to having a framework for the extremisation of difficulty functions. However, the basis $\mathcal{S}$ suggested by the LM model is orthogonal but not orthonormal. As a result, the computational form of the inner–product is not a *canonical form*: it still has weighting terms contained within it. For example, in Eq. (4.1) or Eq. (4.3), there are terms such as $\mathrm{P}_X(x_i)$ or $\langle \bar{P}_i, \bar{P}_i \rangle$. For our purposes canonical forms are the preferred forms to perform calculations in. In order to achieve a canonical form we need to perform a change of basis;

from our current basis to one that has orthogonal, unit vectors. This is the focus of the following section.

## 4.4 Transformation between Bases

Upon using the orthogonal basis $\mathcal{S}$, the inner–product suggested by the LM model has a computational form[11] which contains weighting terms that are the probabilities of demands submitted to the system in operation. In this sense this computational form can be said to be induced by the demand profile. We can transform this into a canonical[12] computational form by a change from an orthogonal basis into one that is orthonormal (see Section A.3). Since the current basis $\mathcal{S}$ is already an orthogonal one[13], in order to define a related orthonormal basis $\hat{\mathcal{S}}$ we only need to use the unit vectors in the same directions as the basis vectors. In this section we define an invertible linear transformation that accomplishes this change of basis – converting components expressed in terms of the orthogonal basis $\mathcal{S}$ to components in terms of the orthonormal basis $\hat{\mathcal{S}}$. The change of basis transformation is simple to define since, by definition, we can relate the basis vectors to their associated unit vectors by $\bar{P}_i = \left\|\bar{P}_i\right\| \hat{P}_i$. From the definition of a basis an arbitrary vector, say $\theta_A^*$, may be written as a unique linear combination of the basis vectors. Recall this defines the components of $\theta_A^*$ with respect to $\mathcal{S}$. From this we can obtain a similar, unique[14], linear combination of the unit vectors as follows:

$$\theta_A^* = \sum_{i=1}^{n} \theta_A^* i \bar{P}_i = \sum_{i=1}^{n} \theta_A^* i \left(\left\|\bar{P}_i\right\| \hat{P}_i\right) = \sum_{i=1}^{n} \left(\theta_A^* i \left\|\bar{P}_i\right\|\right) \hat{P}_i = \sum_{i=1}^{n} \theta_A i \hat{P}_i,$$

where $\theta_A i = \theta_A^* i \left\|\bar{P}_i\right\|$. This defines the components of $\theta_A^*$ in terms of the basis $\hat{\mathcal{S}}$. Even though this change of basis transforms the computational form of the inner–product it does not change the inner–product since, by definition, the inner–product imbues the vector–space with an invariant notion of length. In terms of vector components, there is a relationship between the change of basis we are discussing and the demand profile. To make this relationship apparent, recall that for each $i = 1, \ldots, n$ we have $\left\|\bar{P}_i\right\| = \left\langle \bar{P}_i, \hat{P}_i \right\rangle = \sqrt{\langle \bar{P}_i, \bar{P}_i \rangle} = \sqrt{\mathrm{P}_X(x_i)}$. If we use the shorthand $P_i := \mathrm{P}_X(x_i)$, then

$$\theta_A i = \theta_A^* i \sqrt{P_i}. \tag{4.11}$$

To illustrate the consequences of the basis change, observe two depictions of the same vector–space in Fig. 4.2 and Fig. 4.3. In this example we have defined difficulty functions on a space of 3 demands. Consequently, both depictions of the vector–space contain 3 axes; each axis associated with a given demand. The only difference between the two diagrams is

[11] That is, the sum on the far right–hand–side in Eq. (4.1), which is a result of using the orthogonal basis $\mathcal{S}$ in the inner–product of Postulate 4.3.2

[12] "Canonical" in the sense that the computational form of the inner–product is the same as *the usual Euclidean inner–product.*

[13] One consequence of this is no cross terms appear in the computational form of the inner–product.

[14] Uniqueness follows from $\hat{\mathcal{S}}$ being a basis and, therefore, having linearly independent vectors.

the choice of basis used to describe the vector–space: Fig. 4.2 uses the basis $\mathcal{S}$ to describe the basis $\hat{\mathcal{S}}$, while the relationship is reversed in Fig. 4.3. Recall, in the basis $\mathcal{S}$, vectors of interest have components that are the difficulty functions defined in the EL and LM models. However,

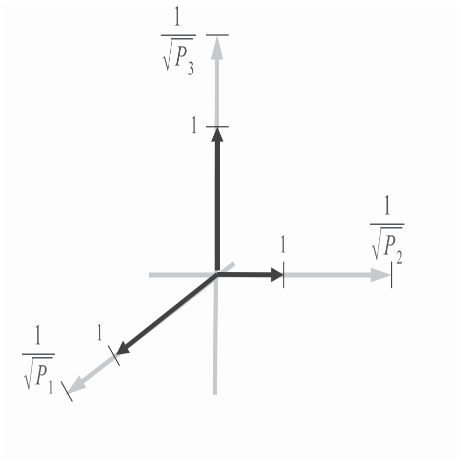


**Figure 4.2:** A depiction of the basis $\hat{\mathcal{S}}$, in terms of the basis $\mathcal{S}$. The components and "lengths" in the diagram are defined with respect to $\mathcal{S}$. For instance, each basis vector in $\mathcal{S}$ is depicted as a dark coloured arrow with relative length "1" since no basis vectors other than itself completely defines it. The basis vectors in $\hat{\mathcal{S}}$ are depicted as light coloured arrows and also lie on the axes, but are relatively larger than the corresponding vectors in $\mathcal{S}$. With respect to the basis $\mathcal{S}$ the coordinates of the vectors in $\hat{\mathcal{S}}$ are $\left(\frac{1}{\sqrt{P_1}}, 0, 0\right), \left(0, \frac{1}{\sqrt{P_2}}, 0\right),$ and $\left(0, 0, \frac{1}{\sqrt{P_3}}\right)$.

this choice of basis is such that with respect to the inner–product the basis vectors do not have unit length. Consequently, even though the basis vectors are represented as arrows parallel to the axes whose "arrow heads" touch the unit marks, these vectors are drawn having different lengths. Contrastingly, the orthonormal basis $\hat{\mathcal{S}}$ consists of unit vectors (unit, with respect to the inner–product) which do not touch the unit marks on the axes. Instead, they are seemingly "longer" with coordinates $\left(\frac{1}{\sqrt{P_1}}, 0, 0\right), \left(0, \frac{1}{\sqrt{P_2}}, 0\right),$ and $\left(0, 0, \frac{1}{\sqrt{P_3}}\right)$. The situation is changed in Fig. 4.3 where the basis $\hat{\mathcal{S}}$, consisting of unit vectors with respect to the inner-product, is used to describe the basis $\mathcal{S}$. Consequently, the basis vectors in $\mathcal{S}$ now have coordinates $\left(\sqrt{P_1}, 0, 0\right), \left(0, \sqrt{P_2}, 0\right),$ and $\left(0, 0, \sqrt{P_3}\right)$.

In a sense, the use of the inner–product in canonical form requires the use of transformed difficulty functions. For, via the transformation in Eq. (4.11), the difficulty function in the LM model, $\theta^*_A(x)$, becomes the "difficulty function" $\theta_A(x)$. As an illustration consider the "worst" possible difficulty function. This arises when for each demand a randomly chosen program is guaranteed to fail. This can also be viewed as the failure region for a version that fails on every demand; that is, the worst possible version is guaranteed to be produced by the development teams for deployment. If we stick with the case of 3 demands then this difficulty function has components $(1, 1, 1)$. This is the vector $\bar{P}$ and is depicted in Fig. 4.4. Here, the orthogonal basis vectors $\mathcal{S}$ are used. These have coordinates $(1, 0, 0), (0, 1, 0), (0, 0, 1)$, despite not being of equal lengths defined by the inner–product. Observe that these are the coordinates of "single versions" (this will be the phrase we use to refer to an equivalence

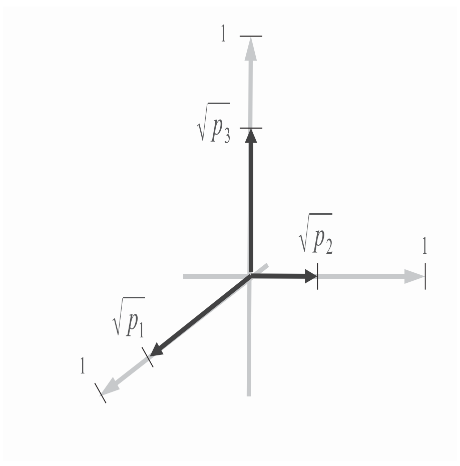


**Figure 4.3:** The vector–space depicted in Fig. 4.2, but depicted here with respect to the basis $\hat{\mathcal{S}}$. The components of the vectors in $\mathcal{S}$ are $\left(\sqrt{P_1}, 0, 0\right)$, $\left(0, \sqrt{P_2}, 0\right)$, and $\left(0, 0, \sqrt{P_3}\right)$.

class of programs that have the same failure regions) with failure regions consisting of single demands $x_1$, $x_2$ and $x_3$ respectively. More generally, in the basis $\mathcal{S}$, we shall call a ***single version*** any vector with each of its components equal to either 1 or 0. By using orthonormal

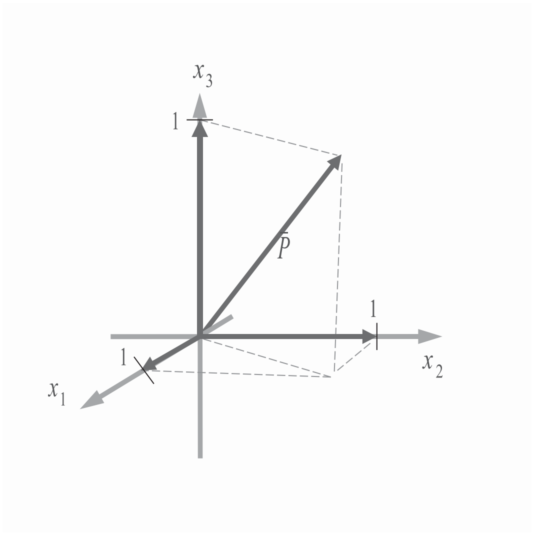


**Figure 4.4:** "Worst" version, $\bar{P}$, depicted in terms of the orthogonal basis $\mathcal{S}$. The vector $\bar{P}$ is a unit vector since its magnitude is 1 (see Eq. (4.9)).

basis $\hat{\mathcal{S}}$ instead we obtain Fig. 4.5. Under this transformation the "single versions" now reflect their proper sizes; versions that are more likely to fail appear longer.

The change of basis is a linear transformation; this follows from two facts. The components of a sum of vectors, under the change of basis, is equal to the sum of the components of each vector, under the change of basis. Also, the components of a scaled vector, under the change of basis, are equal to a scaling by the same amount of the components of the unscaled vector, under the change of basis. Therefore, the example in Fig.s 4.4 and 4.5

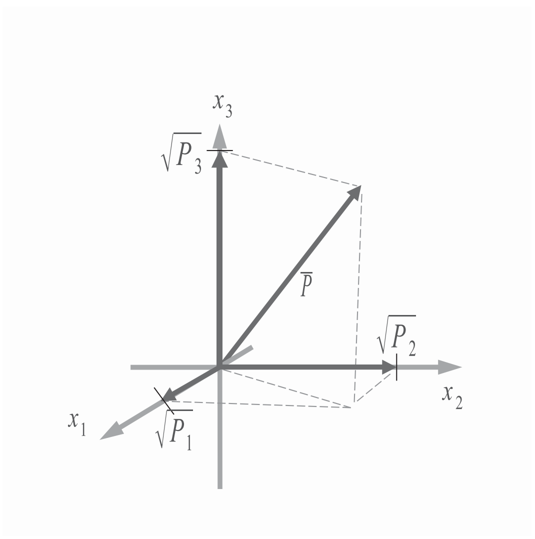


**Figure 4.5:** "Worst" version, $\bar{P}$, depicted in terms of orthonormal basis $\hat{\mathbb{S}}$.

illustrate the effect of the invertible, linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ such that for each $\left(\theta^*_B 1, \theta^*_B 2, \ldots, \theta^*_B n\right) \in \mathbb{R}^n$ the image under $T$ of this vector is given by the matrix product

$$\begin{pmatrix} \sqrt{P_1} & 0 & \ldots & 0 \\ 0 & \sqrt{P_2} & \ldots & 0 \\ \vdots & \ddots & \ldots & \vdots \\ 0 & 0 & \ldots & \sqrt{P_n} \end{pmatrix} \begin{pmatrix} \theta^*_B 1 \\ \theta^*_B 2 \\ \vdots \\ \theta^*_B n \end{pmatrix} = \begin{pmatrix} \theta_B 1 \\ \theta_B 2 \\ \vdots \\ \theta_B n \end{pmatrix} \in \mathbb{R}^n,$$

where the matrix above is an invertible[15] matrix uniquely representing T. Under this transformation the constraints on the difficulty functions – requiring each component of a difficulty to lie inclusively between 0 and 1 – are preserved.[16]

$$\begin{matrix} 0 \le \\ 0 \le \\ \vdots \\ 0 \le \\ 0 \le \end{matrix} \begin{pmatrix} \theta^*_B 1 \\ \theta^*_B 2 \\ \ldots \\ \theta^*_B (n-1) \\ \theta^*_B n \end{pmatrix} \begin{matrix} \le 1 \\ \le 1 \\ \vdots \\ \le 1 \\ \le 1 \end{matrix} \quad \equiv \quad \begin{matrix} 0 \le \\ 0 \le \\ \vdots \\ 0 \le \\ 0 \le \end{matrix} \begin{pmatrix} \theta_B 1 \\ \theta_B 2 \\ \ldots \\ \theta_B (n-1) \\ \theta_B n \end{pmatrix} \begin{matrix} \le \sqrt{P_1} \\ \le \sqrt{P_2} \\ \vdots \\ \le \sqrt{P_{n-1}} \\ \le \sqrt{P_n} \end{matrix}$$

Because $T$ is invertible there exists a related linear, inverse transformation for which similar relationships hold. So, $T^{-1} : \mathbb{R}^n \to \mathbb{R}^n$ is an invertible, linear transformation such that for each $\left(\theta_B 1, \theta_B 2, \ldots, \theta_B n\right) \in \mathbb{R}^n$ the image under $T^{-1}$ of this vector is given by the matrix

[15] The matrix is guaranteed to have an inverse since there are no null demands and, as such, $\mathrm{P}_X(x) > 0$ for all demands $x$.

[16] As a "side note" the matrix representation of this linear transformation is in canonical form – *eigenvalues* are the diagonal elements and all other entries are zero – and the basis vectors are *eigenvectors*. Consequently, a proof of its invertibility is given by observing that there are no zero eigenvalues and, therefore, the *Kernel* of this transformation contains only the zero vector, $\bar{0}$. Equivalently, the determinant of the matrix is non–zero.

product

$$\begin{pmatrix} \frac{1}{\sqrt{P_1}} & 0 & \dots & 0 \\ 0 & \frac{1}{\sqrt{P_2}} & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ 0 & 0 & \dots & \frac{1}{\sqrt{P_n}} \end{pmatrix} \begin{pmatrix} \theta_B 1 \\ \theta_B 2 \\ \vdots \\ \theta_B n \end{pmatrix} = \begin{pmatrix} \theta^*_B 1 \\ \theta^*_B 2 \\ \vdots \\ \theta^*_B n \end{pmatrix} \in \mathbb{R}^n,$$

where the matrix is an invertible matrix uniquely representing $T^{-1}$.

We have demonstrated that the inner–product of an arbitrary pair of vectors does not change under transformation $T$. Therefore, angles and lengths are preserved. This means that any statement about lengths and angles are independent of the basis used: the statements involve only the invariant inner–product, and thus are independent of coordinate system. Now, since difficulty functions are modelled as vectors and expected *pfd*s are the lengths of appropriate vectors, we propose the following steps for approaching extremisation problems involving difficulty functions and expected *pfd*'s:

1. Transform vector components, from being in terms of $\mathcal{S}$ to $\hat{\mathcal{S}}$, using $T$ so that the computational form of the inner–product is the "nice" canonical one. In doing so, angles and vector lengths will appear correct and consistent when depicted. In addition, the extremisations are easier to carry out since all of the information pertaining to lengths and angles is contained in the vector components;
2. Perform extremisation on difficulty functions. This is achieved via geometric manipulations/operations, and results in the extreme value for some objective function. This extreme value will be in terms of either difficulty function components with respect to $\hat{\mathcal{S}}$ or invariant angles/lengths;
3. If necessary transform vector components back, from $\hat{\mathcal{S}}$ to $\mathcal{S}$, using $T^{-1}$.

The diagonal vector, $\bar{P}$, defines the bounding region for the difficulty functions. There are two interpretations of $\bar{P}$, depending on whether the basis $\mathcal{S}$ or $\hat{\mathcal{S}}$ is being used, that will be useful in discussion:

1. the components of $\bar{P}$, in the basis $\mathcal{S}$, can be viewed as the failure set for the "worst" software version (or the "worst" difficulty function possible); the version that fails on all $n$ demands. In a similar spirit, each axis represents the failure set for a version that fails only on a unique demand. Each diagonal formed by 2 of these "axis" single versions represents a version that fails exclusively on a related pair of unique demands. Further still, each diagonal formed by a unique triplet of "axis" single versions is a single version that fails exclusively on a related triplet of unique demands. This "game" can continue to be played until one arrives at the "worst" version, $\bar{P}$. We note briefly that the origin, $\bar{0}$, is the perfect version: the version that does not fail on any demand. Naturally, therefore, an ***imperfect single version*** is a single version that is not the perfect version.
2. "correct" depictions of vector lengths and angles (that is, "correct" according to some

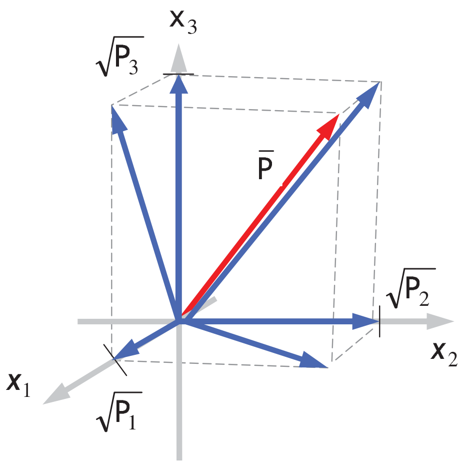


**Figure 4.6:** For a demand space of size 3 there are $2^3$ possible "single versions"/"failure regions", all of which are depicted in this figure. 3 vectors, each of which lie parallel to some unique axes, represent failures on single demands. 3 vectors that are parallel to the diagonals in 2-dimensional planes (sides of the hyperrectangle) represent failures on pairs of demands. 1 vector parallel to the diagonal of the hyperrectangle represents failure on all 3 demands and the origin represents no failure on any demand.

inner–product) can be made by using the basis $\hat{\mathbb{S}}$. For, in this basis, the "importance" placed on each demand by the demand profile is reflected by the length of vectors: bad difficulty functions appear long (the longest of these is $\bar{P}$) and "good" difficulty functions appear short (the shortest of these being the origin, $\bar{0}$). In particular, in the basis $\hat{\mathbb{S}}$, $\bar{P}$'s components are completely defined by the demand profile. Therefore, as shown in Eq. (4.9), the size of $\bar{P}$ is 1; the largest possible size for a difficulty function.

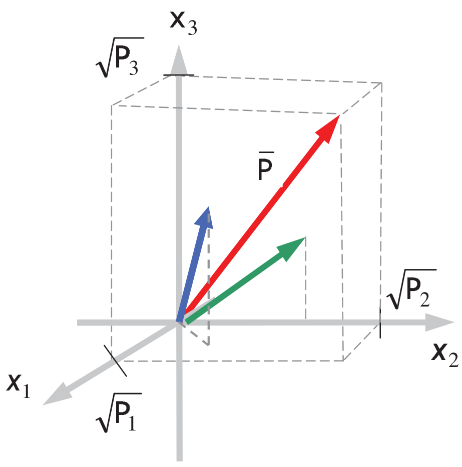


**Figure 4.7:** Given a demand profile the cuboid depicted above represents the region in the 3-dimensional space where each possible difficulty function has a unique vectorial representation. All difficulty functions must lie within this region. The diagonal of the cuboid is completely defined by the demand profile and has coordinates $\left(\sqrt{P_1}, \sqrt{P_2}, \sqrt{P_3}\right)$.

In both viewpoints the bounding region for the difficulty functions is defined by $\bar{P}$: any vector lying on, or within, this bounding region is a difficulty function. Subspaces may be formed by choosing a collection of difficulty functions and, subsequently, forming a set of all possible linear combinations of these vectors. For example, each single version spans a subspace and, in particular, each imperfect single version spans a one–dimensional subspace. This behaviour is mimicked in higher-dimensions so that each pair of distinct, imperfect single versions forms a 2–dimensional subspace, and each triplet of distinct, imperfect single versions forms a 3–dimensional subspace, and so on. While the linear combination of any collection of difficulty functions spans some subspace, the only vectors in that subspace that will be of interest must lie within $\bar{P}$'s bounding box. This requirement can bring added complexity to the extremisation of difficulty functions, as such a task may necessarily require analyses of various, intersecting higher-dimensional surfaces.

Furthermore, *the imperfect single versions are the only difficulty functions with magnitudes equal to cosines of the angles they make with the "demand profile"*, $\bar{P}$. Equivalently, the imperfect single versions are the projections of the demand profile, $\bar{P}$, onto each of the subspaces defined by linear combinations (using equal weights) of unit vectors parallel to the axes. This point will be discussed in more detail later, as it is very important for extremisation.

In this section we have successfully defined a change of basis that ensures the inner–product of any pair of vectors be calculated as the sum of the product of respective vector components. ***From now on we shall assume that, via*** $T$, ***we are working in the transformed basis*** $\hat{\mathcal{S}}$ ***and not the LM model basis*** $\mathcal{S}$; whenever we need to we will transform back to $\mathcal{S}$ via $T^{-1}$. In the final section of this chapter we draw attention to operational aspects of our geometric coincident failure model, detailing the kinds of vector manipulations required for extremisation.

## 4.5 A Brief Clarification On Notation

The symbols $\theta_A$ and $\theta^*_A$ are vectors. For this thesis these symbols refer to the same vector: the vector that in the basis $\mathcal{S} = \left\{\bar{P}_1, \ldots, \bar{P}_n\right\}$ has the difficulties $(\theta^*_A 1, \ldots, \theta^*_A n)$ as its components (These are the difficulties from the LM model). Therefore, we may replace $\theta^*_A$ with $\theta_A$, and vice–versa, anywhere in the geometric models. On the other hand $\theta_A i$ and $\theta^*_A i$ are the $i$th components of a single vector (the vector denoted by either $\theta_A$ or $\theta^*_A$) with respect to the bases $\hat{\mathcal{S}} = \left\{\hat{P}_1, \ldots, \hat{P}_n\right\}$ and $\mathcal{S} = \left\{\bar{P}_1, \ldots, \bar{P}_n\right\}$ respectively. Consequently, $\theta_A i \neq \theta^*_A i$ in general. Therefore, coordinate independent expressions (expressions in which the components of a vector do not appear) can have $\theta^*_A$ replaced by $\theta_A$ in them without changing their meaning. For example, the notation for the inner–product $\left\langle\ ,\ \right\rangle$ acting on a pair of vectors, say $\theta_A$ and $\bar{P}$, does not involve coordinates. This means that

$$\left\langle \theta^*_A, \bar{P} \right\rangle \equiv \left\langle \theta_A, \bar{P} \right\rangle$$

is a perfectly legitimate tautology: it's always true. Furthermore, because $\theta_A^* \equiv \theta_A$ the following is also true.

$$\theta_A^* = \sum_{i=1}^{n} (\theta_A^* i)\, \bar{P}_i = \sum_{j=1}^{n} \left(\theta_A j\right) \hat{P}_j = \theta_A. \tag{4.12}$$

Usually, when the basis $\mathcal{S}$ is used, the vector is denoted as $\theta_A^*$. If the basis $\hat{\mathcal{S}}$ is used instead, the vector is denoted as $\theta_A$. The context will always be made clear concerning which choice of basis is being used. However, Eq. (4.12) demonstrates that it is perfectly legitimate to violate this rule when necessary.

## 4.6 Extremisation via Angles, Magnitudes and Planes

In this section we discuss relationships between vectors that will be useful for the proofs on bounds in Chapter 5. These relationships are consequences of the inner–product being defined as the signed magnitude of the projection of a vector. That is, for an arbitrary vector $\theta_B$, and an arbitrary non–zero vector $\theta_A$ with associated unit vector $\hat{\theta}_A$, $\left\langle \theta_B, \hat{\theta}_A \right\rangle$ is the "amount" of vector $\theta_B$ in the direction of $\theta_A$. We can write this in a slightly alternative form. Recall, for a given non-zero vector, $\theta_A$, the relationship $\theta_A = \|\theta_A\|\, \hat{\theta}_A$ must hold. Therefore, $\left\langle \theta_B, \hat{\theta}_A \right\rangle = \left\langle \theta_B, \frac{\theta_A}{\|\theta_A\|} \right\rangle$. The linear transformation $\left\langle \;, \frac{\theta_A}{\|\theta_A\|} \right\rangle$ defines lengths of projections on the subspace spanned by the $\theta_A$ difficulty. We can depict this using an infinite collection of lines orthogonal to $\theta_A$. The special case of $\theta_A$ being a basis vector, say $\hat{P}_i$, results in a method for determining the $i$th component of an arbitrary vector $\theta_B$ since

$$\left\langle \theta_B, \hat{P}_i \right\rangle = \left\langle \sum_{j=1}^{n} \theta_B j \hat{P}_j, \hat{P}_i \right\rangle = \sum_{j=1}^{n} \theta_B j \left\langle \hat{P}_j, \hat{P}_i \right\rangle = \theta_B i \left\langle \hat{P}_i, \hat{P}_i \right\rangle = \theta_B i. \tag{4.13}$$

So, as expected, the magnitude of the projection of an arbitrary vector with respect to an orthonormal basis vector is the component of the arbitrary vector with respect to the basis vector. If, instead, we consider the inner–product with respect to the single versions that fail on only one demand – the vectors $\mathcal{S} = \left\{\bar{P}_1, \ldots, \bar{P}_n\right\}$ – we obtain

$$\begin{aligned}\langle \theta_B, \bar{P}_i \rangle &= \left\langle \sum_{j=1}^{n} \theta_B j \hat{P}_j, \left\|\bar{P}_i\right\| \hat{P}_i \right\rangle = \sum_{j=1}^{n} \theta_B j \left\|\bar{P}_i\right\| \left\langle \hat{P}_j, \hat{P}_i \right\rangle \\ &= \theta_B i \left\|\bar{P}_i\right\| \left\langle \hat{P}_i, \hat{P}_i \right\rangle \\ &= \theta_B i \left\|\bar{P}_i\right\|. \end{aligned} \tag{4.14}$$

Further still, we may consider projections with respect to single versions that fail on more than one demand. These single versions can be expressed as sums of single versions in $\mathcal{S}$. Consequently, inner–products between arbitrary difficulty functions and single versions will result in sums of terms like $\theta_B i \left\|\bar{P}_i\right\|$ in above. In particular, for the "worst" single version $\bar{P}$ we have

$$\begin{aligned}\left\langle\theta_B, \bar{P}\right\rangle &= \left\langle \sum_{j=1}^{n} \theta_B j \hat{P}_j, \sum_{i=1}^{n} \left\|\bar{P}_i\right\| \hat{P}_i \right\rangle = \sum_{j=1,i=1}^{n} \theta_B j \left\|\bar{P}_i\right\| \left\langle \hat{P}_j, \hat{P}_i \right\rangle \\ &= \sum_{i=1}^{n} \theta_B i \left\|\bar{P}_i\right\| \left\langle \hat{P}_i, \hat{P}_i \right\rangle \\ &= \sum_{i=1}^{n} \theta_B i \left\|\bar{P}_i\right\| .\end{aligned}$$

But, Eq. (4.8) on page 94 describes the magnitude of projections of difficulty functions onto the demand profile, $\bar{P}$, as the expected single version *pfd* related to the difficulty. Therefore,

$$q_B = \left\langle\theta_B, \bar{P}\right\rangle = \sum_{i=1}^{n} \theta_B i \left\|\bar{P}_i\right\| = \sum_{i=1}^{n} \theta_B i \sqrt{P_i}. \tag{4.15}$$

These observations are depicted in Fig. 4.8. Now, appreciate that $\bar{P}$ is a unit vector[17]. Hence, the values of the linear transformation $\left\langle\ , \bar{P}\right\rangle$ on the difficulty functions $\theta_A$ and $\theta_B$ are the distances from the origin, $q_A$ and $q_B$, in the direction of $\bar{P}$. These are the magnitudes

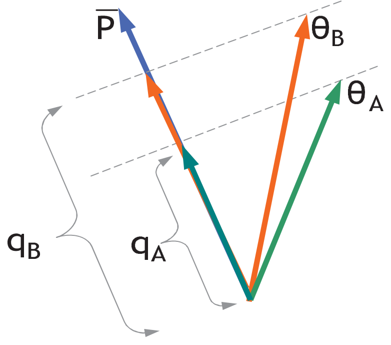


**Figure 4.8:** With respect to the demand profile, $\bar{P}$, the difficulty functions, $\theta_A$ and $\theta_B$, have related projections $q_A\bar{P}$ and $q_B\bar{P}$ respectively. The magnitudes of these projections are the expected *pfd*s related to the difficulties. These difficulty functions are not the only ones that result in these expected *pfd* values. Consequently, any set of difficulty functions that touch a dashed line orthogonal to $\bar{P}$ will have the same projection with respect to $\bar{P}$ and, thus, will share the same expected *pfd*. This figure also shows that the magnitude of a difficulty function is at least as large as the size of the difficulty function's projection in the direction of $\bar{P}$. That is, for an arbitrary difficulty function, $\theta_A$ say, it is the case that $\sqrt{q_{AA}} = \|\theta_A\| \geq \left\langle\theta_A, \bar{P}\right\rangle = q_A$.

of the projections $q_A\bar{P}$ and $q_B\bar{P}$, respectively. Later on we show that $\left\langle\ , \bar{P}\right\rangle$ defines sets of difficulty functions that share a common expected *pfd*. These are *equivalence classes* of difficulty functions and, pictorially, they are subsets of planes in $\mathbb{R}^n$ orthogonal to $\bar{P}$.

The cosine between two unit vectors is given by the inner–product of the vectors[18]. So, given an arbitrary difficulty function $\theta_B$, $\left\langle\hat{\theta}_B, \bar{P}\right\rangle$ is the cosine of the angle between $\theta_B$ and

[17]Unsurprising, since $\bar{P}$ is the demand profile and, consequently, has components that sum to 1.
[18]See Eq. (A.12) on page 204.

$\bar{P}$. There is another useful form of the cosine. Let $\gamma_B$ be the angle $\theta_B$ makes with $\bar{P}$. Then

$$\cos\gamma_B = \left\langle \hat{\theta}_B, \bar{P} \right\rangle = \left\langle \frac{\theta_B}{\|\theta_B\|}, \bar{P} \right\rangle = \frac{1}{\|\theta_B\|} \left\langle \theta_B, \bar{P} \right\rangle = \frac{q_B}{\sqrt{q_{BB}}} \tag{4.16}$$

So, the cosine of the angle between a ***non–zero difficulty function***[19], say $\theta_B$, and the demand profile relates the expected single version *pfd* with the expected system *pfd* of a homogeneous, 1–out–of–2 system. Note that the size of the cosine is inversely related to the size of the angle $\gamma_B$: the larger the cosine the smaller the angle, and vice–versa. Therefore, for difficulty functions $\theta_A$ and $\theta_B$, we have $\cos^{-1}\left(\frac{q_B}{\|\theta_B\|}\right) \leq \cos^{-1}\left(\frac{q_A}{\|\theta_A\|}\right)$ if, and only if, $\frac{q_A}{\|\theta_A\|} \leq \frac{q_B}{\|\theta_B\|}$. This is shown in Fig. 4.9.

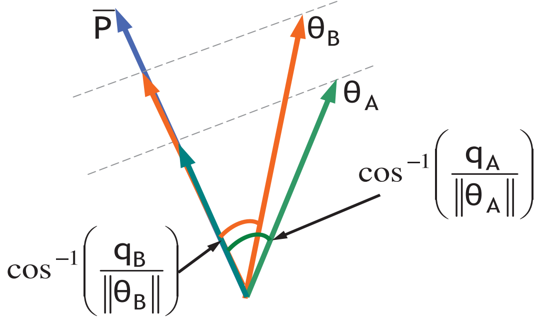

$\bar{P}$
$\theta_B$
$\theta_A$
$\cos^{-1}\left(\frac{q_A}{\|\theta_A\|}\right)$
$\cos^{-1}\left(\frac{q_B}{\|\theta_B\|}\right)$

**Figure 4.9:** Trigonometry justifies the definition of $\frac{q_A}{\|\theta_A\|}$ as the cosine of the angle $\theta_A$ makes with $\bar{P}$. The smaller the value of the cosine of the angle the larger the angle.

Notice that the smaller the value of the cosine, the larger $q_{BB}$ is relative to $q_B^2$. This implies that the angle, $\gamma_B$, encodes part of the same information as the variance of the difficulty function, $\underset{X}{\mathrm{Var}}\left(\theta_B(X)\right)$, since $\underset{X}{\mathrm{Var}}\left(\theta_B(X)\right) = q_{BB} - q_B^2$. This relationship is detailed in the following theorems that are a direct consequence of Eq. (4.16).

**Theorem 4.6.1.** *In a set of non–zero difficulty functions with the same mean the larger the angle a difficulty function makes with the demand profile the greater its variation.*

**Theorem 4.6.2.** *In a set of non–zero difficulty functions of the same size the larger the angle a difficulty function makes with the demand profile the greater its variation.*

For an arbitrary difficulty function, $\theta_B$ say, there is a vector, $\theta_B - q_B\bar{P}$, whose magnitude adequately captures notions of how far away $\theta_B$ is from the demand profile. We call this vector the *variance vector*.

[19]A non–zero difficulty function is any difficulty function other than the one modelled by the origin. That is, any difficulty function with some non–zero components. An implication of this is the difficulty function has a non–zero magnitude.

**Theorem 4.6.3.** *$\theta_B$ and $\bar{P}$ define a vector, $\theta_B - q_B\bar{P}$, orthogonal to $\bar{P}$ and with a magnitude equal to the standard deviation of $\theta_B$.*

*Proof.* $\theta_B - q_B\bar{P}$ is orthogonal to $\bar{P}$ since

$$\left\langle \theta_B - q_B\bar{P}, \bar{P}\right\rangle = \left\langle \theta_B, \bar{P}\right\rangle - \left\langle q_B\bar{P}, \bar{P}\right\rangle = \left\langle \theta_B, \bar{P}\right\rangle - q_B\left\langle \bar{P}, \bar{P}\right\rangle = q_B - q_B = 0$$

Also, the magnitude of $\theta_B - q_B\bar{P}$ is $\underset{X}{\mathrm{Var}}\left(\theta_B\left(X\right)\right)$ since

$$\begin{aligned}\left\|\theta_B - q_B\bar{P}\right\|^2 = \left\langle \theta_B - q_B\bar{P}, \theta_B - q_B\bar{P}\right\rangle &= \left\langle \theta_B, \theta_B\right\rangle - {q_B}^2\left\langle \bar{P}, \bar{P}\right\rangle \\ &= \left\|\theta_B\right\|^2 - {q_B}^2 \\ &= \underset{X}{\mathrm{Var}}\left(\theta_B\left(X\right)\right). \quad \blacksquare\end{aligned}$$

Figure 4.10 shows the variance vector $\theta_B - q_B\bar{P}$. For a fixed expected *pfd*, $q_B$, the length of

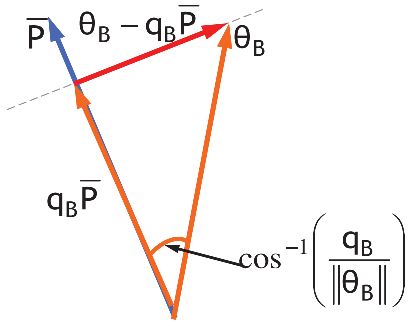


**Figure 4.10:** Variance vector

this vector is seen to increase/decrease depending on whether the angle $\theta_B$ makes with the demand profile is increased/decreased.

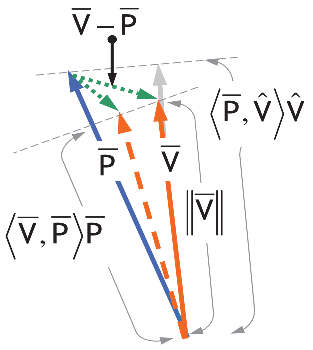
There are special situations where knowing the cosine of the angle between a difficulty function and the demand profile is sufficient for knowing the size of the difficulty function. In fact, imperfect single versions are the only non–zero difficulty functions with the following property.

**Theorem 4.6.4.** *The magnitude of an imperfect single version is equal to the cosine of the angle it makes with the demand profile. That is, for an imperfect single version, $\bar{V}$, and demand profile, $\bar{P}$, we have $\left\langle \bar{V}, \hat{V}\right\rangle = \left\langle \hat{V}, \bar{P}\right\rangle$.*

*Proof.* Let $\bar{V}$ be an imperfect single version. Then, it is a sum of some unique vectors in $\mathcal{S}$, depending on which demands $\bar{V}$ fails on. So, $\bar{V} = \sum_{i=1}^{n} \omega\left(\bar{V}, x_i\right)\bar{P}_i$, where $\omega\left(\bar{V},\right)$ is the score function for the version $\bar{V}$. Then,

$$\begin{aligned}\left\langle \hat{V}, \bar{P} \right\rangle = \left\langle \frac{\bar{V}}{\|\bar{V}\|}, \bar{P} \right\rangle = \frac{1}{\|\bar{V}\|} \left\langle \bar{V}, \bar{P} \right\rangle &= \frac{1}{\|\bar{V}\|} \left\langle \sum_{i=1}^{n} \omega\left(\bar{V}, x_i\right) \bar{P}_i, \sum_{j=1}^{n} \bar{P}_j \right\rangle \\ &= \frac{1}{\|\bar{V}\|} \sum_{i=1,j=1}^{n} \left\langle \omega\left(\bar{V}, x_i\right) \bar{P}_i, \bar{P}_j \right\rangle \\ &= \frac{1}{\|\bar{V}\|} \sum_{i=1}^{n} \left\langle \omega\left(\bar{V}, x_i\right) \bar{P}_i, \bar{P}_i \right\rangle \\ &= \frac{1}{\|\bar{V}\|} \sum_{i=1}^{n} \left\langle \omega\left(\bar{V}, x_i\right) \bar{P}_i, \omega\left(\bar{V}, x_i\right) \bar{P}_i \right\rangle \\ &= \frac{1}{\|\bar{V}\|} \left\langle \sum_{i=1}^{n} \omega\left(\bar{V}, x_i\right) \bar{P}_i, \sum_{j=1}^{n} \omega\left(\bar{V}, x_j\right) \bar{P}_j \right\rangle \\ &= \frac{1}{\|\bar{V}\|} \left\langle \bar{V}, \bar{V} \right\rangle = \left\langle \bar{V}, \hat{V} \right\rangle . \quad \blacksquare \end{aligned}$$

Conversely, if a non–zero difficulty function has a magnitude equal to its cosine, then the



$\bar{V} - \bar{P}$
$\langle \bar{P}, \hat{V} \rangle \hat{V}$
$\bar{P}$
$\bar{V}$
$\|\bar{V}\|$
$\langle \bar{V}, \bar{P} \rangle \bar{P}$

**Figure 4.11:** The magnitude of the projection of $\bar{P}$ in the direction of an arbitrary vector $\bar{V}$ is always larger than $\|\bar{V}\|$, the magnitude of $\bar{V}$. Also, for vectors that touch a line orthogonal to the demand profile $\bar{P}$, the smaller the angle between the vector and $\bar{P}$ the smaller the magnitude of the vector.

difficulty function must be an imperfect single version. This is captured by the following theorem.

**Theorem 4.6.5.** *For an arbitrary non–zero difficulty function, $\theta$, and demand profile, $\bar{P}$, if $\left\langle \theta, \hat{\theta} \right\rangle = \left\langle \hat{\theta}, \bar{P} \right\rangle$ then $\theta$ is an imperfect version.*

*Proof.*

$$\left\langle \theta, \hat{\theta} \right\rangle = \left\langle \hat{\theta}, \bar{P} \right\rangle \Leftrightarrow \left\langle \sum_{i=1}^{n} \theta i \hat{P}_i, \sum_{j=1}^{n} \hat{\theta} j \hat{P}_j \right\rangle = \left\langle \sum_{i=1}^{n} \hat{\theta} i \hat{P}_i, \sum_{j=1}^{n} \|\bar{P}_j\| \hat{P}_j \right\rangle$$
$$\Leftrightarrow \sum_{i=1}^{n} \theta i \hat{\theta} i = \sum_{j=1}^{n} \hat{\theta} j \left\|\bar{P}_j\right\|$$

$$\Leftrightarrow \sum_{i=1}^{n} \hat{\theta} i \left( \theta i - \left\| \bar{P}_i \right\| \right) = 0$$

Now, from the definition of difficulty functions and the change of basis transformation $T$, $0 \leq \theta i \leq \left\| \bar{P}_i \right\|$ and $0 \leq \hat{\theta} i \leq 1$ for $i = 1, \ldots, n$. Therefore, $\sum\limits_{i=1}^{n} \hat{\theta} i \left( \theta i - \left\| \bar{P}_i \right\| \right) = 0$ implies that $\theta i = 0$ or $\left\| \bar{P}_i \right\|$, for all i. So, since $\theta$ is a non–zero difficulty function it must be an imperfect single–version. ■

The combination of the last two theorems is the following.

**Theorem 4.6.6.** *The only non–zero difficulty functions for which $q_{BB} = q_B$ are the imperfect single versions.*

Alternatively, Theorem 4.6.6 can be stated as follows.

**Theorem 4.6.7.** *The imperfect single versions are the only non–zero difficulty functions whose magnitudes are equal to the cosines of the angles they make with the demand profile.*

It was previously indicated that $\left\langle \ , \bar{P} \right\rangle$ can be used to define sets of difficulty functions with common expected *pfd*s. We explore this assertion further. In order to do this we require the definition of a plane in $\mathbb{R}^n$. Given a unit vector $\bar{P}$ and a distance $q$ from the origin along $\bar{P}$ a plane will consist of all vectors, orthogonal to $\bar{P}$, with tails at the "arrow head" of the vector $q\bar{P}$ (see Fig. 4.12). Therefore, if an arbitrary non–zero vector $\bar{V}$ has an "arrow head"

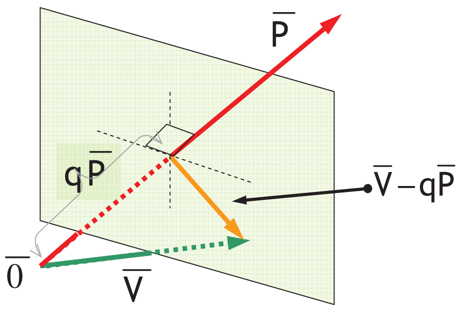


**Figure 4.12:** A plane is defined by a normal, unit vector, say $\bar{P}$, and a preferred projection length $q$, in the direction of $\bar{P}$, for any non–zero vector, $\bar{V}$, that touches the plane.

touching the plane then it must be the case that its projection in the direction of $\bar{P}$ is $q\bar{P}$, and the vector $\bar{V} - q\bar{P}$ must be orthogonal[20] to $\bar{P}$. That is, the equation of the plane is given by $\left\langle \bar{V} - q\bar{P}, \bar{P} \right\rangle = 0$. Consequently, it can be shown that

**Theorem 4.6.8.** *The set of all vectors $\bar{V}$ that touch a plane – where the plane has normal*

[20]Such a plane is said to be orthogonal to $\bar{P}$ since every vector contained in the plane is orthogonal to $\bar{P}$.

*vector $\bar{P}$ and lies at a distance $q$ from the origin in the direction of $\bar{P}$ – must satisfy*

$$q = \left\langle \bar{V}, \bar{P} \right\rangle = \sum_{i=1}^{n} \left(\bar{V}i\right) \sqrt{P_i}.$$

*Proof.* Any vector $\bar{V}$ that touches the plane must satisfy $0 = \left\langle \bar{V} - q\bar{P}, \bar{P} \right\rangle = \left\langle \bar{V}, \bar{P} \right\rangle - \left\langle q\bar{P}, \bar{P} \right\rangle = \left\langle \bar{V}, \bar{P} \right\rangle - q \left\langle \bar{P}, \bar{P} \right\rangle = \left\langle \bar{V}, \bar{P} \right\rangle - q$.

But,

$$\begin{aligned} \left\langle \bar{V}, \bar{P} \right\rangle = \left\langle \sum_{i=1}^{n} \bar{V}i\hat{P}_i, \sum_{j=1}^{n} \left\| \bar{P}_j \right\| \hat{P}_j \right\rangle &= \sum_{i=1}^{n} \bar{V}i \left\| \bar{P}_i \right\| \left\langle \hat{P}_i, \hat{P}_i \right\rangle \\ &= \sum_{i=1}^{n} \bar{V}i \left\| \bar{P}_i \right\| \\ &= \sum_{i=1}^{n} \left(\bar{V}i\right) \sqrt{P_i} \end{aligned}$$

and, consequently, $q = \left\langle \bar{V}, \bar{P} \right\rangle = \sum_{i=1}^{n} \left(\bar{V}i\right) \sqrt{P_i}$ is the equation of the plane. ■

The form of the expected *pfd*, given in Eq. (4.15) on page 103, is very similar to the equation of a plane induced by the mean $q_B$. However, Eq. (4.15) is restricted to vectors that are difficulty functions. So, in this case only a subset of the plane induced by $q_B$ is of interest; the subset intersecting and lying within $\bar{P}$'s bounding region. Vectors lying in this region of the plane are variance vectors. For multiple expected *pfd*s each *pfd* has an associated plane with all of these planes being parallel (see Fig.s 4.13, 4.14).

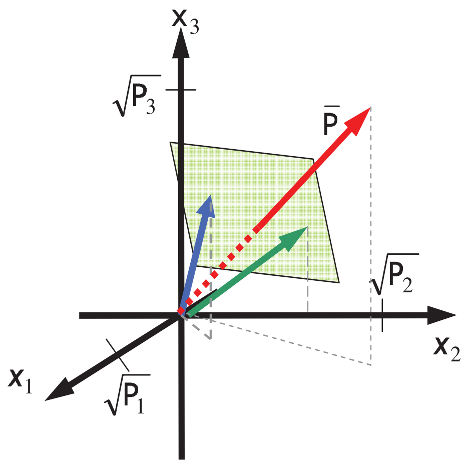


**Figure 4.13:** The expectation of a difficulty function defines an equivalence class of difficulty functions with a striking geometrical relationship; the difficulty functions all touch the same plane which is perpendicular to the demand profile, $\bar{P}$, and lies at a distance from the origin determined by the value of the expected *pfd*.

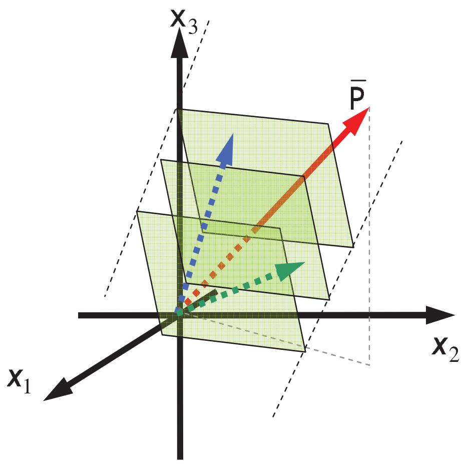


**Figure 4.14:** Different values for the means of different difficulty functions are depicted as a collection of parallel planes.

While we have chosen to use $\bar{P}$ to illustrate the equation of planes, similar planes can be defined with respect to a unit vector in the direction of an arbitrary difficulty function. So, given the non–zero difficulty function $\theta$, exactly the same development should be followed. The linear transformation $\left\langle\ ,\hat{\theta}\right\rangle$ is used to define planes with respect to $\theta$. The equation of the plane a distance $t$ from the origin, in the direction of $\theta$, is $\left\langle\bar{V}-t\hat{\theta},\hat{\theta}\right\rangle = 0$; where the vector $\bar{V}$ is any vector such that $\left\langle\bar{V},\hat{\theta}\right\rangle = t$.

So far we have expanded on the geometry related to the LM model, detailing relationships between difficulty functions that will be useful for the extremisation problems in the current work. As a precursor to the extremisation problems of Chapter 5, the next section details solutions to simple extremisation problems that will prove useful later on.

## 4.7 Preliminary Results of Geometry–based Extremisation

We have established a relationship between the expected *pfd*, the magnitude of the difficulty function and the angle the difficulty function makes with the demand profile. We shall now consider various questions concerning the sizes of these quantities under weak constraints.

### 4.7.1 Angles with Demand Profile under Weak Constraints

Firstly, we consider the extremisation of the angle between a difficulty function and the demand profile, $\bar{P}$. Let $\gamma_B$ be such an angle for the difficulty function $\theta_B$.

How small can $\gamma_B$ get?

If $\gamma_B = 0$, then $\cos\gamma_B = \dfrac{q_B}{\sqrt{q_{BB}}} = 1$ and $\theta_B$ is parallel to $\bar{P}$. Equivalently, this means that:

- $\theta_B$ is parallel to $\bar{P}$. That is, like all constant difficulty functions, $\theta_B = q_B\bar{P}$. To see that the components of this vector are indeed constant, use the transformation $T^{-1}$ on the components $q_B\bar{P} = \left(q_B\sqrt{P_1}, \ldots, q_B\sqrt{P_n}\right)$ to obtain $\left(q_B, \ldots, q_B\right)$. This indicates that $\theta_B$ has the constant value of $q_B$.
- $\operatorname*{Var}_X\left(\theta_B\left(X\right)\right) = 0$.
- the difficulty function is the smallest possible difficulty function with mean $q_B$.

Alternatively, what is the largest possible angle some difficulty function $\theta_B$ can make with $\bar{P}$? It is the angle defined by any difficulty function that is parallel to the imperfect single version with the smallest magnitude. Under the transformation $T$, this is the angle the single version, $\sqrt{P_i}\hat{P}_i$, makes with $\bar{P}$; where $P_i$ is the probability of the least likely demand (that is, $P_i := \min\{P_1, \ldots, P_n\}$).

**Theorem 4.7.1.** *The largest possible angle a difficulty function can make with the demand profile is the angle made by the smallest imperfect single version with the demand profile.*

*Proof.* To prove this recall that all of the imperfect single versions have their magnitudes equal to the cosines of their related angle with $\bar{P}$. Therefore, for imperfect single version $\bar{V}$ the cosine of its related angle is

$$\left\|\bar{V}\right\| = \sqrt{\sum_{j=1}^{n} \omega\left(\bar{V}, x_j\right) P_j} \geq \sqrt{P_i},$$

where $P_i := \min\{P_1, \ldots, P_n\}$. Now, $\sqrt{P_i}$ is the magnitude of the single version, $\sqrt{P_i}\hat{P}_i$, that fails only on the least likely demand, $x_i$. This means this is the imperfect single version furthest from $\bar{P}$. Any difficulty function parallel to this version will share the same cosine value. For if $\theta$ is a difficulty function parallel to this version it will be such that it can be expressed as some scaling of the basis vector $\hat{P}_i$, such as $\theta i\hat{P}_i$, where $0 \leq \theta i \leq \sqrt{P_i}$. Then, the cosine of the angle it makes with $\bar{P}$ is

$$\frac{\left\langle \theta, \bar{P}\right\rangle}{\left\langle \theta, \hat{\theta}\right\rangle} = \frac{P_i}{\sqrt{P_i}} = \sqrt{P_i}.$$

This demonstrates that all of the difficulty functions lying in the 1–dimensional subspace spanned by $\hat{P}_i$ share the same angle with $\bar{P}$. Now, it is impossible to define a 1–dimensional subspace further away from $\bar{P}$ but still contained in the bounding region defined by $\bar{P}$. Hence, the result follows. ■

### 4.7.2 Relationship between Magnitudes, Angles and Projections

There are three ways in which $\cos\gamma_B = \dfrac{q_B}{\sqrt{q_{BB}}}$ can be used in a certain form of constrained extremisation problems. Typically, given $\bar{P}$, these problems will have one of the quantities $q_B, q_{BB}$ or $\cos\gamma_B$ fixed. Using the angle, $\gamma$, as a notion of “distance from the demand profile” in what follows, the possibilities are

- if $q_{BB}$ is kept constant, then an increase in $\cos\gamma_B$ (i.e. a decrease in $\gamma_B$) implies an increase in $q_B$, and vice-versa. ***The closer a difficulty function of a given size is to the demand profile, the larger its mean.*** Geometrically, the surface of possibilities described by this problem is some region on the surface of a sphere[21] with radius $q_{BB}$. The region is contained within, or intersects with, the bounding box defined by $\bar{P}$;
- if $q_B$ is kept constant, then a decrease in $\cos\gamma_B$ (i.e. an increase in $\gamma_B$) implies an increase in $q_{BB}$, and vice-versa. ***The further away a difficulty function with a given mean is from the demand profile, the larger the difficulty function is.*** Geometrically, the surface of interest is the region of a plane orthogonal to $\bar{P}$, at a distance $q_B$ from the origin and intersecting with, or contained within, the bounding region defined by $\bar{P}$ (see Fig. 4.15);

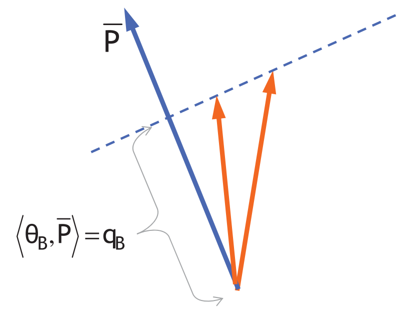


**Figure 4.15:** For a fixed mean $q_B$ the sizes of the difficulty functions increases the further away the difficulty functions are from the demand profile. Fixing the mean is the requirement that the only relevant difficulty functions are the ones with arrowheads touching a plane which is both orthogonal to $\bar{P}$, and lies at a distance $q_B$ from the origin. The dashed blue line lies in such a plane.

- if $\cos\gamma_B$ is kept constant, then an increase in $q_{BB}$ implies an increase in $q_B$, and vice-versa. ***For a difficulty function at a fixed distance from the demand profile, the larger the difficulty function is, the larger its mean.*** This problem defines a cone such that any distinct pair of difficulty functions, both parallel to the surface of the cone, are at an equal angular distance $\gamma$ from $\bar{P}$. As usual, only the surface of the cone within, or on, the bounding region defined by $\bar{P}$ is relevant.

These relationships are very useful in determining the consequences of carrying out geometric operations, such as the rotation or scaling of vectors, in a bid to perform some extremisation.

## 4.7.3 Extremisation in Subspaces

We turn our attention to subspaces defined by the single versions, $\mathcal{S}$, and the sizes of various angles vectors in these subspaces make with the demand profile. These subspaces are important because they define the "walls" of $\bar{P}$'s bounding box. There are $n$ single versions in $\mathcal{S}$ so that there are $2^n - 1$ non-trivial subspaces possible, each of these spanned by some subset of distinct vectors in $\mathcal{S}$. Each of these subspaces has a unique diagonal and *each imperfect single version is parallel to a unique diagonal.* For some concreteness consider $k$ distinct,

[21]see Fig. 5.2 in Chapter 5 for an example of such a surface

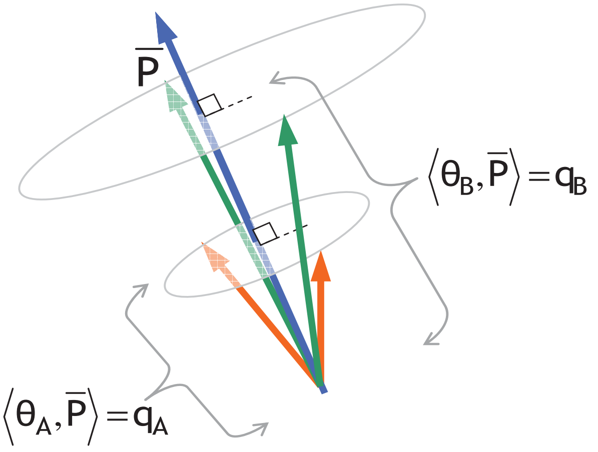

$\bar{P}$
$\langle \theta_B, \bar{P} \rangle = q_B$
$\langle \theta_A, \bar{P} \rangle = q_A$

**Figure 4.16:** When the angle with the demand profile is fixed the set of difficulty functions of interest form a higher dimensional cone, depicted here as a set of circles of increasing radius. Each circle should be regarded as a sphere: in an $n$–dimensional vector–space each of these circles is an $(n-1)$–dimensional sphere with an $(n-2)$–dimensional surface. The radius of each sphere is $\sqrt{1-\frac{q_A^2}{q_{AA}}}$, and it is centered on the point $\underbrace{(q_A, \ldots, q_A)}_{n \text{ times}}$ defined by the constant difficulty function with mean $q_A$. Given a fixed angle with the demand profile the size of a difficulty function and its mean are directly proportional: an increase in size would imply a longer difficulty function on the cone of interest (compare the shorter, orange difficulty functions with the longer, green difficulty functions), which in turn implies a longer projection on the demand profile. Consequently, the mean is also increased (i.e. $q_A \le q_B$).

single versions, $\bar{V}_1, \ldots, \bar{V}_k \in \mathcal{S}$, where $0 < k < n$ and the remaining $n-k$ single versions, $\bar{W}_1, \ldots, \bar{W}_{n-k} \in \mathcal{S}$, such that $\bar{W}_1, \ldots, \bar{W}_{n-k} \neq \bar{V}_1, \ldots, \bar{V}_k$. Then these sets of versions form ***orthogonal subspaces***: each of the vectors in $\text{Span}\big(\{\bar{V}_1, \ldots, \bar{V}_k\}\big)$, the subspace spanned by $\{\bar{V}_1, \ldots, \bar{V}_k\}$, is orthogonal to each of the vectors in $\text{Span}\big(\{\bar{W}_1, \ldots, \bar{W}_{n-k}\}\big)$, the subspace spanned by $\{\bar{W}_1, \ldots, \bar{W}_{n-k}\}$. As a shorthand, we label the sets as $\mathcal{S}_k := \{\bar{V}_1, \ldots, \bar{V}_k\}$ and $\mathcal{S}_{n-k} := \{\bar{W}_1, \ldots, \bar{W}_{n-k}\}$. Note that because $\mathcal{S}$ is a basis then any non-empty subset of $\mathcal{S}$ is a basis for the subspace spanned by that subset. Thus, $\mathcal{S}_k$ and $\mathcal{S}_{n-k}$ are bases. This means an arbitrary vector, $\bar{V} \in \mathcal{S}_k$ for example, can be written as a unique linear combination, $\bar{V} = \big(\bar{V}1\big)\,\bar{V}_1 + \ldots + \big(\bar{V}n\big)\,\bar{V}_k$, for some unique set of scalars $\{\bar{V}1, \ldots, \bar{V}n\}$. In particular, an arbitrary difficulty function in this subspace has the same form, but with the restrictions $0 \le \bar{V}1 \le 1, \ldots, 0 \le \bar{V}n \le 1$ on the components of $\bar{V}$. Recall, these constraints are simply the requirement that a difficulty function must have components with magnitudes in the unit interval. We shall give special attention to the single versions contained in a subspace. $\text{Span}(\mathcal{S}_k)$ contains $2^k$ versions whose failure regions are subsets of the set of demands related to the single versions $\mathcal{S}_k$, these demands having probabilities of occurrence $\|\bar{V}_1\|^2, \ldots, \|\bar{V}_n\|^2$. Similar statements are true for vectors in $\mathcal{S}_{n-k}$.

The “walls” or bounding surfaces of the region defined by $\bar{P}$ consist of all those difficulty functions with some of their components equal to zero. These are very important as many extremisation problems are solved by difficulty functions that either touch, or are contained in, regions parallel to one of such “walls”.

**Definition 4.7.2.** *Let $\mathcal{P}(\mathcal{S})$ be the power set of $\mathcal{S}$, and consider the set $\mathcal{P}(\mathcal{S})/\mathcal{S}$. A "wall" of $\bar{P}$'s bounding box is a region of some subspace,* Span$(s)$*, restricted to only difficulty functions, where $s \in \mathcal{P}(\mathcal{S})/\mathcal{S}$.*

In turn, a similar notion of "walls" may be defined for Span$(\mathcal{S}_k)$ and its related subspaces. The version parallel to the diagonal of Span$(\mathcal{S}_k)$ – that is, parallel to $\bar{V}_1 + \ldots + \bar{V}_k$ – defines a bounding box of difficulty functions in Span$(\mathcal{S}_k)$. Subspaces spanned by proper subsets of $\mathcal{S}_k$, and subsequently restricted to difficulty functions, form the "walls" of this bounding box. So, we see a pattern here, where regions of subspaces form "walls" which are, in turn, bounded by regions of subspaces.

Previously, we stated that the sum of two difficulty functions is not necessarily a difficulty function, since the vector formed by such a sum may not lie in the bounding region defined by the demand profile (see Section 4.3). However, if the difficulty functions lie in orthogonal subspaces such as Span$(\mathcal{S}_k)$ and Span$(\mathcal{S}_{n-k})$, then their sum is guaranteed to be a difficulty function. This suggests the following theorem.

**Theorem 4.7.3.** *Let $\mathcal{S}$ be the basis formed by all of the imperfect single versions that fail only on single demands. So, given $n$ demands, there will be $n$ such versions. Let the sets $\mathcal{S}_{n-k}$ and $\mathcal{S}_k$ be a partition of $\mathcal{S}$. That is, $\mathcal{S}_{n-k}, \mathcal{S}_k \subset \mathcal{S}$ such that $\mathcal{S}_{n-k} \cap \mathcal{S}_k = \emptyset$ and $\mathcal{S}_{n-k} \cup \mathcal{S}_k = \mathcal{S}$. Then $\theta \in$* Span$(\mathcal{S})$ *if, and only if, there exist unique difficulty functions, $\bar{V} \in$* Span$(\mathcal{S}_k)$ *and $\bar{W} \in$* Span$(\mathcal{S}_{n-k})$*, such that*

$$\theta = \bar{V} + \bar{W}.$$

More is true since it is possible to define various partitions[22] of $\mathcal{S}$ consisting of orthogonal sets of imperfect single versions that fail on single demands. Any pair of orthogonal difficulty functions must belong to a pair of orthogonal subspaces spanned by a pair of non–intersecting orthogonal subsets of $\mathcal{S}$. Consequently, any linear combination of vectors in $\mathcal{S}$ is a sum of linear combinations of vectors in the orthogonal subspaces spanned by the partition sets.

**Theorem 4.7.4.** *Any difficulty function can be written as the sum of orthogonal difficulty functions.*

This result will be useful later in Chapter 5 when discussing reliability gains due to forcing diversity.

Given a subspace, such as Span$(\mathcal{S}_k)$, what is the smallest angle a difficulty function within this subspace can make with the demand profile? It turns out that the imperfect single version that fails only on every demand related to the subspace defines the smallest angle.

**Theorem 4.7.5.** *Given a subspace, Span$(\mathcal{S}_k)$, the smallest angle a difficulty function in the subspace can make with the demand profile is the angle the main diagonal of Span$(\mathcal{S}_k)$ makes*

[22]A partition of a set $\mathcal{S}$ is a collection of non–intersecting, proper subsets of $\mathcal{S}$ whose union is $\mathcal{S}$.

*with the demand profile.*

*Proof.* An arbitrary difficulty function, $\theta$, in $\text{Span}(\mathcal{S}_k)$ has the form $(\theta 1)\,\bar{V}_1 + \ldots + (\theta k)\,\bar{V}_k$ and the associated unit vector $\hat{\theta} = \dfrac{\theta}{\|\theta\|} = \dfrac{\sum_{i=1}^{k} (\theta i)\,\bar{V}_i}{\sqrt{\sum_{i=1}^{k} (\theta i)^2 \|\bar{V}_i\|^2}}$. If $\gamma$ is the angle $\theta$ makes with $\bar{P}$ we seek $\theta$ such that $\cos\gamma$ is maximised. That is, maximise

$$\cos\gamma = \left\langle \bar{P}, \frac{\sum_{i=1}^{k} (\theta i)\,\bar{V}_i}{\sqrt{\sum_{i=1}^{k} (\theta i)^2 \|\bar{V}_i\|^2}} \right\rangle = \frac{\sum_{i=1}^{k} (\theta i)\,\|\bar{V}_i\|^2}{\sqrt{\sum_{i=1}^{k} (\theta i)^2 \|\bar{V}_i\|^2}}$$

We know from the Cauchy–Schwarz inequality that

$$\frac{\sum_{i=1}^{k} (\theta i)\,\|\bar{V}_i\|^2}{\sqrt{\sum_{i=1}^{k} (\theta i)^2 \|\bar{V}_i\|^2}\sqrt{\sum_{i=1}^{k} \|\bar{V}_i\|^2}} \le 1 \Leftrightarrow \cos\gamma \le \sqrt{\sum_{i=1}^{k} \|\bar{V}_i\|^2},$$

where $\sqrt{\sum_{i=1}^{k} \|\bar{V}_i\|^2}$ is the magnitude of the version $\bar{V} = \sum_{i=1}^{k} \bar{V}_i$ and, therefore, the cosine of the angle it makes with $\bar{P}$. Since this is the largest possible value for $\cos\gamma$, then $\bar{V}$ makes the smallest angle with $\bar{P}$. Additionally, any difficulty function parallel to $\bar{V}$ – say $x\bar{V}$, for some scalar $x$ such that $0 < x \le 1$ – shares the same angle with $\bar{P}$ as $\bar{V}$, since the cosine of its angle with $\bar{P}$ is

$$\left\langle \bar{P}, \frac{x\bar{V}}{\|x\bar{V}\|} \right\rangle = \left\langle \bar{P}, \frac{x\bar{V}}{x\,\|\bar{V}\|} \right\rangle = \left\langle \bar{P}, \frac{\bar{V}}{\|\bar{V}\|} \right\rangle = \frac{\langle \bar{P}, \bar{V} \rangle}{\|\bar{V}\|} = \frac{\sum_{i=1}^{k} \|\bar{V}_i\|^2}{\sqrt{\sum_{i=1}^{k} \|\bar{V}_i\|^2}} = \sqrt{\sum_{i=1}^{k} \|\bar{V}_i\|^2},$$

which is the same as the cosine value for $\bar{V}$'s angle with $\bar{P}$. Hence, any difficulty parallel to $\bar{V}$ shares the same angle with $\bar{P}$ as $\bar{V}$. ■

Alternatively, what is the largest angle a difficulty function within $\text{Span}(\mathcal{S}_k)$ can make with the demand profile?

**Theorem 4.7.6.** *Given a subspace Span($\mathcal{S}_k$) the largest angle a difficulty function within the subspace can make with the demand profile is the angle made by the imperfect single version that fails solely on the least likely demand related to Span($\mathcal{S}_k$). That is, the angle made by the version in $\mathcal{S}_k$ with the smallest magnitude.*

*Proof.* Let $P_i := \min\left\{\left\|\bar{V}_1\right\|^2, \ldots, \left\|\bar{V}_k\right\|^2\right\}$. Then,

for an arbitrary difficulty function $\theta \in \mathcal{S}_k$ with angle $\gamma$ we have

$$\begin{aligned} P_i \sum_{j=1}^{k} (\theta j)^2 \left\|\bar{V}_j\right\|^2 &\leq \sum_{j=1}^{k} (\theta j)^2 \left\|\bar{V}_j\right\|^4 \\ &\leq \sum_{j=1}^{k} (\theta j)^2 \left\|\bar{V}_j\right\|^4 + 2 \sum_{i<j=1}^{k} \theta j \left\|\bar{V}_j\right\| \left\|\bar{V}_i\right\| \theta i \\ &= \left( \sum_{j=i}^{k} \theta j \left\|\bar{V}_j\right\|^2 \right)^2 . \end{aligned}$$

Therefore,

$$\sqrt{P_i} \leq \frac{\sum_{j=i}^{k} \theta j \left\|\bar{V}_j\right\|^2}{\sqrt{\sum_{j=1}^{k} (\theta j)^2 \left\|\bar{V}_j\right\|^2}} = \cos\gamma.$$

Now, $\sqrt{P_i}$ is the magnitude of the version that fails only on demand $x_i$. Consequently, the cosine of the angle this version makes with the demand profile is also $\sqrt{P_i}$, and so this version makes the largest angle with $\bar{P}$ that is possible for any difficulty function in Span($\mathcal{S}_k$). Furthermore, any difficulty function parallel to this version will make the same angle with $\bar{P}$. ■

So, given any subspace Span($\mathcal{S}_k$), we can determine the difficulty functions within the subspace that are either furthest from, or closest to, the demand profile. When $\mathcal{S}_k = \mathcal{S}$ these results reduce to results we have previously obtained: the version that fails on the least likely demand is furthest from $\bar{P}$, and the version which fails on all demands (and is, therefore, closest to $\bar{P}$) is $\bar{P}$.

As an example of these considerations we reproduce Fig. 4.6 here, in Fig, 4.17. This is the case with a demand space of 3 demands, $\{x_1, x_2, x_3\}$, with 3 versions $\mathcal{S} := \left\{\bar{P}_1, \bar{P}_2, \bar{P}_3\right\}$, each version failing solely on a corresponding demand and having magnitude

$$\left\|\bar{P}_i\right\| = \sqrt{P_i} = \sqrt{\underset{X}{\mathrm{P}}(x_i)} .$$

Consider the subset of $\mathcal{S}$, $\mathcal{S}_2 = \left\{\bar{P}_2, \bar{P}_3\right\}$. The 4 versions in the Span($\mathcal{S}_2$) are $\bar{0}, \bar{P}_2, \bar{P}_3$ and $\bar{P}_2 + \bar{P}_3$. The version, $\bar{P}_2 + \bar{P}_3$, is parallel to the diagonal of Span($\mathcal{S}_2$). The region of Span($\mathcal{S}_2$) that forms a "wall" of $\bar{P}$'s bounding box is bounded by the versions $\bar{P}_2$ and $\bar{P}_3$. The furthest difficulty from $\bar{P}$ in Span($\mathcal{S}_2$) is the version $\bar{P}_2$ with components $\left(0, \sqrt{P_2}, 0\right)$, since $\left\|\bar{P}_2\right\| = \sqrt{P_2} = \min\left\{\sqrt{P_2}, \sqrt{P_3}\right\}$. On the other hand, the difficulty function in Span($\mathcal{S}_2$) closest to $\bar{P}$ is the version $\bar{P}_2 + \bar{P}_3$.

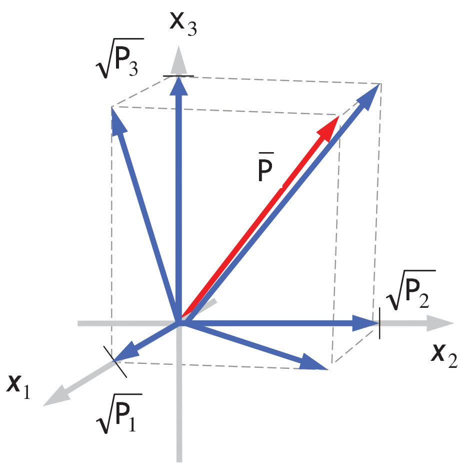
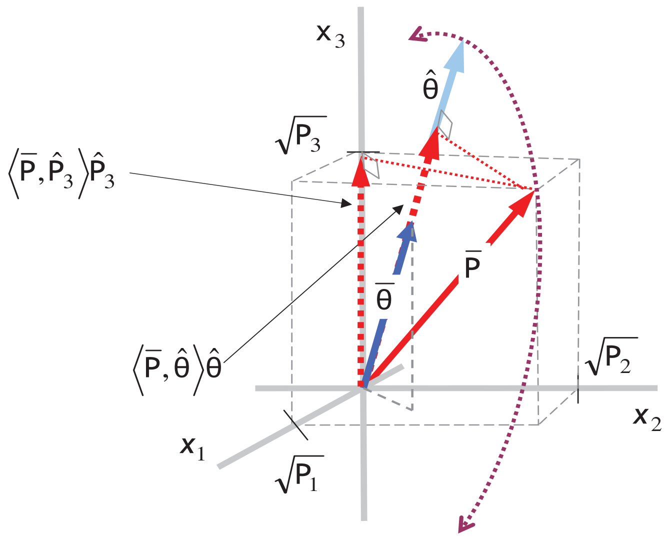


**Figure 4.17:** For a demand space of size 3 there are $2^3$ possible "single versions"/"failure regions", including the perfect version represented by the origin, all of which are depicted in this figure. These versions, apart from the origin and $\bar{P}$, are parallel to diagonals in subspaces that define the "walls" of $\bar{P}$'s bounding box. The versions that fail on only one demand form 1–dimensional boundaries for the "walls" of the bounding box. In the figure $\sqrt{P_1} < \sqrt{P_2} < \sqrt{P_3}$.



$x_3$
$\hat{\theta}$
$\sqrt{P_3}$
$\langle \bar{P}, \hat{P}_3 \rangle \hat{P}_3$
$\bar{P}$
$\bar{\theta}$
$\langle \bar{P}, \hat{\theta} \rangle \hat{\theta}$
$\sqrt{P_2}$
$x_2$
$x_1$
$\sqrt{P_1}$

**Figure 4.18:** The length of the projection of $\bar{P}$ onto a subspace Span($\theta$), which is the span of an arbitrary difficulty function $\theta$, is at least as large as the magnitude of $\theta$, $\left\langle \theta, \hat{\theta} \right\rangle$. This is equivalent to $q_{BB} \leq q_B$. The case of equality is precisely the case where $\theta$ is a version, such as $\bar{P}_3$.

### 4.7.4 Largest Difficulty Function with Specified Mean

Finally, to close this section we consider the following problem. What is the largest expected system *pfd*, $q_{BB}$, given $\bar{P}$ and $q_B$? We know that the mean, $q_B$, induces a plane orthogonal to $\bar{P}$ such that there exists a difficulty function touching this plane with a largest magnitude. This will be a difficulty function that makes the largest possible angle with $\bar{P}$, given that it touches the plane. We already know an upper bound for the magnitude of an arbitrary difficulty function, $\theta_B$, that touches the plane since

$$\|\theta_B\| = \sqrt{q_{BB}} = \sqrt{\langle\theta_B, \theta_B\rangle} = \sqrt{\sum_{i=1}^{n} (\theta_B i)^2} \leq \sqrt{\sum_{i=1}^{n} \theta_B i} = \sqrt{\langle\theta_B, \bar{P}\rangle} = \sqrt{q_B}.$$

This inequality could also be written as $\|\theta_B\| \leq \cos\gamma_B$. From this bound it is easy to see that

$$\langle\theta_B, \theta_B\rangle \leq q_B \Leftrightarrow \langle\theta_B, \theta_B\rangle \leq \left\langle\theta_B, \bar{P}\right\rangle \Leftrightarrow \langle\theta_B, \theta_B\rangle - \left\langle\theta_B, \bar{P}\right\rangle \leq 0 \Leftrightarrow \left\langle\theta_B, \theta_B - \bar{P}\right\rangle \leq 0.$$

Therefore,

> *for fixed $\bar{P}$, the magnitude of $\theta_B$ is maximised if, and only if, the vectors $\theta_B$ and $\bar{P}$ are as close to perpendicular (that is, orthogonal where both vectors are non-zero) as possible.*

In particular, perpendicularity[23] is guaranteed if, and only if, $\theta_B$ is a single version. We proved this equivalence previously, and show the general idea in Fig. 4.18 which depicts projections of $\bar{P}$, both with respect to an arbitrary unit vector $\hat{\theta}$ and with respect to the unit vector $\hat{P}_3$ related to version $\bar{P}_3$. For the case where $\theta_B$ is not an imperfect single version, the difficulty function closest to a suitable imperfect single version will be the difficulty function with the largest magnitude. So, if we find the imperfect single versions that are closest to the plane, then the largest difficulty functions will be in their "neighbourhoods". Locating the versions closest to a plane is not difficult, since the magnitudes of all of the versions can be ordered. We illustrate the idea using a 2–dimensional example, depicted in Fig. 4.19. Here, we consider a case where the versions magnitudes are ordered as $0 \leq \left\|\bar{P}_1\right\| \leq \left\|\bar{P}_2\right\| \leq \left\|\bar{P}_1 + \bar{P}_2\right\|$ or, equivalently, $0 \leq P_1 \leq P_2 \leq P_1 + P_2$. Conceptually, the following thought experiment is useful for carrying out the optimisation. Imagine that we start with an expected version *pfd* value of 0, and then *continuously*[24] increase the value of $q_B$ until it reaches its maximum value of $P_1 + P_2 = 1$. In so doing we can imagine a plane, perpendicular to $\bar{P}$, that rises continuously from the origin as $q_B$ is increased until it reaches the "arrow

[23]Perpendicularity is the property $\left\langle\theta_B, \theta_B - \bar{P}\right\rangle = 0$, where $\theta_B \neq \bar{0}$.

[24]Here we are alluding and appealing to a form of mathematical continuity. Informally, we appeal to the notion that difficulty functions that touch the boundary of a given plane –i.e. where the plane intersects $\bar{P}$'s bounding box – are arbitrarily close to "nearby" difficulty functions that touch the boundary of "nearby" planes. The consequence of this is that magnitudes of such difficulty functions, and the angles they make with $\bar{P}$, are also arbitrarily close.

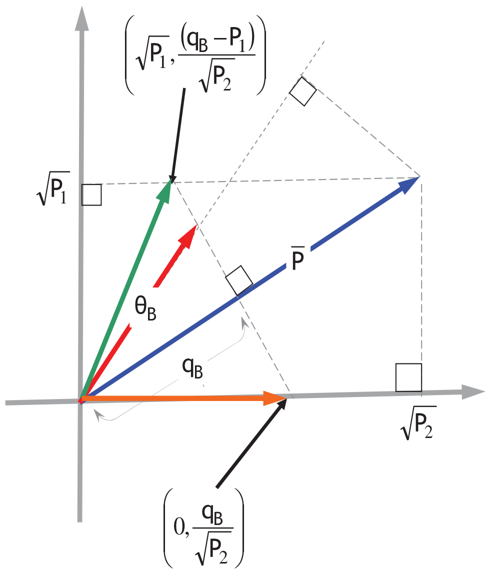

$\left(\sqrt{P_1}, \frac{(q_B - P_1)}{\sqrt{P_2}}\right)$
$\sqrt{P_1}$
$\bar{P}$
$\theta_B$
$q_B$
$\sqrt{P_2}$
$\left(0, \frac{q_B}{\sqrt{P_2}}\right)$

**Figure 4.19:** The plane induced by $q_B$ is a line perpendicular to $\bar{P}$. For $\|\bar{P}_1\| \leq \sqrt{P_1} \leq q_{BB} \leq \sqrt{P_2} \leq \|\bar{P}_2\|$ the difficulty function with the largest $q_{BB}$ value should lie close to either version $\bar{P}_1$, or $\bar{P}_2$. Thus, the difficulty function should either be parallel to $\bar{P}_2$, or as close to $\bar{P}_1$ as possible.

head" of $\bar{P}$. There are 3 cases that are relevant as the plane rises.

*Case 1* $0 \leq q_B \leq P_1$: In this case the versions closest to the $q_B$ plane are $\bar{0}$ and $\bar{P}_1$. Accordingly, the difficulty function with the largest $q_{BB}$ value must be the one parallel to $\bar{P}_1$ that touches the plane which has the coordinates $\left(\frac{q_B}{\sqrt{P_1}}, 0\right)$. In the LM coordinates, via the transformation $T^{-1}$, this has components $\left(\frac{q_B}{P_1}, 0\right)$;

*Case 2* $P_1 \leq q_B \leq P_2$: Here, the versions $\bar{P}_1$ and $\bar{P}_2$ are closest to the $q_B$ plane. Therefore, the difficulty function with largest $q_{BB}$ either has components $\left(\sqrt{P_1}, \frac{(q_B - P_1)}{\sqrt{P_2}}\right)$ or $\left(0, \frac{q_B}{\sqrt{P_2}}\right)$, respectively. Equivalently, in LM coordinates, these are $\left(1, \frac{(q_B - p_1)}{P_2}\right)$ and $\left(0, \frac{q_B}{P_2}\right)$, respectively;

*Case 3* $P_2 \leq q_B \leq P_1 + P_2$: In the final case the versions closest to the $q_B$ plane are $\bar{P}_2$ and $\bar{P}_1 + \bar{P}_2$. Consequently, the form of the desired difficulty function is $\left(\frac{(q_B - P_2)}{\sqrt{P_1}}, \sqrt{P_2}\right)$ as this form is the closest difficulty function to either of the versions. Under $T^{-1}$ the difficulty function is $\left(\frac{(q_B - P_2)}{P_1}, 1\right)$, equivalently.

The forms of the difficulty function given above are obtained in the following manner. We consider a difficulty function that touches a plane, and makes the smallest angle with a version "lying just below" the plane. This difficulty function has components that are almost identical to the components of the version, except the component associated with least likely demand that the version succeeds on. For example, given $P_1 \leq q_B \leq P_2$ we have the version $\bar{P}_1$ lying below the $q_B$ plane. Therefore, the closest difficulty function, $\theta_B$, that touches the plane must be coincident with $\bar{P}_1$ on components that relate to demands on which $\bar{P}_1$

fails. Therefore, it must be $\left(\sqrt{P_1},\ \right)$ to begin with. There is only one remaining component to define. If this difficulty function touches the $q_B$ plane, i.e. if $\langle \theta_B, \bar{P}\rangle = q_B$, then the complete form of the difficulty must be $\left(\sqrt{P_1}, \dfrac{(q_B - P_1)}{\sqrt{P_2}}\right)$. Alternatively, for a version "lying above" a given plane, the difficulty function closest to this version should be identical in terms of all of the components except the component associated with the least likely of the demands on which the version fails. So, in the same example that has just been used, $\bar{P}_2$ lies "just above" the $q_B$ plane. Therefore, the form of the difficulty function should at least be $(0,\ )$, and with the requirement that the difficulty function touch the $q_B$ plane the final form must be $\left(0, \dfrac{q_B}{\sqrt{P_2}}\right)$. After generalising this procedure, we shall discuss the motivation for using the component related to the least likely demands in more detail (see page 120).

This optimisation procedure can be generalised in an unsurprising manner to higher dimensions. The algorithm is as follows.

1. Order the versions in terms of their magnitudes or, equivalently, their angular distances from the demand profile;
2. Find the versions closest to the $q_B$ plane from above and below;
3. Define the forms of difficulty functions, closest to these versions, that touch the boundary of the $q_B$ plane. There are two possibilities. Each possibility ultimately requires only a single component to be determined from the requirment that the difficulty function should touch the $q_B$ plane. For a version "just below" the plane the form of the difficulty function should have components identical to the version except for the component related to the least likely of the demands on which the version succeeds. For a version "just above" the plane the form of the difficulty function should have components identical to the version except for the component related to the least likely of the demands on which the version fails. In both cases if more than one demand is the least likely then any of these demands can be used as a reference;
4. Determine which of these candidate difficulty functions is closest to a version or, equivalently, furthest from the demand profile.

For example, consider a case in 3–dimensions with the following demand profile. $P_1 + P_2 + P_3 = 1$ and $P_1 \le P_2 \le P_3 \le P_1 + P_2 \le P_1 + P_3 \le P_2 + P_3 \le P_1 + P_2 + P_3$. For $P_1 + P_2 \le q_B \le P_1 + P_3$ the $\bar{V}_B$ that results in the maximum $q_{BB}$ is given by either $\begin{pmatrix} \sqrt{P_1} \\ \sqrt{P_2} \\ \dfrac{q_B - P_1 - P_2}{\sqrt{P_3}} \end{pmatrix}$ or $\begin{pmatrix} \dfrac{q_B - P_3}{\sqrt{P_1}} \\ 0 \\ \sqrt{P_3} \end{pmatrix}$, depending on which is further from the demand profile. If we consider the inner–product, $\langle \theta_B, \theta_B - \bar{P}\rangle$, which we wish to minimize for each of these candidate difficulty functions we obtain

$$\frac{q_B - P_1 - P_2}{\sqrt{P_3}}\left(\frac{P_1 + P_2 + P_3 - q_B}{\sqrt{P_3}}\right) \text{ or } \frac{q_B - P_3}{\sqrt{P_1}}\left(\frac{P_1 + P_3 - q_B}{\sqrt{P_1}}\right).$$

So, whichever of these expressions has the smaller value will indicate the difficulty function with the largest $q_{BB}$.

On page 119 we postponed explaining why use was made of the component related to the least likely demand on which a version fails or succeeds? The reasons depend on whether the reference version for the difficulty function lies above or below the $q_B$ plane. In what follows we shall discuss each of these two cases in turn, using a version $\bar{V}$ with the form $\bar{V} = \sum_{i=1}^{k} \bar{V}_i$, where each $\bar{V}_i$ is a unique vector in $\mathcal{S}$ and they form a set $\mathcal{S}_k$. Relatedly, let $\bar{W}_1, \ldots, \bar{W}_{n-k}$ be the remaining versions contained in $\mathcal{S}$, forming the set $\mathcal{S}_{n-k}$.

## Case 1: $\bar{V}$ is a version that lies "just below" the plane

Firstly, consider the case where $\bar{V}$ is a version that lies "just below" the plane. Our aim is to define a difficulty function that is

1. as far away as possible from the demand profile;
2. touches the $q_B$ plane, and;
3. remains in the vicinity of the vector $\bar{V}$.

We proceed in a constructive manner by commencing with the $\bar{V}$ as a template for our difficulty function. To this end, all of the components related to demands on which $\bar{V}$ fails should be the same for our difficulty function. So, we seek a difficulty function $\theta_B = \sum_{i=1}^{k} \bar{V}_i + \sum_{j=1}^{n-k} \theta j \bar{W}_j$, with yet to be determined scalars $\theta 1, \ldots, \theta n - k$ such that

$$\langle \theta_B, \bar{P} \rangle = \sum_{i=1}^{k} \|\bar{V}_i\|^2 + \sum_{j=1}^{n-k} \theta j \|\bar{W}_j\|^2 = q_B \ .$$

We may rewrite this relationship in the slightly more suggestive form,

$$\acute{q} := \sum_{j=1}^{n-k} \theta j \|\bar{W}_j\|^2 = q_B - \sum_{i=1}^{k} \|\bar{V}_i\|^2.$$

That is, we seek a difficulty function $\bar{W} = \sum_{j=1}^{n-k} \theta j \bar{W}_j$ in the subspace Span($\mathcal{S}_{n-k}$), such that the projection of $\bar{W}$ with respect to $\bar{P}$ has magnitude $\acute{q}$, and $\bar{W}$ is as far away as possible from $\bar{P}$. In fact, the second requirement of largest possible angular distance from $\bar{P}$ is equivalent to the requirement that $\bar{W}$ have the largest possible angular distance from the version $\sum_{j=1}^{n-k} \bar{W}_j$, which is itself parallel to the main diagonal of Span($\mathcal{S}_{n-k}$). Two points will be relevant in specifying $\bar{W}$. Firstly, since $\bar{V}$ lies "just below" the $q_B$ plane, this implies that $\acute{q} \leq \|\bar{W}_i\|$ for any $\bar{W}_i \in \mathcal{S}_{n-k}$. Secondly, we have demonstrated previously that a version $\bar{W}_{\min} \in \mathcal{S}_{n-k}$, whose magnitude is equal to $\min\left\{\|\bar{W}_i\| : \bar{W}_i \in \mathcal{S}_{n-k}\right\}$, makes the largest possible angle with the version $\sum_{j=1}^{n-k} \bar{W}_j$. The version $\bar{W}_{\min}$ is the version in $\mathcal{S}_{n-k}$ that fails only on the least likely demand associated with the versions in $\mathcal{S}_{n-k}$. With these observations, $\bar{W}$ must be parallel to $\bar{W}_{\min}$ while touching the plane induced by $\acute{q}$. This plane is perpendicular to the version $\sum_{j=1}^{n-k} \bar{W}_j$. So, $\bar{W} = \phi\bar{W}_{\min}$ for some scalar $\phi$, such that

$$\langle \bar{W}, \bar{P} \rangle = \left\langle \phi\bar{W}_{\min}, \sum_{j=1}^{n-k} \bar{W}_j \right\rangle = \phi \left\| \bar{W}_{\min} \right\|^2 = \acute{q}.$$

Therefore, $\phi = \dfrac{\acute{q}}{\left\| \bar{W}_{\min} \right\|^2}$ and, hence, the complete required form of $\theta_B$ is

$$\theta_B = \sum_{i=1}^{k} \bar{V}_i + \sum_{j=1}^{n-k} \theta j \bar{W}_j = \sum_{i=1}^{k} \bar{V}_i + \left( \frac{q_B - \sum\limits_{i=1}^{k} \left\| \bar{V}_i \right\|^2}{\left\| \bar{W}_{\min} \right\|^2} \right) \bar{W}_{\min} \ . \tag{4.17}$$

This potential solution has the same form as the solution difficulty functions in the 2–dimensional and 3–dimensional examples begun on page 118, and it suggests that one candidate for the maximum value of $q_{BB}$ is

$$q_{BB} = \langle \theta_B, \theta_B \rangle = \sum_{i=1}^{k} \left\| \bar{V}_i \right\|^2 + \left( \frac{q_B - \sum\limits_{i=1}^{k} \left\| \bar{V}_i \right\|^2}{\left\| \bar{W}_{\min} \right\|} \right)^2 . \tag{4.18}$$

This potential solution for $q_{BB}$ will be compared with the potential solution from the second case, which we now discuss.

### Case 2: $\bar{V}$ is a version that lies "just above" the plane

Secondly, we consider the alternative case, where $\bar{V}$ is a version that lies "just above" the plane. Again, our aim is to define a difficulty function, $\theta_B$, that

1. is in $\bar{V}$'s "neighbourhood";
2. is as far away as possible from $\bar{P}$, and;
3. touches the $q_B$ plane.

Since $\bar{V}$ lies "just above" the $q_B$ plane, any version that is a linear combination of a proper subset of versions in $\mathcal{S}_k$ must lie below the $q_B$ plane. The largest of these versions must be the sum of all of the versions in $\mathcal{S}_k$, excluding the smallest imperfect single version in $\mathcal{S}_k$ (we denote this smallest version as $\bar{V}_{\min}$). This sum of versions will be a version that forms the longest diagonal on the "walls" of a bounded region defined by $\bar{V}$. Consequently, we have a version that lies below the $q_B$ plane, and using this version we wish to determine the largest difficulty function (say $\theta_B$) in the subspace Span$(\mathcal{S}_k)$ that touches the $q_B$ plane. But, because this version is a version in Span$(\mathcal{S}_k)$ lying "just below" $q_B$, this is identical to the previous case we discussed. Therefore, we seek a difficulty function that is the sum of the version lying "just below" the plane, and some vector parallel to $\bar{V}_{\min}$. This sum should be a vector that touches the $q_B$ plane. That is, we seek $\theta_B = \left(\sum_{i=1}^{k} \bar{V}_i - \bar{V}_{\min}\right) + \phi\bar{V}_{\min}$, for some scalar $\phi$, such that

$$q_B = \langle \theta_B, \bar{P} \rangle = \left\langle \left(\sum_{i=1}^{k} \bar{V}_i - \bar{V}_{\min}\right) + \phi\bar{V}_{\min}, \bar{V} \right\rangle = \sum_{i=1}^{k} \left\|\bar{V}_i\right\|^2 + (\phi - 1)\left\|\bar{V}_{\min}\right\|^2.$$

Therefore,

$$\phi = \frac{\left(q_B + \left\|\bar{V}_{\min}\right\|^2 - \sum_{i=1}^{k} \left\|\bar{V}_i\right\|^2\right)}{\left\|\bar{V}_{\min}\right\|^2} \, ,$$

which implies that the form of $\theta_B$ is exactly the same as the form of solution difficulty functions in the 2–dimensional and 3–dimensional examples begun on page 118. That is,

$$\theta_B = \sum_{i=1}^{k} \bar{V}_i + \left(\frac{\left(q_B - \sum_{i=1}^{k} \left\|\bar{V}_i\right\|^2\right)}{\left\|\bar{V}_{\min}\right\|^2}\right) \bar{V}_{\min} \, . \tag{4.19}$$

As a consequence, our second candidate for the largest value of $q_{BB}$ is

$$q_{BB} = \langle \theta_B, \theta_B \rangle = \sum_{i=1}^{k} \left\|\bar{V}_i\right\|^2 - \left\|\bar{V}_{\min}\right\|^2 + \left(\frac{\left(q_B + \left\|\bar{V}_{\min}\right\|^2 - \sum_{i=1}^{k} \left\|\bar{V}_i\right\|^2\right)}{\left\|\bar{V}_{\min}\right\|}\right)^2 \, . \tag{4.20}$$

A comparison of the values of Eq.'s (4.18) and (4.20) determines which of these is the actual largest value for $q_{BB}$ and, therefore, determines the difficulty function with the largest size.

## 4.8 Summary

This chapter, in conjunction with Appendix A, has focused on developing a geometric model of coincident failure in multi–version software, using the LM model as the starting point for the endeavour[25]. Various aspects of the LM model have striking geometric interpretations. For instance, given a finite set of demands, an arbitrary difficulty function was shown to uniquely define the non–negative components of a vector in a finite–dimensional vector–space. The vector so defined is, at most, of unit size. Similarly, software failure sets and the demand profile were both shown to be vectors of, at most, unit size (see Section 4.4).

[25]In Appendix A, by presenting a formulation of the mathematics of finite–dimensional vector–spaces in a heuristic manner, justification is given for many of the notions and constructs used in developing the geometric model (see Sections A.1, A.2, and A.3). Not least of these is the construction and use of the inner–product as a multilinear transformation that imbues a vector–space with universal and consistent lengths and angles (see Section A.2).

The size or magnitude of these vectors are expected *pfd*s, and the expectation of the square of a difficulty function (that is, the expected system *pfd* in the EL model) was shown to be the same as the square of the magnitude of the vector that is defined by the difficulty function. This is because there is a natural identification of mathematical expectation, on the one hand, with inner–product on the other hand[26]. So,

> *the longer the magnitude of a vector, the larger the expected system* pfd *resulting from a pair of independent, identically distributed version development processes*[27] *related to the vector's associated difficulty function.*

Also, the mean of a difficulty function – that is, the expected *pfd* resulting from a version development process related to the difficulty function – defines a plane that is orthogonal to the vector defined by the demand profile. In this sense we may state that

> *the expected* pfd *of a difficulty function is the magnitude of a vector's projection – where the vector is defined by the difficulty function – in the direction of the vector defined by the demand profile.*

Put another way, the mean defines an entire family of difficulty functions that "cast the same shadow" on the demand profile (see Section 4.6).

The chapter also explored a number of theorems[28] that are relevant for the extremisation problems of Chapter 5. To illustrate an application of the theorems, consider the challenge of finding a difficulty function (with a specified mean) that has the largest possible value for its associated expected system *pfd*. This is a quadratic optimisation problem with linear (inequality) constraints. It turns out that to solve this problem there are two relevant geometric facts that we prove:

1. if the given mean equals the expected *pfd* of some difficulty function with components equal to either 0 or 1, then this difficulty function is a solution to the problem and the given mean is the largest value for the expected system *pfd*. Indeed, difficulty functions whose components are either 0 or 1 are the only difficulty functions with the property that their means are equal to their associated expected system *pfd*;
2. more generally, given a set of difficulty functions that share the same mean, a solution to the problem can be found by considering those difficulty functions that make the largest angle with the demand profile. Theorem 4.6.1, on page 104, states this, in terms of a relationship between the magnitude of a difficulty function and the cosine of the angle it makes with the demand profile[29]. This approach, of using cosines of angles to

[26]see Section 4.3, starting on page 88.
[27]see Section 2.5, starting on page 43.
[28]see Sections 4.6 and 4.7
[29]For our purposes, the cosine of an angle is an invertible function since we are only concerned with angles between 0 and $\frac{\pi}{2}$ radians. Consequently, we may state a relationship between the magnitude of a difficulty function and the angle it makes with the demand profile, as a relationship between the magnitude of a difficulty function and the cosine of the angle it makes with the demand profile.

maximise or minimize quadratic functions, has not been applied to software reliability modelling before, and we have not come across explicit reference to such a technique in the literature.

Related to this are the largest and smallest angles a difficulty function can make with the demand profile, which we derive in Theorems 4.7.1 (on page 110), 4.7.5 (on page 114) and 4.7.6 (on page 114).

Finally, some of the theorems exemplify notions related to the degree of diversity between difficulty functions. For instance, the angle between difficulty functions is informative in the following way. For a pair of difficulty functions with given means and second moments (and, therefore, fixed sizes), the larger the angle between the difficulty functions the smaller the covariation between them. This casts an old notion – the covariation between a pair of difficulty functions – in a new light – the angle between them. However, angles can be used to capture more than just covariation. For there are families of difficulty functions that are awkward to study using notions like covariance, since the difficulty function pairs thereof have seemingly unrelated means and covariation. However, such a set of difficulty function pairs may be described in terms of an angle. For instance, suppose the sizes of a pair of difficulty functions, and the angle between them, are given. Then, all difficulty function pairs with these properties have the same expected system *pfd* related to forcing diversity, despite distinct pairs possibly having difficulty functions with different means and covariation.

# Chapter 5

# Bounds on Expected System PFD

In Chapter 4 a geometric model of coincident failure was developed, with the aim of using it to determine bounds on expected system *pfd*. In particular, Section 4.2 (page 87) details the 5 steps to be followed when using the model to obtain bounds. In summary, obtaining the solution to a given extremisation problem[1] proceeds as follows:

1) Start with a model of coincident failure in multi–version software: either the LM model, or an extension with the canonical form discussed in Chapter 3.
2) From the model of coincident failure, define an equivalent vector–space representation (a *finite–dimensional, real, inner–product space* in a given *basis*).
3) Use the constraints of the extremisation problem to define the region of the vector–space containing all potential solutions.
4) Solve the extremisation problem by performing transformations on vectors contained in the region of potential solutions, in a bid to to obtain a vector (or a collection of vectors) which achieve the extreme.
5) Reinterpret the solutions in terms of the LM model, or a suitable extension.

The techniques for representing a practical scenario as an LM model extension – that is, step 1 indicated above – were covered in Chapter 3, while Chapter 4 focused on the methodology that accomplishes steps 2 and 3. In this chapter, we concentrate on steps 4 and 5: that is, we derive various attainable bounds on the expected *pfd*s of versions and systems, under various conditions.

[1]The extremisation problems presented in this chapter are not an exhaustive problem set, and were chosen for being representative of the type of problems that can arise. The chapter will focus largely on extremising the following quantity: Given the difficulty functions $\theta_A$ and $\theta_B$, such that $\theta_A \neq \theta_B$, we extremise the expected system *pfd* induced by forcing diversity, which is

$$q_{AB} := \langle \theta_A, \theta_B \rangle \ .$$

In this sense $q_{AB}$ is the inner–product of 2 distinct vectors, and as such many bounds related to $q_{AB}$ follow naturally from the properties of inner–products.

This chapter contains four sections: Sections 5.1, 5.2, 5.3, and 5.4. Section 5.1 gives reasons for bounding the expected system *pfd*. The focus of Section 5.2 is defining conditions under which forcing diversity cannot result in an expected system *pfd* that is worse than it would be if diversity was not forced. Section 5.3 focuses on deriving upper and lower bounds for the expected *pfd* of a system built by forcing diversity, $q_{AB}$, under various conditions of available knowledge. For our purposes, knowledge comes in the form of having available various probabilities that, if true, constrain the magnitude of $q_{AB}$. Such constraining probabilities include the demand profile $\bar{P}$, the expected system *pfd*s for systems built by allowing diversity to occur naturally, difficulty functions and single version expected *pfd*s. Finally, a summary of the results of the chapter is given in Section 5.4.

## 5.1 Reasons for Bounding the Expected System PFD

One of the useful consequences of the LM model assumptions is that comparisons can be made between an expected *pfd* for a single–version system and the expected *pfd* for a related 1–out–of–2 system. This was highlighted in Chapter 2, and is particularly useful when using *pfd* estimates of single version systems to bound the possible *pfd* estimates for related multi–version systems. In general, the expected *pfd* for a single version system **cannot be used** to bound expected system *pfd* in this way[2]. However, this is possible under the LM model, and generalisations of the LM model that adhere to observability Criterion 2.4.1. For in these situations, the expected *pfd* for a channel (built using a given methodology) is identical to the expected *pfd* for a single version system (built using the same methodology). So, for development processes that are adequately modelled via the LM (or LM-*like*) model, we may bound the expected system *pfd* by using the expected *pfd*s for the channels[3]. In this chapter, we consider scenarios in which the expected *pfd*s for single–version systems are known, and use these to bound the expected *pfd* for a related 1–out–of–2 system built by forcing diversity.

Obtaining accurate estimates of difficulty functions (and, therefore, expected *pfd*s) may be infeasible or unlikely to achieve in practice. This is because in Chapter 2 we defined difficulty functions as expectations of suitable score functions with respect to *version sampling* distributions (for instance, see page 36), and these distributions are unknowable in many cases. However, by using bounds on expected system *pfd*, an assessor of a system can be conservative about what system reliability to expect: bounds facilitate "worst–case" analysis. An example of this is the use of *Beta–factors* in practice; in particular, when reasoning about common–cause failure and its potential impact on the reliability of safety critical systems. In essence, a beta–factor is the relative size of the reliabilities associated with two systems, expressed in form of a quotient. For instance, if the expected system *pfd*

[2]See Appendix C for a discussion.

[3]For instance, it is plausible that estimates for the reliability of Commercial off–the–shelf (COTS) software, based on historical failure data, might be readily available. Also, there are situations where it is reasonable to argue that COTS software have been developed by development teams that have had no interaction with one another during development[4]. Consequently, given reliability estimates for COTS software to be used in a fault–tolerant configuration, it is possible to define attainable bounds on the expected system *pfd*. This is because the marginal expected *pfd*s are the expected *pfd*s of the single version COTS developments.

is $q_{AA}$ and the expected *pfd* for a constituent channel is $q_A$, we may define a beta–factor $\beta_A$ as follows.

$$\beta_A := \frac{q_{AA}}{(q_A)^2} .$$

Consequently, this beta–factor gives an idea of the error involved in expecting the channels to fail independently. Suppose only the value of $q_A$ is known to an assessor of this "*A*" system. Then, by choosing hypothetical values for $\beta_A$, the assessor may consider the different consequences of naive assumptions of failure independence. Results presented in this thesis bound $q_{AA}$ when $q_A$ is given, so that the assessor has an upper bound for her range of hypothetical beta–factor values:

$$\beta_A \leq \frac{max\,\{q_{AA}\}}{(q_A)^2} .$$

Additionally, *forcing diversity* – the requirement that each channel be developed using a unique development process methodology – has been advocated as a possible way of encouraging failure diversity between versions. However, forcing diversity cannot always be expected to result in better system reliability than if diversity was allowed to occur naturally instead. Nevertheless, there are cases when forcing diversity results in expected system *pfd* that cannot be worse, and may be better, than if diversity is not forced. In fact, in Chapter 2 (see Section 2.5 on page 42) we discussed two such cases under the LM model, using two related notions of indifference: indifference in methodologies and indifference in expected *pfd* values. In this chapter, we reiterate the close relationship between these 2 notions of indifference, and prove a generalisation of an earlier result where forcing diversity was shown to be beneficial under indifference in methodologies. Also, we give a necessary and sufficient geometric condition for forcing diversity to result in expected reliability that is no worse than otherwise.

Even when forcing diversity is shown to be "a good thing" as it results in improving expected reliability, the extent of this improvement might still be relevant. For instance, it might be the case that the largest expected reliability gain brought about by forcing diversity may not justify the added cost that may accompany such a policy. We do not take into account cost considerations for the purposes of our analyses[5], partly because thresholds for cost–benefit trade-offs are likely to be decided on a case–by–case basis in general. Nevertheless, in any such decision model, it is useful to know what the bounds on the expected reliability would be. Consequently, by specifying attainable bounds on the reliability of a sys-

[5]See Appendix C for a brief discussion of why "how to conduct such cost–benefit considerations" is unobvious.

tem built by forcing diversity, we indicate the limits on any expected reliability improvement obtained thereby.

Furthermore, the methodology we use for bounding expected *pfd*s can also be used to bound other averages that may be of interest. For instance, recall that the expected system pfd for an "AB" system built with the common influence $E$ between the channel development processes may be written as

$$q_{AB} = \sum_{all\ x,\, all\ e} \theta_{A;e}(x)\,\theta_{B;e}(x)\,P(X = x)\,P(E = e)\ , \tag{5.1}$$

where $X$ is a random demand and $\theta_{A;e}(x)$, for instance, is the probability that a randomly chosen "A" version, developed when the common influence had a value $e$ during development, fails on the demand $x$. Each of the following pair of means may be averaged to obtain Eq. (5.1):

- $q(e) = \sum_{all\ x} \theta_{A;e}(x)\,\theta_{B;e}(x)\,P(X = x)$ is the probability that an "AB" system, built under the common influence value $e$ during its development, fails on a random demand. Viewed as a function of the common influence $E$, the expectation of this average with respect $E$ gives Eq. (5.1). Consequently, for each value $e$ we may treat $\theta_{A;e}(x)$ and $\theta_{B;e}(x)$ as difficulty functions, and using the ideas of Chapter 4 we may define a representative geometric model (with axes defined by the demands) and extremise $q(e)$.
- $q(x) = \sum_{all\ e} \theta_{A;e}(x)\,\theta_{B;e}(x)\,P(E = e)$ is the probability that an "AB" system, built under an unknown value for the common influence, fails on a demand $x$. Viewed as a function of the random demand $X$, the expectation of this average with respect $X$ gives Eq. (5.1). Therefore, for each demand $x$ the quantities $\theta_{A;e}(x)$ and $\theta_{B;e}(x)$ can be treated in an analogous fashion to difficulty functions, and using the ideas of Chapter 4 we may define a representative geometric model (with axes defined by the values of the common influence) and extremise $q(x)$.

The bounds also give information about the distributions of expected system *pfd*s, in that they specify the interval of values over which the distribution assigns non–zero probability (see Fig. 5.1).

## 5.2 Forced Diversity vs Natural Diversity

Recall from Chapter 2 that in building a 1–out–of–2 system, diversity between channel development processes can either occur naturally – due to the respective development teams making their respective decisions in isolation, albeit under similar circumstances – or it may be "forced" – by imposing different conditions under which the different teams make their respective decisions in isolation. Since we demonstrated in Section 2.5 that, under the LM model, forcing diversity does not always ensure better expected reliability than if diversity were allowed to occur naturally[6], we are faced with a problem. A manager of the develop-

[6] That is, given a pair of development process methodologies – say methodologies $A$ and $B$ – the minimum of the three expected system *pfd*s $q_{AA}$, $q_{BB}$ and $q_{AB}$ is not always $q_{AB}$.

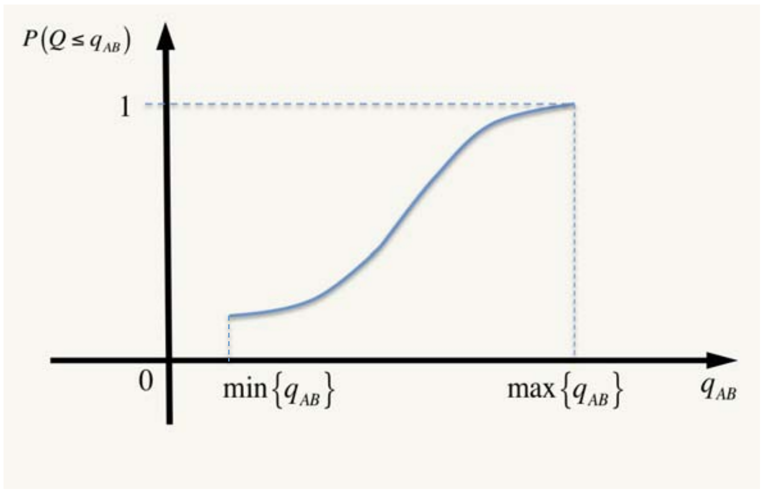


**Figure 5.1:** The techniques of this chapter give bounds on the expected system *pfd*. Consequently, they define the interval over which the distribution of expected system *pfd* may assign non–zero probability values.

ment of a 1–out–of–2 system, trying to decide whether to force diversity or not, cannot rely on there being a particular order between the expected system *pfd*s of these alternatives, in general. This prompts the following question.

$$\text{Under what conditions is } q_{AB} \leq \min\{q_{AA}, q_{BB}\}\,? \tag{5.2}$$

The result in Eq. (2.18) on page 47 gives a particular case where this holds: that is, under indifference between the expected system *pfd*s when diversity is not forced, forcing diversity gives a lower bound on expected system reliability. We generalise this result in the next sub–section. In addition, in sub–section 5.2.2, we tackle the problem of whether to force diversity or not, given that an assessor has no compelling reason to prefer one methodology over another from a given set of methodologies.

### 5.2.1 Preliminary Bounds when Forcing Diversity

When is Eq. (5.2) certain to hold in practice? Assume that the manager of a system development process has estimates of either the relative values or the actual values of the expected system *pfd*s for homogeneous systems (that is, estimates of either $\frac{q_{AA}}{q_{BB}}$, or $q_{AA}$ and $q_{BB}$). Consider two arbitrary non–zero difficulty functions[7], $\theta_A$ and $\theta_B$. Without loss of generality, we assume that the inequality $\sqrt{q_{AA}} = \|\theta_A\| \leq \|\theta_B\| = \sqrt{q_{BB}}$ holds. Then, Eq. (5.2) holds if, and only if, the following inequality holds.

$$\left\langle \theta_B - \theta_A, \hat{\theta}_A \right\rangle \leq 0 \qquad \text{or} \qquad \|\theta_B\| \cos\gamma \leq \|\theta_A\|\,, \tag{5.3}$$

[7]Note, we are interested in only non–zero difficulty functions[8], since a zero difficulty function guarantees $q_{AB} = 0$.

where $\gamma$ is the angle between $\theta_A$ and $\theta_B$. So, the projection of $\theta_B$ in the direction of $\theta_A$ must be smaller than the magnitude of $\theta_A$. The border line case is when the difficulty functions form a "right–angled triangle", with the larger difficulty function as the hypotenuse. This suggests the following theorem.

**Theorem 5.2.1.** *Let $\theta_A$ and $\theta_B$ be a pair of difficulty functions such that $\|\theta_A\| \leq \|\theta_B\|$ and let $\gamma$ denote the angle between them. Forcing diversity will result in expected system* pfd *that is no greater, in general, than any of the pair of related expected system* pfd*s that result from allowing diversity to occur naturally $\big($that is, $\langle\theta_A,\ \theta_B\rangle \leq \|\theta_A\|\big)$ if, and only if,*

$$\frac{\pi}{2} \geq \gamma \geq \cos^{-1}\left(\frac{\|\theta_A\|}{\|\theta_B\|}\right) \geq 0 \ . \tag{5.4}$$

So, for any orientation of $\theta_A$ and $\theta_B$ such that $\gamma$ is sufficiently large, forcing diversity will not worsen expected reliability. Note that inequality (5.4) does not depend on the values of difficulty functions, *it only depends on the expected system* pfd*s for homogeneous systems.* This is important since, in practice, it might be easier to specify a reliability estimate than it is to specify an entire difficulty function. Also, given that you might have some information about a pair of difficulty functions, the theorem specifies "how good" the unknown parts of the difficulty functions need to be to ensure obtaining the best reliability as a consequence of forcing diversity.

Three special cases of Theorem 5.2.1 include:

*Case* 1 $\|\theta_A\| = \|\theta_B\|$: This captures the idea of **indifference between the expected system** *pfd*s resulting from building both channels of a system by using methodology $A$ or $B$ exclusively. That is, for a system assessor who is indifferent between the expected system *pfd*s resulting from allowing diversity to occur naturally, none of these expected system *pfds* is better than the expected system *pfd* resulting from forcing diversity. This result was proved in [13], and is a consequence of the fact that the projection of one difficulty function in the direction of the other difficulty function must be smaller than either difficulty function. Put another way, the condition $\|\theta_A\| = \|\theta_B\|$ is equivalent to requiring that the difficulty functions touch the surface of the same sphere. Consequently, it is impossible to orient the difficulty functions – that is, create a sufficiently small angle between them – in a way that results in a projection that is sufficiently long enough for diversity not to be beneficial. Therefore, Eq. (5.3) always holds[9].

Equality is attained if, and only if, the difficulty functions thereof are collinear. Since the difficulty functions are required to have the same magnitude, collinearity implies that the difficulty functions are identical. This means that the channels are developed in such a way that the teams are identical in how difficult they find each demand. This is unlikely to be the case in practice: one can expect differences to exist between the teams

[9]This result is a consequence of the Cauchy-Schwarz inequality since, from Eq. (5.3), we have

$$q_{AB} = |\langle\theta_A, \theta_B\rangle| \leq \|\theta_A\|\,\|\theta_B\| = \|\theta_B\|^2 = \langle\theta_B, \theta_B\rangle = q_{BB} = \min\{q_{AA}, q_{BB}\}\,. \tag{5.5}$$

(e.g. different individuals with different software development experience make up the different teams), even if such differences may have only a small impact on the variation in difficulty between the teams. Consequently, it is unlikely that the equality holds in practice, in which case forcing diversity gives the best expected system reliability.

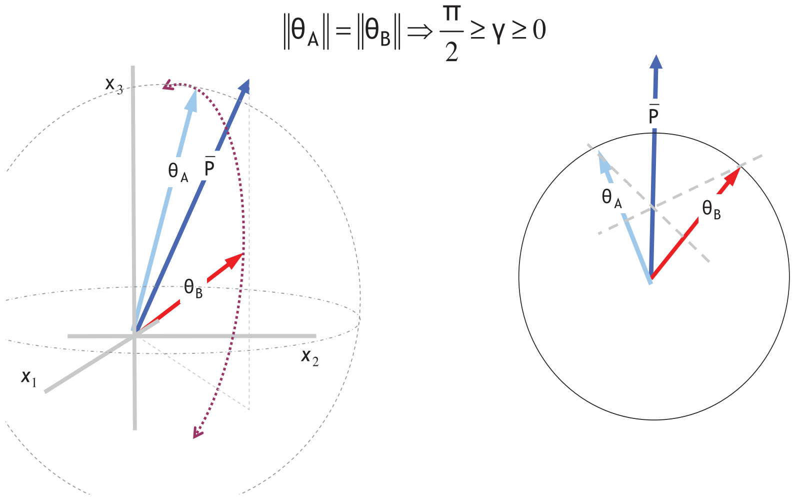

$\|\theta_A\| = \|\theta_B\| \Rightarrow \frac{\pi}{2} \geq \gamma \geq 0$
$x_3$
$\theta_A$
$\bar{P}$
$\theta_B$
$x_1$
$x_2$
$\bar{P}$
$\theta_A$
$\theta_B$

**Figure 5.2:** This figure depicts two viewpoints of the set of all difficulty functions considered when maximizing $q_{AB}$, subject to the constraint that the expected system *pfds* for homogeneous "AA" and "BB" systems are assumed equal. All of the difficulty functions of interest have their tips touching the surface of a sphere of radius $\|\theta_A\|$. Intuitively, from the figure on the right above, we see that it is impossible to orient an arbitrary pair of difficulty functions that both touch the sphere in such a way that the projection of one vector in the direction of the other vector is longer than the other vector. This implies that the angle $\gamma$ must be at least 0, which is the case for any pair of relevant difficulty functions touching the sphere, and at most $\frac{\pi}{2}$.

*Case* 2 *$\theta_A$ and $\theta_B$ are orthogonal*: In particular, if the difficulty functions are perpendicular then their mutual projections are $\bar{0}$, and thus Eq. (5.3) holds with

$$q_{AB} = 0 \leq \min\{q_{AA}, q_{BB}\} \ .$$

If there exist imperfect single versions, say $\bar{V}_1$ and $\bar{V}_2$, such that they do not fail coincidentally on any demands (therefore, they are orthogonal) and $\|\theta_A\| \leq \|\bar{V}_1\|$, $\|\theta_B\| \leq \|\bar{V}_2\|$, then $q_{AB} = 0$ can be attained. Otherwise, the attainable lower bound is tighter. However, $q_{AA}, q_{BB} << 1$ for high reliability systems, so that such a pair of orthogonal versions can occur in practice.

*Case* 3 *A pair of difficulty functions, one of which is constant, with the same mean*: Without loss of generality, consider the difficulty functions $\theta_A = q_B\bar{P}$ and $\theta_B$, so that

$$q_{AB} = \langle\theta_A, \theta_B\rangle = \langle q_B\bar{P}, \theta_B\rangle = q_B\langle\bar{P}, \theta_B\rangle = (q_B)^2 \leq \min\{q_{AA}, q_{BB}\}.$$

Of course, for a specified demand profile $\bar{P}$, its related bounding region constrains the size of the maximum possible angle, $\gamma_{\max}$, between an arbitrary pair of difficulty functions with specified sizes $\|\theta_A\|$ and $\|\theta_B\|$. From Theorem 5.2.1 the minimum angle required for forced

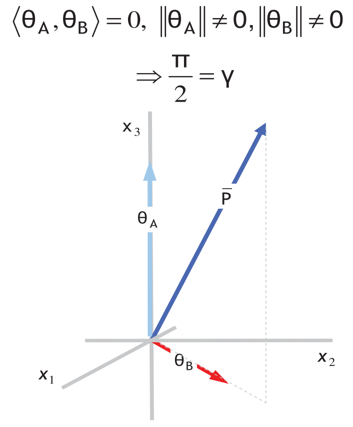

$\langle \theta_A, \theta_B \rangle = 0,\ \|\theta_A\| \neq 0, \|\theta_B\| \neq 0$
$\Rightarrow \frac{\pi}{2} = \gamma$
$x_3$
$\theta_A$
$\bar{P}$
$\theta_B$
$x_1$
$x_2$

**Figure 5.3:** In this figure the candidate difficulty functions are orthogonal and, consequently, must lie on the boundaries of the region defined by $\bar{P}$.

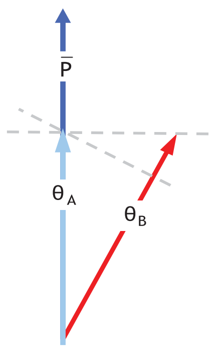


**Figure 5.4:** Here a pair of difficulty functions, one of them being constant, is assumed to have the same mean (so they have the same projection lengths on the demand profile). The constant difficulty function is parallel to the demand profile, $\bar{P}$. Consequently, the projection of any other difficulty function in the direction of $\bar{P}$ has a length that is the shared mean and, hence, must be the length of the constant difficulty function.

diversity to guarantee an expected system *pfd* that cannot be worse is

$$\gamma = \cos^{-1}\left(\frac{\|\theta_A\|}{\|\theta_B\|}\right) = \cos^{-1}\left(\sqrt{\frac{q_{AA}}{q_{BB}}}\right) ,$$

which may be too large (that is, $\gamma_{\max} < \gamma$) given the restrictions of $\bar{P}$. In such a case it is impossible for forcing diversity to be beneficial. The problem of determining the value of $\gamma_{\max}$ given $\bar{P}$, $\|\theta_A\|$ and $\|\theta_B\|$, is the focus of ongoing research.

We can use the special cases above to specify some more examples of when forcing diversity is beneficial. Consider the case when a manager of a development process has reason to believe that the difficulty functions for the channels' development processes are identical on some subset of the demand space. So, given a demand profile $\bar{P}$, suppose $\theta_A$ and $\theta_B$ are such that they are identical on a subset of demands. Then, according to Theorem 4.7.4 there are 3 difficulty functions, $\bar{V}_1$, $\bar{V}_2$ and $\bar{V}_3$, such that $\bar{V}_2$ is orthogonal to both $\bar{V}_1$ and $\bar{V}_3$,

$$\theta_A = \bar{V}_1 + \bar{V}_2 \text{ and } \theta_A = \bar{V}_2 + \bar{V}_3.$$

This means that $q_{AB} = \langle\theta_A, \theta_B\rangle = \langle\bar{V}_2, \bar{V}_2\rangle + \langle\bar{V}_1, \bar{V}_3\rangle$. Similarly, $q_{AA} = \langle\bar{V}_2, \bar{V}_2\rangle + \langle\bar{V}_1, \bar{V}_1\rangle$ and $q_{BB} = \langle\bar{V}_2, \bar{V}_2\rangle + \langle\bar{V}_3, \bar{V}_3\rangle$. Consequently, forcing diversity will not worsen reliability (that is, $q_{AB} \leq \min\{q_{AA}, q_{BB}\}$) if, and only if $\langle\bar{V}_1, \bar{V}_3\rangle \leq \min\{\langle\bar{V}_1, \bar{V}_1\rangle, \langle\bar{V}_3, \bar{V}_3\rangle\}$. So, the three cases outlined above as particular examples of Theorem 5.2.1 can be applied here to give three more cases in which forcing diversity is beneficial. If the parts of the difficulty functions that are not identical have any of the following properties:

1. If they have the same size, then $\langle\bar{V}_1, \bar{V}_1\rangle = \langle\bar{V}_3, \bar{V}_3\rangle$. This is indifference between the sizes of the difficulty functions on a subset of the demands. Consequently, via the Cauchy–Schwarz inequality, $\langle\bar{V}_1, \bar{V}_3\rangle \leq \langle\bar{V}_1, \bar{V}_1\rangle = \langle\bar{V}_3, \bar{V}_3\rangle = \min\{\langle\bar{V}_1, \bar{V}_1\rangle, \langle\bar{V}_3, \bar{V}_3\rangle\}$;
2. If they are orthogonal, then $\langle\bar{V}_1, \bar{V}_3\rangle = 0 \leq \min\{\langle\bar{V}_1, \bar{V}_1\rangle, \langle\bar{V}_3, \bar{V}_3\rangle\}$;
3. If they have equal means, with either $\bar{V}_1$ or $\bar{V}_3$ being a constant difficulty function, then $\langle\bar{V}_1, \bar{V}_3\rangle = \langle\bar{V}_1, \bar{P}\rangle\langle\bar{V}_3, \bar{P}\rangle \leq \min\{\langle\bar{V}_1, \bar{V}_1\rangle, \langle\bar{V}_3, \bar{V}_3\rangle\}$.

### 5.2.2 Forced Diversity under Indifference between Methodologies

In the last sub–section we indicated how *indifference between expected system pfd*s (for instance, $q_{AA} = q_{BB}$) may be used to justify forcing diversity. Similarly, there is another form of indifference – *indifference between methodologies* – that may be used to justify forcing diversity. A manager of a development process is said to be **indifferent between methodologies** if, amongst alternative combinations of methodologies that may be used to build the channels of the system, she is equally likely to use any of these combinations in building the system[10]. Such a situation could arise for different reasons, including:

[10] *Indifference between pfd*s and *indifference between methodologies* are closely related, but different. Regard "the expected system *pfd* resulting from building a system by employing a randomly chosen methodology" as a random variable. Then indifference between methodologies defines a distribution of expected system *pfds*, while indifference between expected system *pfds* is a statement about equal realisations of a random variable. To illustrate this relationship further, suppose an engineer is presented with two seemingly identical balls that are coloured differently – perhaps, one blue and one green. Further suppose that the

- The manager may be convinced that the methodology combinations do not differ significantly, in terms of both the cost incurred in their use and their associated expected system *pfd*s;
- The manager has no evidence available to suggest one combination being better, in some relevant sense, than another;
- The manager may suspect that certain combinations of methodology indeed result in better expected system reliability. However, she may be unsure about whether the expected reliability gains from using these combinations justify the potential extra economic cost incurred in using these combinations.

Given such uncertainty about preferences between methodologies, our aim is to illustrate a scenario where forcing diversity is the desired option. To this end we consider a coordinator/manager of the development of some 1–out–of–N system, who is faced with the choice of whether to force diversity (that is, require that each channel be built using a unique methodology) or not. Suppose:

1. The manager has a choice of $N$, unique software development methodologies, $m_1, m_2, \ldots, m_N$, say;
2. The manager is indifferent amongst the methodologies that may be employed in the development of each channel. So, any of $m_1, m_2, \ldots, m_N$ is equally likely to be assigned to the development of each channel. In practice, this would be the case if the manager has no evidence to justify preferring one methodology over another. For instance, the methodologies may be relatively novell and as such historical, empirical or anecdotal evidence to justify the use of one methodology over another may be unavailable. Even if the methodologies have been applied extensively it might be the case that they have not been applied under sufficiently similar circumstances, similar to the current proposed development process. For instance, the systems that have been built using the methodologies might have architectures that are sufficiently different from the proposed system architecture to be built. So, available empirical evidence may not be applicable in choosing which methodologies should be preferred. Note, however, that this does not mean the difficulty functions, induced by applying the methodologies, are the same; our assessor might readily accept that it is plausible some methodologies are more suited for use in tackling certain issues during development than others, despite not knowing which. Indeed, as was pointed out in [13], it should be expected that diverse methodologies should induce diverse difficulty functions.

Under these conditions the following theorem holds.

manufacturer of the balls has told the engineer that one of these seemingly identical balls has a significantly higher bounce, according to an industry–standard definition for the bounce of a ball, without indicating which of these balls has a higher bounce. The engineer is then tasked with guessing which of the two balls has the lower bounce. Since the balls are seemingly identical, the engineer would assign equal probability to each ball having the lower bounce, despite being told that one of the balls indeed has a higher bounce. The bounce of each ball is analogous to the expected system *pfd* for a homogeneous system: in both experiments these are unknown quantities with a possible ordering, and a choice of lowest of these unknown quantities must be made. The balls having equal probability of being the one with the lowest bounce is akin to being indifferent between methodologies. However, if by running tests on the balls in her laboratory, the engineer finds out that the balls actually have equal bounce heights, then she becomes indifferent between the heights of the ball bounces. This is similar to being indifferent between expected system *pfd*s.

**Theorem 5.2.2.** *Building a 1–out–of–N system by forcing diversity results in a system with expected* pfd *that is less than or equal to the expected* pfd *obtained if diversity were not forced. That is,*

$$P\begin{pmatrix}\textit{1–out–of–N system, built by not}\\ \textit{forcing diversity, fails on } X\end{pmatrix} \geq P\left(\Pi_{m_1},\ldots,\Pi_{m_N} \textit{ fail on } X\right)$$

*Proof.* We assume an LM–type model of the development and operation of a 1–out–of–N system. So, each channel is developed in isolation, the model of system development is the product probability space comprised of the models for each channel and $\left(\mathcal{X},\Sigma_{\mathcal{X}},\mathrm{P}_X(\cdot)\right)$ is a model of demand occurrence. However, because there is no mathematical difference between the LM–model and a generalisation of the LM model where independence is modelled via conditional independence, the following proof applies in both cases. That is, this result also applies in cases where the development of the channels is isolated except for common activities which induce conditional failure independence between the channels. Let $i_1,\ldots,i_N$ be indices associated with the channels, $1,\ldots,N$, of a 1–out–of–N system such that $i_j = m_1,\ldots,m_N$ for $j=1,\ldots,N$. These indices will be used as a notational device in the proof to assign methodologies to the development of the channels. So, for example, given 3 unique methodologies, $\{m_1,m_2,m_3\}$, a 1–out–of–3 system whose development is modelled by the random vector $(\Pi_{m_3},\Pi_{m_1},\Pi_{m_2})$ is equivalent to $(\Pi_{i_1},\Pi_{i_2},\Pi_{i_3})$ where channel 1 is developed using methodology 3, i.e. $i_1=m_3$, channel 2 is developed using methodology 1, i.e. $i_2=m_1$, and channel 3 is developed using methodology 2, i.e. $i_3=m_2$. Each methodology, say $m_i$, induces a related difficulty function, $\theta_{m_i}(x)$. Now, there are $N^N$ ways of assigning the $N$ methodologies to the development of the channels. Each of these assignments are equally likely. Furthermore, $N!$ of these are assignments in which diversity is forced. That is, there are $N!$ ways of assigning the $N$ unique methodologies to the developments of the $N$ channels such that no two channels are developed using the same methodology. Therefore, there are $N^N-N!$ ways in which a manager may assign the methodologies such that at least two of the channels are developed using the same methodology. As a consequence of the manager having no preference in which of these possible assignments should be employed if diversity is not forced, then

$$P\begin{pmatrix}\text{1–out–of–N system, built by not}\\ \text{forcing diversity, fails on } X\end{pmatrix} = \frac{\sum\limits_{i_1,i_2,\cdots,i_N}\left(\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{i_j}\mathrm{P}_X(dx)\right) - N!\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{m_j}\mathrm{P}_X(dx)}{N^N-N!},$$

where the *Lebesgue–Stieltjes* integrals used here collapse to finite sums for our purposes, so that

$$\begin{aligned}\int\limits_{\mathcal{X}}\prod_{j=1}^{N}\theta_{i_j}\mathrm{P}_X(dx) = \int\limits_{\mathcal{X}}\theta_{1_j}\ldots\theta_{N_j}\mathrm{P}_X(dx) &= \int\limits_{\mathcal{X}}\theta_{1_j}(x)\ldots\theta_{N_j}(x)\mathrm{P}_X(dx)\\ &= \sum_{\mathcal{X}}\theta_{1_j}(x)\ldots\theta_{N_j}(x)\mathrm{P}_X(x),\end{aligned}$$

and $\theta_{i_j}(x)$ is the difficulty function induced by the methodology $i_j$. Since the integrals above are finite valued (by definition, these integrals are probabilities) we may view the right–hand–side of the last equation as the arithmetic mean of real numbers. Consequently, using the well known result that the arithmetic mean of a finite set of non–negative real numbers is greater than or equal to their geometric mean[11] [46, 47, 48], we see that

$$\frac{\sum\limits_{i_1,i_2,\cdots,i_N}\left(\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{i_j}\mathrm{P}_X(dx)\right)-N!\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{m_j}\mathrm{P}_X(dx)}{N^N-N!}\geq\left(\frac{\prod\limits_{i_1}\cdots\prod\limits_{i_N}\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{i_j}\mathrm{P}_X(dx)}{\left(\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{m_j}\mathrm{P}_X(dx)\right)^{N!}}\right)^{\frac{1}{N^N-N!}}$$

Upon rewriting the right–hand–side of the inequality above, and applying a generalisation of the Cauchy-Schwarz inequality, this implies

$$\left(\frac{\prod\limits_{i_1}\cdots\prod\limits_{i_N}\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{i_j}\mathrm{P}_X(dx)}{\left(\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{m_j}\mathrm{P}_X(dx)\right)^{N!}}\right)^{\frac{1}{N^N-N!}}=\left(\frac{\prod\limits_{i_1}\cdots\prod\limits_{i_N}\int\limits_{\mathcal{X}}\left[\prod\limits_{j=1}^{N}\theta_{i_j}^{\frac{1}{N^N-N!}}\right]^{N^N-N!}\mathrm{P}_X(dx)}{\left(\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{m_j}\mathrm{P}_X(dx)\right)^{N!}}\right)^{\frac{1}{N^N-N!}}$$

$$\geq\int\limits_{\mathcal{X}}\prod\limits_{j=1}^{N}\theta_{m_j}\mathrm{P}_X(dx)=P\left(\Pi_{m_1},\ldots,\Pi_{m_N}\text{ fail on }X\right)$$

Therefore,

$$P\begin{pmatrix}\text{1–out–of–N system, built by not}\\ \text{forcing diversity, fails on }X\end{pmatrix}\geq P\left(\Pi_{m_1},\ldots,\Pi_{m_N}\text{ fail on }X\right).\quad\blacksquare$$

Note: for its use in practice the theorem does not require the manager to have estimates of the expected *pfd*s used in the theorem; whatever the values of the expected *pfd*s they must imply the theorem holds. This is ideal because such estimates may not be readily available for various reasons, including the typical complexity of software used in safety critical systems as well as the "*structure*" of the demand space which may make exhaustive testing infeasible. Indeed, partly because such estimates may be unavailable in practice, managers of development processes may not have strong justification to prefer some methodology over some other methodology. In essence, the manager is indifferent among these methodologies.

[11] A sketch proof of this can be given by taking the natural logarithm of a representative arithmetic mean, accepting that the natural logarithm function is concave, monotonic increasing and therefore, by Jensen's inequality, the result follows.

## 5.3 Optimisation of Expected System Pfd under Forced Diversity

So far we have explored conditions for forcing diversity to be beneficial. Now, we concern ourselves with trying to determine the limits on system reliability under a policy of forcing diversity. While, in practice, it may be difficult to obtain complete descriptions of difficulty functions and demand profiles, it is possible that partial knowledge of such functions and estimates of expected *pfd*s are available. These estimates might be based on historical data of system operation and failure behaviour, where such systems operate in similar environments and have been developed using similar development process methodologies. Such available knowledge may be used to constrain the value of expected system *pfd*. For instance, COTS software that has been used extensively – such as part of airplane flight control systems (typically fly–by–wire) [49] [34] or Database Management Systems – will potentially have data concerning several hours of use. So, estimates for the *pfd*s of such software may be available. One viewpoint of such estimates is that these are expected *pfd*s. Expectations, because of uncertainty about the existence of unseen demands on which the software would fail. So, if such COTS components were created using different methodologies, and these components are to be used in a fault–tolerant configuration, then the *pfd* estimates constrain the reliability of the system to be built.

In this section we shall consider a number of constrained optimisation problems in which we will seek a pair of difficulty functions, $\theta_A$ and $\theta_B$, that result in an extreme value for some given function. The ***objective function*** – that is, the function to be extremized – will usually be the expected system *pfd* under a development process policy of forcing diversity, $q_{AB}$. The set of difficulty function pairs that achieve the extreme value of the objective function will be constrained. In fact, both equality and inequality constraints will always be imposed, and sometimes one of the difficulty functions will be given. So, the aim will be to find a difficulty function (or a pair of difficulty functions) that results in the maximum/minimum value of $q_{AB}$ being attained. The equality constraints will either be linear (such as a given value for the mean of an unknown difficulty function) or quadratic (such as a given value for the expected system *pfd* for a homogeneous system). The inequality constraints will only be used to ensure that the vectors of interest in the solution of the optimisation problems are indeed difficulty functions; the vectors lie in a region of the vector space defined by the demand profile $\bar{P}$. In summary, given two channels of a 1–out–of–2 system (labeled $A$ and $B$), the constraints on the optimisation problems that follow will be functions of the following:

1. Either a difficulty function $\theta_A$, or (in an exclusive sense) $\theta_B$;
2. The demand profile, $\bar{P}$;
3. The expected *pfd*s ($q_A$ or $q_B$);
4. The expected system *pfd*s ($q_{AA}$ or $q_{BB}$).

We take each constrained optimisation problem in turn.

### 5.3.1 Extremisation of $q_{AB}$ (given a Demand Profile)

**Theorem 5.3.1.** *Given a demand profile, $\bar{P}$, the maximum/minimum possible values for $q_{AB}$ resulting from a pair of non–zero difficulty functions are 1 and 0.*

*Proof.* Since there is no constraint on which pair of difficulty functions we may choose, the problem is trivial. For the minimum we choose any pair of difficulty functions that lie in orthogonal subspaces, thus ensuring that their inner–product is zero. On the other hand, for the maximum, the largest difficulty function is the one that guarantees failure on every demand; that is, the version $\bar{P}$. Consequently, by defining $\theta_A := \bar{P}$ and $\theta_B := \bar{P}$ we ensure that $\theta_A$ and $\theta_B$ are collinear and, therefore, their inner–product is the maximum possible value. ■

### 5.3.2 Extremisation of $q_{AB}$ (given a Demand Profile, $q_B$ and $q_{BB}$)

Given $\bar{P}$, $q_B \neq 0$ and $q_{BB} \neq 0$ what is the maximum/minimum possible $q_{AB}$ resulting from a pair of non–zero difficulty functions, one of them with associated means $q_B$ and $q_{BB}$? We shall deal with the maximum and the minimum in turn.

**Theorem 5.3.2.** *Let $\theta_B$ be some difficulty function that has associated means $q_B$ and $q_{BB}$. Let $\theta_A$ be a non–zero vector that we seek such that with $\theta_B$ maximises $q_{AB}$. Given the demand profile $\bar{P}$ the maximum value of $q_{AB}$ is $q_B$, obtained by $\theta_A = \bar{P}$ for any appropriate $\theta_B$.*

*Proof.* Firstly we need to check that $\bar{P}$, $q_B$ and $q_{BB}$ are consistent, in the sense that there exists $\theta_B$ with associated means $q_B$ and $q_{BB}$. Such a $\theta_B$ exists if, and only if, the largest difficulty function that touches the $q_B$ plane has a magnitude of at least $\sqrt{q_{BB}}$. Therefore, using the algorithm for obtaining the largest difficulty that touches the $q_B$ plane discussed in the previous chapter, we can verify $\theta_B$'s existence. Suppose we have verified $\theta_B$'s existence and, consequently, defined a suitable $\theta_B$. Since $q_{AB} = \langle \theta_A, \theta_B \rangle = \left\langle \theta_A, \hat{\theta}_B \right\rangle \|\theta_B\|$ then, in order to maximise $q_{AB}$, we proceed in 2 steps:

1. Find a vector, $\theta_A$, with the largest projection in the direction of $\theta_B$;
2. Find the $\theta_B$ that results in the largest of these largest projections.

From the computational form of the inner–product it is easy to see that $\bar{P}$ gives the largest projection in the direction of $\theta_B$. But the projection of $\bar{P}$ in the direction of $\theta_B$ must have length $\left\langle \bar{P}, \hat{\theta}_B \right\rangle = \frac{q_B}{\sqrt{q_{BB}}}$, irrespective of which $\theta_B$ we use. Therefore, $\theta_A = \bar{P}$ gives the maximum value of $q_{AB}$ as

$$q_{AB} = \left\langle \bar{P}, \theta_B \right\rangle = q_B \quad ■$$

For the lower bound there are 2 possibilities: $\theta_B$ may, or may not, lie completely in a subspace spanned by the versions in some proper subset of $\mathcal{S}$. That is, $\theta_B$ may, or may not, lie in a "wall" of the bounding box defined by the demand profile $\bar{P}$. If $\theta_B$ does not lie in a "wall", then it is impossible to define non–zero $\theta_A$ that ensures a lower bound for

$q_{AB}$. This is because due to the continuity of $\mathbb{R}^n$ it is always possible to define a smaller, non–zero $\theta_A$ that ensures a smaller, non–zero value for $q_{AB}$. If, on the other hand, $\theta_B$ lies in a "wall" then it lies in a subspace, say $\text{Span}(\mathcal{S}_k)$, that has related orthogonal subspaces, such as $\text{Span}(\mathcal{S}_{n-k})$. Therefore, any difficulty function $\theta_A \in \text{Span}(\mathcal{S}_{n-k})$ will ensure that $q_{AB} = 0$.

## 5.3.3 Extremisation of $q_{AB}$ (given a Demand Profile and Difficulty Function)

Given a demand profile, $\bar{P}$, and a difficulty function, $\theta_B$, we can define the set of all difficulty functions such that the value of $q_{AB} = \langle \theta_B, \theta_A \rangle$ is some yet to be determined $q$. An arbitrary member of this set is denoted as $\theta_A$. In particular, specifying such a set tells us which difficulty functions are relevant in three cases: 1) those that result in the maximum $q_{AB}$ value; 2) those that result in the minimum $q_{AB}$ value, and; 3) those that result in $q_{AB} = q_A q_B$ (this is the case where the channels fail independently). In solving the problem we will use the components of vectors with respect to the orthonormal basis $\hat{\mathcal{S}}$, where these components may be obtained from the components with respect to the orthogonal basis $\mathcal{S}$ via the transformation $T$ of Section 4.4. We proceed as follows. An arbitrary difficulty function has the form

$$\theta_A := \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}, \quad \text{subject to the constraints} \quad \begin{matrix} 0 \leq \\ 0 \leq \\ 0 \leq \\ \vdots \\ 0 \leq \end{matrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \begin{matrix} \leq \sqrt{P_1} \\ \leq \sqrt{P_2} \\ \leq \sqrt{P_3} \\ \vdots \\ \leq \sqrt{P_n} \end{matrix} . \tag{5.6}$$

For an arbitrary member of the set of interest the value of $q_{AB}$ is $q$. That is,

$$q_{AB} = \langle \theta_A, \theta_B \rangle = x_1 \theta_B 1 + \ldots + x_n \theta_B n = q \tag{5.7}$$

which implies that for some non–zero $\theta_B$ component, say $\theta_B j \neq 0$, we have

$$x_j = \frac{1}{\theta_B j}\big(q - (x_1 \theta_B 1 + \ldots + x_{j-1}\theta_B(j-1) + x_{j+1}\theta_B(j+1) + \ldots + x_n \theta_B n)\big). \tag{5.8}$$

This equation constrains the values of $\{x_1, \ldots, x_n\}$, the $n$ unknown components of the arbitrary difficulty function $\theta_A$. Consequently, using Eq. (5.8) in Eq. (5.6), the form of a difficulty function that satisfies such a constraint is

$$\begin{pmatrix} x_1 \\ \vdots \\ x_{j-1} \\ \frac{1}{\theta_B j}\big(q - \ (x_1\theta_B 1 + \ \ldots \ + x_{j-1}\theta_B(j-1) + x_{j+1}\theta_B(j+1) + \ \ldots \ + x_n\theta_B n)\big) \\ x_{j+1} \\ \vdots \\ x_n \end{pmatrix} \quad . \tag{5.9}$$

The jth component of this difficulty must satisfy the constraints of Eq. (5.6). This implies that the value $q$ is constrained by the bounds

$$\begin{aligned} & x_1\theta_B 1 + \ \ldots \ + x_{j-1}\theta_B(j-1) + x_{j+1}\theta_B(j+1) + \ \ldots \\ & \qquad \ldots \ + x_n\theta_B n \leq q \leq \theta_B j\sqrt{P_j} + x_1\theta_B 1 + \ \ldots \\ & \qquad\qquad \ldots \ + x_{j-1}\theta_B(j-1) + x_{j+1}\theta_B(j+1) + \ \ldots \\ & \qquad\qquad \ldots \ + x_n\theta_B n \quad . \end{aligned}$$

These bounds suggest an approach for obtaining the maximum, or minimum, $q$ values. We demonstrate the approach by obtaining the maximum; the minimum is obtained by following an analogous argument. Note: so far the jth component has been chosen arbitrarily as the component determined by Eq. (5.7) apart from the requirement that $\theta_B j$ be non–zero. However, if we further require that $\theta_B j\sqrt{P_j}$ be the largest of the set $\{\theta_B 1\sqrt{P_1}, \ldots, \theta_B n\sqrt{P_n}\}$, then for the maximum $q$ to be obtained it is necessary that

$$\acute{q} = x_1\theta_B 1 + \ \ldots\ldots \ + x_{j-1}\theta_B(j-1) + x_{j+1}\theta_B(j+1) + \ \ldots \ + x_n\theta_B n, \tag{5.10}$$

where $\acute{q} = q - \theta_B j\sqrt{P_j}$. But, this has a similar form to the constraint in Eq. (5.7) except that the number of unknown components is reduced by one: that is, this constraint does not involve $x_j$. Thus, we may approach this in a similar way we approached Eq. (5.7). That is, the $n-1$ components $\{x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n\}$ of $\theta_A$ are constrained by Eq.'s (5.10) and (5.6). By continuing in this fashion until there are no more components to determine, and then backwards substituting to obtain the form of $\theta_A$ and the maximum value of $q_{AB}$, we have

$$q_{AB} = \theta_B 1\sqrt{P_1} + \ \ldots \ + \theta_B n\sqrt{P_n} = q_B \tag{5.11}$$

and the form of the difficulty function that makes this maximum $q_{AB}$ value possible is

$$\theta_A = \begin{pmatrix} \sqrt{P_1} \\ \sqrt{P_2} \\ \sqrt{P_3} \\ \vdots \\ \sqrt{P_n} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \quad . \tag{5.12}$$

This result is an alternative proof of Theorem 5.3.2 on page 138. Observe that Eq.'s (5.11) and (5.12) do not depend on the magnitude of $\theta_B$ implying that in this case the expected system *pfd* for a homogeneous, methodology–B system is unimportant in determining the maximum value of $q_{AB}$.

Instead of maximizing $q_{AB}$ we could have used this approach to determine the maximum covariation between an unknown difficulty function, $\theta_A$, and the given difficulty function, $\theta_B$. That is, given a demand profile $\bar{P}$ and a difficulty function $\theta_B$, we maximise

$$\underset{X}{\mathrm{Cov}}\left(\theta_A\left(X\right), \theta_B\left(X\right)\right) := \langle \theta_A, \theta_B - q_B \bar{P} \rangle. \tag{5.13}$$

The result of this is the following theorem. The components of vectors in this theorem are expressed with respect to the orthonormal basis $\hat{\mathbb{S}}$.

**Theorem 5.3.3.** *Given a demand profile, $\bar{P}$, and a difficulty function, $\theta_B$ let the related variance vector be $\theta_B - q_B\bar{P}$. There exist imperfect single versions, $\bar{V}$ and $\bar{W}$, with components $\left\{\bar{V}1, \ldots, \bar{V}n\right\}$ and $\left\{\bar{W}1, \ldots, \bar{W}n\right\}$, such that*

$$\bar{V}i = \begin{cases} \sqrt{P_i}, \text{ if } \theta_B i - q_B\sqrt{P_i} < 0 \\ 0, \text{ otherwise} \end{cases}, \quad \bar{W}i = \begin{cases} \sqrt{P_i}, \text{ if } \theta_B i - q_B\sqrt{P_i} > 0 \\ 0, \text{ otherwise} \end{cases},$$

*and the maximum and minimum values for a covariance of difficulty functions, one of the difficulty functions being $\theta_B$, are $\left\langle \theta_B - q_B\bar{P}, \bar{V}\right\rangle$ and $\left\langle \theta_B - q_B\bar{P}, \bar{W}\right\rangle$ respectively.*

As a consequence of Theorem 5.3.3, we see that Eq. (5.12) may be written as

$$\theta_A = \bar{V} + \bar{W} + \bar{U},$$

where the imperfect single version $\bar{U}$ is defined as having vector components, $\left\{\bar{U}1, \ldots, \bar{U}n\right\}$, such that

$$\bar{U}i := \begin{cases} \sqrt{P_i}, \text{ if } \theta_B i - q_B\sqrt{P_i} = 0 \\ 0, \text{ otherwise} \end{cases} \tag{5.14}$$

That is, *the difficulty function that results in the maximum expected system* pfd *under forced diversity, $q_{AB}$, is the sum of three difficulty functions that, respectively, result in the maximum, minimum and zero covariation with the given difficulty function $\theta_B$.*

### 5.3.4 Extremisation of $q_{AB}$ (given Demand Profile, $q_A$, Difficulty Function)

Given a demand profile $\bar{P}$, an expected version *pfd* $q_A$, and a difficulty function $\theta_B$, what is the maximum/minimum possible $q_{AB}$? We seek a difficulty function $\bar{V}_A$ with related mean

$q_A$ such that $\langle \bar{V}_A, \theta_B \rangle$ is minimised/maximised. Let

$$\bar{V}_A := \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}, \quad \text{subject to the constraints} \quad \begin{matrix} 0 \leq \\ 0 \leq \\ 0 \leq \\ \vdots \\ 0 \leq \end{matrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \begin{matrix} \leq \sqrt{P_1} \\ \leq \sqrt{P_2} \\ \leq \sqrt{P_3} \\ \vdots \\ \leq \sqrt{P_n} \end{matrix} \tag{5.15}$$

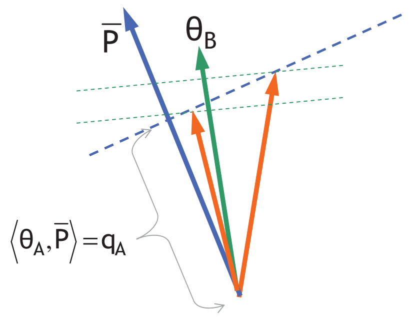


**Figure 5.5:** Maximisation of $q_{AB}$, given $\theta_B$. For a fixed mean, $q_A$, we seek an appropriate difficulty function by moving away from the demand profile in a direction that increases the value of its inner–product with $\theta_B$. This increase in inner–product value is illustrated by the fact that moving a difficulty function close to $\bar{P}$ to a difficulty function further away from $\bar{P}$, while keeping the mean fixed at $q_A$ (this is the requirement that the arrow head always touch the thick dashed–line), will cause the difficulty function to touch increasingly "higher" thin dashed–lines, each of these thin dashed–lines being orthogonal to $\theta_B$. So, the inner–product of the difficulty function that is further away is greater.

Recall, these constraints are equivalent, under transformation $T$ of Section 4.4, to the usual constraints of difficulty function values necessarily lying in the closed unit interval. Extremising $q_{AB}$, given difficulty $\theta_B$ and $q_A$, is an n-dimensional problem with the 2 constraints,

$$\begin{aligned} \langle \bar{V}_A, \theta_B \rangle &= x_1\theta_B 1 + x_2\theta_B 2 + \ldots + x_n\theta_B n = q_{AB} \\ \langle \bar{V}_A, \bar{P} \rangle &= x_1\sqrt{P_1} + x_2\sqrt{P_2} + \ldots + x_n\sqrt{P_n} = q_A \end{aligned} \tag{5.16}$$

For this set of equations to really represent 2 constraints we require $\theta_B \neq \bar{P}$. In the case where $\theta_B$ and $\bar{P}$ are collinear, any $\bar{V}_A$ ensures the value $q_A$ for $q_{AB}$ and the problem is solved. Appreciate that in stating these constraints we have written the unknown $q_{AB}$ as though it was a known value constraining the problem. The purpose of this "trick" is to ultimately put $q_{AB}$ at the center of the constraints on $\bar{V}_A$ detailed in Eq. (5.15). The advantage of this is that it ensures that attainable bounds can be defined on $q_{AB}$, and the bounds necessarily take these constraints on difficulty functions into account. This set of linear equations can be cast in the form of a matrix, and the usual methods of row operations can be used to "eliminate" two of the variables $x_1, \ldots, x_n$. In so doing Eq. (5.15) becomes

$$\begin{array}{c} 0 \leq \\ 0 \leq \\ 0 \leq \\ \vdots \\ 0 \leq \end{array} \left( \begin{array}{c} \frac{q_{AB}\sqrt{P_2} - q_A\theta_B 2}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} - \left( \frac{\theta_B 3\sqrt{P_2} - \theta_B 2\sqrt{P_3}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_3 + \ldots + \frac{\theta_B n\sqrt{P_2} - \theta_B 2\sqrt{P_n}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_n \right) \\ \frac{q_A\theta_B 1 - q_{AB}\sqrt{P_1}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} - \left( \frac{\theta_B 1\sqrt{P_3} - \theta_B 3\sqrt{P_1}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_3 + \ldots + \frac{\theta_B 1\sqrt{P_n} - \theta_B n\sqrt{P_1}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_n \right) \\ x_3 \\ \vdots \\ x_n \end{array} \right) \begin{array}{c} \leq \sqrt{P_1} \\ \leq \sqrt{P_2} \\ \leq \sqrt{P_3}, \\ \vdots \\ \leq \sqrt{P_n} \end{array} \tag{5.17}$$

where $\theta_B 1\sqrt{P_2} \neq \theta_B 2\sqrt{P_1}$. It is always possible to choose components that satisfy $\theta_B 1\sqrt{P_2} \neq \theta_B 2\sqrt{P_1}$ since $\theta_B$ and $\bar{P}$ are not collinear and, consequently, $\theta_B$ is not a constant difficulty function. Let us choose the components so that $\theta^*_B 1 = \frac{\theta_B}{\sqrt{P_1}}$ and $\theta^*_B 2 = \frac{\theta_B}{\sqrt{P_2}}$ are the highest and lowest $\theta^*_B$ values, respectively. In this case, $\theta^*_B 1 > \theta^*_B 2$ and therefore $\theta_B 1\sqrt{P_2} > \theta_B 2\sqrt{P_1}$. Using the following shorthand notation,

$$()_1 := \left( \frac{\theta_B 3\sqrt{P_2} - \theta_B 2\sqrt{P_3}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_3 + \ldots + \frac{\theta_B n\sqrt{P_2} - \theta_B 2\sqrt{P_n}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_n \right)$$

and

$$()_2 := \left( \frac{\theta_B 1\sqrt{P_3} - \theta_B 3\sqrt{P_1}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_3 + \ldots + \frac{\theta_B 1\sqrt{P_n} - \theta_B n\sqrt{P_1}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_n \right),$$

observe that from the constraints on $x_1$ and $x_2$ in Eq. (5.17) we may define the following upper bounds on $q_{AB}$.

$$q_{AB} \leq \frac{1}{\sqrt{P_2}} \left[ \left( ()_1 + \sqrt{P_1} \right) \left( \theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1} \right) + q_A\theta_B 2 \right] \tag{5.18}$$

$$q_{AB} \leq \frac{-1}{\sqrt{P_1}} \left[ ()_2 \left( \theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1} \right) - q_A\theta_B 1 \right] \tag{5.19}$$

Similarly, we can also define lower bounds on $q_{AB}$.

$$q_{AB} \geq \frac{1}{\sqrt{P_2}} \left[ ()_1 \left( \theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1} \right) + q_A\theta_B 2 \right]$$

$$q_{AB} \geq \frac{-1}{\sqrt{P_1}} \left[ \left( ()_2 + \sqrt{P_2} \right) \left( \theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1} \right) - q_A\theta_B 1 \right]$$

Here, we see the advantage of treating $q_{AB}$ as if it were a known quantity. For, the upper bound on $q_{AB}$ is manifestly dependent on the unknown quantities $x_1$, ..., $x_n$ which, in turn, are subject to the difficulty function constraints in Eq. (5.15). As a result, all of the constraints in our optimisation problem are brought to bear on the value of $q_{AB}$ in an obvious way. Also, the symmetry of these upper and lower bounds hint at a more general point about the current optimisation problem: the minimisation of $q_{AB}$ follows analogous steps to the maximisation of $q_{AB}$. So, while we choose to concentrate on the maximisation problem from here on, the minimisation problem develops in an almost identical manner.

An attainable upper bound for $q_{AB}$ is given by the smaller of the two upper bounds stated in Eq.'s (5.18) and (5.19). We are free to choose which upper bound is the smaller of the two as either case will lead us to the same answer. Suppose the upper bound in Eq. (5.18) is less than, or equal to, the upper bound in Eq. (5.19). This would be the case if, and only if,

$$q_A - P_1 \geq x_3\sqrt{P_3} + \ \ldots \ + x_n\sqrt{P_n} \geq 0 \tag{5.20}$$

In particular, this suggests that the unknown quantity $x_3\sqrt{P_3} + \ \ldots \ + x_n\sqrt{P_n}$ should equal $q_A - P_1$, at least. Observe that the form of the upper bound in Eq. (5.18) is such that it is linear in $()_1$. Also, $()_1$ is itself linear in the unknowns $x_1, \ldots, x_n$. So, optimising this upper bound on $q_{AB}$ results in an n–2 dimensional Linear optimisation problem with 2 constraints. That is,

$$\begin{aligned} \acute{q} &= \frac{\theta_B 3\sqrt{P_2} - \theta_B 2\sqrt{P_3}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_3 + \ldots + \frac{\theta_B n\sqrt{P_2} - \theta_B 2\sqrt{P_n}}{\theta_B 1\sqrt{P_2} - \theta_B 2\sqrt{P_1}} x_n \\ q_A - P_1 &= x_3\sqrt{P_3} + x_4\sqrt{P_4} + \ldots + x_n\sqrt{P_n}, \end{aligned} \tag{5.21}$$

for some as yet to be determined real number $\acute{q}$. Compare this with the original n–dimensional problem in Eq. (5.16). The problems have identical forms and so can be approached in the same manner. The obvious advantage now is that the problem stated in Eq. (5.21) is of lower dimensionality. Solving this lower dimensional problem by using the approach already utilised for the higher dimensional problem results in yet another lower dimensional problem: this process is iterative. There are two possible end points to this process: either the dimensionality of the problem ultimately reaches 2–dimensions (for even number $n$) or 3–dimensions (for odd number $n$). Upon reaching any of these stages the problem becomes deterministic, in which case components can be determined in the lower dimensional problems, and backward substitution can be used to determine components in higher dimensional problems until all of the components are determined. Ultimately, this can lead to a number of different forms for $\bar{V}_A$ in Eq. (5.15), including:

$$\bar{V}_A = \begin{pmatrix} \sqrt{P_1} \\ 0 \\ \sqrt{P_3} \\ \vdots \\ 0 \\ \dfrac{q_A - \sum\limits_{i=1}^{\frac{j-1}{2}} P_{2i-1}}{\sqrt{P_j}} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} \sqrt{P_1} \\ 0 \\ \sqrt{P_3} \\ \vdots \\ 0 \\ \sqrt{P_j} \\ \dfrac{q_A - \sum\limits_{i=1}^{\frac{j+3}{2}} P_{2i-1}}{\sqrt{P_{j+1}}} \\ \sqrt{P_{j+2}} \end{pmatrix}, \begin{pmatrix} \sqrt{P_1} \\ 0 \\ \sqrt{P_3} \\ \vdots \\ 0 \\ \sqrt{P_j} \\ \dfrac{q_A - \sum\limits_{i=1}^{n} P_i + \sum\limits_{k=1}^{\frac{j-1}{2}} P_{2k}}{\sqrt{P_{j+1}}} \\ \sqrt{P_{j+2}} \\ \vdots \\ \sqrt{P_{n-1}} \\ \sqrt{P_n} \end{pmatrix}, \begin{pmatrix} \sqrt{P_1} \\ 0 \\ \sqrt{P_3} \\ \vdots \\ 0 \\ \sqrt{P_j} \\ \dfrac{q_A - \sum\limits_{i=1}^{\frac{j+1}{2}} P_{2i-1}}{\sqrt{P_{j+1}}} \end{pmatrix}, \tag{5.22}$$

where $j$ is some odd number. Under the transformation $T^{-1}$ we may write these vectors in LM–model coordinates as

$$\bar{V}_A = \begin{pmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 0 \\ \dfrac{q_A - \sum_{i=1}^{\frac{j-1}{2}} P_{2i-1}}{P_j} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ \dfrac{q_A - \sum_{i=1}^{\frac{j+3}{2}} P_{2i-1}}{P_{j+1}} \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ \dfrac{q_A - \sum_{i=1}^{n} P_i + \sum_{k=1}^{\frac{j-1}{2}} P_{2k}}{P_{j+1}} \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ \dfrac{q_A - \sum_{i=1}^{\frac{j+1}{2}} P_{2i-1}}{P_{j+1}} \end{pmatrix}. \tag{5.23}$$

This illustrates that there are a number of possible forms for the difficulty. So, in these cases maximum $q_{AB}$ has the following forms.

$$q_{AB} = \langle \bar{V}_A, \theta^*_B \rangle = P_1\theta^*_B 1 + \ldots + P_{j-2}\theta^*_B(j-2) + \left( q_A - \sum_{i=1}^{\frac{j-1}{2}} P_{2i-1} \right) \theta^*_B j,$$

$$q_{AB} = \langle \bar{V}_A, \theta^*_B \rangle = P_1\theta^*_B 1 + \ldots + P_{j+2}\theta^*_B(j+2) + \left( q_A - \sum_{i=1}^{\frac{j+3}{2}} P_{2i-1} \right) \theta^*_B (j+1),$$

$$q_{AB} = \langle \bar{V}_A, \theta^*_B \rangle = P_1\theta^*_B 1 + \ldots + P_n\theta^*_B n + \left( q_A - \sum_{i=1}^{n} P_i + \sum_{k=1}^{\frac{j-1}{2}} P_{2k} \right) \theta^*_B (j+1),$$

$$q_{AB} = \langle \bar{V}_A, \theta^*_B \rangle = P_1\theta^*_B 1 + \ldots + P_j\theta^*_B j + \left( q_A - \sum_{i=1}^{\frac{j+1}{2}} P_{2i-1} \right) \theta^*_B (j+1).$$

We could have approached the same problem another way. From Fig. 5.5 we can appreciate that the further away a non–zero vector, $\theta_A$, is from $\bar{P}$ – in a preferred direction determined by another vector $\theta_B$ – the larger the projection of $\theta_A$ on $\theta_B$. This preferred direction is specified by requiring that the projection of $\theta_A$ in $\theta_B$'s direction should be larger than the projection of $\theta_A$ if it was collinear with $\theta_B$. The condition that $\theta_A$ be collinear with $\theta_B$ means that $\theta_A$ is a scalar multiple of the unit vector in the direction of $\theta_B$. That is, $\theta_A = q\dfrac{\theta_B}{\|\theta_B\|}$ for some non–zero real number $q$. The scaling $q$ may be determined by the expectation of $\theta_A$ having a value $q_A$. So,

$$q_A = \langle \theta_A, \bar{P} \rangle = \langle q\frac{\theta_B}{\|\theta_B\|}, \bar{P} \rangle = q\langle \frac{\theta_B}{\|\theta_B\|}, \bar{P} \rangle = q\frac{q_B}{\|\theta_B\|}.$$

Therefore, $q = \frac{q_A \|\theta_B\|}{q_B}$ and $\theta_A = \frac{q_A \theta_B}{q_B}$. Consequently, the length of the projection of $\theta_A$ in the direction of a collinear $\theta_B$ is $\left\langle \theta_A, \frac{\theta_B}{\|\theta_B\|} \right\rangle = \frac{q_A\|\theta_B\|}{q_B}$. Accordingly, we are interested in those difficulty functions with projections in the direction of $\theta_B$ that are larger than this. That is, we seek $\theta_A$ such that

$$\left\langle \theta_A, \frac{\theta_B}{\|\theta_B\|} \right\rangle \geq \frac{q_A\|\theta_B\|}{q_B} \Leftrightarrow \left\langle \theta_A, \frac{q_B}{\|\theta_B\|^2}\theta_B - \bar{P} \right\rangle \geq 0$$

So, a proof along these lines will require obtaining a difficulty function, $\theta_A$, which has a mean $q_A$ and a projection in the direction of the known vector $\frac{q_B}{\|\theta_B\|^2}\theta_B - \bar{P}$ that is as large as possible.

### 5.3.5 Extremisation of $q_{AB}$ (given values for $q_A, q_B, q_{AA}$, and $q_{BB}$)

Given $q_A$, $q_B$, $q_{AA}$ and $q_{BB}$ what is the maximum/minimum possible $q_{AB}$? Again, we ignore the trivial cases when either $q_{AA}$ or $q_{BB}$ equals zero. By ignoring these cases, we are asserting that $q_A$ and $q_B$ are non-zero as well. Let $\theta_A$ and $\theta_B$ be the non–zero, arbitrary difficulty functions with these means, respectively. Also, suppose $\bar{P}$ is some applicable demand profile. The size of $q_{AB}$ is dependent on how close together, or far apart, the difficulty functions are. In particular, this angular distance is minimized and maximised if $\theta_A$, $\theta_B$ and $\bar{P}$ are coplanar. That is, extremisation occurs when these three vectors lie in a 2-dimensional plane spanned by any two of these vectors that are not collinear. For minimisation, we require $\theta_A$ and $\theta_B$ to lie on either side of $\bar{P}$. Contrastingly, for maximisation, the difficulty functions should lie on the same side of $\bar{P}$. This makes sense since, via the Cauchy-Schwarz inequality, perpendicularity results in a $q_{AB}$ that is zero, and collinearity results in $q_{AB}$ that is largest. Additionally, this relationship between geometry and extremisation lends itself to "natural language" expressions. For example, difficulty functions that are close together (i.e. possess a sufficiently small angle between them) can be said to be "similar", "positively correlated" or "not very diverse". Alternatively, if a sufficiently large angle exists between them they can be said to be "dissimilar", "negatively correlated" or "very diverse". Let $\gamma_A$ and $\gamma_B$ be the respective angles the difficulty functions make with $\bar{P}$. Using the condition for minimisation/maximisation of $q_{AB}$ the angle between the difficulty functions should be $\gamma_A \pm \gamma_B$. Therefore,

$$\begin{aligned}
\langle \theta_A, \theta_B \rangle &= \|\theta_A\| \|\theta_B\| \cos(\gamma_A \pm \gamma_B) = \|\theta_A\| \|\theta_B\| \cos\left(\cos^{-1}\left(\frac{q_A}{\|\theta_A\|}\right) \pm \cos^{-1}\left(\frac{q_B}{\|\theta_B\|}\right)\right) \\
&= \|\theta_A\| \|\theta_B\| \left( \frac{q_A q_B}{\|\theta_A\| \|\theta_B\|} \mp \frac{\sqrt{\left(\|\theta_A\|^2 - {q_A}^2\right)\left(\|\theta_B\|^2 - {q_B}^2\right)}}{\|\theta_A\| \|\theta_B\|} \right) \\
&= q_A q_B \mp \sqrt{(q_{AA} - {q_A}^2)(q_{BB} - {q_B}^2)}
\end{aligned}$$

Consequently, the minimum/maximum value of $q_{AB}$ is

$$\langle \theta_A, \theta_B \rangle = q_A q_B \mp \sqrt{\left(q_{AA} - {q_A}^2\right)\left(q_{BB} - {q_B}^2\right)} \tag{5.24}$$

In terms of the probabilistic LM model Eq. (5.24) uses the maximum value of the magnitude of $\underset{X}{\mathrm{Cov}}\left(\theta_A(X), \theta_B(X)\right)$, which is $\sqrt{\left(q_{AA} - {q_A}^2\right)\left(q_{BB} - {q_B}^2\right)}$, to maximise $q_{AB}$. That is, Eq. (5.24) is a special case of the LM equation, Eq. (2.15), with maximum magnitude for the covariance.

Actually, the bounds given above are always attainable in the general context of an inner–product space. For the bounds to be particularly relevant to difficulty functions they need to be restricted to some region defined by some $\bar{P}$. By definition, a non–zero difficulty function must make a non–negative angle of size strictly less than 90° with some $\bar{P}$. In order for the bounds to be attainable we require a pair of difficulty functions, with mean characteristics as given, that fit in some appropriate bounding region. In what follows we treat the lower and the upper bounds in turn, defining necessary and sufficient conditions for attaining these bounds in a region defined by some $\bar{P}$.

For the lower bound, the largest possible angle between the difficulty functions should be 90°, as this is the largest separation between difficulty functions that is possible in a bounding region. In fact, this is the condition for the lowest possible lower bound of the inner–product of two difficulty functions: it is the case for which $q_{AB} = 0$. If $\gamma_A + \gamma_B \geq \frac{\pi}{2}$ then it is possible [12] to construct 2 or 3 dimensional demand profiles such that $\langle \theta_A, \theta_B \rangle = q_{AB} = 0$. In fact, if $\gamma_A + \gamma_B > \frac{\pi}{2}$ then the relevant vectors are

$$\theta_A := \begin{pmatrix} \|\theta_A\| \\ 0 \\ 0 \end{pmatrix}, \ \theta_B := \begin{pmatrix} 0 \\ \|\theta_B\| \\ 0 \end{pmatrix}, \ \text{and } \bar{P} := \begin{pmatrix} \dfrac{q_A}{\|\theta_A\|} \\ \\ \dfrac{q_B}{\|\theta_B\|} \\ \\ \sqrt{1 - \left(\left(\dfrac{q_A}{\|\theta_A\|}\right)^2 + \left(\dfrac{q_B}{\|\theta_B\|}\right)^2\right)} \end{pmatrix}.$$

Similarly, if $\gamma_A + \gamma_B = \frac{\pi}{2}$, then an example of a set of vectors that satisfy the zero bound is

$$\theta_A := \begin{pmatrix} \|\theta_A\| \\ 0 \end{pmatrix}, \ \theta_B := \begin{pmatrix} 0 \\ \|\theta_B\| \end{pmatrix}, \ \text{and } \bar{P} := \begin{pmatrix} \dfrac{q_A}{\|\theta_A\|} \\ \\ \dfrac{q_B}{\|\theta_B\|} \end{pmatrix}.$$

[12] As a side–note the condition $\gamma_A + \gamma_B \geq \frac{\pi}{2}$ is equivalent to either of the requirements

$$\cos\left(\cos^{-1}\left(\frac{q_A}{\|\theta_A\|}\right) + \cos^{-1}\left(\frac{q_B}{\|\theta_B\|}\right)\right) \leq 0, \quad \text{or} \quad \frac{{q_A}^2}{q_{AA}} + \frac{{q_B}^2}{q_{BB}} \leq 1. \tag{5.25}$$

Alternatively, when $\gamma_A+\gamma_B < \frac{\pi}{2}$ we can define a pair of difficulty functions that are coplanar with the demand profile and, therefore, satisfy the lower bound given by Eq. (5.24) *if, and only if,*

$$\sin(\gamma_A+\gamma_B) \leq \frac{\max\left\{\sqrt{\|\theta_A\|^2 - q_A{}^2}, \sqrt{\|\theta_B\|^2 - q_B{}^2}\right\}}{\|\theta_A\|\,\|\theta_B\|}. \tag{5.26}$$

This follows from the requirement that vectors in the direction of $\theta_A$ and $\theta_B$ lie in a 2–dimensional bounding box with a unit vector diagonal if, and only if, such a bounding box itself lies inbetween 2 extreme bounding boxes with sides

$$\left\{\begin{pmatrix} 0 \\ \frac{q_A}{\|\theta_A\|}\end{pmatrix}, \begin{pmatrix} \sqrt{1-\left(\frac{q_A}{\|\theta_A\|}\right)^2} \\ 0 \end{pmatrix}\right\} \text{ and } \left\{\begin{pmatrix} \sqrt{1-\left(\frac{q_B}{\|\theta_B\|}\right)^2} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \frac{q_B}{\|\theta_B\|}\end{pmatrix}\right\}.$$

An example of this is depicted in Fig. 5.6. If the condition in Eq. (5.26) holds, then we

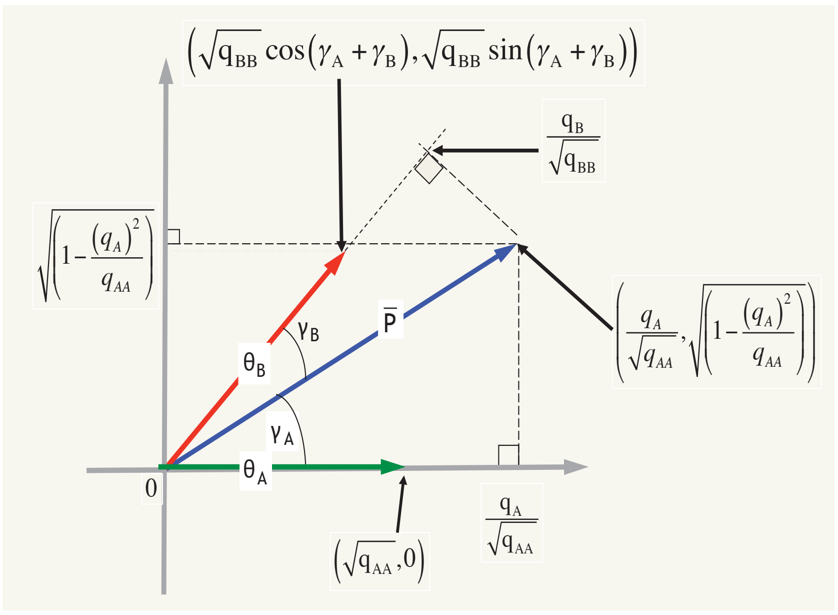

$\left(\sqrt{q_{BB}}\cos(\gamma_A+\gamma_B), \sqrt{q_{BB}}\sin(\gamma_A+\gamma_B)\right)$
$\frac{q_B}{\sqrt{q_{BB}}}$
$\sqrt{1-\left(\frac{q_A}{q_{AA}}\right)^2}$
$\bar{P}$
$\gamma_B$
$\theta_B$
$\left(\frac{q_A}{\sqrt{q_{AA}}}, \sqrt{1-\left(\frac{q_A}{q_{AA}}\right)^2}\right)$
$\gamma_A$
$\theta_A$
0
$\frac{q_A}{\sqrt{q_{AA}}}$
$\left(\sqrt{q_{AA}}, 0\right)$

**Figure 5.6:** An example of a pair of difficulty functions, $\theta_A$ and $\theta_B$, for which $q_{AB}$ is minimum.

may define the following set of vectors that hit the lower bound specified by Eq. (5.24).

$$\theta_A := \begin{pmatrix} \|\theta_A\| \\ 0 \end{pmatrix}, \; \theta_B := \begin{pmatrix} \|\theta_B\|\cos(\gamma_A+\gamma_B) \\ \|\theta_B\|\sin(\gamma_A+\gamma_B) \end{pmatrix} \text{ and } \bar{P} := \begin{pmatrix} \frac{q_A}{\|\theta_A\|} \\ \sqrt{1-\left(\frac{q_A}{\|\theta_A\|}\right)^2} \end{pmatrix}. \tag{5.27}$$

We switch our attention to the upper bound. For the upper bound to be attained it is necessary that the angle between each vector and the demand space is strictly less than 90°. Without loss of generality suppose $\frac{\pi}{2} > \gamma_A \geq \gamma_B$. Therefore, $\frac{\pi}{2} > \gamma_A \geq \gamma_A - \gamma_B$ which

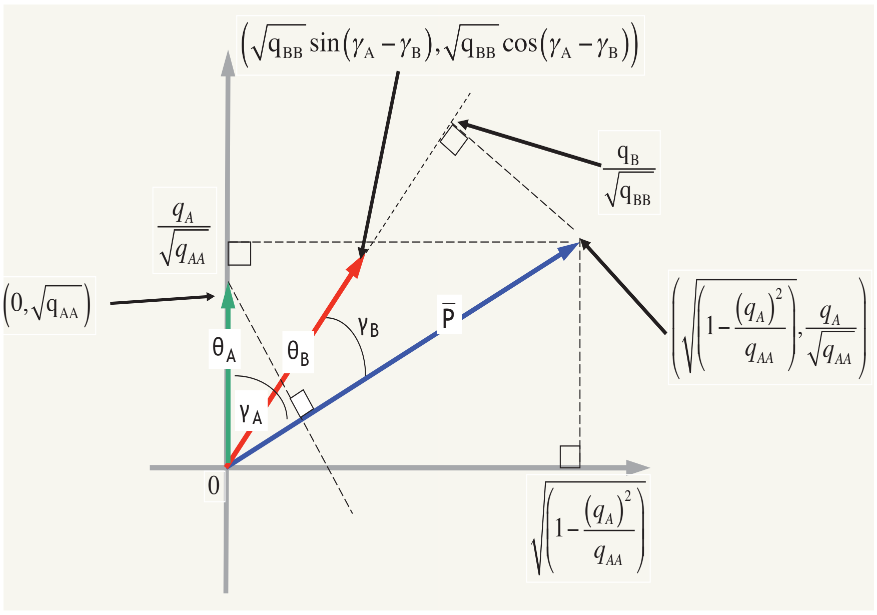

$\left(\sqrt{q_{BB}}\sin(\gamma_A-\gamma_B),\sqrt{q_{BB}}\cos(\gamma_A-\gamma_B)\right)$
$\frac{q_B}{\sqrt{q_{BB}}}$
$\frac{q_A}{\sqrt{q_{AA}}}$
$(0,\sqrt{q_{AA}})$
$\left(\sqrt{\left(1-\frac{(q_A)^2}{q_{AA}}\right)},\frac{q_A}{\sqrt{q_{AA}}}\right)$
$\gamma_B$
$\bar{P}$
$\theta_A$
$\theta_B$
$\gamma_A$
0
$\sqrt{\left(1-\frac{(q_A)^2}{q_{AA}}\right)}$

**Figure 5.7:** An example of a pair of difficulty functions, $\theta_A$ and $\theta_B$, for which $q_{AB}$ is maximum.

implies $\frac{q_A}{\|\theta_A\|} \leq \cos(\gamma_A - \gamma_B)$; that is, in the orientation where $\theta_A$ and $\theta_B$ are coplanar with, and on the same side of, $\bar{P}$ it is necessary that $\theta_A$ be closer to $\theta_B$ than it is to $\bar{P}$. Observe that vectors in the direction of $\theta_A$ and $\theta_B$ can be contained in a 2–dimensional bounding region defined by a unit vector if and only if such a region lies between two extremes, one of which is a valid bounding box and the other is a degenerate, unattainable bounding box. The attainable bounding box has some unit vector, $\bar{P}$, as its diagonal while the other extreme is the vector $\bar{P}$ and the zero–vector. The extreme cases are "separated" by an orientation angle of $\frac{\pi}{2} - (\gamma_A - \gamma_B)$ and the sides of the attainable extreme bounding box are

$$\left\{ \begin{pmatrix} \sqrt{1 - \left(\frac{q_A}{\|\theta_A\|}\right)^2} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \frac{q_A}{\|\theta_A\|} \end{pmatrix} \right\}.$$

An example of such a bounding box is given in Fig. 5.7. So, both vectors $\theta_A$ and $\theta_B$ are contained in some acceptable bounding region so that the upper bound on $q_{AB}$ given by Eq. (5.24) is attained *if, and only if,*

$$\cos(\gamma_A - \gamma_B) \leq \frac{q_A}{\|\theta_A\| \, \|\theta_B\|}. \tag{5.28}$$

This is the requirement that the projection of $\theta_B$ in the direction of $\theta_A$ must be shorter than the projection of $\bar{P}$ in the direction of $\theta_A$. Using this we may define a pair of difficulty functions, subject to the constraints on their means, such that their inner–product is a maximum.

$$\theta_A := \begin{pmatrix} 0 \\ \|\theta_A\| \end{pmatrix}, \ \theta_B := \begin{pmatrix} \|\theta_B\| \sin(\gamma_A - \gamma_B) \\ \|\theta_B\| \cos(\gamma_A - \gamma_B) \end{pmatrix} \text{ and } \bar{P} := \begin{pmatrix} \sqrt{1 - \left(\frac{q_A}{\|\theta_A\|}\right)^2} \\ \frac{q_A}{\|\theta_A\|} \end{pmatrix}.$$

## 5.3.6 Maximisation of $q_{AB}$ (given a Demand Profile, $q_A$ and $q_B$)

Given a demand profile $\bar{P}$, and values for both expected *pfd*s $q_A$ and $q_B$, what is the maximum/minimum attainable value for $q_{AB}$? The perfect version, i.e. the zero difficulty function, is the only difficulty function orthogonal to $\bar{P}$. As a consequence either $q_A$ or $q_B$ equal to zero implies that the difficulty functions thereof must be the zero vector, which guarantees that $q_{AB} = 0$. Therefore, we focus on non–zero difficulty functions and, equivalently, $q_A, q_B \neq 0$. In seeking a pair of difficulty functions that ensure the maximum value of $q_{AB}$ we consider four cases in turn:

*Case 1*: If $q_A = q_B$, then both difficulty functions must touch the same plane. We know that for fixed magnitudes a pair of difficulty functions will have their maximum inner–product when they are collinear. Furthermore, since the difficulty functions have the same first moment, $q_A$, they must be no larger than the largest possible difficulty function touching the $q_A$ plane. Consequently, the solution to the maximisation problem in this case is to assign both difficulty functions to be the largest difficulty function touching the $q_A$ plane. Therefore, the largest value for $q_{AB}$ is the value of $q_{AA}$ for the largest difficulty function (see Section 4.7.4) with mean $q_A$.

*Case 2* Suppose that the values of $q_A$ and $q_B$ are such that the largest difficulty functions – say $\theta_A$ and $\theta_B$ – touching the $q_A$ and $q_B$ planes, lie in the same 1–dimensional subspace $\text{Span}(\bar{V})$, where $\bar{V} \in \mathcal{S}$. So, $\theta_A$ and $\theta_B$ are collinear. Consequently, $q_{AB}$ is maximised by these difficulty functions and its value is

$$q_{AB} = \langle \theta_A, \theta_B \rangle = \left\langle \frac{q_A}{\|\bar{V}\|}\hat{V}, \frac{q_B}{\|\bar{V}\|}\hat{V} \right\rangle = \frac{q_A}{\|\bar{V}\|}\frac{q_B}{\|\bar{V}\|}\left\langle \hat{V}, \hat{V} \right\rangle = \frac{q_A}{\|\bar{V}\|}\frac{q_B}{\|\bar{V}\|}.$$

*Case 3*: If there exists a version $\bar{V}$ such that, without loss of generality, $q_A < \|\bar{V}\| < q_B$, then the candidate pair of difficulty functions to be found, $\theta_A$ and $\theta_B$, touch different planes. Obtain the largest difficulty function that touches the $q_A$ plane and define this as $\theta_A$. Now we have $\bar{P}$, $q_B$ and a difficulty function and we wish to obtain another difficulty function $\theta_B$ that together with $\theta_A$ maximises $q_{AB}$. But this is precisely the extremisation problem dealt with in Section 5.3.4. Therefore, using this technique we obtain a difficulty function $\theta_B$ that indeed maximises $q_{AB}$ with $\theta_A$. For example, suppose $q_A < P_1 + P_3 + \ldots + P_{j-2} + P_j < q_B$, and $\theta_A$ has the form:

$$\theta_A = \begin{pmatrix} \sqrt{P_1} \\ 0 \\ \sqrt{P_3} \\ \vdots \\ 0 \\ \sqrt{P_{j-2}} \\ 0 \\ \dfrac{\left(q_A - \sum_{i=1}^{(j-1)/2} P_{2i-1}\right)}{\sqrt{P_j}} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \theta_B = \begin{pmatrix} \sqrt{P_1} \\ 0 \\ \sqrt{P_3} \\ \vdots \\ 0 \\ \sqrt{P_j} \\ 0 \\ \vdots \\ \dfrac{\left(q_B - \sum_{i=1}^{(r-1)/2} P_{2i-1}\right)}{\sqrt{P_r}} \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The largest $q_{AB}$ is then

$$\langle \theta_A, \theta_B \rangle = P_1 + P_3 + \ldots + P_{j-2} + \frac{\left(q_A - \sum_{i=1}^{(j-1)/2} P_{2i-1}\right)}{\sqrt{P_j}} \sqrt{P_j} + 0 \cdot \sqrt{P_{j+2}} + \ldots$$
$$\ldots + 0 \cdot \frac{\left(q_B - \sum_{i=1}^{(r-1)/2} P_{2i-1}\right)}{\sqrt{P_r}} = q_A.$$

That is, the maximum $q_{AB}$ is the lower of the two expected version *pfd*s.

*Case 4* The algorithm used in the previous case, where $\theta_A$ was maximised and $\theta_B$ was defined using the algorithm of Section 5.3.4, is quite general. It gives the correct solution in *case*s' *1, 2* and *3*. We suspect that this algorithm gives the correct solution in the general case where no restrictions are placed on $q_A$ and $q_B$. One reason is the fact that *case 1* and *case 3* are "extremes": in *case 1* the planes are identical and in *case 3* the planes are separated by a version. If, starting from the situation with overlapping planes, a plane is moved by a "very small" amount we may ask for a pair of difficulty functions that maximise $q_{AB}$ in this configuration. Since $\mathbb{R}^n$ is continuous it seems plausible that the pair of vectors which give the largest $q_{AB}$ value will be "close" to the pair of vectors that maximised $q_{AB}$ when the planes were overlapping. However, while we suspect this algorithm to result in the correct solution we have not proved this and it remains the focus of future work.

### 5.3.7 Maximisation of $q_{AB}$ (given a Demand Profile, $q_{AA}$ and $q_{BB}$)

**Theorem 5.3.4.** *Given a demand profile $\bar{P}$, and expected* pfd*s $q_{AA}$ and $q_{BB}$, the maximum possible value for the expected* pfd *$q_{AB}$ is $\sqrt{q_{AA}q_{BB}}$.*

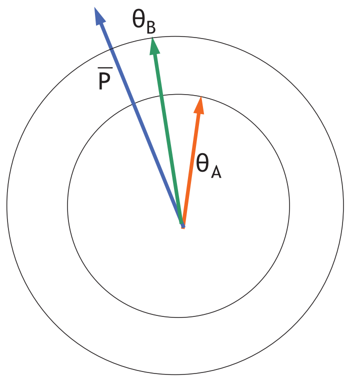


**Figure 5.8:** We seek a pair of difficulty functions that maximise $q_{AB}$, with the requirement that one of the vectors touches the sphere of radius $\sqrt{q_{BB}}$ and the other touches the sphere of radius $\sqrt{q_{AA}}$. From the diagram collinearity of the vectors will give the largest inner–product of the vectors. Since $\|\bar{P}\| = 1$ it is always possible to define a pair of collinear difficulty functions of the given sizes collinear with the demand profile.

*Proof.* We deal exclusively with the case where both $q_{AA}$ and $q_{BB}$ are non–zero; for if one of these quantities is zero, then $q_{AB}$ is zero. For non–zero $q_{AA}$ and $q_{BB}$, constant difficulty functions with magnitudes $\sqrt{q_{AA}}$ and $\sqrt{q_{BB}}$ attain the maximum $q_{AB}$ given by the Cauchy–Schwarz upper bound. These are the difficulty functions $\theta_A = \sqrt{q_{AA}}\bar{P}$ and $\theta_B = \sqrt{q_{BB}}\bar{P}$. ■

## 5.4 Summary

It is tempting to use the equations of the LM model as a practical recipe for obtaining numerical values of reliability measures. However, the LM model is a conceptual model with measures of interest, such as difficulty functions, that may be unknowable in practice. This can make direct use of the LM model – as a tool for inference – problematic. Furthermore, even if one had estimates for some of the measures (for instance, the expected *pfd* of the channels $q_A$), using these estimates in the LM model may not be enough to obtain the values of other measures of interest (for instance, the expected system *pfd* $q_{AA}$). The problem here is that the relationship between a known $q_A$ and an unknown $q_{AA}$ defines *a family of difficulty functions with the same mean, but possibly different variation over the demand space*[13]. This is easily seen on the right–hand–side of Eq. (2.13) on page 43, which we reproduce here, where the variance term's value is constrained by knowing the value of $q_A$ but still unknown.

$$q_{AA} = (q_A)^2 + \underset{X}{\mathrm{Var}}\left(\theta_A(X)\right)$$

[13]An equivalent geometric characterisation of this is that $q_A$ defines a family of difficulty functions, with possibly different magnitudes, that touch the same plane. The plane in question is orthogonal to the demand profile, and lies at a distance $q_A$ from the origin.

So, the value of $q_{AA}$ is not uniquely determined by $q_A$. Nevertheless, $q_{AA}$ is constrained by $q_A$ so that it is possible to obtain difficulty functions that result in extreme values for $q_{AA}$; in effect bounding the possible values of $q_{AA}$. This chapter has been concerned with defining such bounds on the expected system *pfd* resulting from forcing diversity under various scenarios. In each scenario a set of parameter values are assumed as known, and the related expected system *pfd* resulting from forcing diversity is either maximised (worst case) or minimized (best case). The scenarios differ in the constraints on the optimisation problem: the latter scenarios in the chapter being more constrained than the former scenarios. Nevertheless, in each extremisation problem, the bound obtained is attainable, and a pair of difficulty functions that attain the bound is given. Ordered by sub–section, we summarize the results of some of the constrained extremisation of $q_{AB}$:

5.3.4 **Maximise $q_{AB}$ (given a demand profile, $q_A$, and difficulty function $\theta_B$)**: Given that a manager has knowledge about the frequency of demands and which demands a given team finds "difficult", in choosing another team should the manager concentrate on diversifying the teams with respect to the most likely demands or with respect to the difficult demands for the first team? More precisely, suppose the manager has estimates of the expected *pfd*s for each channel ($q_A$ and $q_B$), some knowledge of the frequency of demands, as well as an idea of the difficult and easy demands for team "B". On page 141, we showed that the upper bound for $q_{AB}$ is attained by a difficulty function, $\theta_A$, with mean $q_A$ and the property that it assigns the value 1 on the most difficult demands according to $\theta_B$. That is, the worst case $q_{AB}$ is given by a team "A" that will always fail to develop a version that succeeds on the demands that team "B" finds difficult. This confirms an intuitive notion – that during software development, if the difficult demands for one team are known, then the benefits of forcing diversity are undermined considerably if the teams have the same "most difficult" demands. This is still true even if the most difficult demands according to $\theta_B$ are extremely unlikely. In this situation, if every version team "A" can produce fails on the most likely demands, then this results in a very bad expected *pfd* for the channel being built, but does not necessarily result in the worst–case expected system *pfd* if these demands are not the most difficult demands for team "B" also.

5.3.5 **Maximise $q_{AB}$ (given $q_A, q_B, q_{AA}$, and $q_{BB}$)**: This is an example of how an upper bound can provide a consistency check for an assessor using beta–factors. In this case, the assessor has estimates for the channel *pfd*s ($q_A$ and $q_B$) and beta–factor values for *homogeneous systems*, and she is interested in determining the worst–case expected system *pfd* resulting from forcing diversity. *Homogeneous systems* are those systems where both channels are built with the same methodology, either methodology "A" (so we may define beta–factor $\beta_A := \frac{q_{AA}}{{q_A}^2}$) or methodology "B" (so beta–factor $\beta_B := \frac{q_{BB}}{{q_B}^2}$). On page 146, we derived the worst case expected system *pfd*:

$$\max\{q_{AB}\} := q_A q_B \left(1 + \sqrt{(\beta_A - 1)(\beta_B - 1)}\right), \tag{5.29}$$

whenever the beta–factors satisfy both of the following conditions[14]:

$$(a)\ \beta_B \leq \beta_A \text{ (without loss of generality)},$$
$$(b)\ 1 \leq \beta_A + \beta_B - \beta_A\beta_B + \left(\frac{1-q_B}{q_B}\right)^2 .$$

Consequently, by choosing beta–factors that satisfy (a) and (b), an assessor can be sure of the worst case bound given in Eq. (5.29). In essence, the assessor has put a bound on the beta–factor for the system under forced diversity, $\beta_{AB} := \dfrac{q_{AB}}{q_A q_B}$, in terms of $\beta_A$ and $\beta_B$:

$$\max\{\beta_{AB}\} = \left(1 + \sqrt{(\beta_A - 1)(\beta_B - 1)}\right). \tag{5.31}$$

What do the difficulty functions and demand profile that maximise $q_{AB}$ in this way look like? An example of such a collection of distributions[15] is illustrated in Fig. 5.12. For added clarity, these distributions are shown separately in Fig.s 5.9, 5.10 and 5.11. As expected, the worst–case is given by a pair of difficulty functions that "agree" on which demands are difficult. This is despite the hypothetical development teams – whose actions are described by these difficulty functions – having very different probabilities of making mistakes on very likely demands: this would be the case in this example, if the parameters were such that:

$$q_A\beta_A \ll q_B\left(1 - \frac{\sqrt{(\beta_B - 1)}}{\sqrt{(\beta_A - 1)}}\right).$$

5.3.6 **Maximise $q_{AB}$( given a demand profile, $q_A$, and $q_B$)**: In this scenario, the manager has some knowledge of the frequency of system demands, as well as estimates of the expected *pfd*s for the channels, $q_A$ and $q_B$. Here, the pair of difficulty functions that attain the worst case $q_{AB}$ depend on the demand profile, and how large $q_A$ and $q_B$ are. We derived upper–bounds on $q_{AB}$, as well as pairs of difficulty functions that attain these bounds, in the following three cases:

(a) If $q_A = q_B$, then the largest $q_{AB}$ is given by a pair of identical difficulty functions with the largest possible second moment. We showed, in section 4.7.4 on page 117, that by ordering subsets of the demand space the difficulty functions with the

[14]This is an example of how a seemingly unobvious relationship between expected *pfd*s can have a simple geometric meaning. The algebraic condition in (b), i.e. $1 \leq \beta_A + \beta_B - \beta_A\beta_B + \left(\frac{1-q_B}{q_B}\right)^2$, was originally stated in Eq. (5.28) on page 149 as:

$$\cos(\gamma_A - \gamma_B) \leq \frac{q_A}{\|\theta_A\|\,\|\theta_B\|}. \tag{5.30}$$

To see this, use the substitution

$$\|\theta_A\|\,\|\theta_B\|\cos(\gamma_A - \gamma_B) = q_A q_B + \sqrt{(q_{AA} - q_A{}^2)(q_{BB} - q_B{}^2)},$$

in Eq.(5.30), and expand the terms. This is the geometric requirement that the two difficulty functions fit inside a "bounding" box defined by the demand profile, $\bar{P}$, in Fig. 5.7.

[15]These distributions were depicted earlier as vectors in Fig. 5.7 on page 149.

largest second moment can be obtained[16]. This means, if the least likely demand has a probability $q$ with the property that $q_B \leq q$, then the worst–case is given by a pair of identical difficulty functions such that they assign a difficulty $\frac{q_B}{q}$ to the least likely demand, and zero difficulty to every other demand. Consequently,

$$\max\{q_{AB}\} = \frac{{q_B}^2}{q}.$$

The lesson here is that for systems with sufficiently small expected system *pfd*s and indistinguishable expected channel *pfd*s, teams which are likely to fail on the unlikely demands undermine expected system reliability.

(b) If $q_A \neq q_B$ and the largest "A" and "B" difficulty functions with these means are "parallel", then the worst–case is given as:

$$\max\{q_{AB}\} = \frac{q_A q_B}{q},$$

for some $q$ that is the mean of a difficulty function with the following two properties: this difficulty function is parallel to the largest "A" and "B" difficulty functions, and it takes only the values 0 or 1 on each demand. This suggests that those difficulty functions that take on the value 0 or 1, and have expected values very close to $q_A$ and $q_B$, define all of the most difficult demands for pairs of difficulty functions that attain the worst–case expected system *pfd*. Therefore, if a pair of teams has probabilities of making mistakes that are diverse with respect to these demands, then the resulting expected system *pfd* from forcing diversity between these teams cannot be the worst–case.

(c) If $q_A \neq q_B$ and there exists a difficulty function $\theta$ with mean $q$ such that (without loss of generality) $q_A < q < q_B$. Further, suppose that $\theta$ assigns only the values 0 or 1 as difficulties to the demands. Then, we showed that the worst–case is the smaller of the two expected *pfd*s for the channels:

$$\max\{q_{AB}\} = q_A.$$

Again, all of the difficulty functions that satisfy the properties of $\theta$ define all of the most difficult demands for any pair of difficulty functions that attain the worst–case expected system *pfd*.

In this Chapter, we also considered the consequences of different notions of indifference, for an observer charged with the task of choosing development process methodologies that result in the best expected system *pfd*s. In particular, two notions of indifference were considered: indifference between methodologies and indifference between expected *pfd*s. We

[16]The order is according to the single–version *pfd*s related to the subsets, related by assuming the subsets define the failure–sets for single–versions

illustrate these in turn, as follows. Given the alternative methodologies "A" and "B" for developing the channels of a system suppose that, upon using any of these methodologies to develop both system channels, there is no evidence available to the observer to indicate which of the resulting expected system *pfd*s ($q_{AA}$ or $q_{BB}$) is worse. Then, if asked to assign a probability to the event "$q_{AA} > q_{BB}$", the observer may specify a probability value that is the same as the value she would give if she was considering either of the alternative events "$q_{BB} > q_{AA}$" or "$q_{BB} = q_{AA}$" instead. In this sense the observer is indifferent amongst the possible, alternative orderings of the expected system *pfd*s that result from building the system using either methodology "A" or methodology "B" exclusively. We say the observer is indifferent between the methodologies. However, if the observer has reason to believe that the expected system *pfd*s are indeed identical in value – that is, she is convinced that the event "$q_{BB} = q_{AA}$" holds – then she is indifferent between the expected system *pfd*s[17].

Using these notions of indifference in section 5.2, we considered the question of whether diversity should be forced or allowed to occur naturally. In sub–section 5.2.1 (see page 129) we gave geometric arguments for a result first stated in [13], concerning why indifference between expected system *pfd*s for homogeneous systems implies that forcing diversity cannot worsen, and may improve, expected system *pfd*. In addition, we stated a necessary and sufficient condition for forcing diversity to not worsen expected system *pfd*, noting that this condition depended only on the ratio of $q_{AA}$ and $q_{BB}$. Further, we showed in sub–section 5.2.2 and Theorem 5.2.2 (see page 135) that similar results hold when forcing diversity under indifference between the methodologies. Unlike indifference between expected system *pfd*s – which in practice requires evidence that these possibly unknowable expected system *pfd*s are equal – use of Theorem 5.2.2 does not require knowledge of expected system *pfd* values. Instead, not having evidence that indicates a preference amongst alternative methodologies is the sufficient requirement for using the theorem.

[17]We note that these notions of indifference can be argued to be the same. They both imply that when asked to choose the best expected system *pfd* the observer will choose either $q_{AA}$ or $q_{BB}$ with equal probability. However, we take the viewpoint that there is a relevant difference here, since one experiment is conditional on the event "$q_{BB} = q_{AA}$", while the other experiment considers this event as only one of three possible outcomes.

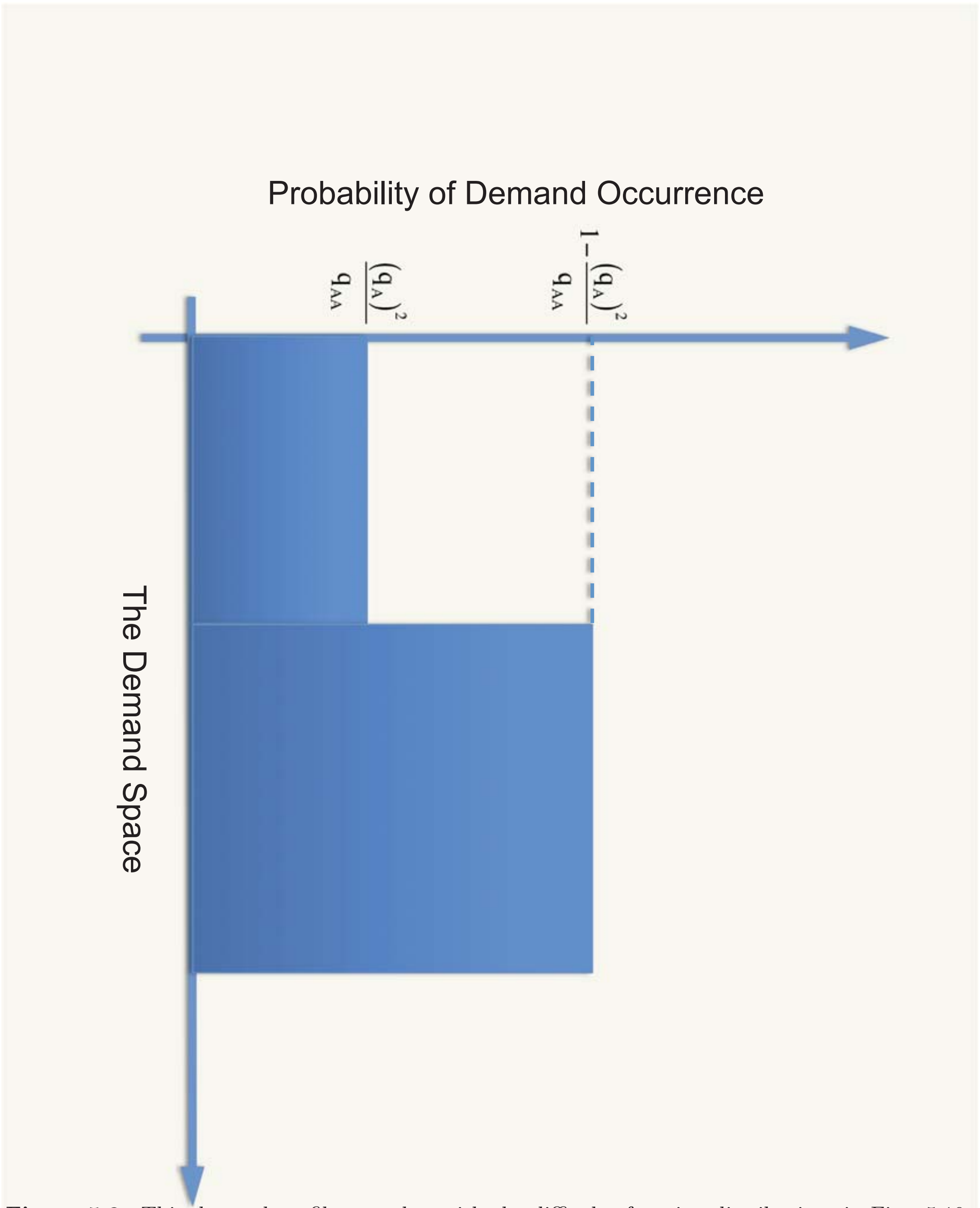


**Figure 5.9:** This demand profile, together with the difficulty function distributions in Fig.s 5.10 and 5.11, maximise $q_{AB}$. Upon using $\beta_A := \frac{q_{AA}}{{q_A}^2}$ in this simple "existence" example, we see that there are two kinds of demands: those that occur with probability ${\beta_A}^{-1}$, and those that occur with probability $1-{\beta_A}^{-1}$.

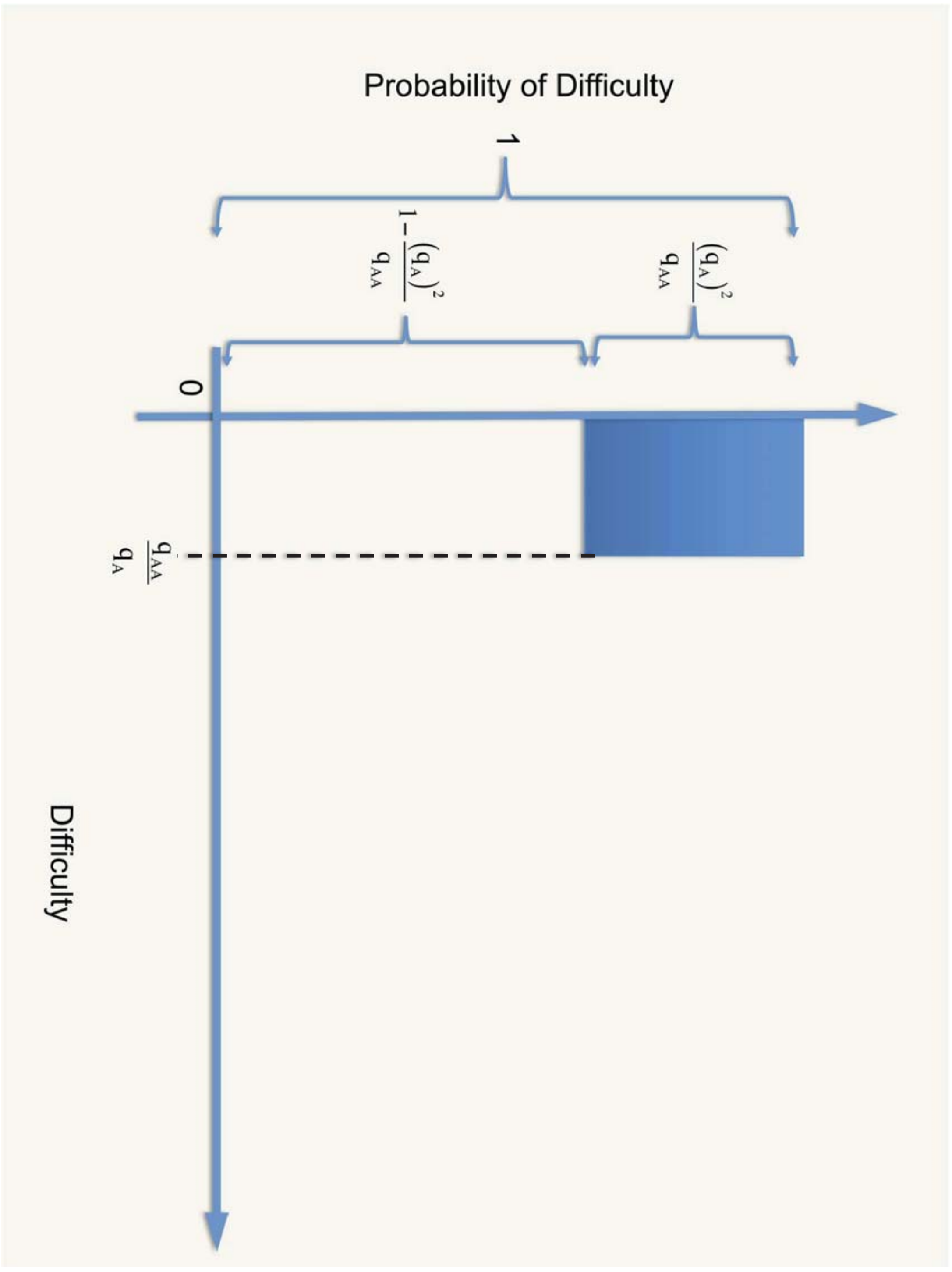


**Figure 5.10:** This difficulty function distribution, $\theta_A$, is part of the solution that maximises $q_{AB}$. Upon using $\beta_A := \frac{q_{AA}}{{q_A}^2}$, this distribution of $\theta_A$ has the following four properties: 1) Those demands which occur during system operation with probability ${\beta_A}^{-1}$ will result in failure with probability $q_A\beta_A$; 2) Those demands which occur during system operation with probability $1-{\beta_A}^{-1}$ do not result in failure; 3) The average difficulty is $q_A$; 4) The second moment of the distribution is ${q_A}^2\beta_A$.

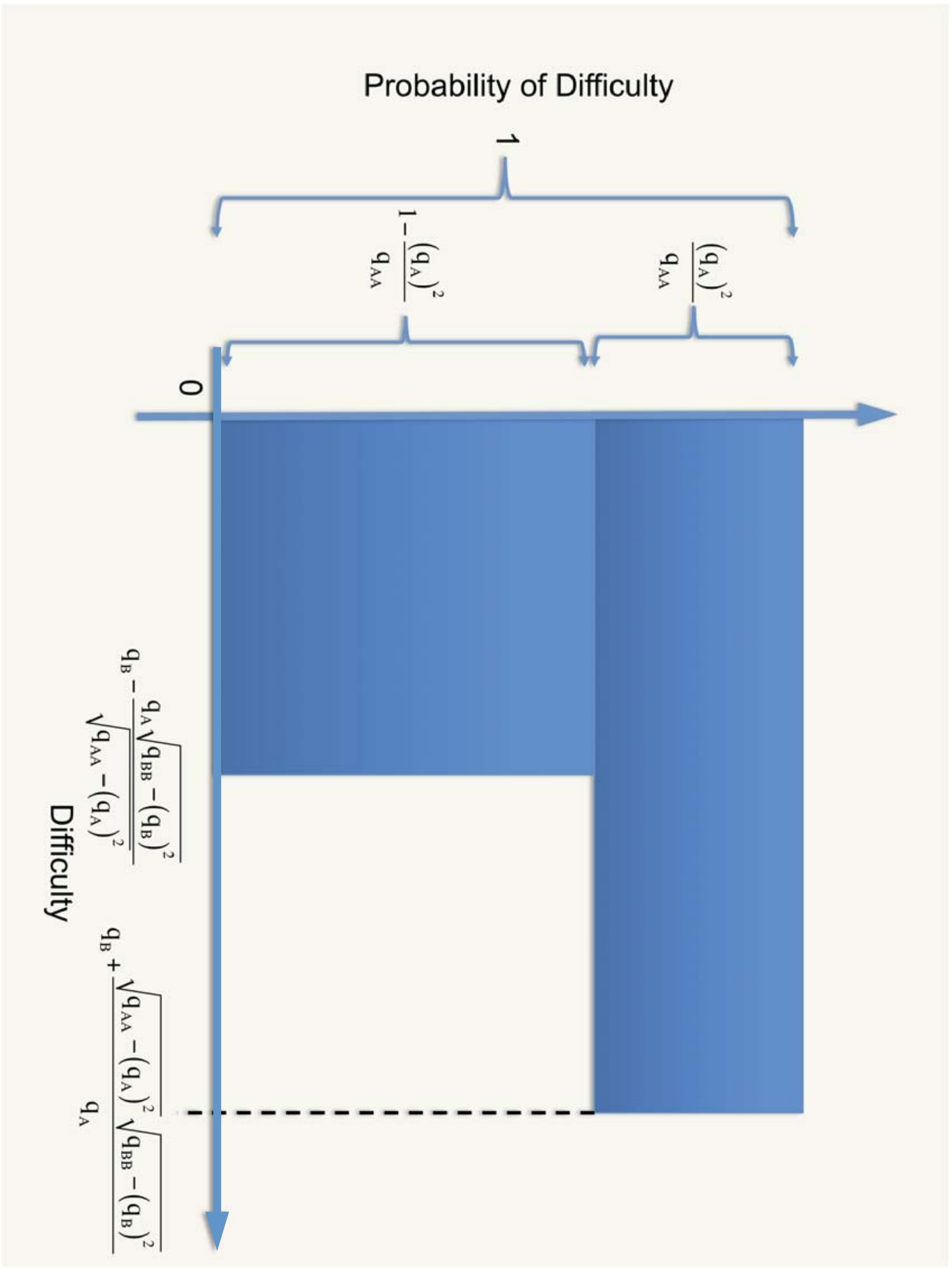


**Figure 5.11:** This difficulty function distribution, $\theta_B$, is part of the solution that maximises $q_{AB}$. Upon using both $\beta_A := \frac{q_{AA}}{{q_A}^2}$ and $\beta_B := \frac{q_{BB}}{{q_B}^2}$, this distribution of $\theta_B$ has the following four properties: 1) Those demands which occur during system operation with probability ${\beta_A}^{-1}$ will result in failure with probability $q_B\left(1+\sqrt{(\beta_A-1)(\beta_B-1)}\right)$; 2) Those demands which occur during system operation with probability $1-{\beta_A}^{-1}$ will result in failure with probability $q_B\left(1-\frac{\sqrt{(\beta_B-1)}}{\sqrt{(\beta_A-1)}}\right)$; 3) The average difficulty is $q_B$; 4) The second moment of the distribution is ${q_B}^2\beta_B$.

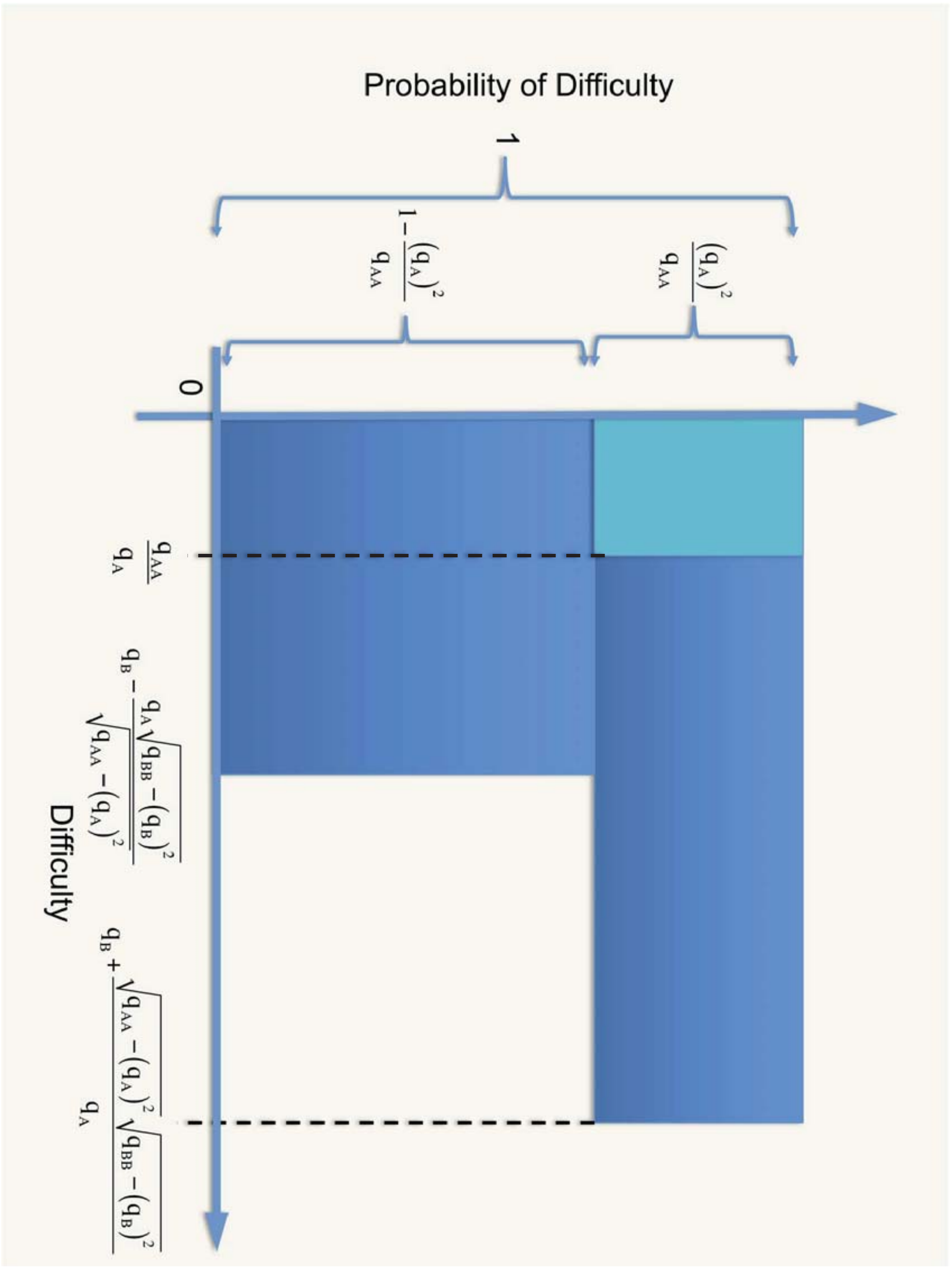

Probability of Difficulty
1
$1-\frac{(q_A)^2}{q_{AA}}$
$\frac{(q_A)^2}{q_{AA}}$
0
$\frac{q_{AA}}{q_A}$
$q_B-\frac{q_A\sqrt{q_{BB}-(q_B)^2}}{\sqrt{q_{AA}-(q_A)^2}}$
$q_B+\frac{\sqrt{q_{AA}-(q_A)^2}\sqrt{q_{BB}-(q_B)^2}}{q_A}$
Difficulty

**Figure 5.12:** A pair of difficulty functions, $\theta_A$ and $\theta_B$, and a demand profile that together attain the maximum value $q_Aq_B + \sqrt{(q_{AA} - q_A{}^2)\,(q_{BB} - q_B{}^2)}$ for $q_{AB}$.

# Chapter 6

# Summary of Main Conclusions

In this chapter, we discuss the results of the extensions to the EL and LM models, detailed in Chapters 3, 4 and 5. Broadly speaking, there are two ways in which the LM model has been extended: weakening of the "perfectly isolated teams" assumption in the model, and developing two alternative visual (graphical and geometric) representations of the LM model. Along these lines the chapter is divided into two main sections: a discourse on generalised models of coincident software failure (in the context of modelling controlled team interaction) in Section 6.1, and section 6.2 concludes the chapter with a summary of optimisation results using geometric models of coincident failure.

## 6.1 Models of Controlled Team Interaction: Modelling Dependence via Conditional Independence

What changes in the LM model when the "perfectly isolated teams" assumption is weakened? To answer this the scenario we chose to model was the development of a 1–out–of–2 system, where each channel of the system is built by a unique team. The teams are allowed to interact with each other, and the teams could share *common influences* (such as common educational backgrounds, or shared development time schedules). It turns out that such dependent software development may be modelled using conditional probabilistic independence, in a similar spirit to how the LM model characterizes failure dependence between independently developed software. Indeed, in this sense the models we have developed are a natural extension of the LM model: mathematically, the effect of a common influence between the teams is very similar to the effect of a demand in the LM model, since both cases result in conditionally independent channel failure (see Section 3.2). This is not the most general form of model we could have explored. However, in Chapter 3 we pointed out that the most general form of a probabilistic model of coincident failure is too general to be useful. Instead, we considered a model of a system development process in which the channel development processes are kept isolated, except for certain points during development where activity outcomes affect all of the channel development processes. Such a system development process has the property that activities in one channel development process are conditionally independent of activities in the other channel development process, conditional on dependence–creating influences such as "back–to–back" testing. In effect this model uses conditional independence to model dependence between the developments of the channels. This characteristic can be seen visually if the scenario is depicted as a BBN. Figures 6.1, 6.2 and 6.4 are depictions of three such scenarios. The property of conditionally independent random variables, $X$ and $Y$ say, conditional on some random variable $Z$, is equivalent to the graphical property that the $Z$ node in the BBN *blocks* all paths between the nodes representing $X$ and $Y$. We say that $Z$ *d–separates* $X$ and $Y$ (see Chapter 2 : Section 2.6 (beginning on page 48) for a definition of *d–separation*). For example, in Fig. 6.1 the development process activities "coding" for channel A (with outcome represented by node "High level design and initial version code" for channel A ) and "coding" for channel B (with outcome represented by "High level design and initial version code" for channel B) are conditionally independent, conditional on the common influence "specification clarification" since this node *d–separates* the other two nodes. Using graphical models (of which BBNs are a special case) as we do here to reason about the models of coincident failure is useful for two reasons: they aid intuition by visualizing relationships of conditional independence, and the implications of such dependence (e.g. "given a set of dependent activities during the development process are the channels developed independently?") can be recognised by applying simple rules based on the graph's topology (e.g. concerning the existence of common ancestor nodes or marginalisation justified by *d–separation*).

### 6.1.1 Decoupling Channel Development Processes

There are two scenario types that may be modelled as controlled team interaction: scenarios where decoupling is possible, and scenarios where decoupling is not possible. ***Decoupling** is the act of substituting a common influence with a pair of influences, one for each channel development process, that have the same distribution as the original common influence.* A necessary condition for decoupling to be possible is **observability criterion 3.2.3** (on page 71) which states:

*For a practical scenario in which decoupling is possible an observer, embedded in a channel's development process, cannot confirm or refute whether an activity is a common influence by observing the outcomes of activities in the development process she is embedded in..*

This is similar to, but not the same as, **observability criterion 2.4.1** (see page 38), which states:

*An observer embedded in the development process of a perfectly isolated development team should not be able to confirm, or refute, the existence of any other development process by observing activity outcomes in the process she is embedded in.*

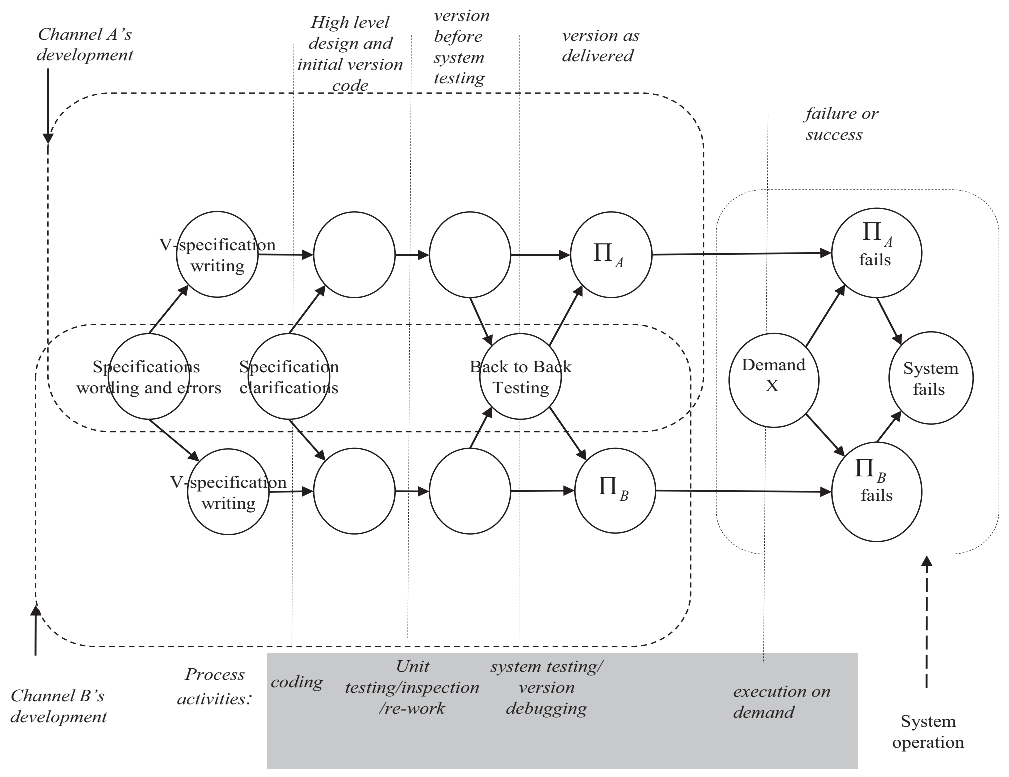


**Figure 6.1:** Example of the generalised model of coincident failure where the versions are developed under dependent development processes which exhibit conditional independence with respect to some common influences. For illustrative purposes we have indicated "Back to Back testing" as the preferred mode of testing in the process; in general there are additional forms of testing that may be applied to the versions. Note that "Back to Back testing" cannot be decoupled since, by definition, there necessarily needs to exist two versions to perform the test non–trivially and, consequently, the common activity violates observability criterion 3.2.3.

Indeed, criterion 3.2.3 is a necessary consequence of, but not a sufficient condition for, criterion 2.4.1 holding. As an example of how the criterion may be applied in practice consider that in N–version programming a *coordinating team*'s objective is to enact the *Communication and Documentation* (C & D) protocol [8, 14]. The goal of the C & D protocol is to avoid opportunities for one team to influence another team in an uncontrollable, and unnoticed manner. An extreme viewpoint of this objective is "to ensure that observability criterion 2.4.1 holds", and an alternative less extreme viewpoint is "to ensure that observability criterion 3.2.3 holds" (so that the teams do not influence each other despite dependencies between them). In particular, consider an activity of issuing specification clarifications (see Fig. 6.2) to the teams in a manner that seeks to preserve the independence between the channel development processes. For instance, consider a simple scenario where a dedicated team might be charged with studying the specification, and issuing clarifications to both channel development teams if and only if ambiguities are discovered by this dedicated team. Such a team may be replaced by a pair of teams, one for each channel development process, tasked with performing the same job but only issuing clarifications to their respective channel development process. So, the single node "Specification clarifications" in Fig. 6.1 becomes the related pair of nodes in Fig. 6.2: the channel development processes have been decoupled with respect to the "Specification clarifications" activity[1].

Note that an alternative protocol has been advocated and used in practice where clarifications have only been issued to a development team upon request, and only to the team requesting the clarification[2] [8, 14, 11]. If a single team performs this duty for both teams then it might be tempting to expect decoupling can be achieved by replacing this team with a pair of teams, each dedicated to a unique channel development process. However, the problem is that while the clarifications issued by the common team may conform to observability criterion 3.2.3 the common team *interacts with both* of the development teams. Consequently, there is no guarantee that questions asked by one team won't influence answers given to the other team. Hence, decoupling cannot be justified. Mathematically, while the sample space for the common influence obeys observability criterion 3.2.3 – a necessary condition for decoupling to be possible – the probability distribution for the common influence is "affected" by the common team answering questions from both development teams[3].

[1]Strictly speaking, for decoupling to be possible the probability measures for the marginal "Specification clarification" activities should be identical to the probability measure for the common "Specification clarifications" activity. It is useful to ask whether such probability measures differ depending on whether the team issuing specification clarifications know that they are giving updates to one or multiple teams. While we do not expect substantial difference to occur based on such knowledge we can avoid such considerations by requiring that the team not be told how many teams are receiving any updates they produce.

[2]Unless there is an error in the specification in which case an update is broadcast to all teams.

[3]This demonstrates that while updates to the specification do not necessarily constitute a violation of observability criterion 3.2.3, and thus may be a point of decoupling between the processes, defining such a protocol in practice can be challenging. Specification updates were broadcast to all teams in NASA's *4 universities* experiment [11] and Knight and Leveson's experiment [10]. The following excerpt is taken from [49] and discusses how clarifications affected the independent development of the channels in the Boeing 7J7.

"*... Boeing experience is that among sources of errors it is most often the basic requirements which are erroneous or misinterpreted. ........ The errors due to misinterpretation can be reduced by very close communication between the system requirements engineers and the software designers. ..... the software designers can help the engineers recognise limitations in the software design when the requirements are being written. There is much benefit from this interactive relationship, which is precluded by the*

As an example of a common influence with respect to which decoupling is not possible consider “Back to Back” testing. For each input given to a pair of versions, “Back to Back” testing involves comparing the outputs of the versions to determine if a mismatch occurs. The rationale being that a mismatch of the outputs may be indicative of the occurrence of failure. So, by definition “Back to Back testing” requires one version from one team and a second version from the other team in our Fig. 6.1 scenario. This means that the outcomes of the common activity of testing – which inputs cause failure in the versions – may conform to observability criterion 3.2.3, but the distribution thereof is *a result of interaction between the versions*. Therefore, it is impossible to define independent, identically distributed analogues of such a testing procedure that necessarily uses dependence. Consequently, “Back to Back” testing cannot be used for decoupling. Alternatively, however, if one considers the random generation of test suites to be used in testing both channels’ versions then such random generation need not “interact” with the channel development processes and its outcomes – some subset of the input space – conform to observability criterion 3.2.3. Therefore, such an activity may be used for decoupling. This form of decoupling was studied in [38].

### 6.1.2 The Independent Sampling Assumption

In Chapter 1 we suggested that the ISA is a necessary consequence of perfect team isolation, but not a sufficient condition since it can be used in cases where certain information is shared between the development processes. We expound on this point here. Recall that the depicted BBNs only depict sources of uncertainty. This means that if the outcome of a common activity is known then this common activity will not be depicted in the figures as it won’t be a source of uncertainty. *The act of fixing the outcome of a common activity decouples the development processes.* It is, therefore, theoretically possible that the channels’ development processes be completely decoupled by a mixture of fixing the outcomes of common influences that cannot be used for decoupling (because these influences have outcomes that “transfer” information between the teams), and decoupling with those common influences that can be used. This is not always possible because there are some common influences that by definition require interaction between the processes (see “Back to Back” testing above). However, when it is possible the consequence is that the teams develop their versions independently. Thus, the ISA holds. The topology of the resulting BBN will be very similar to the BBN for the LM model depicted in Fig. 6.3. However, there is a subtle difference which cannot be seen from the BBNs, precisely because the BBN only depicts sources of uncertainty. In general, a pair of decoupled processes may contain instantiated common influences where these common influences could not, otherwise, be used for decoupling. For instance, if the teams are required to exchange the details of the algorithms (excluding implementation

*dissimilar software design approach, where systems and software teams must be kept segregated. [...] the 7J7 program confirmed that the three separate teams [...] were having to ask Boeing so many questions for clarification of the requirements that the independence of the three teams was irreparably compromised. This is the reason why Boeing elected to revert to the usual and customary method of creating and certifying flight critical source code. It was determined that there is a net gain in total system integrity with the single software design approach.*”

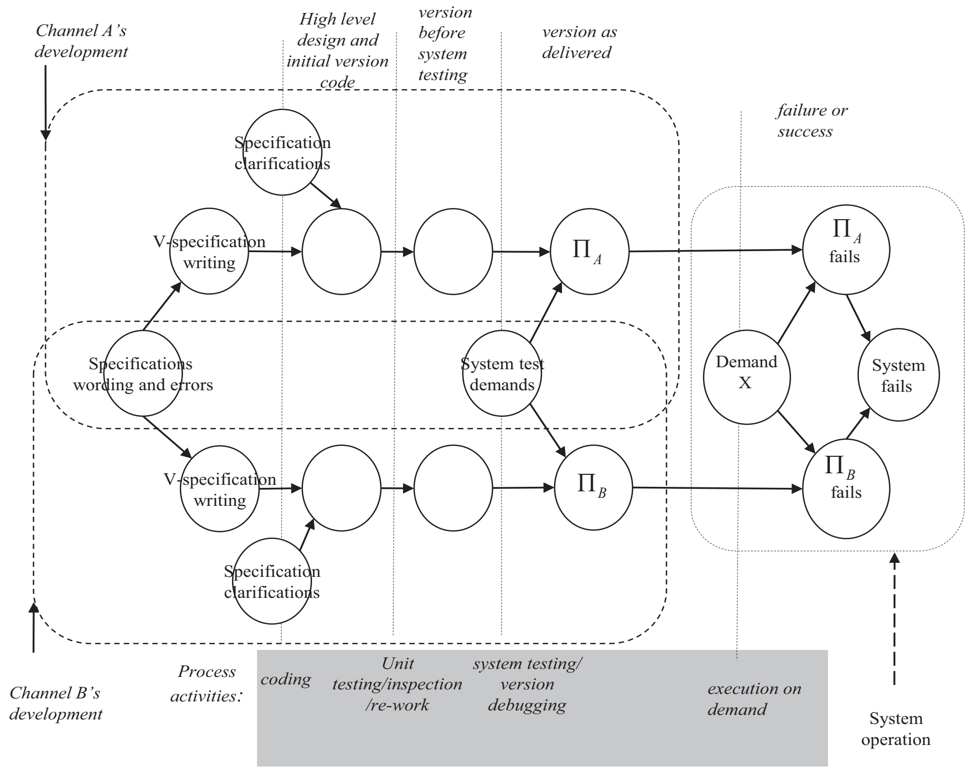


**Figure 6.2:** Decoupling of the specification clarifications may be possible since such information may be sanitized, ensuring that no information passes between the teams and, thus, observability criterion 3.2.3 holds for the activity. Also, note that the "Back to Back testing" in Fig. 6.1 has been replaced with testing using a randomly generated test suite to be used to test both versions. This common influence satisfies observability criterion 3.2.3 and decoupling with respect to this activity/influence is possible.

details) they have respectively implemented during development then this will not be a source of uncertainty during the development process if the algorithms used by the teams have been predetermined. Such common influence outcomes do not conform to observability criterion 2.4.1 since information exchange occurs between the teams. Therefore, despite the teams being conditionally independent in how they develop their versions (conditional on the values of common influences), the teams are not *perfectly isolated* and violate one of the assumptions of the LM model. We say the model is LM–*like*.

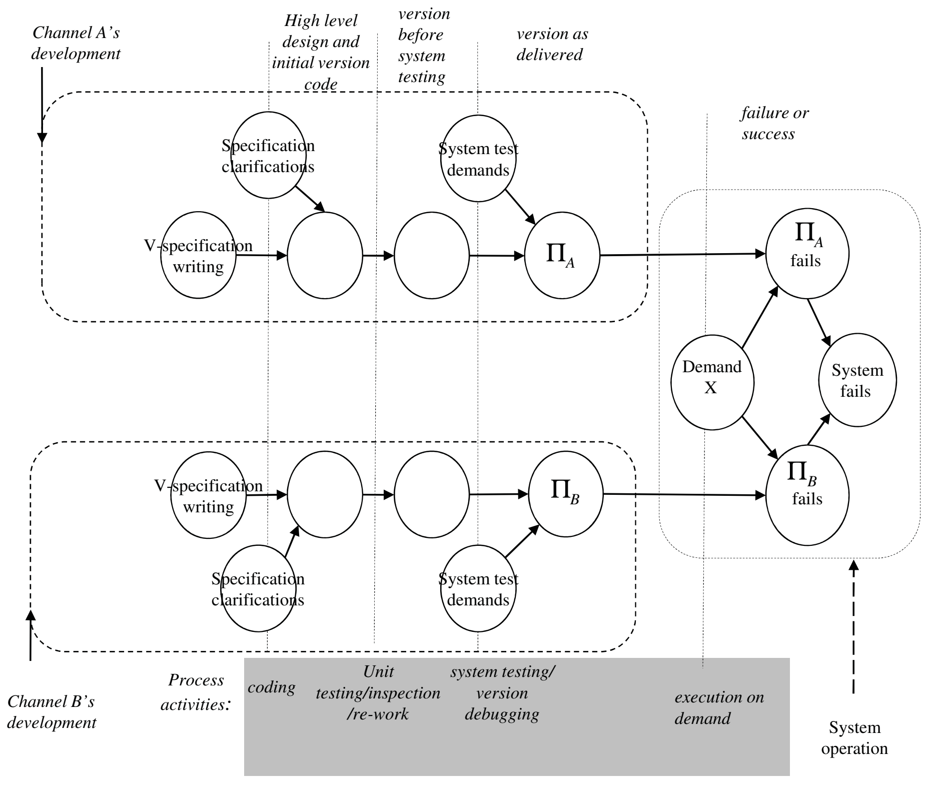


**Figure 6.3:** An example of a development process that may be modelled via the LM model. Such a diagram does not distinguish between a scenario where there are no common influences during system development, and a scenario where all of the common influences are instantiated (that is, they have values that remain fixed throughout the development process.)

Another example of how the teams may be independent (ISA holds) but may still violate the conditions of the LM model is given by considering the system specification. Since the system specification gives details about the 1–out–of–2 system being developed this is a source of dependence between the development processes. However, the nature of this dependence and its implications for whether the LM model is applicable is determined by whether the teams have access to the system specification (as opposed to some modified version of this that is tailored towards a channel's development). If the teams have access to the system specification then this could result in one team being able to infer details about the other development process. This is certainly a violation of observability criterion

2.4.1, and thus LM is not applicable. Despite this violation the teams are still conditionally independent in how they develop their versions, conditional on the specification. So, given a system specification, a BBN very similar to (and subtly different from) the one in Fig. 6.3 is still applicable. Such a model is LM-*like* because the ISA holds, but is not LM because fixed information passes between the teams. One might attempt to reduce commonality between the development processes resulting from the system specification by using a pair of so–called *V–specifications*; one for each channel. The idea is that the V–specification is more channel–focused than the system specification, with the aim of reducing unnecessary common–viewpoints between the teams while maintaining enough detail and affording the teams enough flexibility in designing and implementing their respective versions[4]. Of course, whether this will achieve the LM conditions will vary from scenario to scenario.

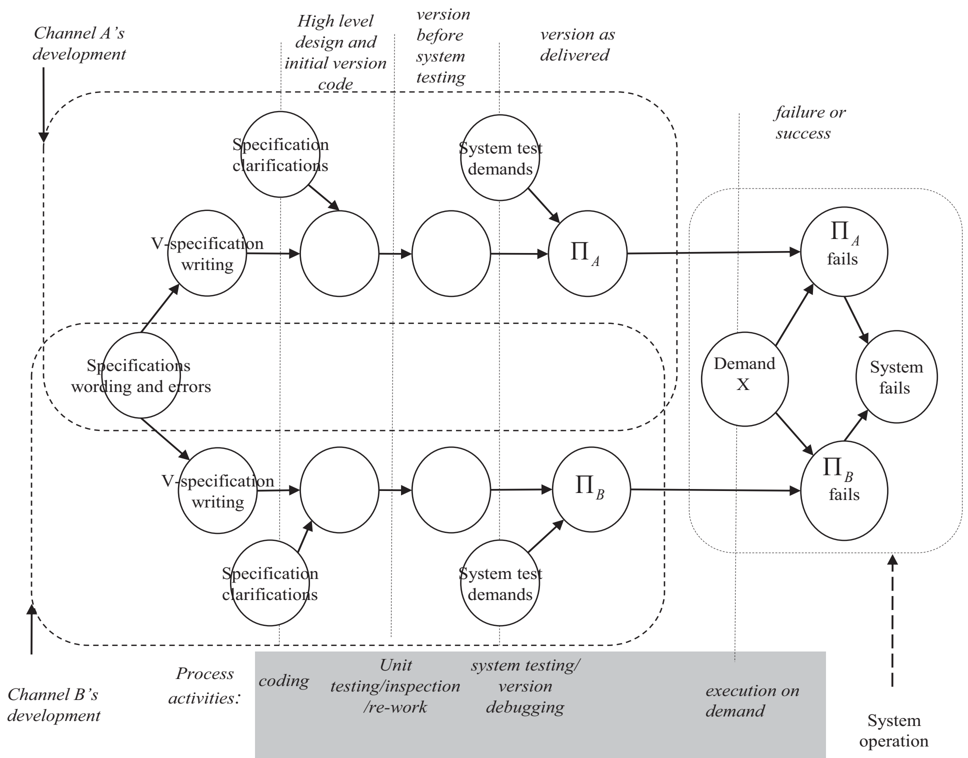


**Figure 6.4:** The channel development processes in Fig. 6.2 have been decoupled with respect to all of the common influences that are sources of uncertainty, except for the system specification. In order to reduce this to the EL/LM model a system specification will need to be written/defined/instantiated, thus removing the uncertainty from this activity. Additionally, the written specification and any other common influence that has been instantiated (and therefore are not depicted in the figure) should conform to observability criterion 2.4.1. Then, the BBN reduces to the LM model as depicted in Fig. 6.3.

[4]The original notion of V–specification was suggested by proponents of *N–version software* [8, 50, 14].

### 6.1.3 Common Influences

Note that the "common influences" need not be limited to the development phase. Indeed, mathematically there is no difference between the demands and common influence outcomes in that both of these create correlation between the channel development processes' respective difficulty functions. Other examples of common influences "outside" of the development process are possible. For example, if we are interested in both physical and software-caused failures our BBNs can include extra nodes that are parents of the nodes "$\Pi_A$ fails" and "$\Pi_B$ fails", to represent common stress factors like ambient temperature or common shocks. A note of caution is sounded here, however. Justification for conditional independence between the channel failures, conditional on these influences, will need to be provided and this forms part of future work.

### 6.1.4 Modelling Results and Practical Considerations

The extensions to the LM models clarify the relationship between the intuitive ideas of "separation", " independence", "diversity of process", formal concepts of probabilistic independence and correlation, and measures of interest such as *pfd*. In addition, these models offer some direct, practical help for decision making: we were able to derive three "preference criteria" among processes for developing two-version systems, based on sufficient conditions which we think people will recognise match the characteristics of practical situations they may face. Fig. 6.5 summarizes how our results enlarge the set of scenarios in which mathematically founded preferences can be stated between alternate ways of running multiple-version development. Our "preference criteria" describe changes that improve the system development process by shifting it from one domain to another as depicted in Fig. 6.5. In particular, an improvement over previous theory is in addressing questions such as, "When does combining multiple ways of 'forcing' diversity bring more benefit than simply 'forcing' diversity in one way (see [15, 51])?" Preference criterion 3.2.8, when used iteratively, gives a sufficient condition in this regard. To illustrate, suppose we have at our disposal a choice of two programming languages, C++ and Pascal, such that the use of each language, when used exclusively to develop both channels of a 1–out–of–2 system, results in the same expected system pfd. That is, we are indifferent between the expected system *pfd*s resulting from using, exclusively, either C++ or Pascal in developing both channels of the system. Then, forcing diversity (using both languages, one for each channel's development) cannot worsen, and may improve, reliability. Given that we force diversity in this way suppose we identify another dimension of diversity, such as choice of algorithm to implement. Then, as long as we are indifferent between the expected system *pfd*s resulting from either applying one algorithm to both channels or the other algorithm to both channels the preference criteria holds: forcing diversity with respect to algorithms cannot worsen reliability.

We summarize the assumptions, and results, of these generalised models in Fig. 6.6 and Fig. 6.7 respectively. Does the insight derived from the earlier "EL" and "LM" models (both of which assume the version development processes to be strictly "independent") remain valid, despite the fact that independent development can be difficult to guarantee

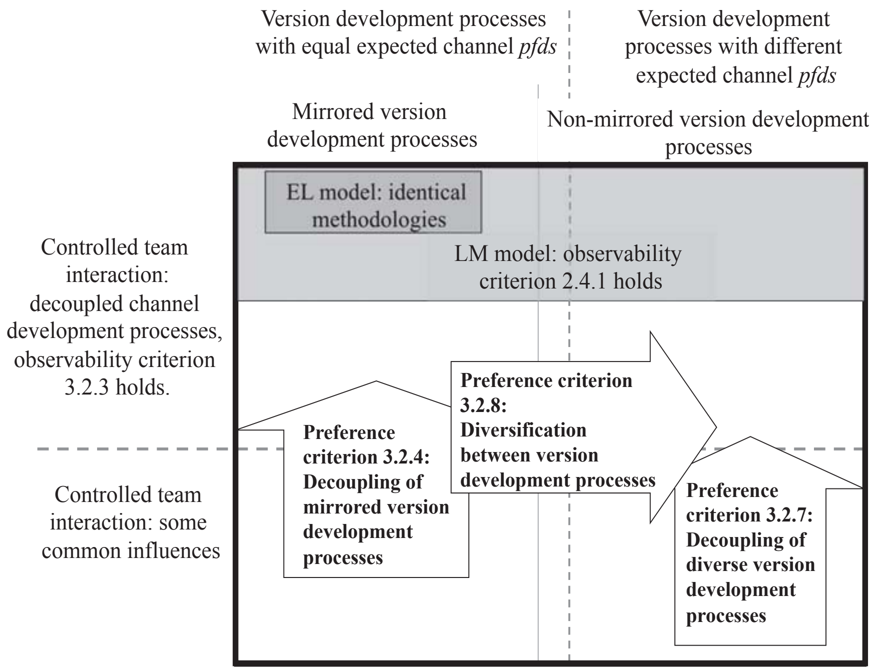
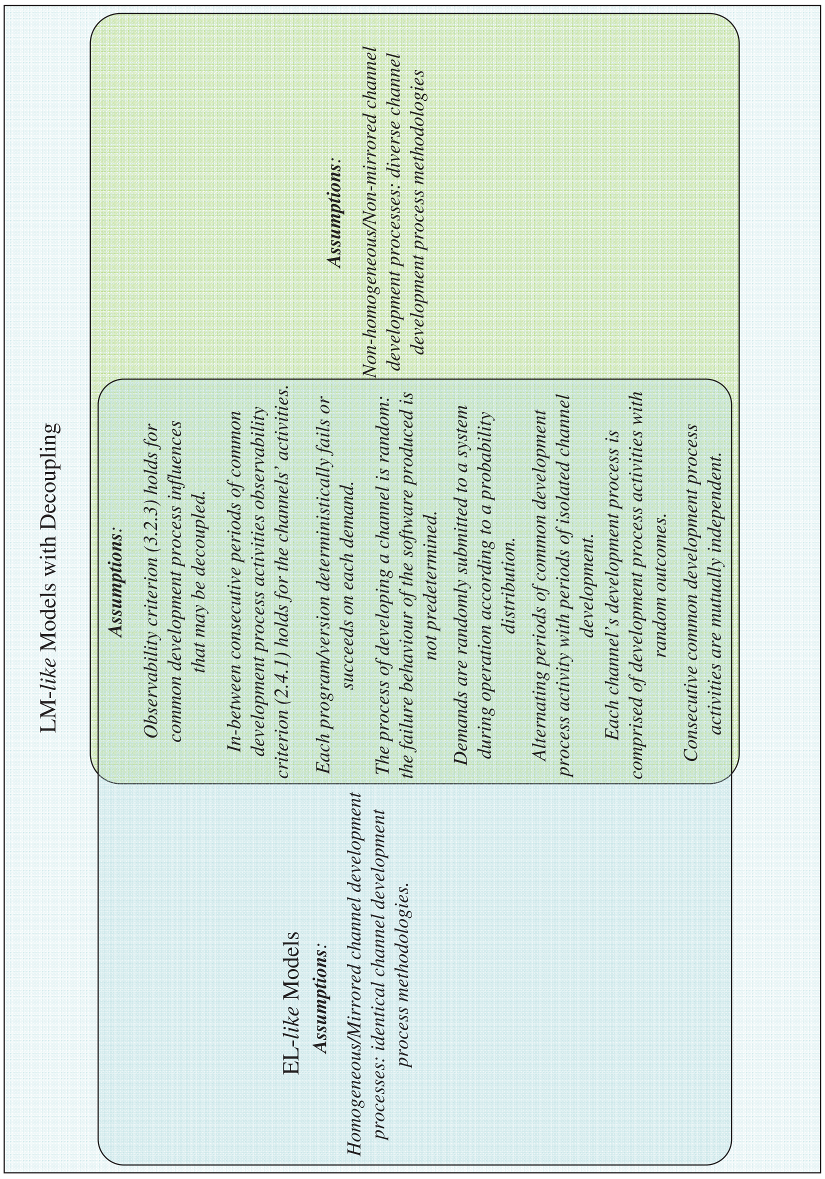

Version development processes with equal expected channel *pfds*
Version development processes with different expected channel *pfds*
Mirrored version development processes
Non-mirrored version development processes
EL model: identical methodologies
LM model: observability criterion 2.4.1 holds
Controlled team interaction: decoupled channel development processes, observability criterion 3.2.3 holds.
Controlled team interaction: some common influences
Preference criterion 3.2.4: Decoupling of mirrored version development processes
Preference criterion 3.2.8: Diversification between version development processes
Preference criterion 3.2.7: Decoupling of diverse version development processes

**Figure 6.5:** A venn diagram showing the space of scenarios we have considered (each scenario involves some system development process employing controlled team interaction), and the subsets on which the various results recalled or derived in Chapter 3 apply. Observe that the LM model can be viewed as a special case of the model of controlled team interaction with complete decoupling: the additional requirement is observability criterion 2.4.1 holds. The arrows indicate the “preference criteria” of Chapter 3: following an arrow from a subset of scenarios into another one cannot worsen, and may improve, the expected system *pfd*. Note that the arrow on the far right indicates an improvement in the positive covariance case of preference criterion 3.2.7. This is the case when the common influence used to decouple the processes induces positive failure correlation between the channels. Alternatively, if the common influence induced negative failure correlation, then decoupling cannot improve reliability and the arrow in the figure would point downwards instead.



Models (based on Conditional Independence) of Coincident Failure in Dependently Developed Multi-version Software: Model Assumptions

LM-*like* Models with Decoupling

***Assumptions***:

*Non-homogeneous/Non-mirrored channel development processes: diverse channel development process methodologies*

***Assumptions***:

*Observability criterion (3.2.3) holds for common development process influences that may be decoupled.*

*In-between consecutive periods of common development process activities observability criterion (2.4.1) holds for the channels' activities.*

*Each program/version deterministically fails or succeeds on each demand.*

*The process of developing a channel is random: the failure behaviour of the software produced is not predetermined.*

*Demands are randomly submitted to a system during operation according to a probability distribution.*

*Alternating periods of common development process activity with periods of isolated channel development.*

*Each channel's development process is comprised of development process activities with random outcomes.*

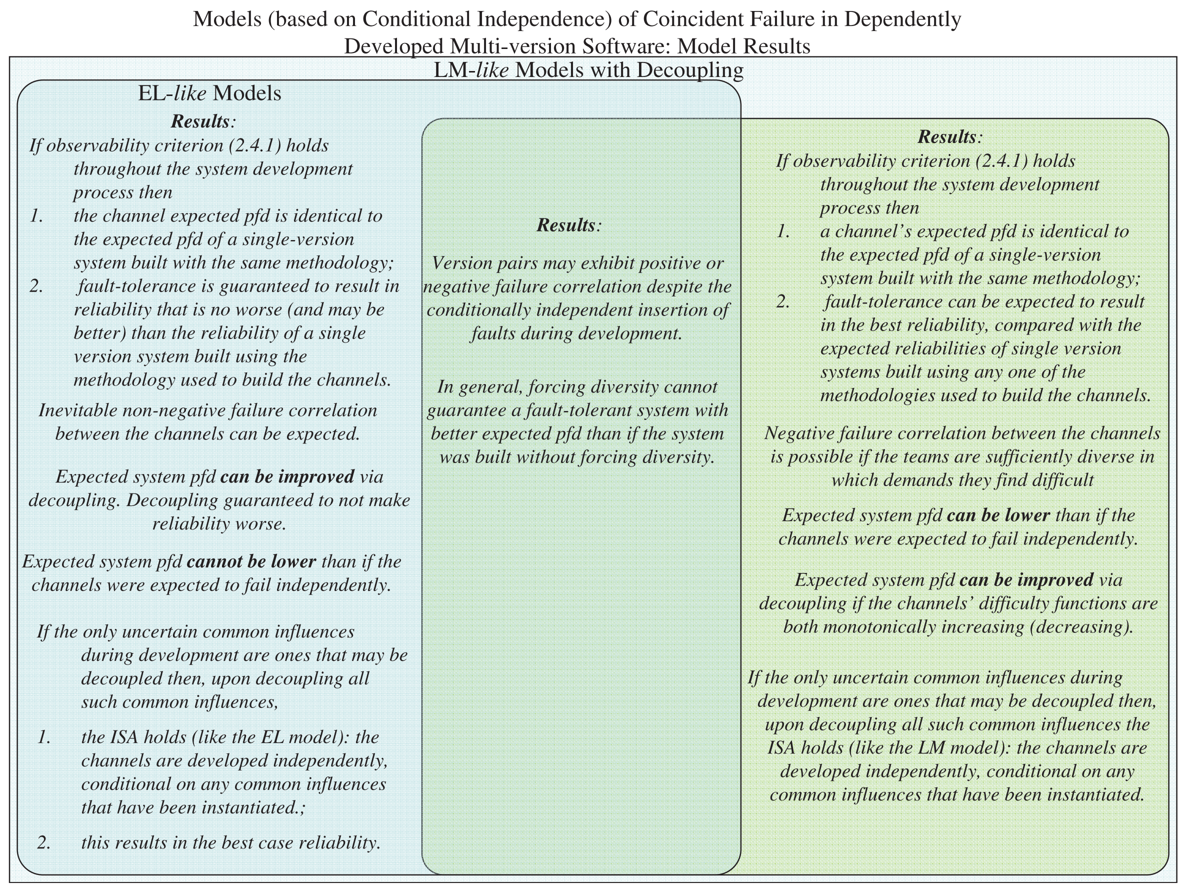
*Consecutive common development process activities are mutually independent.*

EL-*like* Models

***Assumptions***:

*Homogeneous/Mirrored channel development processes: identical channel development process methodologies.*

**Figure 6.6:** This table summarizes the assumptions of the LM model extensions for which decoupling is possible, and can be viewed as a venn diagram with intersecting sets of assumptions for the models. These models are collectively referred to as LM–*like* models with decoupling. LM–*like*, because the *Independent Sampling Assumption* (ISA) holds. A subset of these models are EL–*like* because the ISA holds, and the channel development processes are identically distributed. EL–*like* model assumptions are a special case of LM–*like* model assumptions. Each assumption in the table applies to each region that the assumption is contained in. Consequently, all of the assumptions apply to LM–*like* models, and a subset of these apply to EL–*like* models.



Models (based on Conditional Independence) of Coincident Failure in Dependently Developed Multi-version Software: Model Results

LM-*like* Models with Decoupling

EL-*like* Models

***Results***:

*If observability criterion (2.4.1) holds throughout the system development process then*

1. *the channel expected pfd is identical to the expected pfd of a single-version system built with the same methodology;*
2. *fault-tolerance is guaranteed to result in reliability that is no worse (and may be better) than the reliability of a single version system built using the methodology used to build the channels.*

*Inevitable non-negative failure correlation between the channels can be expected.*

*Expected system pfd* ***can be improved*** *via decoupling. Decoupling guaranteed to not make reliability worse.*

*Expected system pfd* ***cannot be lower*** *than if the channels were expected to fail independently.*

*If the only uncertain common influences during development are ones that may be decoupled then, upon decoupling all such common influences,*

1. *the ISA holds (like the EL model): the channels are developed independently, conditional on any common influences that have been instantiated.;*
2. *this results in the best case reliability.*

***Results***:

*Version pairs may exhibit positive or negative failure correlation despite the conditionally independent insertion of faults during development.*

*In general, forcing diversity cannot guarantee a fault-tolerant system with better expected pfd than if the system was built without forcing diversity.*

***Results***:

*If observability criterion (2.4.1) holds throughout the system development process then*

1. *a channel's expected pfd is identical to the expected pfd of a single-version system built with the same methodology;*
2. *fault-tolerance can be expected to result in the best reliability, compared with the expected reliabilities of single version systems built using any one of the methodologies used to build the channels.*

*Negative failure correlation between the channels is possible if the teams are sufficiently diverse in which demands they find difficult*

*Expected system pfd* ***can be lower*** *than if the channels were expected to fail independently.*

*Expected system pfd* ***can be improved*** *via decoupling if the channels' difficulty functions are both monotonically increasing (decreasing).*

*If the only uncertain common influences during development are ones that may be decoupled then, upon decoupling all such common influences the ISA holds (like the LM model): the channels are developed independently, conditional on any common influences that have been instantiated.*

**Figure 6.7:** Results from the LM–*like* models with decoupling possible. The results from the EL-*like* models are a subset of the results applicable to LM–*like* models.

in practice and at times it is even advisable to violate it intentionally? For instance, does the main message obtained from the EL result still hold, i.e. that a prudent assessor, given identical version development processes, should assume positive correlation between version failures? Indeed it does because with mirrored version development processes the ISA was indeed the most optimistic assumption (cf. Eq. (3.3) in Chapter 3), so that the conclusion applies *a fortiori* if it is violated.

A limitation of the preference criteria is that they depend on simple sufficient conditions (e.g. they support decisions about removing a common influence if it induces positive covariance between difficulty functions for any values of the other influences), and when these are not satisfied the criteria no longer help: to decide between alternative system development processes one must then look for empirical evidence.

These generalised models make it easy to include in the descriptions of "cause of dependence" the cases of diversity-reducing influences that yet improve system *pfd* (e.g., testing with a common, randomly chosen test suite); as well as the possibility, at least in theory, of common influences that improve system *pfd* by *increasing* diversity. Dependence in the system development process is not in itself good or bad. What matters are questions like: does the common influence that "creates" the dependence affect the teams' difficulty functions in "the same way"? Do both teams become more or less likely to make mistakes on given demands due to a change in an influence?

## 6.2 Geometric Model of Coincident Failure

The results of Chapter 3 are "qualitative": they specify an ordering between expected system *pfd*s resulting from different ways of organising the development process, without indicating how large these orderings might be. Consequently, while these results suggest preferable ways of organising the development process, they give no indication of how much benefit a given preference can bring. So, despite expected *pfd*s possibly being either difficult to estimate or unknowable in practice, is it still possible to gain an appreciation of how big orderings between expected *pfd*s might be? In Chapter 4, starting from the models of Chapter 3, we developed a geometric model of coincident software failure. The main motivation for this was to understand the effect of forcing diversity on system reliability and, in particular, how much benefit or loss (in terms of system reliability) forcing diversity can bring under different practical situations. Consequently, the problems of interest involved the optimisation of the expected system *pfd* resulting from forcing diversity, $q_{AB}$, subject to constraints. In performing such optimisations we required a formulation of the LM model that was both flexible and descriptive enough to allow different extremisation problems to be approached in an intuitive and unified way. A typical form of the optimisation problems we considered is the following:

*Obtain a pair of difficulty functions, $\theta_A$ and $\theta_B$, such that they maximise the expected system pfd, $q_{AB}$, subject to some constraints on the demand profile and the expected pfds $q_A$, $q_B$,*

*$q_{AA}$ and $q_{BB}$.*

To this end we developed a geometric formulation of the LM model, inspired by so–called *coordinate–free* approaches to applied geometry[5]. Using this model, important relationships in the LM model can be stated in terms of the behaviour of the inner–product on pairs of difficulty functions[6].

### 6.2.1 Benefits of Geometry-based Analyses

This novel approach to modeling and analysing coincident failure in multiversion software presents a number of benefits, including:

1. It provides a unified approach to the constrained extremisation of expected *pfd*s;
2. It is based on one of the oldest branches of mathematics and, consequently, many proofs of relationships between difficulty functions and expected *pfd*s are simple consequences of well known results;
3. Proofs of theorems stating constrained relationships between expected (system) *pfd*s can be done using whatever coordinates make the problem easier to analyse: if the theorem is proved in one coordinate system, it will also be true in any alternative coordinate system. This is a consequence of the inner–product being invariant – under a change of basis – when acting on an arbitrary pair of difficulty functions. That is, the inner–product's values are unchanging under a change of the coordinate system used to describe the underlying vector space. In Chapter 5, under a basis transformation $T$ that is defined in Section 4.4, multiple theorems were proved using convenient coordinate systems. Some of these results are summarized in Fig.s 6.10, 6.11 and 6.14;
4. The inner–product defines important geometric relationships between pairs of difficulty functions. So, there is a rigorous sense in which difficulty functions have magnitudes, projections, and make angles with each other. Consequently, theorems stated in terms of the

[5]The essence of such approaches, used widely in Mathematical–physics, is that when using geometry to solve practical problems there are important relationships that characterise/solve the problem (such as angles between vectors and lengths of vectors), and these relationships should be independent of the choice of basis/coordinates used to describe the problem. This principle was followed in developing the model as follows. For a fixed number of demands, $n$ say, we postulated the existence of an ***n–dimensional inner–product space*** (An inner–product space is a vector space, together with an inner–product defined with respect to it. See Postulate 4.3.2 on page 92, and page 190) such that a subset of the space has vectors that model difficulty functions, and the action of the inner–product on pairs of these vectors gives expected (system) *pfd*s. See Chapter 4 : Section 4.3 (beginning on page 88) for a more precise statement of the geometric models definition.

[6]As an example of how the LM model results find expression in the geometric models consider, in particular, the EL model result which shows that identical difficulty functions (e.g. for teams which develop their versions in such a way that they have identical probabilities of making any given mistake) imply that the expected system *pfd* is no better than it would be if the versions were expected to fail independently. For the difficulty function $\theta$ this is characterized by (see Eq. (2.13) on page 43)

$$\mathbb{E}\left[\theta^2\right] \geq \mathbb{E}\left[\theta\right]^2 .$$

In geometric terms this is the result

$$\|\theta\| \geq \langle \theta, \bar{P} \rangle.$$

That is, the difficulty function $\theta$ has a size, $\|\theta\|$, which is never smaller than the projection of $\theta$ in the direction of the demand profile, $\bar{P}$. This is depicted in Fig. 4.8 on page 103 for two difficulty functions, $\theta_A$ and $\theta_B$.

inner–product also "suggest" what relationships involving difficulty functions and expected *pfd*s are (im)possible. This aids intuition when performing constrained optimisation, since such geometric relationships indicate sufficient conditions for a maximum/minimum of some objective function to be obtained. This is especially useful when the optimisation problems involve either non–linear constraints, or non–linear objective functions. For instance, one might wish to find a difficulty function, $\theta_A$, such that the following 3 constraints are satisfied: 1) it has a related expected system *pfd* value of $q_{AA}$; 2) a demand profile is specified (that is, for each demand an associated probability that the demand will occur in practice is specified), and; 3) together with a given difficulty function, $\theta_B$ say, the pair $\theta_A$ and $\theta_B$ results in a maximum value for $q_{AB}$.

The quadratic constraint in this example problem is the specified expected system *pfd*, $q_{AA}$. This means $\sqrt{q_{AA}}$ is the size of the difficulty function $\theta_A$ to be found. Hence, the set of potential solutions is the set of all difficulty functions that touch the surface of an $n$–dimensional sphere, where the sphere has radius $\sqrt{q_{AA}}$ and is centered at the origin. Consequently, in order to maximise $q_{AB}$, we seek members of this set that are **as close to** being **collinear** with $\theta_B$ **as possible**. This is a concise statement of an algorithm that results in a solution. It is cumbersome, at best, to state this recipe for solving the optimisation problem using alternative formalisms, such as probability theory (the basic formalism used in the LM model)[7].

5. Relationships between difficulty functions, where these relationships are stated solely in terms of an inner–product, are not dependent on the size of the demand space: that is, such relationships hold for any set of difficulty functions defined on a finite number of demands. This is because the proofs of these results are independent of the dimensionality

[7]For the interested reader we proceed to solve the problem for the case where $\sqrt{q_{AA}}$ is not larger than the size of the largest difficulty function that is collinear with $\theta_B$. Consequently, we may obtain $\theta_A$ by multiplying the unit vector in the direction of $\theta_B$ by $\sqrt{q_{AA}}$. But, which difficulty function is the largest difficulty function that is collinear with $\theta_B$? Given the components of the demand profile $\left(\sqrt{P_1}, \ldots, \sqrt{P_n}\right)$ in the basis $\hat{\mathcal{S}}$, the largest difficulty function is

$$\frac{\sqrt{P_j}}{\theta_B j}\begin{pmatrix} \theta_B 1 \\ \theta_B 2 \\ \vdots \\ \theta_B (n-1) \\ \theta_B n \end{pmatrix},$$

where $\theta_B j = \theta_B^*(x_j)\sqrt{P_j}$ is the non–zero component such that $\dfrac{\sqrt{P_j}\sqrt{q_{BB}}}{\theta_B j}$ is the smallest quotient out of all similar quotients that involve components of the difficulty function $\theta_B$. This quotient is the magnitude of the largest difficulty function that is collinear with $\theta_B$. Consequently, as long as $\sqrt{q_{AA}} \leq \dfrac{\sqrt{q_{BB}}}{\theta_B^*(x_j)}$, the difficulty function $\theta_A$ that we seek is

$$\frac{\sqrt{q_{AA}}}{\sqrt{q_{BB}}}\begin{pmatrix} \theta_B^*(x_1) \\ \theta_B^*(x_2) \\ \vdots \\ \theta_B^*(x_{n-1}) \\ \theta_B^*(x_n) \end{pmatrix},$$

where we have written the components in the LM–basis $\mathcal{S}$. So, using this $\theta_A$ together with $\theta_B$ gives the maximum value of $q_{AB}$ as $\sqrt{q_{AA}q_{BB}}$. Obtaining the solution in the case where $\sqrt{q_{AA}} \geq \dfrac{\sqrt{q_{BB}}}{\theta_B(x_j)}$ is more involving, utilising results from earlier chapters in the thesis.

of the underlying vector–space used to model the difficulty functions. Consequently, such relationships can be depicted in 2 or 3 dimensions, and these diagrams may then be used to guide intuition about what the implications of such relationships are in higher dimensions. Each such diagram may depict sets of difficulty functions and, therefore, the diagram is useful for comparing these sets. As an illustration consider Theorem 4.6.6 stated on page 107, which we reproduce here. Let $q_B$ and $q_{BB}$ designate the first and second moments, respectively, for an arbitrary difficulty function.

*In the basis $\mathcal{S}$ suggested by the LM model, the only difficulty functions for which $q_{BB} = q_B$ are the "single versions": those difficulty functions with each of their components equal to either 1 or 0.*

This theorem equates two values, $q_B$ and $q_{BB}$, of some inner–product. This means that to understand the consequences we may depict this relationship in a 3–dimensional diagram, such as Fig. 4.17 on page 116, and use this to aid intuition when reasoning about the relationship in higher dimensions. We reproduce the diagram here in Fig. 6.8 for the reader's convenience. Here, the set of all possible difficulty functions in 3–dimensions is

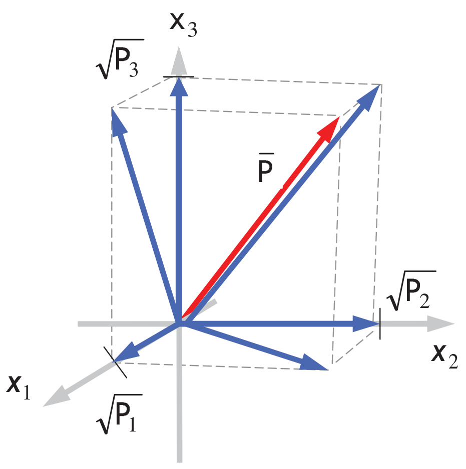


**Figure 6.8:** For a demand space of size 3 there are $2^3$ possible "single versions"/"failure regions", including the perfect version represented by the origin, all of which are depicted in this figure. The imperfect single versions are the largest difficulty functions with their associated means, and this is still true in higher finite–dimensions.

depicted as a subset of the non–negative region. In particular, we see that there are $2^3$ difficulty functions with the property $q_B = q_{BB}$. While we may not be able to depict this property in higher–dimensions, the coordinate–independent nature of the theorem suggests that in $n$ dimensions there are $2^n$ difficulty functions with this property. Furthermore, consequences of this property – such as, the single versions being the largest difficulty functions with their respective means – also hold in higher dimensions.

6. There are optimisation problems that are relatively easy to solve using this geometric formalism, compared with using alternative formalisms. For instance, consider the arguably

non–trivial optimisation problems discussed in Section 5.3.6 (beginning on page 150) which are solved by appealing to simple geometric relationships between sets of difficulty functions.

7. Relationships in the LM model that are relevant for constrained optimisation, but do not have an immediate probabilistic interpretation, can have geometric meaning. For instance, the quantity $\frac{q_A}{\sqrt{q_{AA}}}$ arises when trying to maximise the value of the expected system *pfd* $q_{AB}$, given values for $q_A, q_B, q_{AA}$ and $q_{BB}$. This dimensionless[8] quantity is not a probability; it has the form
$$\frac{\text{probability}}{\sqrt{\text{probability}}}.$$
However, the ratio is the cosine of the angle the difficulty function makes with the demand profile. The difficulty functions that satisfy this constraint touch the surface of an $(n-1)$–dimensional "sphere", where the sphere is centered on the point $(\underbrace{q_A, \ldots, q_A}_{n \text{ times}})$ with radius
$$\sqrt{1 - \frac{q_A^2}{q_{AA}}}.$$
Everything else being equal, the smaller this cosine the more variation of difficulty exists. Alternatively, the larger the cosine the closer the difficulty function is to being constant. See Fig. 4.9 on page 104, and Fig. 4.16 on page 112, for depictions of this relationship. In general, such relationships can be depicted in representative 2–dimensional and 3–dimensional diagrams (using arrows hinged at the origin) that illustrate how these constraints limit the objective function $q_{AB}$.

8. Furthermore, these geometric models also apply to extensions of the LM model that relax the ISA. Recall, in Chapter 3 we extended the LM model using conditional independence to model dependent channel development processes. One consequence of this is that given a demand, and the outcomes of all common influences between the development teams of a 1–out–of–2 system, the channels fail independently. This is mathematically similar in form to the difficulty functions in the LM model, the difference being that instead of conditioning solely on a demand the conditioning now also takes into account all of the outcomes of common activities during the development process. Therefore, like the LM model, the expected system pfds are expectations of products of difficulty functions. In terms of the geometric model this means that given $n$ demands and $m$ common influences – the $i$th common influence having $n_i$ possible outcomes – we may represent a difficulty function as an $N$–dimensional vector in a $N$–dimensional inner–product space, where $N = n + n_1 + \ldots + n_m$. Consequently, the results of geometrically modelling the LM model are also applicable as the consequences of a dependent software development process that is modelled via conditional independence.

[8]This quantity is dimensionless: the numerator and denominator are of the same degree.

### 6.2.2 Results and Practical Considerations

The EL/LM models were primarily developed to explain possible failure correlation between independently developed software versions. It is tempting to use these models by estimating the expected *pfd*s they refer to, and using these estimates directly in making decisions about actual development processes. However, one hindrance to doing so is that some of the probability distributions used in the models, such as demand profiles and version sampling distributions, may not be completely known in a given practical scenario. Consequently, making expected *pfd*s possibly unknowable as well. On the other hand, bounds on expected *pfd* are a weaker requirement than knowing the actual expected *pfd* values. In this regard the bounds we obtained are a first–step in estimating possibly unknowable expected *pfd*s, by constraining their values using other model parameters that may be known.

An assessor of a system may gain confidence in her estimates of expected *pfd*s if these estimates satisfy relationships that the actual, unknown values of the expected *pfd*s must satisfy. The bounds on expected system *pfd* define such consistency checks. For instance, if the assessor had estimates of the expected channel *pfd*s and the demand profile, then her possible estimates of expected system pfd are bounded as indicated in Fig. 6.10. Many more of such relationships are summarised in Fig.s 6.10, 6.12, and 6.14.

Related to the previous point is the use of beta–factors when reasoning about expected system *pfd*. The beta–factors we are interested in are the relative sizes of expected *pfd*s, and as such they indicate the "orders of magnitude" difference between expected *pfd*s. Consequently, by casting the extremisation problems in terms of beta–factors, we can specify how many "orders of magnitude" improvement in expected system reliability may be obtained from organising the development process in some preferred way. For example, in Fig.s 6.11 and 6.12 (see pages 183 and 184) we summarize some bounds derived in terms of the expected *pfd* values $q_A$, $q_{AA}$, $q_B$ and $q_{BB}$. Equivalently, when these results are put in terms of beta–factors (as in Fig. 6.13 on page 185), they also indicate bounds on how many "orders of magnitude" the expected system *pfd* may improve by,

We also explored sufficient conditions for forcing diversity to result in the least expected system *pfd* amongst alternatives. In particular, Theorem 5.2.1 (on page 130) gives a necessary and sufficient condition in this regard, and is based solely on the relative sizes of the pair of expected system *pfd*s for homogeneous systems, $q_{AA}$ and $q_{BB}$. This result generalises a particular case first shown in [13], where forcing diversity was argued to be beneficial when an assessor was indifferent between $q_{AA}$ and $q_{BB}$; that is, $q_{AA} = q_{BB}$ for the assessor. The result also indicates a certain irrelevance of the expected channel *pfd*s and the demand profile in ensuring that forcing diversity is beneficial. Admittedly, in order to use the result in practice, either knowledge of the values of $q_{AA}$ and $q_{BB}$ or justification for the equality of these values is required. However, another result comes to a similar conclusion, but with less stringent conditions for its use in practice, is given by Theorem 5.2.2 (on page 135). Here, instead of indifference between the values of the expected system *pfd*s, the assessor is indifferent amongst the different possible methodologies that may be used to build a homogeneous system. Under this indifference the theorem states that the expected system *pfd* resulting from not forcing diversity cannot be better than the expected system *pfd* resulting from forcing diversity.

In the course of developing the model, many relationships were derived between difficulty functions, their sizes, their means and the angles they make. As a flavour of the kind of relationships that are possible we offer the following three examples. For many more relationships along similar lines the reader is referred to Chapter 4.

1. The models make apparent how the sizes of difficulty functions (and, therefore, their related expected system *pfd*s) are constrained by the vector $\bar{P}$. This vector has components, in the basis $\bar{\hat{S}}$, that are completely defined by the demand profile and, consequently, is referred to in this thesis as the demand profile (see the definition on page 94). By using the normalised basis $\hat{\mathbb{S}} := \left\{\hat{P}_1, \ldots, \hat{P}_n\right\}$ (defined on page 92) an arbitrary difficulty function can be expressed as a linear combination of these basis vectors, where the weights in the linear combination (that is the vector components) can be no larger than thresholds defined by the demand profile. That is, an arbitrary difficulty function $\theta$ may be written as
$$\theta = \sum_{i=1}^{n} \theta(x_i)\hat{P}_i,$$
where $\theta(x_i) \leq \sqrt{P(X = x_i)}$ for $i = 1, \ldots, n$. The case where $\theta(x_i) = \sqrt{P(X = x_i)}$ for all $i$ defines the largest difficulty function, $\bar{P}$, which is the difficulty function that represents failure on all demands[9].
2. For a set of difficulty functions with identical means, the larger the angle a difficulty function makes with the demand profile the larger the difficulty function. On a related note, the difficulty functions that are furthest away from the demand profile are those that are collinear with the basis vector associated with the least likely demand.
3. There is a notion of "degree of diversity" between an arbitrary pair of difficulty functions which is characterized by the angle between them. Given a set of difficulty function pairs, where all of the difficulty functions have the same size, the larger the angle between a pair of such difficulty functions the smaller the correlation between them. For instance, as an extreme case, if the difficulty functions are perpendicular then the related value of the expected system *pfd* (for a system built by channel development processes characterised by those difficulty functions) is 0.

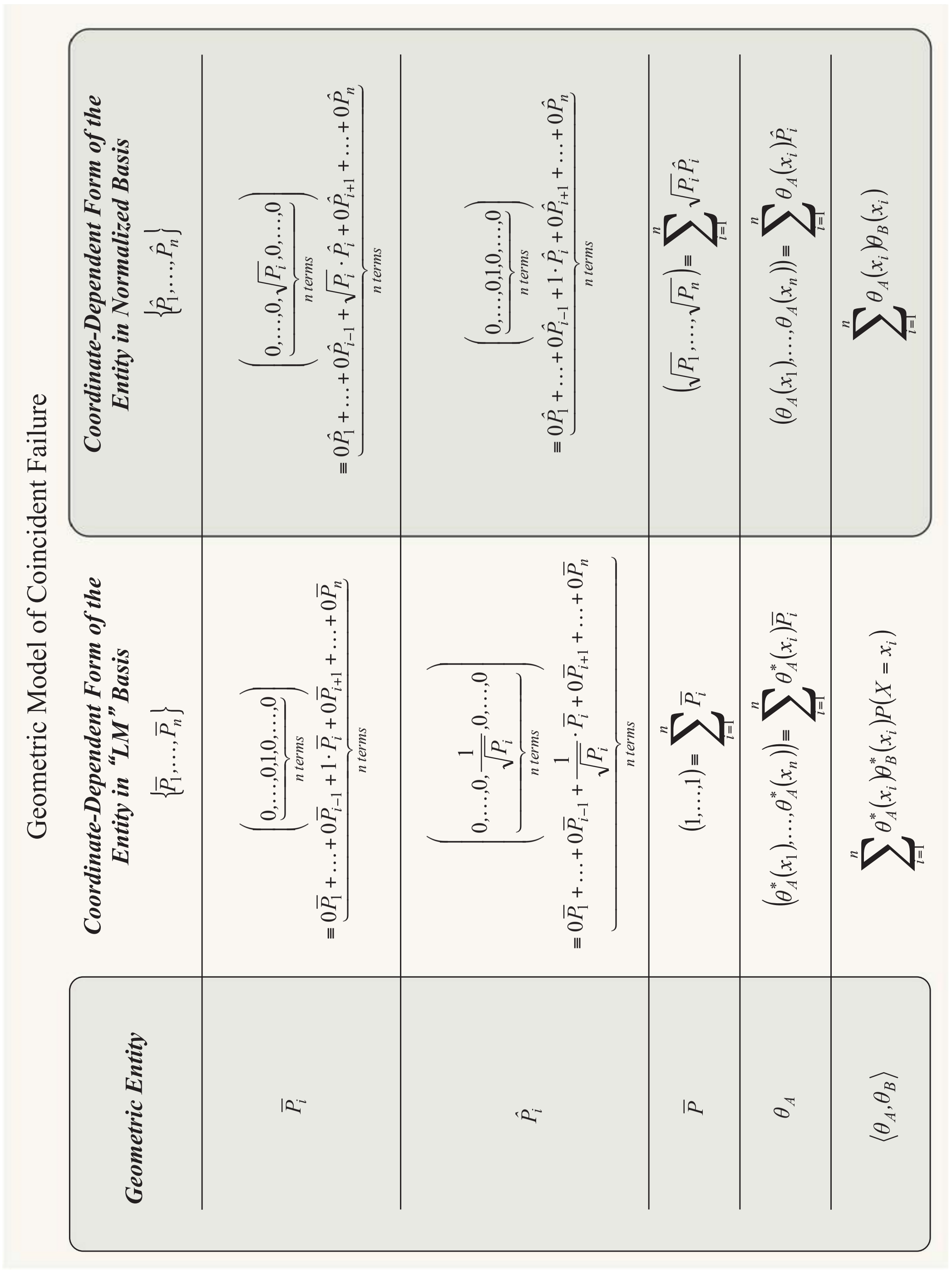
[9]The size of $\bar{P}$ is one since
$$\begin{aligned}\langle \bar{P}, \bar{P}\rangle &= \langle \sum_{i=1}^{n} \sqrt{P(X = x_i)}\hat{P}_i, \sum_{j=1}^{n} \sqrt{P(X = x_j)}\hat{P}_j\rangle \\ &= \sum_{i=1,j=1}^{n,n} \sqrt{P(X = x_i)}\sqrt{P(X = x_j)}\langle \hat{P}_i, \hat{P}_j\rangle \\ &= \sum_{i=1}^{n} P(X = x_i) \\ &= 1,\end{aligned}$$
where we used the bilinearity property of the inner–product and the fact that the basis vectors are an orthonormal set of vectors.

We now summarize the solutions to the extremisation problems of Chapter 5 in Fig.s 6.10, 6.11 and 6.14. But first, in Fig. 6.9, we express coordinate dependent forms of entities in the model using the bases

$$\mathcal{S} := \left\{\bar{P}_1, \ldots, \bar{P}_n\right\} \text{ and } \hat{\mathcal{S}} := \left\{\hat{P}_1, \ldots, \hat{P}_n\right\}$$

defined on pages 91 and 92, in Section 4.3. The entities used in the geometric model are:

1. $\bar{P}_i$ is the vector that models the failure set for the software version that fails only on the ith demand;
2. $\hat{P}_i$ is the unit vector in the same direction as the vector $\bar{P}_i$;
3. $\bar{P}$ is the vector such that in the basis $\mathcal{S}$ it models the software version that fails on all demands, and in the basis $\hat{\mathcal{S}}$ its components are completely defined by the demand profile (hence, why this vector is referred to as the demand profile);
4. $\theta_A$ is a vector whose components in the basis $\mathcal{S}$ are completely defined by a difficulty function $\theta_A(x)$. Consequently, we refer to such a vector as a difficulty function;
5. $\left\langle\theta_A, \theta_B\right\rangle$ is the inner–product of the vectors $\theta_A$ and $\theta_B$.



Geometric Model of Coincident Failure

| ***Geometric Entity*** | ***Coordinate-Dependent Form of the Entity in "LM" Basis*** $\left\{\bar{P}_1,\ldots,\bar{P}_n\right\}$ | ***Coordinate-Dependent Form of the Entity in Normalized Basis*** $\left\{\hat{P}_1,\ldots,\hat{P}_n\right\}$ |
|---|---|---|
| $\bar{P}_i$ | $\left(\underbrace{0,\ldots,0,1,0,\ldots,0}_{n\ terms}\right) \equiv \underbrace{0\bar{P}_1+\ldots+0\bar{P}_{i-1}+1\cdot\bar{P}_i+0\bar{P}_{i+1}+\ldots+0\bar{P}_n}_{n\ terms}$ | $\left(\underbrace{0,\ldots,0,\sqrt{P_i},0,\ldots,0}_{n\ terms}\right) \equiv \underbrace{0\hat{P}_1+\ldots+0\hat{P}_{i-1}+\sqrt{P_i}\cdot\hat{P}_i+0\hat{P}_{i+1}+\ldots+0\hat{P}_n}_{n\ terms}$ |
| $\hat{P}_i$ | $\left(\underbrace{0,\ldots,0,\frac{1}{\sqrt{P_i}},0,\ldots,0}_{n\ terms}\right) \equiv \underbrace{0\bar{P}_1+\ldots+0\bar{P}_{i-1}+\frac{1}{\sqrt{P_i}}\cdot\bar{P}_i+0\bar{P}_{i+1}+\ldots+0\bar{P}_n}_{n\ terms}$ | $\left(\underbrace{0,\ldots,0,1,0,\ldots,0}_{n\ terms}\right) \equiv \underbrace{0\hat{P}_1+\ldots+0\hat{P}_{i-1}+1\cdot\hat{P}_i+0\hat{P}_{i+1}+\ldots+0\hat{P}_n}_{n\ terms}$ |
| $\bar{P}$ | $(1,\ldots,1)\equiv\sum_{i=1}^{n}\bar{P}_i$ | $\left(\sqrt{P_1},\ldots,\sqrt{P_n}\right)\equiv\sum_{i=1}^{n}\sqrt{P_i}\hat{P}_i$ |
| $\theta_A$ | $\left(\theta_A^*(x_1),\ldots,\theta_A^*(x_n)\right)\equiv\sum_{i=1}^{n}\theta_A^*(x_i)\bar{P}_i$ | $\left(\theta_A(x_1),\ldots,\theta_A(x_n)\right)\equiv\sum_{i=1}^{n}\theta_A(x_i)\hat{P}_i$ |
| $\langle\theta_A,\theta_B\rangle$ | $\sum_{i=1}^{n}\theta_A^*(x_i)\theta_B^*(x_i)P(X=x_i)$ | $\sum_{i=1}^{n}\theta_A(x_i)\theta_B(x_i)$ |

**Figure 6.9:** This is a list of entities in the geometric model expressed in two bases: the orthogonal basis $\mathcal{S}$, suggested by the LM model, and the related orthonormal basis $\hat{\mathcal{S}}$, which consists of unit vectors that are collinear with the $\mathcal{S}$ basis vectors. In $\mathcal{S}$ the vectors of interest have components that are completely defined by the difficulty functions used in the LM model. In the alternative $\hat{\mathcal{S}}$ basis the vectors have components that are not difficulty functions, but are related to the difficulty functions by scalings (see Section 4.4 for the definition of a transformation $T$ that implements this scaling). Recall the following notation: given $n$ demands, and the demand profile $\{P(X=x_1),\ldots,P(X=x_n)\}$, for the sake of brevity we write $P_i = P(X=x_i)$, for each $i=1,\ldots,n$. This notation is distinct from the related vectors $\hat{P}_i$, $\bar{P}_i$ and $\bar{P}$.

Optimisation of Expected System *pfd* , $q_{AB}$, under Forcing Diversity

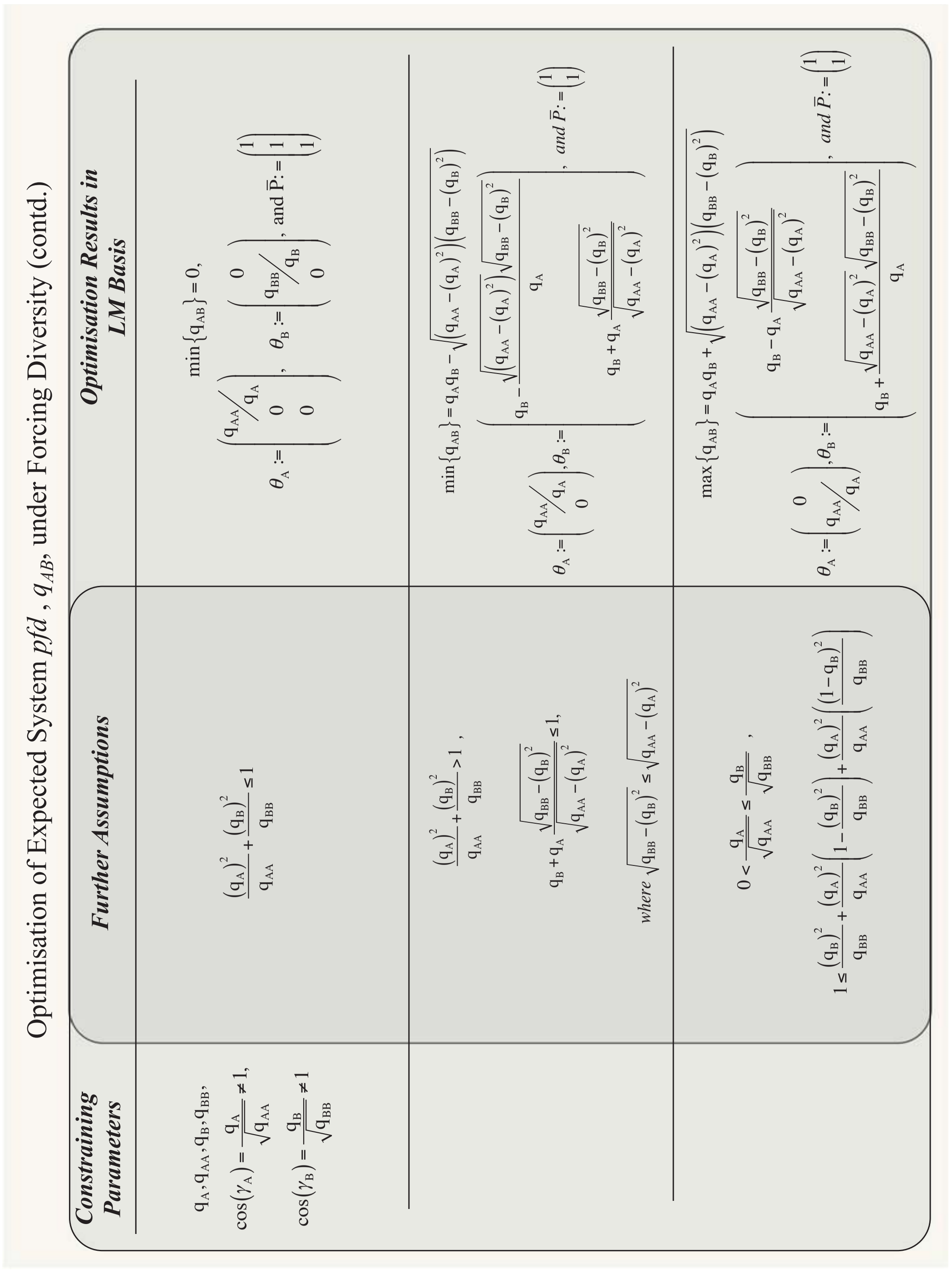
| ***Constraining Parameters*** | ***Further Assumptions*** | ***Optimisation Results*** |
|---|---|---|
| $\overline{P}, q_A, \theta_B$ | | $max\{q_{AB}\} = \sum_{i=1}^{j} P_i \theta_B(x_i) + \left(q_A - \sum_{i=1}^{j} P_i\right)\theta_B(x_{j+1}),$ $where\ \theta_B(x_1) \geq \ldots \geq \theta_B(x_n)\ and\ \sum_{i=1}^{j} P_i \leq q_A \leq \sum_{i=1}^{j+1} P_i$ |
| $\overline{P}$ | | $max\{q_{AB}\} = 1\ and\ min\{q_{AB}\} = 0$ |
| $\overline{P}, q_B, q_{BB}$ | | $max\{q_{AB}\} = q_B\ and\ min\{q_{AB}\} = 0$ |
| $\overline{P}, \theta_B$ | | $max\{q_{AB}\} = q_B\ and\ min\{q_{AB}\} = 0$ |
| $\overline{P}, q_A, q_B$ | $q_A = q_B$ | $max\{q_{AB}\} = the\ largest\ value\ for\ q_{AA}$ |
| | *The largest difficulty functions lie in the same 1-dimensional subspace spanned by the imperfect single version* $\overline{V}$. | $max\{q_{AB}\} = \frac{q_A}{\lVert\overline{V}\rVert}\frac{q_B}{\lVert\overline{V}\rVert}$ |
| | *There exists an imperfect, single version* $\overline{V}$ *such that* $q_A \leq \lVert\overline{V}\rVert \leq q_B$ | $max\{q_{AB}\} = q_A$ |
| $\overline{P}, q_{AA}, q_{BB}$ | | $max\{q_{AB}\} = \sqrt{q_{AA}}\sqrt{q_{BB}}$ |

**Figure 6.10:** The results of a number of constrained optimisation problems with the same objective function $q_{AB}$ (being the expected system *pfd* when diversity is forced during software development) are summarized. This summary is only intended to be a brief overview as the precise conditions under which each result holds are given in Chapter 5. Furthermore, these bounds are all attainable and the difficulty functions that ultimately result in these bounds are also given in Chapter 5.

Optimisation of Expected System *pfd* , $q_{AB}$, under Forcing Diversity (contd.)

| ***Constraining Parameters*** | ***Further Assumptions*** | ***Optimisation Results in Normalized Basis*** |
|---|---|---|
| $q_A, q_{AA}, q_B, q_{BB}$, $\cos(\gamma_A) = \frac{q_A}{\sqrt{q_{AA}}} \neq 1$, $\cos(\gamma_B) = \frac{q_B}{\sqrt{q_{BB}}} \neq 1$ | $\gamma_A + \gamma_B \geq \frac{\pi}{2}$ | $min\{q_{AB}\} = 0,\ \theta_A := \begin{pmatrix} \sqrt{q_{AA}} \\ 0 \\ 0 \end{pmatrix},\ \theta_B := \begin{pmatrix} 0 \\ \sqrt{q_{BB}} \\ 0 \end{pmatrix},$ $and\ \overline{P} := \begin{pmatrix} cos(\gamma_A) \\ cos(\gamma_B) \\ \sqrt{1-\left((cos(\gamma_A))^2 + (cos(\gamma_B))^2\right)} \end{pmatrix}$ |
| | $\gamma_A + \gamma_B < \frac{\pi}{2}$ , $\sin(\gamma_A + \gamma_B) \leq \frac{\max\left\{\sqrt{q_{AA} - (q_A)^2}, \sqrt{q_{BB} - (q_B)^2}\right\}}{\sqrt{q_{AA}}\sqrt{q_{BB}}}$, *where* $\max\left\{\sqrt{q_{AA} - (q_A)^2}, \sqrt{q_{BB} - (q_B)^2}\right\} = \sqrt{q_{AA} - (q_A)^2}$ | $min\{q_{AB}\} = q_A q_B - \sqrt{\left(q_{AA} - (q_A)^2\right)\left(q_{BB} - (q_B)^2\right)}$ $\theta_A := \begin{pmatrix} \sqrt{q_{AA}} \\ 0 \end{pmatrix}, \theta_B := \begin{pmatrix} \sqrt{q_{BB}}\ cos(\gamma_A + \gamma_B) \\ \sqrt{q_{BB}}\ sin(\gamma_A + \gamma_B) \end{pmatrix},$ $and\ \overline{P} := \begin{pmatrix} cos(\gamma_A) \\ \sqrt{1-(cos(\gamma_A))^2} \end{pmatrix}$ |
| | $\frac{\pi}{2} > \gamma_A \geq \gamma_B$ , $cos(\gamma_A - \gamma_B) \leq \frac{q_A}{\sqrt{q_{AA}}\sqrt{q_{BB}}}$ | $max\{q_{AB}\} = q_A q_B + \sqrt{\left(q_{AA} - (q_A)^2\right)\left(q_{BB} - (q_B)^2\right)}$ $\theta_A := \begin{pmatrix} 0 \\ \sqrt{q_{AA}} \end{pmatrix}, \theta_B := \begin{pmatrix} \sqrt{q_{BB}}\ sin(\gamma_A - \gamma_B) \\ \sqrt{q_{BB}}\ cos(\gamma_A - \gamma_B) \end{pmatrix},$ $and\ \overline{P} := \begin{pmatrix} \sqrt{1-(cos(\gamma_A))^2} \\ cos(\gamma_A) \end{pmatrix}$ |

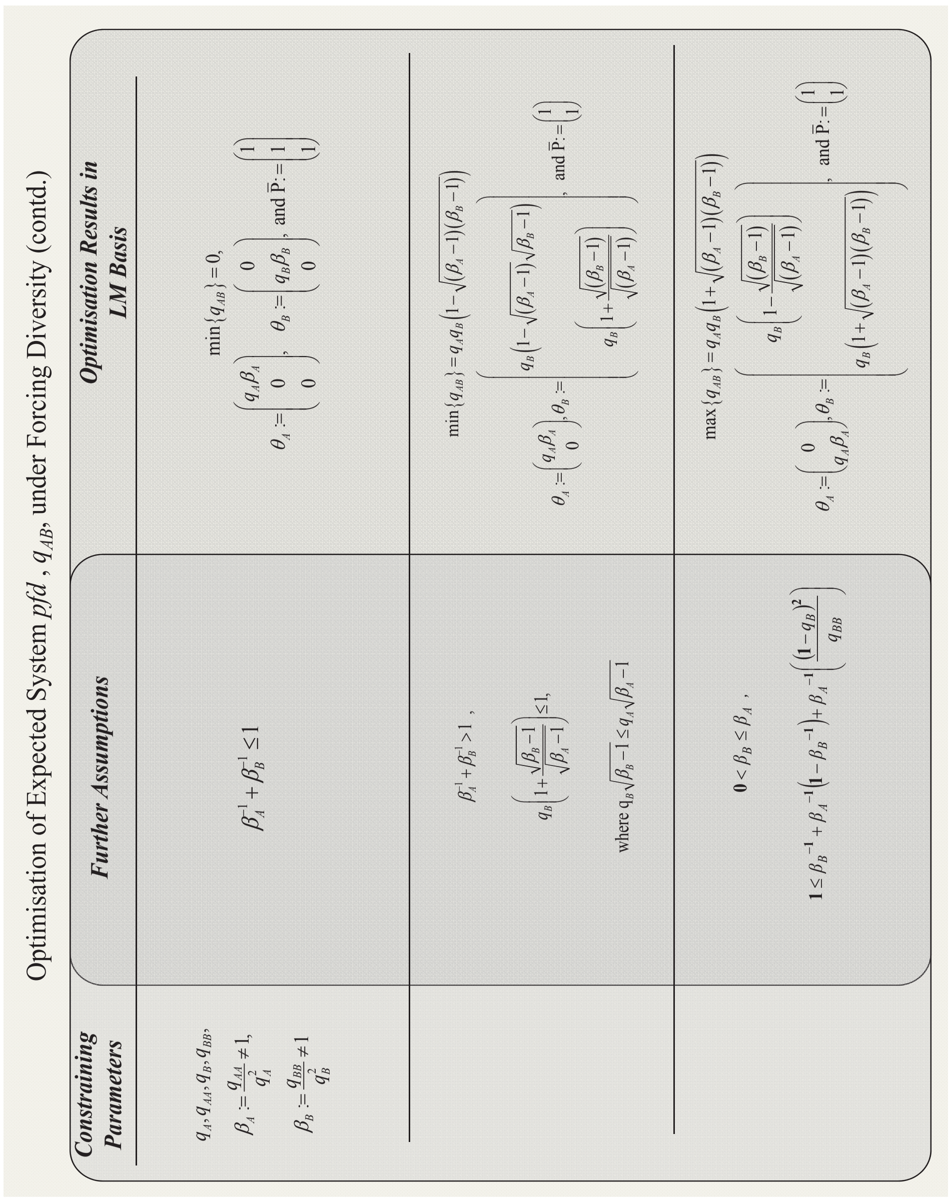
**Figure 6.11:** Further results of constrained optimisation problems with the objective function $q_{AB}$. Again, these bounds are attainable and details of the difficulty functions that assure the attainment of these bounds can be found in Chapter 5.



Optimisation of Expected System *pfd* , $q_{AB}$, under Forcing Diversity (contd.)

| Constraining Parameters | Further Assumptions | Optimisation Results in LM Basis |
|---|---|---|
| $q_A, q_{AA}, q_B, q_{BB}$, $\cos(\gamma_A) = \frac{q_A}{\sqrt{q_{AA}}} \neq 1$, $\cos(\gamma_B) = \frac{q_B}{\sqrt{q_{BB}}} \neq 1$ | $\frac{(q_A)^2}{q_{AA}} + \frac{(q_B)^2}{q_{BB}} \leq 1$ | $\min\{q_{AB}\} = 0$, $\theta_A := \begin{pmatrix} q_{AA}/q_A \\ 0 \\ 0 \end{pmatrix}$, $\theta_B := \begin{pmatrix} 0 \\ q_{BB}/q_B \\ 0 \end{pmatrix}$, and $\bar{P} := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ |
| | $\frac{(q_A)^2}{q_{AA}} + \frac{(q_B)^2}{q_{BB}} > 1$ , $q_B + q_A \frac{\sqrt{q_{BB} - (q_B)^2}}{\sqrt{q_{AA} - (q_A)^2}} \leq 1$, *where* $\sqrt{q_{BB} - (q_B)^2} \leq \sqrt{q_{AA} - (q_A)^2}$ | $\min\{q_{AB}\} = q_A q_B - \sqrt{\left(q_{AA} - (q_A)^2\right)\left(q_{BB} - (q_B)^2\right)}$ $\theta_A := \begin{pmatrix} q_{AA}/q_A \\ 0 \end{pmatrix}, \theta_B := \begin{pmatrix} q_B - \frac{\sqrt{\left(q_{AA} - (q_A)^2\right)}\sqrt{q_{BB} - (q_B)^2}}{q_A} \\ q_B + q_A \frac{\sqrt{q_{BB} - (q_B)^2}}{\sqrt{q_{AA} - (q_A)^2}} \end{pmatrix}$, *and* $\bar{P} := \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ |
| | $0 < \frac{q_A}{\sqrt{q_{AA}}} \leq \frac{q_B}{\sqrt{q_{BB}}}$ , $1 \leq \frac{(q_B)^2}{q_{BB}} + \frac{(q_A)^2}{q_{AA}}\left(1 - \frac{(q_B)^2}{q_{BB}}\right) + \frac{(q_A)^2}{q_{AA}}\left(\frac{(1-q_B)^2}{q_{BB}}\right)$ | $\max\{q_{AB}\} = q_A q_B + \sqrt{\left(q_{AA} - (q_A)^2\right)\left(q_{BB} - (q_B)^2\right)}$ $\theta_A := \begin{pmatrix} 0 \\ q_{AA}/q_A \end{pmatrix}, \theta_B := \begin{pmatrix} q_B - q_A \frac{\sqrt{q_{BB} - (q_B)^2}}{\sqrt{q_{AA} - (q_A)^2}} \\ q_B + \frac{\sqrt{q_{AA} - (q_A)^2}\sqrt{q_{BB} - (q_B)^2}}{q_A} \end{pmatrix}$, *and* $\bar{P} := \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ |

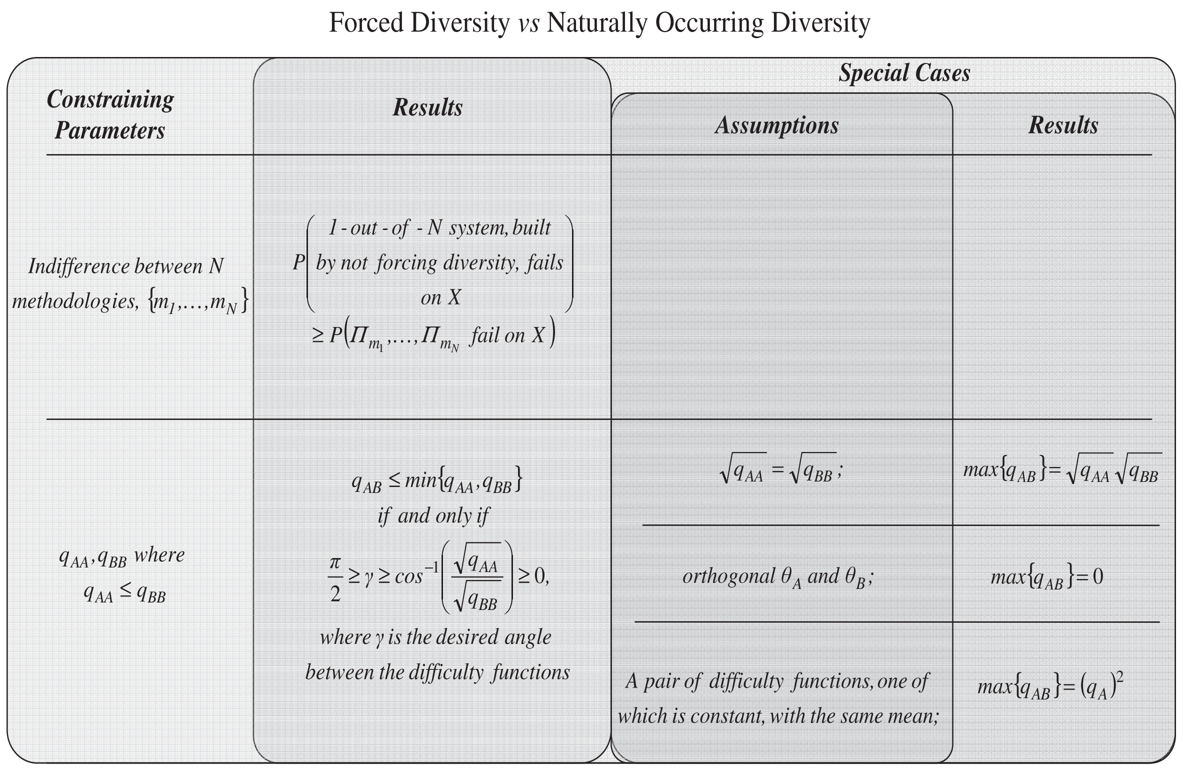
**Figure 6.12:** This figure is equivalent to the summary of Fig. 6.11. Here, however, the results are in terms of the original LM model formulation. Note that the constraints of the optimisation problems may not have an easily identifiable probabilistic form, but may have a simple geometric one. For instance, the requirement $1 \leq \frac{{q_B}^2}{q_{BB}} + \frac{{q_A}^2}{q_{AA}}\left(1 - \frac{{q_B}^2}{q_{BB}}\right) + \frac{{q_A}^2}{q_{AA}}\left(\frac{(1-q_B)^2}{q_{BB}}\right)$ is equivalent to the requirement $\cos(\gamma_A - \gamma_B) \leq \frac{q_A}{\sqrt{q_{AA}}\sqrt{q_{BB}}}$, which is the requirement that the dot product of a pair of difficulty functions (separated by an angle $\gamma_A - \gamma_B$) is less than or equal to $q_A$.



Optimisation of Expected System *pfd* , $q_{AB}$, under Forcing Diversity (contd.)

| ***Constraining Parameters*** | ***Further Assumptions*** | ***Optimisation Results in LM Basis*** |
|---|---|---|
| $q_A, q_{AA}, q_B, q_{BB}$, $\beta_A := \frac{q_{AA}}{q_A^2} \neq 1$, $\beta_B := \frac{q_{BB}}{q_B^2} \neq 1$ | $\beta_A^{-1} + \beta_B^{-1} \leq 1$ | $\min\{q_{AB}\} = 0$, $\theta_A := \begin{pmatrix} q_A\beta_A \\ 0 \\ 0 \end{pmatrix}$, $\theta_B := \begin{pmatrix} 0 \\ q_B\beta_B \\ 0 \end{pmatrix}$, and $\bar{\text{P}} := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ |
| | $\beta_A^{-1} + \beta_B^{-1} > 1$ , $q_B\left(1+\frac{\sqrt{\beta_B - 1}}{\sqrt{\beta_A - 1}}\right) \leq 1$, where $q_B\sqrt{\beta_B - 1} \leq q_A\sqrt{\beta_A - 1}$ | $\min\{q_{AB}\} = q_A q_B\left(1-\sqrt{(\beta_A - 1)(\beta_B - 1)}\right)$ $\theta_A := \begin{pmatrix} q_A\beta_A \\ 0 \end{pmatrix}, \theta_B := \begin{pmatrix} q_B\left(1-\sqrt{(\beta_A - 1)}\sqrt{\beta_B - 1}\right) \\ q_B\left(1+\frac{\sqrt{(\beta_B - 1)}}{\sqrt{(\beta_A - 1)}}\right) \end{pmatrix}$, and $\bar{\text{P}} := \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ |
| | $\mathbf{0} < \beta_B \leq \beta_A$ , $\mathbf{1} \leq \beta_B^{\mathbf{-1}} + \beta_A^{\mathbf{-1}}\left(\mathbf{1} - \beta_B^{\mathbf{-1}}\right) + \beta_A^{\mathbf{-1}}\left(\frac{(\mathbf{1} - q_B)^{\mathbf{2}}}{q_{BB}}\right)$ | $\max\{q_{AB}\} = q_A q_B\left(1+\sqrt{(\beta_A - 1)(\beta_B - 1)}\right)$ $\theta_A := \begin{pmatrix} 0 \\ q_A\beta_A \end{pmatrix}, \theta_B := \begin{pmatrix} q_B\left(1-\frac{\sqrt{(\beta_B - 1)}}{\sqrt{(\beta_A - 1)}}\right) \\ q_B\left(1+\sqrt{(\beta_A - 1)(\beta_B - 1)}\right) \end{pmatrix}$, and $\bar{\text{P}} := \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ |

**Figure 6.13:** The results here are equivalent to the results of Fig.s 6.11 and 6.12. The emphasis here is on beta–factors: that is, the relative sizes of expected *pfd*s. In the optimisation problem, the upper bound on $q_{AB}$ suggests that the difference between forcing diversity and allowing it to occur naturally by using methodology "B" say, is at most $\left|\log_{10}\left(\frac{q_A\left(1+\sqrt{(\beta_A - 1)(\beta_B - 1)}\right)}{q_B\beta_B}\right)\right|$ orders of magnitude.



Forced Diversity *vs* Naturally Occurring Diversity

| Constraining Parameters | Results | Special Cases: Assumptions | Special Cases: Results |
|---|---|---|---|
| *Indifference between N methodologies,* $\{m_1,\dots,m_N\}$ | $P\left(\begin{array}{c}\textit{1-out-of-N system, built}\\ \textit{by not forcing diversity, fails}\\ \textit{on } X\end{array}\right) \geq P\left(\Pi_{m_1},\dots,\Pi_{m_N}\ \textit{fail on } X\right)$ | | |
| $q_{AA}, q_{BB}$ *where* $q_{AA} \leq q_{BB}$ | $q_{AB} \leq min\{q_{AA}, q_{BB}\}$ *if and only if* $\frac{\pi}{2} \geq \gamma \geq cos^{-1}\left(\frac{\sqrt{q_{AA}}}{\sqrt{q_{BB}}}\right) \geq 0,$ *where* $\gamma$ *is the desired angle between the difficulty functions* | $\sqrt{q_{AA}} = \sqrt{q_{BB}}\,;$ | $max\{q_{AB}\} = \sqrt{q_{AA}}\sqrt{q_{BB}}$ |
| | | *orthogonal* $\theta_A$ *and* $\theta_B$*;* | $max\{q_{AB}\} = 0$ |
| | | *A pair of difficulty functions, one of which is constant, with the same mean;* | $max\{q_{AB}\} = (q_A)^2$ |

**Figure 6.14:** Conditions under which forcing diversity guarantees a lower bound on the expected system *pfd*, compared with if diversity is not forced.

# Chapter 7

# Suggestions for Future Work

It might be beneficial to explore the implications of changes to a development process, where such changes are modeled as transformations of vectors representing the difficulty functions. Can changes to the development process manifest as rotations, scalings and, or projections? And would this mean that via such changes to the development process one might be able to "navigate" the space of difficulty functions in preferred ways? Even if a given change to the development process does not result in a vector transformation, perhaps it induces a constraining surface in the vector space, such a surface representing the set of difficulty functions that could potentially result from the given change. In such cases there might even be orderings between the sizes of expected *pfd*s resulting from different changes. In this sense some changes are "better"[1] than others, or are simply improvements. For instance, in [52] the consequence of software reliability growth on the ratio of expected single–version *pfd* to expected system *pfd* (that is, the ratio $\frac{q_A}{q_{AA}}$ say ) was modelled thus: for each demand, $x$, the value of some relevant difficulty function, $\theta_A(x)$ say, is scaled by some non–negative amount $f(x)$. In essence, reliability growth is achieved by practical activities that, in effect, transform one difficulty function into another. An alternative approach would be to model the effects of practical activities that result in reliability growth as geometric transformations on difficulty functions. Then, the practical activities of interest are those with associated geometric transformations that ensure the ratio $\frac{q_A}{q_{AA}}$ satisfies some property, e.g. the ratio increases.

Additionally, a duality between the demands and the programs exists in the LM model. For a given demand there is an associated "difficulty" which is the probability a version will be developed that fails on the demand. This characterises how different development processes can result in the same demand having different difficulty: some ways of organising the system development result in the demand being easy, while others result in the demand being difficult. Similarly, for a given program, there is its associated *pfd* which is the probability that in operation the next randomly occurring demand will cause the version to fail.

[1]Changes may be better because they result in a lowering of the expected system pfd. However, in a wider practical context, the economic implications of a change for the system development process might imply a change is not preferable, even if it does result in improved reliability.

This implies that different demand profiles may result in different *pfd*s: some profiles make the demands in the failure set for a given version very unlikely, while others make them more likely. Now, the geometric model was developed from the viewpoint of different development processes and a fixed demand profile, with the demands defining the dimensions of the vector space and difficulty functions defining the $n$–tuples. It might be useful to explore the consequences of a "dual" geometric formulation: one from the viewpoint of different demand profiles and a fixed development process, where the dimensionality of the space is equal to the number of possible versions and the $n$–tuples consist of version *pfd*s. In such a formulation, by performing transformations on vectors, one can study the effect of different demand profiles on the reliability of software produced from a given development process. Such analyses is useful when reasoning about the behaviour of systems under different environmental conditions. For instance, the potential difference between the demand profile used when testing the system and the demand profile when the system has been deployed.

Another area for future work is in extending the modeling beyond 1–out–of–N systems. While these are an important category of systems, and illustrate the basic problems in managing "diversity", the results obtained in this context may not scale as expected, or may not transfer altogether to other fault–tolerant architectures. Certainly, in the case of the LM model, Littlewood and Miller [13] showed that not all results extend in intuitive ways from the 1-out-of-2 case to voted (k–out–of–n) architectures, and further subtle, counter–intuitive effects are possible.

The expected *pfd*s are averages so care should be taken in their interpretation and use. In particular, a 1–out–of–N system's actual *pfd*, when built, will normally differ from the expected *pfd*. Nevertheless, before the 1–out–of–N system is built it is unknown which programs will actually be developed to make up the channels. Therefore, there is uncertainty about the failure behaviour of the actual system that is built. It is this uncertainty that makes the system pfd "*expected*". That is, given some configuration of methodologies to be used during the developments of the channels, the expected system *pfd* is an average over all of the system *pfd*s of the possible 1–out–of–N systems that could be built. This is described in more detail in [34]. Models that take into account the variation of the *pfd*s of 1–out–of–2 systems that could possibly be built have been explored in the literature [53]. An extension of the current work would be to consider bounds on the distributions of the versions and version pairs.

The LM model has been applied to numerous practical applications [54]: from the modeling of the consequences of diversity to security [18], to the modeling of the impact of Human–machine diversity on the effectiveness of breast–cancer screening approaches [55]. By extending the LM model to cases of dependent software development as we have done, there is plenty of scope to revisit these applications of the LM model and study what the implications of dependence would be. For instance, the model of the fault–creation process used in [53] may be extended, relaxing the assumption of the independent insertion of faults. As a consequence could dependence result in much better, or much worse, reliability gains than the authors predict would occur from "improving" development processes?

Also, like the LM model, there is significant potential for applying both the generalised models of dependent development and the geometric models beyond modeling coincident failure in software. The reason is that at the heart of the models is the idea that dependence may be modelled via conditional independence. Any system which may be conceptualised this way could benefit from the model results. It would be interesting to apply both models to other areas of Computer Science and Engineering applications – wherever dependence between processes has been ignored by assuming independence.

Further analysis of the generalised models may reveal more "preference criteria", with sufficient conditions that are clear enough to be recognised in practical situations. It would be especially interesting to find more examples of system development policies that should create useful, negative covariance between the difficulty functions of the two channel development processes.

In this thesis, forcing diversity has been demonstrated to be beneficial, but only under certain conditions and only in the sense that it cannot result in worsening expected system *pfd*. This would be sufficient to employ forced diversity in practice if the economic cost of forcing diversity is not a constraint. However, in general, judgments about whether to force diversity should also take into account the cost of implementing a regime of forcing diversity, and consider the trade-offs between cost and reliability gains. An economic/utility model of the benefits of forcing diversity could take our theorems and bounds into account, thereby aiding system developers in deciding whether to force diversity under more general settings.

Finally, there are particular results that could benefit from further generalisation or require more rigorous proof. For instance, does Theorem 5.2.2 (see Chapter 5, page 135) still hold if the number of methodologies is larger than the number of channels? Initial investigation suggests that the result should still hold for a larger pool of methodologies since it is true for any set of n–distinct methodologies, and the more general case would involve "simply" sampling without replacement n–times from a larger methodology pool. Also, a rigourous proof of case 4 in Section 5.3.6 is a point of current investigation. For the reasons stated in Section 5.3.6 the result is expected to be provable.

# Appendix A

# Finite–Dimensional, Real Inner–Product Spaces

This appendix is a review of vector spaces and linear algebra. The aim is to provide a sufficient amount of geometric background for work presented in the thesis. Some texts that provide an excellent introduction and comprehensive treatment of some of the material presented here include [40, 41, 42, 43, 44]. However, the development of vector spaces presented here is a heuristic development tailored specifically with the framework of the EL/LM models in mind, and with the aim of performing extremisation using geometric manipulations. We enumerate the 3 sections according to their section numbers, giving details of each section's focus.

A.1 *Vectors, Vector Spaces and Basis Vectors*: Defines fundamental geometric entities and concepts such as vectors, spans, linear independence and subspaces;

A.2 *Real Inner–Product*: Discusses the definition of a very important construct; the so–called *inner–product*. The inner–product imbues vector spaces with useful geometric notions such as angles and lengths;

A.3 *Orthogonality and Collinearity*: Defines the concept of an angle between a pair of vectors in a finite–dimensional vector space. In particular, the notion of perpendicularity used in 2–dimensional Euclidean geometry is generalised to any finite–dimensional vector space.

## A.1 Vectors, Vector Spaces and Basis Vectors

We are interested in defining and manipulating vectors; these are entities defined by a ***vector space***. For our purposes a *vector space* is a set $\mathbf{V}$, the set of real numbers $\mathbb{R}$ and a pair of *binary operations* [1] called vector addition, denoted by $+$, and scalar multiplication, denoted by $\cdot$, that adhere to the eight axioms given below. *The elements of* $\mathbf{V}$ *are called* ***vectors*** and *the real numbers* – elements of $\mathbb{R}$ – *are called* ***scalars***. Let $\bar{U}, \bar{V}, \bar{W} \in \mathbf{V}$ and let $a, b \in \mathbb{R}$.

[1]Suppose we have defined a set of entities $\mathcal{S}$. A Binary operation is a function $f : (\mathcal{S} \times \mathcal{S}) \to \mathcal{S}$.

The sum of two vectors is denoted $\bar{V}+\bar{W}$ while the product of a scalar, a ,and a vector, $\bar{V}$, is denoted $a\cdot\bar{V}$ or $a\bar{V}$. The eight axioms are:

1. *Associativity of vector addition*: $\bar{U}+\left(\bar{V}+\bar{W}\right)=\left(\bar{U}+\bar{V}\right)+\bar{W}$.
2. *Commutativity of vector addition*: $\bar{V}+\bar{W}=\bar{W}+\bar{V}$.
3. *Identity element of vector addition*: There exists an element, $\bar{0}\in\mathbf{V}$, called the zero vector such that $\bar{V}+\bar{0}=\bar{V}$, for all $\bar{V}\in\mathbf{V}$.
4. *Inverse elements of vector addition*: For all $\bar{V}\in\mathbf{V}$ there exists an element $\bar{W}\in\mathbf{V}$, called the additive inverse of $\bar{V}$, such that $\bar{V}+\bar{W}=\bar{0}$. The additive inverse is denoted $\left(-\bar{V}\right)$.
5. *Distributivity of scalar multiplication with respect to vector addition*: $a\left(\bar{V}+\bar{W}\right)=a\bar{V}+a\bar{W}$.
6. *Distributivity of scalar multiplication over the addition of real numbers*: $(a+b)\,\bar{V}=a\bar{V}+b\bar{V}$.
7. *Compatibility of scalar multiplication with the multiplication of real numbers*: $a\left(b\bar{V}\right)=(ab)\,\bar{V}$.
8. *Identity element of scalar multiplication*: The multiplicative identity, $1\in\mathbb{R}$, is such that for arbitrary vector $\bar{V}\in\mathbf{V}$, $1\bar{V}=\bar{V}$ .

A number of results follow from these axioms. For instance, the zero vector, $\bar{0}$, is unique[2]. Also[3], $\bar{0}=0\bar{V}$. From this it follows[4] that $\left(-\bar{V}\right)=-1\cdot\bar{V}$; that is, the additive inverse of a vector is the vector multiplied by the scalar $-1$. From here on we shall write $-1\cdot\bar{V}$ as $-\bar{V}$. Finally, we can define the difference of two vectors and division of a vector by a (non-zero) scalar, since $\bar{V}-\bar{X}=\bar{V}+\left(-\bar{X}\right)$, $\bar{V}/a=(1/a)\cdot\bar{V}$. The definition of vectors given above may appear fairly abstract. However, there is a plethora of vector space examples, some of which are:

- The set containing only the zero vector, $\{\bar{0}\}$. Both vector addition and scalar multiplication result in $\bar{0}$;
- The set of real numbers, $\mathbb{R}$, is a vector space with vector addition being the usual addition of numbers and scalar multiplication is the usual multiplication of real numbers. This is a very important vector space, partly because any practical activity that is adequately modelled by the real number line is therefore a vector space e.g. credit status, angles, displacement, e.t.c.;
- The velocity of a body. Vector addition and scalar multiplication are defined most easily by representing these vectors in $\mathbb{R}^n$;
- The set of all $m\times n$ matrices with real number entries. Vector addition is matrix addition and scalar multiplication is the multiplication of each entry of the matrix by a real number;

[2]Since if we had two distinct zero vectors, $\bar{0}_1$ and $\bar{0}_2$, then by definition $\bar{0}_1=\bar{0}_1+\bar{0}_2=\bar{0}_2$, which is a contradiction.

[3]Since $0\bar{V}=0\bar{V}+\bar{V}+\left(-\bar{V}\right)=(0+1)\,\bar{V}+\left(-\bar{V}\right)=\bar{V}+\left(-\bar{V}\right)=\bar{0}$.

[4]Since $\left(-\bar{V}\right)=\bar{0}+\left(-\bar{V}\right)=0\bar{V}+\left(-\bar{V}\right)=(1-1)\,\bar{V}+\left(-\bar{V}\right)=\bar{V}-\bar{V}+\left(-\bar{V}\right)=-1\cdot\bar{V}$.

- The set of polynomials with coefficients in $\mathbb{R}$ and degree less than or equal to $n$ is a vector space over $\mathbb{R}$, denoted by $\mathbf{P}_n$. Vector addition and scalar multiplication are defined by adding corresponding polynomial terms and distributing scalar multiplications over the coefficients of polynomials, respectively. For example, $(8x^3 + 6x^2 - 3) + (-7x^2 + 2x) = 8x^3 - x^2 + 2x - 3$ and $4(5x^2 + 3x + 2) = 20x^2 + 12x + 8$. This is an (n+1)–dimensional vector space;

Vector spaces are said to be *closed* under vector addition and scalar multiplication, which means that the results of such operations are vectors. Consequently, a combination of these operations also results in a vector. For instance, given the vectors $\bar{V}_1, \ldots, \bar{V}_j \in \mathbf{V}$ and scalars $a_1, \ldots, a_j \in \mathbb{R}$ we may obtain the vector $\bar{W} = a_1\bar{V}_1 + \ldots + a_j\bar{V}_j$. $\bar{W}$ is referred to as a "*linear combination*" of the vectors $\bar{V}_1, \ldots, \bar{V}_j$. The ***span*** of a set of vectors is the set of all the possible linear combinations of the vectors. We shall be concerned only with the spans of finite sets of vectors. Consider the set of vectors $\mathcal{S} = \{\bar{P}_1, \ldots, \bar{P}_n\}$, and its associated span denoted by Span($\mathcal{S}$). If it is impossible to define a proper subset of $\mathcal{S}$ that also spans Span($\mathcal{S}$), then we say that the members of $\mathcal{S}$ are ***linearly independent***. There is an equivalent way of stating this via the following theorem.

**Theorem A.1.1.** *The members of $\mathcal{S}$ are linearly independent if, and only if,*

$$a_1\bar{P}_1 + \ldots + a_n\bar{P}_n = \bar{0} \implies a_1, \ldots, a_n = 0.$$

*Proof.* To prove sufficiency we shall assume that there exists a non-trivial linear combination of the vectors in $\mathcal{S}$ that results in the zero vector, and show that this implies there exists a proper subset of $\mathcal{S}$ that also spans Span($\mathcal{S}$). Assume, for some $\bar{P}_i$ we can write $a_1\bar{P}_1 + \ldots + a_{i-1}\bar{P}_{i-1} + a_{i+1}\bar{P}_{i+1} + \ldots + a_n\bar{P}_n = -a_i\bar{P}_i$, for $a_i \neq 0$. Then it is possible to write $\bar{P}_i$ as a linear combination of the other vectors in $\mathcal{S}$. This means that these other vectors – a proper subset of $\mathcal{S}$ – span the vector space.

To show necessity we shall assume that there exists a proper subset of $\mathcal{S}$ that also spans Span($\mathcal{S}$), and show that this implies there exists a non-trivial linear combination of the vectors in $\mathcal{S}$ that results in the zero vector. Assume that some proper subset of $\mathcal{S}$, say $\mathcal{M}$, also spans Span($\mathcal{S}$). Then, for a given vector in $\mathcal{S}$ not contained in $\mathcal{M}$, say $\bar{P}_i$, there exists some linear combination of vectors in $\mathcal{M}$ that yields $\bar{P}_i$. However, this means that it is possible to write $\bar{0}$ as a linear combination of the vectors in $\mathcal{M}$ and $\bar{P}_i$ where not all of the coefficients are zero. ■

For our purposes a ***subspace***, $\mathbf{S}$, of a vector space[5] $\mathbf{V}$ is a subset of $\mathbf{V}$ that is, itself, a vector space with respect to $\mathbb{R}$ and the same vector space operations as $\mathbf{V}$. For instance, examples of subspaces in $\mathbb{R}^3$ include $\mathbb{R}^3$ itself and lines, or planes, that pass through the origin. In order to show that $\mathbf{S}$ is a vector space we need only show that for any $\bar{U}, \bar{V} \in \mathbf{S}$ and any $a, b \in \mathbb{R}$ we have $a\bar{U} + b\bar{V} \in \mathbf{S}$. For example,

[5]The vector space is actually the quadruplet $\{\mathbf{V}, \mathbb{R}, +, \cdot\}$. Nevertheless, it is usual to refer to $\mathbf{V}$ as the vector space when the context is clear.

**Theorem A.1.2.** *Let $\bar{V} \in \mathbf{V}$. Then, Span$\big(\{\bar{V}\}\big)$ is a subspace.*

*Proof.* By definition, $\text{Span}\big(\{\bar{V}\}\big) = \big\{\bar{W} \in \mathbf{V} | \bar{W} = a\bar{V} \text{ for some } a \in \mathbb{R}\big\}$. Choose any $\bar{X}, \bar{Y} \in \text{Span}\left(\{\bar{V}\}\right)$ and any $a, b \in \mathbb{R}$. From the definition of $\text{Span}\big(\{\bar{V}\}\big)$ there exist $c, d \in \mathbb{R}$ such that $\bar{X} = c\bar{V}$ and $\bar{Y} = d\bar{V}$. Therefore,

$$a\bar{X} + b\bar{Y} = ac\bar{V} + bd\bar{V} = \ (ac + bd)\,\bar{V} \in \text{Span}\left(\{\bar{V}\}\right),$$

by definition of $\mathbf{V}$ and $\text{Span}\left(\{\bar{V}\}\right)$. ■

In fact, for any set of vectors $\bar{V}_1, \ldots, \bar{V}_m \in \mathbf{V}$, $\text{Span}\big(\{\bar{V}_1, \ldots, \bar{V}_m\}\big)$ is a subspace. Trivially, $\mathbf{V}$ is a subspace with respect to itself.

The ability to obtain vectors by taking linear combinations of others raises the question: "does there exist a set of vectors whose span is the entire vector space, $\mathbf{V}$?". Such a set is said to span the vector space. Trivially, the set of all of the vectors in the vector space spans the vector space. A set of linearly independent vectors that span the entire vector space is called a ***basis*** for the vector space. It may be possible to define more than one basis for a vector space. However, the number of vectors in any of these bases is the same, and defines the dimension of the vector space. A finite dimensional vector space is one whose related bases have a finite number of vectors, e.g. an $n$–dimensional vector space has $n \in \mathbb{N}$ vectors in any related basis. The definition of a basis for a finite dimensional vector space is very useful, not least because it implies *every finite dimensional vector space is equivalent to some* $\mathbb{R}^n$. Consequently, vectors can be depicted as arrows in some $\mathbb{R}^n$ and calculations/computation can be carried out. See Fig.'s A.1, A.2, and A.3. We give some examples of basis:

- the empty set, $\emptyset$, is defined to be the basis for the 0–dimensional vector space $\{\bar{0}\}$;
- any real number is a basis for $\mathbb{R}$;
- the set of $m \times n$ matrices such that each matrix has the value "1" in a unique position and zero everywhere else is a basis for the vector space of $m \times n$ matrices;
- One possible basis for $\mathbf{P}_n$ is a monomial basis. For instance, $\{1, x^1, x^2, x^3\}$ is a basis for $\mathbf{P}_3$.

Given a basis an $n$–dimensional vector space is equivalent to some $\mathbb{R}^n$: vector manipulations in such a vector space are equivalent to manipulations on *n–tuples* in some appropriate $\mathbb{R}^n$. To see this appreciate that:

**Theorem A.1.3.** *Every vector in a finite–dimensional vector space can be written as a unique, linear combination of the basis vectors.*

*Proof.* This follows directly from the definition of linearly independent vectors. Assume that there are 2 linear combinations, $a_1\bar{P}_1 + \ldots + a_n\bar{P}_n$ and $b_1\bar{P}_1 + \ldots + b_n\bar{P}_n$, of the basis vectors

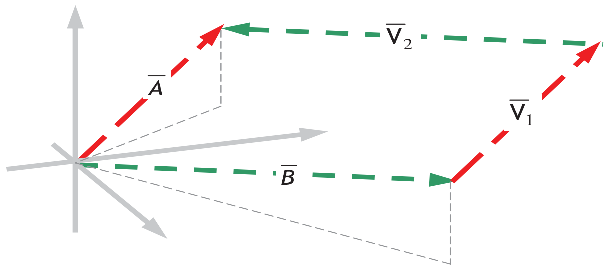


**Figure A.1:** A vector may be depicted as "an arrow of given length pointing in a given direction". Given a vector with tail at the "origin" any other arrow that is parallel to this vector, of the same length and pointing in the same direction, is the same vector. This is very useful because any finite dimensional vector space is equivalent to some $\mathbb{R}^n$ and, as such, may be visualised as a collection of arrows.

$\{\bar{P}_1, \ldots, \bar{P}_n\}$ that result in the same vector. Then,

$$a_1\bar{P}_1 + \ldots + a_n\bar{P}_n = b_1\bar{P}_1 + \ldots + b_n\bar{P}_n \Leftrightarrow (a_1 - b_1)\bar{P}_1 + \ldots + (a_n - b_n)\bar{P}_n = \bar{0}$$
$$\implies a_1 = b_1, \ldots, a_n = b_n. \qquad \blacksquare$$

This means that given a basis, say $\mathcal{S}$, we can associate with each vector, $\bar{V}$, a unique set of coefficients, $a_1, \ldots, a_n \in \mathbb{R}$, such that $\bar{V} = a_1\bar{P}_1 + \ldots + a_n\bar{P}_n$. $a_1, \ldots, a_n$ are the *components* of the vector, $\bar{V}$ with respect to the basis $\mathcal{S}$. Because there is one–to–one correspondence between components of a vector and a vector it is usual to refer to the *n–tuple* $(a_1, \ldots, a_n)$ as the vector $\bar{V}$. It can be shown that all the properties of vectors carry over to manipulations

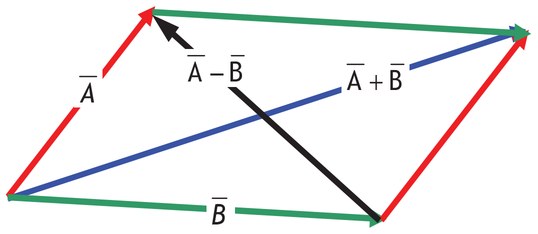


**Figure A.2:** Vector addition can be carried out with visual consequences. The visual effect of the sum of two vectors is to produce the relevant diagonal of the parallelogram produced by the two vectors.

on n–tuples. In particular, vector addition is achieved by adding together corresponding vector components – that is, $(a_1, \ldots, a_n) + (b_1, \ldots, b_n) = (a_1 + b_1, \ldots, a_n + b_n)$ – and scalar multiplication is achieved by multiplying each component by the relevant scalar – that is, $c(a_1, \ldots, a_n) = (ca_1, \ldots, ca_n)$, where $c \in \mathbb{R}$. So, manipulations on components of vectors is actually shorthand for the equivalent manipulations in terms of linear combinations of basis vectors. We can represent a given basis, say $\mathcal{S}$, of an n–dimensional vector space as a set of n–tuples in $\mathbb{R}^n$. Since the basis vectors are linearly independent for $\bar{P}_i \in \mathcal{S}$ we must have $\bar{P}_i = 0 \cdot \bar{P}_1 + \ldots + 0 \cdot \bar{P}_{i-1} + 1 \cdot \bar{P}_i + 0 \cdot \bar{P}_{i+1} + \ldots + 0. \cdot \bar{P}_n$. Therefore, $\bar{P}_i$ has the components $(0, \ldots, 0, \underbrace{1}_{ith}, 0, \ldots, 0)$ with respect to the basis $\mathcal{S}$. This can be done for each basis vector

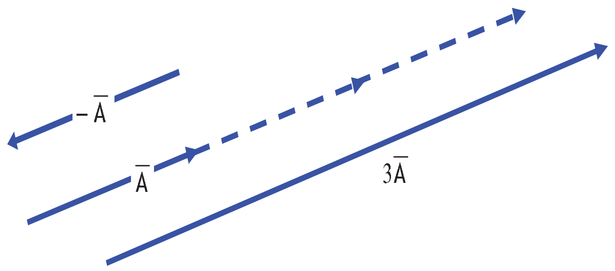


**Figure A.3:** Scalar Multiplication is intimately linked to vector addition. Importantly, scalar multiplication imbues the vector space with the notion of direction. If a vector is depicted as an arrow than multiplying by a scalar produces an arrow, parallel to the original, that is scaled by the relevant amount in the direction indicated by the sign of the multiplying scalar, as shown.

to obtain the $n$ n–tuples that represent the usual basis in $\mathbb{R}^n$. We follow the convention of writing the vectors in $\mathbb{R}^n$ as column vectors. So, in the basis $\mathcal{S}$, the n–tuples associated with the basis vectors $\{\bar{P}_1, \ldots, \bar{P}_n\}$ are:

$$\{ \underbrace{\begin{pmatrix}1\\0\\0\\\vdots\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\\vdots\\0\end{pmatrix}, \ldots, \begin{pmatrix}0\\0\\\vdots\\0\\1\end{pmatrix}}_{n \text{ vectors}} \} \subset \mathbb{R}^n, \tag{A.1}$$

respectively. These n–tuples are important and always form a basis of $\mathbb{R}^n$. They are called the *standard/usual basis* in $\mathbb{R}^n$. To see that these vectors actually form a basis observe that they span $\mathbb{R}^n$, since

$$\text{for} \begin{pmatrix}c_1\\c_2\\c_3\\\vdots\\c_n\end{pmatrix} \in \mathbb{R}^n,\ c_1\begin{pmatrix}1\\0\\0\\\vdots\\0\end{pmatrix} + c_2\begin{pmatrix}0\\1\\0\\\vdots\\0\end{pmatrix} + \ldots + c_{n-1}\begin{pmatrix}0\\0\\\vdots\\1\\0\end{pmatrix} + c_n\begin{pmatrix}0\\0\\\vdots\\0\\1\end{pmatrix} = \begin{pmatrix}c_1+0+0+\ldots+0\\0+c_2+0+\ldots+0\\\vdots\\0+\ldots+0+c_{n-1}+0\\0+\ldots+0+0+c_n\end{pmatrix}$$

$$= \begin{pmatrix}c_1\\c_2\\\vdots\\c_{n-1}\\c_n\end{pmatrix},$$

and they are linearly independent, since

$$c_1\begin{pmatrix}1\\0\\0\\\vdots\\0\end{pmatrix}+c_2\begin{pmatrix}0\\1\\0\\\vdots\\0\end{pmatrix}+\ldots+c_{n-1}\begin{pmatrix}0\\0\\\vdots\\1\\0\end{pmatrix}+c_n\begin{pmatrix}0\\0\\\vdots\\0\\1\end{pmatrix}=\begin{pmatrix}c_1+0+0+\ldots+0\\0+c_2+0+\ldots+0\\\vdots\\0+\ldots+0+c_{n-1}+0\\0+\ldots+0+0+c_n\end{pmatrix}$$

$$=\begin{pmatrix}c_1\\c_2\\\vdots\\c_{n-1}\\c_n\end{pmatrix}=\begin{pmatrix}0\\0\\\vdots\\0\\0\end{pmatrix}$$

implies that $c_1,\ldots,c_n=0$. If we were to draw the usual basis vectors as arrows, then these would be the arrows with "unit length" on each axis in a right–handed coordinate system; such as the usual $x$–$y$–$z$ Cartesian coordinate system (see Fig. A.4).

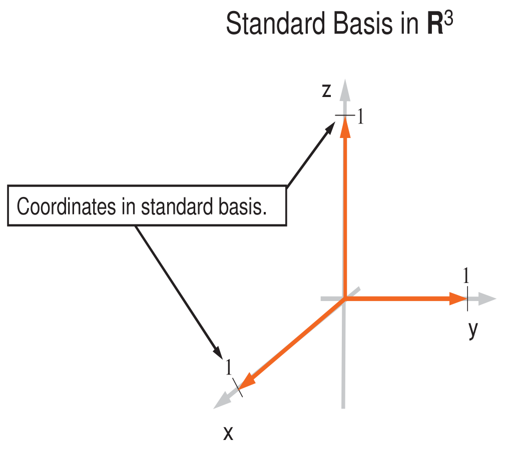


**Figure A.4:** The Standard/Usual Basis in n-dimensional Euclidean space, $\mathbb{R}^n$. Note that for this figure the "unit lengths" are intentionally drawn differently. This is because the basis vectors only define primitive notions of length along each axis. That is, the value of "1" at a point on an axis indicates that only "1" of the basis vector lying parallel to the axis completely defines that point. However, this is a relative notion of distance that can change if a different basis is used: a new basis vector parallel to the former basis vector, and $\frac{1}{2}$ it's length, will result in the point being indicated with the value "2". An inner–product is required to define a more general notion of length and angles.

Here a very important point must be made. By choosing a basis we choose a set of "arrows" with respect to which all other "arrows" are described. Consequently, these basis "arrows" have components equivalent to the usual basis in $\mathbb{R}^n$. However, the basis "arrows" need not have the same "length": these arrows only define primitive "unit lengths". So, in drawing these arrows all that is relevant is that each arrow should lie in the positive direction of a unique axis, with the head of the arrows touching the "1" mark. You could say the axes need not have the same scale. It is more precise to say that at this level of description only a very primitive notion of length exists: only relative length along each axis is defined, relative to some unique basis vector. Defining length that is independent of the basis used requires extra structure to be imposed on the vector space. Indeed, extra structure is also necessary for defining appropriate notions of nearness and angles. To deal with such matters in the following section we define a construct called the *inner–product*.

## A.2 Real Inner–Product

In this section we develop a universal notion of length and angles in a vector space using a largely heuristic approach. Ideally, a universal notion of length should be independent of the basis vectors chosen to perform calculations in. Our aim is to define a function of vectors that results in the lengths of the vectors, irrespective of the basis chosen. One vector whose invariant length we should be able to specify for our function right away is the zero vector, $\bar{0}$, whose length we define as 0. In addition, it is reasonable that such a function will determine a consistent notion of unit length, globally. If such a function were defined then for each vector, $\bar{A}$, there would exist a vector, $\hat{A}$, in the same direction as $\bar{A}$ such that this vector has unit length. We call such a vector a ***unit vector*** and we must have the relationship $\bar{A} = \|\bar{A}\|\,\hat{A}$, for some unique, non-negative scalar $\|\bar{A}\|$ which we define *as the length or magnitude of* $\bar{A}$. From this relationship we infer that *the zero vector is the only vector with zero length*, since if we assume that some vector $\bar{A} \neq \bar{0}$ has zero length then $\bar{A} = \|\bar{A}\|\,\hat{A} = 0\hat{A} = \bar{0}$, which is a contradiction and, therefore, the result follows. We observe that the magnitude of a vector parallel to $\bar{A}$, say $a\bar{A}$ for some scalar $a$, is $a\,\|\bar{A}\|$ since

$$\|a\bar{A}\|\,\hat{A} = a\bar{A} = a\left(\|\bar{A}\|\,\hat{A}\right) = \left(a\,\|\bar{A}\|\right)\hat{A},$$

from which we conclude

$$\|a\bar{A}\| = a\,\|\bar{A}\| \tag{A.2}$$

We expect other useful relationships. For instance, we know that the sum of vectors parallel to a given vector is a vector parallel to the given vector. So, for the sum of a given pair of vectors parallel to $\bar{A}$ – for instance, $x\bar{A}, y\bar{A}$ and the sum $x\bar{A} + y\bar{A}$ for some scalars $x$ and $y$ – we see that

$$\begin{aligned}\left(x\,\|\bar{A}\| + y\,\|\bar{A}\|\right)\hat{A} &= (x+y)\,\|\bar{A}\|\,\hat{A}\\ &= (x+y)\,\bar{A}\\ &= x\bar{A} + y\bar{A}\\ &= \|x\bar{A} + y\bar{A}\|\,\hat{A}\end{aligned}$$

$$= \left\|(x+y)\,\bar{A}\right\|\hat{A}$$

holds. Therefore, since $\hat{A}$ is not the zero vector by definition, we have

$$\left\|(x+y)\,\bar{A}\right\| = x\left\|\bar{A}\right\| + y\left\|\bar{A}\right\|. \tag{A.3}$$

That is, *the magnitude of the sum of a pair of parallel vectors is equal to the sum of the magnitudes of the vectors.* This result may be generalised to the sum of any finite number of parallel vectors via induction. Note that by choosing a vector $\bar{A}$ with respect to which other vectors have been defined as parallel we have been manipulating vectors in $\mathrm{Span}\big(\{\bar{A}\}\big)$, which we have previously shown to be a subspace. Therefore, the results in Eq.'s (A.3), (A.2) are really a statement of *linearity* with respect to this subspace: given a vector $\bar{A}$ our "length function" is a *linear transformation*[6] from $\mathrm{Span}\big(\{\bar{A}\}\big)$ to $\mathbb{R}$. So, given a vector $\bar{A}$, we seek a real–valued linear function, $\left\langle\ ,\hat{A}\right\rangle : \mathrm{Span}\left(\{\bar{A}\}\right) \to \mathbb{R}$, such that:

1. for $\bar{V} \in \mathrm{Span}\left(\{\bar{A}\}\right)$, the magnitude of $\bar{V}$ with respect to the unit vector $\hat{A}$ is $\left\langle\bar{V},\hat{A}\right\rangle$;
2. for an arbitrary pair of vectors parallel to $\bar{A}$ – say $\bar{V},\bar{W} \in \mathrm{Span}\left(\{\bar{A}\}\right)$ – and for some scalars – say $x, y \in \mathbb{R}$ – we have

$$\left\langle x\bar{V}+y\bar{W},\hat{A}\right\rangle = x\left\langle\bar{V},\hat{A}\right\rangle + y\left\langle\bar{W},\hat{A}\right\rangle.$$

Linearity is a useful property for our length function to have if we are to ultimately perform calculations with vectors. Thus far our main requirement has been: given a vector, say $\bar{A}$, we are able to calculate the length of any vector parallel to $\bar{A}$ in a linear fashion, relative to the unit vector $\hat{A}$. What similar requirement will be useful for vectors not parallel to $\bar{A}$? We need a notion of *length in the direction of $\hat{A}$ for an arbitrary vector.* If our function defines such a length in a globally consistent and linear way, then this will be ideal. Fig. A.5 suggests such a notion, inspired by a similar idea used in elementary trigonometry. We postulate that a conceptual "shadow" of $\hat{B}$ can be cast on the line defined by $\hat{A}$. The *signed length* of this "shadow" is the length, up to sign and with respect to $\hat{A}$, of the dotted green vector and we write this as $\left\langle\hat{B},\hat{A}\right\rangle$. One way of viewing this is our function, relative to $\hat{A}$, should return the "amount" of $\hat{B}$ in the direction of $\hat{A}$. That is, $\left\langle\hat{B},\hat{A}\right\rangle$ is the value of our function, relative to $\hat{A}$, on $\hat{B}$. We expect that the closer $\hat{B}$ is to being parallel to $\hat{A}$ the closer this length will be to either the value 1 or -1, the *signed length*s if $\hat{B}$ was parallel to either $\hat{A}$ or $-\hat{A}$ respectively. Put another way we expect that the further $\hat{B}$ is from being parallel to $\hat{A}$ the closer this length is to zero (since the dotted green vector approaches the

[6]Let $\mathbf{V}$ and $\mathbf{W}$ be vector spaces with respect to $\mathbb{R}$. A function $T : \mathbf{V} \to \mathbf{W}$ is a *linear transformation* if for scalars $a, b$ and vectors $\bar{X}, \bar{Y} \in \mathbf{V}$ we have $T\left(a\bar{X}+b\bar{Y}\right) = aT\left(\bar{X}\right) + bT\left(\bar{Y}\right)$. Linear transformations are important, partly because they allow movement from one vector space to another in a manner that is adapted to the underlying vector space structures. It is precisely this property that allows numerical manipulations/calculations that are equivalent to the transformation to be carried out, upon choosing a pair of bases for the respective vector spaces.

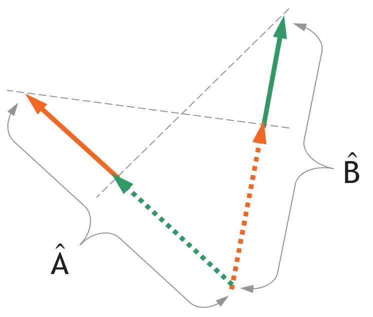


**Figure A.5:** Unit vectors are projected unto each other

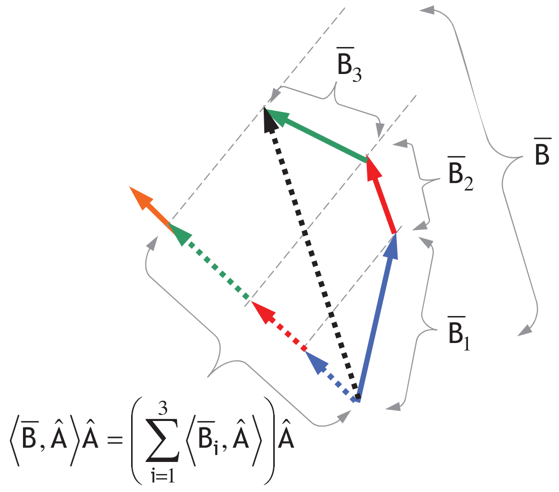
zero vector). So, our function should return *the signed length of the projection of the vector* $\hat{B}$ *onto the subspace defined by* $\hat{A}$, $\mathrm{Span}(\{\bar{A}\})$. In particular, $\|\bar{A}\| = \left\langle \bar{A}, \hat{A} \right\rangle$.

Also, we expect the symmetric relationship

$$\left\langle \hat{B}, \hat{A} \right\rangle = \left\langle \hat{A}, \hat{B} \right\rangle \tag{A.4}$$

to hold: the length of $\hat{B}$ in terms of $\hat{A}$ is equal to the length $\hat{A}$ in terms of $\hat{B}$, since $\hat{A}$ and $\hat{B}$ have equal lengths.

Another requirement for our function is that *the magnitude of a finite sum of vectors should equal the sum of the individual vector magnitudes, relative to* $\hat{A}$. This is suggested by examples such as in Fig. A.6. Three vectors, $\bar{B}_1, \bar{B}_2$ and $\bar{B}_3$ are summed to give the



$\bar{B}_3$
$\bar{B}_2$
$\bar{B}$
$\bar{B}_1$
$\left\langle \bar{B}, \hat{A} \right\rangle \hat{A} = \left( \sum_{i=1}^{3} \left\langle \bar{B}_i, \hat{A} \right\rangle \right) \hat{A}$

**Figure A.6:** The projection of a sum of vectors is shown to be identical to the sum of the projections of the individual vectors; a manifestation of an aspect of linearity.

vector $\bar{B}$. The diagram suggests that the projection of $\bar{B}$ onto $\mathrm{Span}(\{\bar{A}\})$ should equal the sum of the projections of $\bar{B}_1, \bar{B}_2, \bar{B}_3$. That is, we expect $\left\langle \bar{B}, \hat{A} \right\rangle \hat{A} = \sum_{i=1}^{3} \left( \left\langle \bar{B}_i, \hat{A} \right\rangle \hat{A} \right) =$

$\left(\sum_{i=1}^{3}\left\langle \bar{B}_i, \hat{A}\right\rangle\right)\hat{A}$. Thus, since $\hat{A} \neq \bar{0}$, we require

$$\left\langle \sum_{i=1}^{3} \bar{B}_i, \hat{A}\right\rangle = \sum_{i=1}^{3}\left\langle \bar{B}_i, \hat{A}\right\rangle .$$

Yet another useful requirement is that *the length of the projection of a vector should be proportional to the length of the vector*: in a sense the longer a vector is the longer its projection should be. Ideally, for any vector $\bar{B}$ this is the relationship

$$\left\langle b\bar{B}, \hat{A}\right\rangle = b\left\langle \bar{B}, \hat{A}\right\rangle, \tag{A.5}$$

for any scalar $b$. The requirements specified so far imply that our function, with respect to unit vectors, operates on vectors in a linear way. That is, for any unit vector $\hat{A}$,

$$\left\langle\ , \hat{A}\right\rangle : \mathbf{V} \to \mathbb{R}$$

is a linear transformation. An immediate consequence of this is the following interesting relationship. Recall that for non–zero vectors $\bar{A}$ and $\bar{B}$, $\bar{A} = \|\bar{A}\|\,\hat{A}$ and $\bar{B} = \|\bar{B}\|\,\hat{B}$ so that

$$\frac{1}{\|\bar{A}\|}\left\langle \bar{A}, \hat{B}\right\rangle = \left\langle \hat{A}, \hat{B}\right\rangle = \left\langle \hat{B}, \hat{A}\right\rangle = \frac{1}{\|\bar{B}\|}\left\langle \bar{B}, \hat{A}\right\rangle,$$

where we have used both linearity, to move scalars in and out of the first "slot" of our function, and the symmetry of $\left\langle \hat{A}, \hat{B}\right\rangle$. Therefore,

$$\|\bar{B}\|\left\langle \bar{A}, \hat{B}\right\rangle = \|\bar{A}\|\,\|\bar{B}\|\left\langle \hat{A}, \hat{B}\right\rangle = \|\bar{A}\|\left\langle \bar{B}, \hat{A}\right\rangle. \tag{A.6}$$

Not only does this relate projections with respect to $\hat{A}$ and projections with respect to $\hat{B}$, it does so in a symmetric way. This is because the middle expression is symmetric in $\bar{A}$ and $\bar{B}$. To check our intuition we consider the behaviour of this relationship under different conditions?

- If $\bar{A}$ and $\bar{B}$ have the same magnitude, then we expect them to result in projections with equal magnitude. Indeed, this symmetric relationship is predicted by Eq. (A.6) since $\|\bar{A}\| = \|\bar{B}\|$ implies $\left\langle \bar{A}, \hat{B}\right\rangle = \left\langle \bar{B}, \hat{A}\right\rangle$. In particular, if $\|\bar{A}\| = \|\bar{B}\| = 1$ then we obtain the symmetry requirement of Eq. (A.4).
- If $\bar{B}$ is a unit vector, i.e. $\|\bar{B}\| = 1$ or $\bar{B} = \hat{B}$, then we expect that the projection of $\bar{A}$ on $\bar{B}$'s subspace should have a signed magnitude that is proportional to the magnitude of $\bar{A}$. In fact Eq. (A.6) tells us that the magnitude of the projection is $\left\langle \bar{A}, \hat{B}\right\rangle = \|\bar{A}\|\left\langle \hat{A}, \hat{B}\right\rangle$; the signed length of the projection of $\hat{A}$ scaled by length of $\bar{A}$.
- If $\bar{A} = \bar{B}$, then the projections should be identical; the length of the vector. Eq. (A.6) confirms this since it implies $\|\bar{A}\| = \|\bar{B}\|$. Note that an arbitrary non–zero vector is parallel to itself in the same direction. Therefore, its projection into its subspace should

be itself, with positive length. This is manifest from Eq. (A.6) since $\|\bar{A}\| = \left\langle \bar{A}, \hat{A} \right\rangle$.

- When is the magnitude of a projection equal to zero? If and only if either the vector being projected is the zero vector, or the subspace being projected to is Span($\{\bar{0}\}$), or the vectors are as far from being parallel as possible. Each of these cases holds in Eq. (A.6). Recall that we proved only the zero vector has magnitude 0 and this is reaffirmed here. For, an arbitrary vector either is parallel to itself, or is the zero vector. Consequently, the only conditions that result in its length, $\left\langle \bar{A}, \hat{A} \right\rangle$, being zero are precisely those for which $\bar{A} = \bar{0}$.

This relationship captures, as special cases, all of the properties we have specified so far for our function. To emphasize this we may define the relationship as

$$\left\langle \bar{A}, \bar{B} \right\rangle = \|\bar{A}\| \, \|\bar{B}\| \left\langle \hat{A}, \hat{B} \right\rangle, \tag{A.7}$$

which is *symmetric in arbitrary non–zero vectors* $\bar{A}$ and $\bar{B}$. The symmetry has yet another important consequence. Because our function is required to be linear in the first "slot" we can show that our function is also linear in the second slot as well. For,

$$\left\langle \bar{A}, x\bar{X} + y\bar{Y} \right\rangle = \left\langle x\bar{X} + y\bar{Y}, \bar{A} \right\rangle = x \left\langle \bar{X}, \bar{A} \right\rangle + y \left\langle \bar{Y}, \bar{A} \right\rangle = x \left\langle \bar{A}, \bar{X} \right\rangle + y \left\langle \bar{A}, \bar{Y} \right\rangle,$$

where $x, y \in \mathbb{R}$ and $\bar{X}, \bar{Y} \in \mathbf{V}$. This shows that our function is necessarily *Bilinear.*

In summary, we have specified necessary properties for a function that should define for us a consistent, global notion of length. Our function should be a real–valued function of pairs of vectors: $\langle \ , \ \rangle \colon \mathbf{V} \times \mathbf{V} \to \mathbb{R}$. Furthermore, for $\bar{A}_1$, $\bar{A}_2$, $\bar{A}$, $\bar{B} \in \mathbf{V}$ and $a$,$b \in \mathbb{R}$, the following properties should hold.

$$\begin{array}{rc}
\text{Linearity:} & \left\langle a\bar{A}_1 + b\bar{A}_2, \bar{B} \right\rangle = a \left\langle \bar{A}_1, \bar{B} \right\rangle + b \left\langle \bar{A}_2, \bar{B} \right\rangle \\
\text{Symmetry:} & \left\langle \bar{A}, \bar{B} \right\rangle = \left\langle \bar{B}, \bar{A} \right\rangle \\
\text{Positive definiteness:} & \left\langle \bar{A}, \bar{A} \right\rangle = \left\|\bar{A}\right\|^2 \geq 0, \text{ for all } \bar{A} \text{ and} \\
 & \left\langle \bar{A}, \bar{A} \right\rangle = \left\|\bar{A}\right\|^2 = 0 \text{ if, and only if, } \bar{A} = 0
\end{array}$$

These properties are necessary and sufficient for our purposes. They define a function, the so–called ***real inner–product***, that imbues the vector space with notions of angles[7] and length. A vector space over $\mathbb{R}$ together with an associated real inner–product is called a *real inner–product space*. Although we motivated the definition by appealing to trigonometric relationships in a 2–dimensional plane the definition is sufficiently robust and applicable in infinite–dimensional vector spaces. However, we will be concerned with finite–dimensional vector spaces.

A consequence of defining an inner–product for a vector space is that a notion of mathematical *continuity* is also defined[8]. Intuitively, this makes sense since continuity requires a

[7]Although we have implicitly utilised notions of angles in discussing parallel and non-parallel vectors we shall make this explicit later on when discussing *orthogonality* and *collinearity*.

[8]Formally, a real inner–product space is necessarily a *normed linear space* which, in turn, defines a *topological space* which, in turn, defines some notion of continuity. Alternatively, a real inner–product space

concept of "arbitrary closeness of entities" and inner–products define distances. Continuity is a necessary requirement for the operations of differentiation, and Riemann integration on closed, bounded intervals. So it comes as no surprise that inner–product spaces are very important as the foundation of many models and methods in Physics, Engineering and Mathematics using multidimensional calculus. However, one of the achievements of the present work is that for the extremisation problems we do not use calculus. This is pertinent for the *quadratic optimisation* problems since the nature of such problems can make the search for turning points, and thus the use of calculus, an irrelevant exercise. Therefore, the solution of such problems using only the inner–product space structure is a clear demonstration of the power afforded by this approach.

In the motivation for the definition of the inner–product it was suggested that its properties are carefully chosen to facilitate calculation. Also, we stated that an important reason for a choice of a basis in a vector space is to allow calculations. We demonstrate the "coming together" of these two observations. Let $\mathcal{S} = \left\{\bar{P}_1, \bar{P}_2, \ldots, \bar{P}_n\right\}$ form a basis for $\mathbf{V}$. Then, for arbitrary vectors $\theta_A$, $\theta_B$ we may write these as unique linear combinations of the basis vectors; that is, $\theta_A = \sum_{i=1}^{n} \theta_A i \bar{P}_i$ and $\theta_B = \sum_{j=1}^{n} \theta_B j \bar{P}_j$. The inner–product of the vectors is

$$\langle \theta_A, \theta_B \rangle = \left\langle \sum_{i=1}^{n} \theta_A i \bar{P}_i, \sum_{j=1}^{n} \theta_B j \bar{P}_j \right\rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \left\langle \theta_A i \bar{P}_i, \theta_B j \bar{P}_j \right\rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \theta_A j \theta_B i \left\langle \bar{P}_i, \bar{P}_j \right\rangle .$$

As a consequence of linearity the apparently abstract entity, $\langle \theta_A, \theta_B \rangle$, is evaluated as a linear combination, $\sum_{i=1}^{n} \sum_{j=1}^{n} \theta_A j \theta_B i \left\langle \bar{P}_i, \bar{P}_j \right\rangle$, of the products of the vector components defined by the choice of basis, $\mathcal{S}$. The sum can be processed further. Recall that in defining the inner–product we postulated the existence of unit vectors in every possible direction. So, each vector $\bar{P}_i$ has an associated unit vector $\hat{P}_i$ such that $\bar{P}_i = \left\langle \bar{P}_i, \hat{P}_i \right\rangle \hat{P}_i = \left\|\bar{P}_i\right\| \hat{P}_i$. Therefore, using Eq. (A.7), this allows us to write

$$\begin{aligned} \langle \theta_A, \theta_B \rangle \;&=\; \sum_{i=1}^{n} \sum_{j=1}^{n} \theta_A j \theta_B i \left\langle \bar{P}_i, \bar{P}_j \right\rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \theta_A j \theta_B i \left\|\bar{P}_i\right\| \left\|\bar{P}_j\right\| \left\langle \hat{P}_i, \hat{P}_j \right\rangle \\ &= \sum_{i=1}^{n} \theta_A i \theta_B i \left\|\bar{P}_i\right\|^2 + 2 \sum_{i<j=1}^{n} \theta_A j \theta_B i \left\|\bar{P}_i\right\| \left\|\bar{P}_j\right\| \left\langle \hat{P}_i, \hat{P}_j \right\rangle . \end{aligned} \tag{A.8}$$

The set $\hat{\mathcal{S}} = \left\{\hat{P}_1, \hat{P}_2, \ldots, \hat{P}_n\right\}$ forms a basis for $\mathbf{V}$, since by definition each unit vector is proportional or "linearly related" to a unique basis vector in $\mathcal{S}$ and, consequently, they inherit linear independence and spanning properties of $\mathcal{S}$. If we use $\hat{\mathcal{S}}$ as the basis in Eq. (A.8) we obtain

$$\langle \theta_A, \theta_B \rangle \;=\; \sum_{i=1}^{n} \theta_A i \theta_B i + 2 \sum_{i<j=1}^{n} \theta_A j \theta_B i \left\langle \hat{P}_i, \hat{P}_j \right\rangle . \tag{A.9}$$

defines a *metric space* which, in turn, defines a topological space which, in turn, defines some notion of continuity.

The observation here is that the use of unit, basis vectors simplifies the form of the expression for evaluating the inner–product of a pair of vectors. We shall return to this form to simplify it further, after we have discussed the notion of angles, *collinearity*, and *orthogonality* in inner–product spaces.

## A.3 Orthogonality and Collinearity

How does one define the angle between a pair of non–zero vectors in a vector space? On the one hand there is no reason to expect that we can orient possibly infinite dimensional vectors to obtain an intended angular separation between them. On the other hand vector spaces by definition carry with them a natural notion of vectors being parallel or collinear: any pair of vectors such that one of the vectors is a scaling of the other can be considered to be collinear. In addition, at the heart of the definition for the inner–product presented in this work lies the notion of projections of one vector onto the subspace spanned by another vector. When defining this what we are actually doing is defining a mapping between a vector space – a construct which can be difficult to visualise in higher dimensions – and trigonometry which we are very comfortable with visually. An example of this sort of relationship is shown in Fig (A.7). These diagrams are drawn to illustrate how the angle between the non–zero vectors

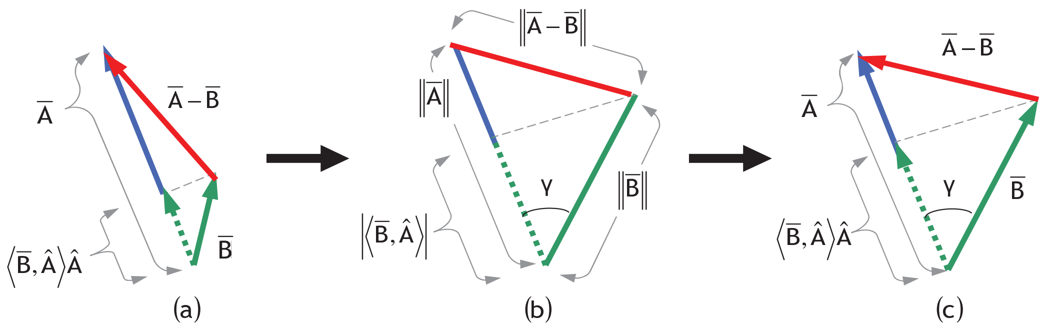


**Figure A.7:** The inner-product is used to define the angle between two non–zero vectors by creating a 2–dimensional triangle whose sides are the lengths of the relevant vectors. In figure (a) only arrows are used to depict multidimensional vectors in some abstract vector space. In figure (b) a 2–dimensional triangle with sides equal to the lengths of the vectors drawn in figure (a). The angle between the vectors $\bar{A}$ and $\bar{B}$ is the one defined by the angle between the corresponding sides of the triangle in (b); the angle is obtained by employing the *cosine rule*. Upon having this understanding of the relationship between figures (a) and (b) it is not necessary to depict them as separate; instead, a single picture, such as (c), in which arrows with the corresponding vector lengths separated by the correct angles may be drawn.

$\bar{A}$ and $\bar{B}$ are defined. Diagram A.7a should be viewed as a very primitive representation of the vectors in some orientation. Note that although the arrows define a triangle the lengths of the arrow have been deliberately drawn to be different from the corresponding triangle in A.7b to emphasize the point that the triangle in A.7a contains only very primitive orientation information: the addition of $\bar{A}$ and $\left(-\bar{B}\right)$ defines $\bar{A}-\bar{B}$. In contrast, the triangle in A.7b is constructed with sides whose lengths are defined by the inner–product. Triangles are simple to both visualize and manipulate. In particular, the angle between the sides corresponding to the vectors $\bar{A}$, $\bar{B}$ is determined using the *cosine rule*,

$$\cos\gamma = \frac{\left(\|\bar{A}\|^2 + \|\bar{B}\|^2\right) - \|\bar{A} - \bar{B}\|^2}{2\,\|\bar{A}\|\,\|\bar{B}\|} \tag{A.10}$$

The angle between the vectors $\bar{A}$ and $\bar{B}$ is defined as the value of $\gamma$. Once this relationship is determined the ideas in A.7a and A.7b can be coalesced into a single figure, A.7c. Equation (A.10) states a relationship that is actually much simpler than it looks. Upon using the properties of the inner–product we have

$$\begin{aligned}\cos\gamma &= \frac{\left(\langle \bar{A}, \bar{A}\rangle + \langle \bar{B}, \bar{B}\rangle\right) - \langle \bar{A} - \bar{B}, \bar{A} - \bar{B}\rangle}{2\left\langle \bar{A}, \hat{A}\right\rangle \left\langle \bar{B}, \hat{B}\right\rangle} \\ &= \frac{\left(\langle \bar{A}, \bar{A}\rangle + \langle \bar{B}, \bar{B}\rangle\right) - \left(\left(\langle \bar{A}, \bar{A}\rangle + \langle \bar{B}, \bar{B}\rangle\right) + 2\langle \bar{A}, \bar{B}\rangle\right.}{2\left\langle \bar{A}, \hat{A}\right\rangle \left\langle \bar{B}, \hat{B}\right\rangle} \\ &= \frac{\langle \bar{A}, \bar{B}\rangle}{\left\langle \bar{A}, \hat{A}\right\rangle \left\langle \bar{B}, \hat{B}\right\rangle}.\end{aligned} \tag{A.11}$$

Equation (A.11) is commonly used in the literature as the definition of the angle between the vectors $\bar{A}$ and $\bar{B}$. This is because the cosine is completely determined by the relevant values of the inner–product. This is no accident and should come as no surprise since it is precisely this sort of reasoning that was used to motivate the definition of the inner–product in the first place.

A comparison of Eq.'s (A.11) and (A.7) illuminates an alternative view of "atomic" projections; the projection of one unit vector onto the subspace spanned by another unit vector. It is the case that

$$\cos\gamma = \left\langle \hat{A}, \hat{B}\right\rangle. \tag{A.12}$$

That is, *the cosine of the angle between two vectors is the inner–product of their respective unit vectors.* Another way of saying this is that the cosine of the angle between the non–zero vectors $\bar{A}$ and $\bar{B}$ is the signed length of the projection of $\hat{A}$ on $\mathrm{Span}\left(\left\{\hat{B}\right\}\right)$. Manifestly, we see why the inner–product is primarily used to return signed lengths of projections. It is because the atomic projections – that is, the inner–product of unit vectors – are really cosines. That is why the requirement,

$$\left|\left\langle \hat{A}, \hat{B}\right\rangle\right| \leq 1, \tag{A.13}$$

we stated in the motivation for the inner–product is an important one. This is the ***Cauchy–Schwarz inequality***. An equivalent way of seeing that this requirement is encapsulated in the definition of the inner–product is via the following proof.

*Proof.* Consider an arbitrary pair of non–zero vectors, $\bar{A}$ and $\bar{B}$. For any scalar $x$ and from the properties of the inner–product we must have

$$x^2 \left\langle \bar{B}, \bar{B} \right\rangle - 2x \left\langle \bar{A}, \bar{B} \right\rangle + \left\langle \bar{A}, \bar{A} \right\rangle = \left\langle \bar{A} - x\bar{B}, \bar{A} - x\bar{B} \right\rangle = \left\| \bar{A} - x\bar{B} \right\|^2 \geq 0.$$

If the quadratic expression above is viewed as a graph, then the inequality is equivalently depicted as the graph not lying below the "$x$–axis". This would be the case if and only if the related quadratic equation has only one or no real roots. That is, the quadratic's associated discriminant is non-positive. So, we require

$$4\left\langle \bar{A}, \bar{B} \right\rangle^2 - 4 \left\langle \bar{B}, \bar{B} \right\rangle \left\langle \bar{A}, \bar{A} \right\rangle \leq 0$$

and the result follows. ■

This inequality points out yet another important property of projections. By definition, the "further" a vector is from a 1–dimensional subspace the smaller the magnitude of its projection onto the subspace – for instance, $\left| \left\langle \bar{B}, \hat{A} \right\rangle \right| \leq \left\langle \bar{B}, \hat{B} \right\rangle$. We shall show later on that this relationship remains true for projections of vectors onto higher dimensional subspaces.

Now that we have a definition of angles, we may consider different orientations between pairs of vectors. For the vectors $\bar{A}$ and $\bar{B}$, we use the form $\left\langle \bar{A}, \bar{B} \right\rangle = \left\| \bar{A} \right\| \left\| \bar{B} \right\| \cos\gamma$ of the inner–product to facilitate the discussion.

1. First, we consider which orientations are relevant when the inner–product is 0. We pointed out previously that as a consequence of the inner–product definition there are three, possibly not exclusive, possibilities: $\left\| \bar{A} \right\| = 0$, $\left\| \bar{B} \right\| = 0$ or $\cos\gamma = 0$. The first two possibilities imply $\bar{A} = \bar{0}$ and $\bar{B} = \bar{0}$ respectively. The third implies that the angle between the vectors is $90°$; the vectors are ***perpendicular***. These are the three conditions under which the pair of vectors are said to be ***orthogonal***. From the foregoing it is clear that the zero vector is the only vector that is orthogonal to every other vector and itself.
2. Next, we consider what happens when a pair of non–zero vectors are parallel or ***collinear***. In this case $\gamma = 0$ and, consequently $\left\langle \bar{A}, \bar{B} \right\rangle = \left\| \bar{A} \right\| \left\| \bar{B} \right\|$; the Cauchy–Schwarz inequality becomes an equality. While this shows that collinearity is sufficient for the Cauchy–Schwarz inequality becoming an equality it is also necessary. This can be shown either by appealing to the "single–root" condition in the proof of the Cauchy–Schwarz inequality given above, or observing that if $\left\langle \hat{B}, \hat{A} \right\rangle = 1$ then $\left\langle \bar{B}, \hat{A} \right\rangle = \left\| \bar{B} \right\| \left\langle \hat{B}, \hat{A} \right\rangle = \left\| \bar{B} \right\|$.

Armed with the notion of orthogonality, we revisit the expression for the evaluation of the inner–product of two vectors given in Eq. (A.9). We had obtained this form by using unit, basis vectors. In addition to this we now require that the basis vectors be mutually orthogonal. This means that $\left\langle \hat{P}_i, \hat{P}_j \right\rangle = 0$ for $i \neq j$. Consequently, we obtain further simplification to the canonical form for the inner–product,

$$\left\langle \theta_A, \theta_B \right\rangle = \sum_{i=1}^{n} \theta_A i \theta_B i. \tag{A.14}$$

Thus, we have demonstrated that *the computational form of the inner–product of two vectors* *can be written as the sum of the products of the vectors' corresponding components.* This is possible because we chose to evaluate the inner–product using a basis of unit vectors that are orthogonal. Such a basis is called an ***orthonormal basis***. If an orthonormal basis is used as the usual basis in $\mathbb{R}^n$, then Eq. (A.14) is the applicable form of the inner–product and is referred to as the *usual inner–product* in n–dimensional Euclidean space. The usual inner–product is the easiest form of the inner–product to manipulate in calculation since it only involves products of corresponding vector components. A subtle point to make here is that by choosing an appropriate basis – that is, a basis adapted to the inner–product – *all of the information that characterizes geometric relationships between vectors is contained in the vectors' components.* Another way to put this is lengths relative to an orthonormal basis are manifestly equivalent to lengths defined by the inner–product.

# Appendix B

# Negative Failure Correlation in Diversity Experiments

Several experiments to study the impact of (design and process) diversity on system reliability have been carried out. Some examples include the UCLA/Honeywell *six language* project [56], the UCI/UVA experiment conducted by Knight and Leveson[1] [10], the *four universities* experiment sponsored by NASA[2] [11], and the university of Iowa/Rockwell NVS project[3]. More recently there have been further attempts to study diversity but not under controlled conditions. Thousands of programs submitted to online programming competitions have been analysed and the potential for diversity to result in significant reliability improvements has been explored [57]. Also, database diversity and its potential for non–crash failure detection has also been extensively studied [58, 59]. In this appendix we recount one of the more popular diversity experiments, the Knight and Leveson experiment, and clarify the relationship between the result reported in [10] and the LM result about correlated coincident failure, on average, resulting from independently developed versions. We also note that in this experiment there are no version pairs that fail together on some demands and, despite this, exhibit negative failure correlation. In this regard we give a sufficient condition which might explain why this is the case. This sufficient condition may also explain a similar lack of coincidentally failing, negatively correlated pairs in another diversity experiment ([57]).

The Knight and Leveson experiment was used to test the hypothesis that the independent development of multiple software versions, developed to the same requirements specification, results in the independent failure of the versions that are developed. Twenty seven software

[1]This involved advanced undergraduate and graduate students from two universities developing programs to solve the "*Launch interceptor*" problem. The problem is to determine whether data from radar reflections off a flying object indicate a hostile flying object, to which an a signal for an interceptor must be given.

[2]This study involved forty programmers from four universities in teams of two (one programmer, one mathematician), producing twenty versions of software to be used in the sensor management of a redundant strapped down inertial unit. The acceptance procedure consisted of sixty test cases, fifty of which were functional tests and the remaining randomly generated. Specification clarifications were given to the teams as question/answer pairs.

[3]forty students (thirty–three Electrical and Computer Engineering (ECE) and Computer Science (CS) students from University of Iowa and seven from Rockwell international) formed fifteen programming teams to independently design, code and test computerized airplane landing systems.

versions were created by advanced undergraduate and graduate students in computer science from the University of Virginia and University of California. Attempts were made to ensure the independent creation of the versions. Each version was ultimately subjected to a million inputs/demands generated according to an operational profile that had been validated by domain experts. The failure behaviour of each program on these inputs was determined. From this, estimates can be obtained for both the *pfd*s of the programs and the probability that at least two of the versions fail on a randomly submitted input. Given twenty seven *pfd* estimates, obtained from observing the failure behaviour of the versions on the test inputs, the programs were assumed to fail independently on a random test input. In essence, if we label the *pfd*s as $pfd_1, \ldots, pfd_{27}$, this independence assumption defines a collection of twenty seven independent Bernoulli trials (the $i$th trial has parameter $pfd_i$), resulting in a probability distribution with the property

$$1 = \prod_{i=1}^{27} (pfd_i + psd_i) = \prod_{i=1}^{27} pfd_i + \frac{1}{26!} \sum_{\sigma \in S_{27}} pfd_{\sigma(1)} psd_{\sigma(2)} \ldots psd_{\sigma(27)} + \ldots + \prod_{i=1}^{27} psd_i \ ,$$

where $S_{27}$ is the set of all permutations of the indices $\{1, \ldots, 27\}$ and $pfd_i = 1 - psd_i$ , for all $i = 1, \ldots, 27$. Consequently, the probability of at least two of the versions failing on an arbitrary input is given by

$$P_0 := P \begin{pmatrix} \text{At least two programs} \\ \text{fail simultaneously} \\ \text{on a given test input} \end{pmatrix} = 1 - \prod_{i=1}^{27} psd_i - \frac{1}{26!} \sum_{\sigma \in S_{27}} pfd_{\sigma(1)} psd_{\sigma(2)} \ldots psd_{\sigma(27)}.$$

"Arbitrary input", because we assume that this probability is the same irrespective of what input is under consideration. Therefore, we may define a Bernoulli distributed random variable, $F \sim \text{Ber}(P_0)$; this is the experiment "an arbitrary input is submitted to all of the versions, and the failure behaviour of the versions is observed". In order to statistically test whether this model is accurate one needs to compare the estimates of probabilities of coincident failure (obtained from the observed experiment sample) with a so-called *region of acceptance* which describes the set of outcomes that are "most likely", assuming the model $F \sim \text{Ber}(P_0)$ is adequate. "Most likely" is defined by a chosen confidence level for the test (99% was used). Of the one million test inputs, 1255 were observed to have caused at least two of the versions to fail. This was compared with the endpoints of a 99% confidence interval of a distribution that describes a sequence of $n$ independent samplings from the distribution $F \sim \text{Ber}(P_0)$. Such a sequence of $n$ ( = 1 million) independent Bernoulli trials – where each trial is the submission of an input to all of the versions, and at least two versions fail on the input with probability $P_0$ – defines a binomial distribution, $\text{Bin}(n, P_0)$. The probability that in $n$ such trials there were $s$ instances where at least two of the versions failed is given as

$$P \begin{pmatrix} s \text{ instances out of } n \text{ where} \\ \text{at least two of the versions failed} \end{pmatrix} = \binom{n}{s} {P_0}^s \left(1 - P_0\right)^{n-s} \simeq P\left(K = s\right) \ ,$$

where $K \sim \mathrm{N}\big(nP_0, nP_0\,(1-P_0)\big)$ is the random variable distributed according to the Normal approximation to the Binomial distribution $\mathrm{B}(n, P_0)$. This Normal approximation is justified because the number of test inputs, $n$, is large at one million. The model, $F$, was rejected at 99% confidence level and thus an assumption of independent failure of the versions, resulting from the independent development of the versions, was refuted. We now know via the EL/LM models that a more appropriate model would have been the following. Given the input $x$ and the twenty seven versions $\pi_1, \ldots, \pi_{27}$, the indicator function for the event of interest is

$$1 - \prod_{i<j} \big(1 - \omega(\pi_i, x)\omega(\pi_j, x)\big) = \begin{cases} 1, & \text{if some pair of the 27 versions fails on } x \\ 0, & \text{otherwise} \end{cases}$$

where $i, j = 1, \ldots, 27$. Consequently, given the model $(\mathcal{X}, \Sigma_{\mathcal{X}}, \mathrm{P}_X(\cdot))$ for the occurrence of demands the probability that at least two of the versions fail on a randomly occurring demand is

$$P\begin{pmatrix} \text{At least two programs} \\ \text{fail simultaneously} \\ \text{on a random test input} \end{pmatrix} = \sum_{x \in \mathcal{X}} \left(1 - \prod_{i<j}\big(1 - \omega(\pi_i, x)\omega(\pi_j, x)\big)\right) \mathrm{P}_X(x).$$

If only a given pair of versions, $\pi_1$ and $\pi_2$, was under scrutiny then this simplifies to

$$P\,(\pi_1 \text{ and } \pi_2 \text{ fail a random demand}) = \sum_{x \in \mathcal{X}} \big(\omega(\pi_1, x)\omega(\pi_2, x)\big)\,\mathrm{P}_X(x)\ . \tag{B.1}$$

which is the system *pfd* for a given pair of versions in a 1–out–of–2 configuration.

So, the experiment results suggest that despite attempts to independently develop multiple software versions (and in so doing make the insertion of faults into the versions by the development teams independent) positive failure correlation between the developed versions can still occur. Essentially, for the versions that are actually developed, the independent insertion of faults during the creation of these versions does not preclude the overlapping of failure regions associated with inserted faults. It is such overlaps that result in coincident failure between the developed versions. This result is related to, but different from the EL/LM model result "independently developed versions cannot be *expected* to fail independently" (see Section 2.5). The difference is that the EL/LM result is a statement about average system pfd, where the average is calculated over all possible pairs of independently developed versions. On the other hand the Knight and Leveson result is a statement about the system pfd for a system made out of a given set of versions, and the uncertainty from the versions' development processes is not directly taken into account[4]. This difference is illustrated by Eq.'s (2.2) and (2.15) in Chapter 2, which we reproduce here as Eq.'s (B.2) and (B.3) respectively. It is the difference between the probability that a given pair of ver-

[4]In order to account for this uncertainty a large sample of programs would need to be developed under "identical" development process conditions. The prohibitively large cost associated with this can make this infeasible, as was pointed out in the NASA funded software diversity experiment [11] involving four universities. It is the case, however, that there are repositories such as those studied in [57] that contain thousands of programs written to the same relatively "simple" requirements specifications. In such cases a feel for version sampling distributions might be obtained.

sions, $\pi_1$ and $\pi_2$, fail on a randomly chosen demand, computationally given by expanding Eq. (B.1) as

$$\begin{aligned} P\left(\begin{array}{c}\pi_1 \text{ and } \pi_2 \text{ fail}\\ \text{on a random demand } X\end{array}\right) &= \mathop{\mathbb{E}}_{X}\left(\omega\left(\pi_1, X\right)\omega\left(\pi_2, X\right)\right)\\ &= pfd_1 pfd_2 + \mathop{\mathrm{Cov}}_{X}\left(\omega\left(\pi_1, X\right), \omega\left(\pi_2, X\right)\right) \end{aligned} \tag{B.2}$$

and the probability that a randomly chosen pair of versions fail on a randomly chosen demand, given as

$$\begin{aligned} P\left(\begin{array}{c}\Pi_A \text{ and } \Pi_B \text{ fail}\\ \text{on a random demand } X\end{array}\right) &= \mathop{\mathbb{E}}_{X,\Pi_A,\Pi_B}\left(\omega\left(\Pi_A, X\right)\omega\left(\Pi_B, X\right)\right)\\ &= \mathop{\mathbb{E}}_{X}\left(\theta_A\left(X\right)\right)\mathop{\mathbb{E}}_{X}\left(\theta_B\left(X\right)\right) + \mathop{\mathrm{Cov}}_{X}\left(\theta_A\left(X\right), \theta_B\left(X\right)\right) . \end{aligned} \tag{B.3}$$

In both of these equations the covariance terms suggest deviations from failure independence. These results are clearly related, but different results. If independence holds in Eq. (B.2) for every possible version pair[5] then independence is guaranteed in Eq. (B.3). This is an argument of sufficiency but not of necessity, for the equations suggest that it is possible not all pairs of versions exhibit failure independence and yet the channels may yet still fail independently, on average. Admittedly, how to recognise such a situation in practice is a non-trivial problem.

It turns out that, apart from pairs of versions that exhibited no coincident failure (in this sense these versions are considered to be *orthogonal*), no negative failure correlation was exhibited by pairs of versions sampled from the twenty seven versions produced in the experiment [60]. A similar situation was observed for the expected system pfds reported in an analysis of thousands of programs submitted to an online programming competitions website [57]. There, thousands of programs were submitted to solve three programming challenges (referred to as the "$3n+1$", "*factovisors*" and "*prime–time*" problems in the study)[6]. The development of a 1–out–of–2 system with improving channel development processes was simulated by removing the most unreliable versions from pools of versions that could be created for each channel. The channels' development processes were diverse in that different programming languages were used for their creation (C,C++ and Pascal). After each improvement to the development processes, randomly chosen version pairs were combined and expected system pfd calculated. This was compared with the expected channel pfd to

[5] Admittedly, this would be a rather strange set of possible programs.

[6] The three programming challenges were:

- **3n+1**: A number sequence is built as follows. Start with a given number; if it is odd, multiply by 3 and add 1; if it is even, divide by 2. The sequence length is the number of these steps to arrive at a result of 1. Determine the maximum sequence length for the numbers between two given integers $0 < i, j \leq 100,000$;
- ***factovisors***: For two given integers $0 \leq i, j \leq 2^{31}$, determine whether $j$ divides factorial $i$.
- ***prime–time***: Euler discovered that the formula $n^2 + n + 41$ produces a prime for $0 \leq n \leq 40$. However, it does not always produce a prime. Write a program that calculates the percentage of primes the formula generates for $n$ between two integers $i$ and $j$ with $0 \leq i \leq j \leq 10,000$.

observe the order of reliability improvement due to diversity. It was the case that to begin with the expected system pfd was close to, but worse than it would be under independent failure of the versions. Upon improving the development processes the positive failure correlation between the versions becomes significant, resulting in reliability improvements that are not as large as they would be under independent channel failures. This was true for all 3 programming challenges. However, the results from the "*factovisors*" challenge differ from the "$3n + 1$" challenge when the processes in each case produced very reliable programs. For the "$3n + 1$" challenge positive failure correlation still significantly undermined the benefits from diversity, while in the "*factovisors*" challenge the positive failure correlation was greatly reduced with improving development processes, resulting in significant reliability gains due to programming language diversity. These experiments suggest that particular pairs of versions, and version pairs on average, benefit from diversity when most of the version pairs are essentially orthogonal and, otherwise, positive failure correlation undermines the benefits from diversity.

Why would this be the case? Certainly, the EL and LM models tell us that if the development teams are "similar" in which tasks they find difficult to develop software for then the resulting set of versions can be expected to be positively correlated in their behaviour. This would explain the results of the programming challenge (differences in programming language does not induce sufficiently diverse development processes for negative failure correlation to be observed). This, however is a statement on average which does not preclude the possibility of producing particular version pairs that are negatively correlated: Eq. (B.2) clearly suggests the possibility of zero or negative failure correlation. Yet, in the experiment none of the pairs of coincidentally failing versions sampled from the twenty seven versions exhibit negative failure correlation, given that these versions are imperfect and sometimes fail together on the same demand. On the other hand note that if positively correlated version pairs are sufficiently likely then positive failure correlation for average pairs of versions unsurprisingly occurs (as an extreme example if positive covariance in Eq. (B.2) occurs for all version pairs this implies positive covariance in Eq. (B.3)). To shed some light on this it is instructive to deconstruct the covariance term in Eq. (B.2). Using the geometric framework developed in Chapter 4 the version $\pi_1$ can be modelled by a vector $V_1$, and similarly some vector $V_2$ models $\pi_2$. If $V_1$ is not the perfect version $\bar{0}$ (that is, the version that succeeds on all demands) then it will be a sum of orthogonal, imperfect single versions according to Theorem 4.7.4. We may choose these imperfect single versions such that each fails on a unique, single demand. $V_2$ may be similarly decomposed. In fact, we may write

$$V_1 = P_1 + P \text{ and } V_2 = P_2 + P,$$

where $P_1$ is a sum of imperfect single versions (and is thus an imperfect single version) that fail on those demands that $V_1$ fails on but $V_2$ does not. $P_1$ is the "exclusive failure" part of $V_1$. Similarly, $P_2$ models the "exclusive failure" part of $V_2$. In contrast to $P_1$ and $P_2$, $P$ is the sum of imperfect single versions that fail only on the demands that both $V_1$ and $V_2$ fail on. Note that $\{P_1, P_2, P\}$ is an orthogonal set of vectors. This allows us to compute the following probabilities (recall that the demand profile is modelled by the vector $\bar{P}$):

$$P\begin{pmatrix}\pi_1 \text{ fails and } \pi_2 \text{ succeeds}\\ \text{on a random demand } X\end{pmatrix} = \langle P_1, \bar{P}\rangle, \quad P\begin{pmatrix}\pi_2 \text{ fails and } \pi_1 \text{ succeeds}\\ \text{on a random demand } X\end{pmatrix} = \langle P_2, \bar{P}\rangle,$$

$$P\begin{pmatrix}\pi_1 \text{ and } \pi_2 \text{ fail}\\ \text{on a random demand } X\end{pmatrix} = pfd_{12} = \langle P, P\rangle$$

and, consequently, the covariance between the versions $V_1$ and $V_2$ can be written as[7]

$$\langle V_1, V_2 - q\bar{P}\rangle = \langle P, P\rangle\langle Y, Y\rangle - \langle P_1, P_1\rangle\langle P_2, P_2\rangle \tag{B.4}$$

where $\langle Y, Y\rangle = \big(1 - \langle P_1, P_1\rangle - \langle P_2, P_2\rangle - \langle P_2, P_2\rangle - \langle P, P\rangle\big)$ is the probability that both versions succeed in operation. Note that

$$\underbrace{\langle P, P\rangle\langle Y, Y\rangle}_{\substack{\text{A product of probabilities}\\ \text{where one probability concerns}\\ \text{coincident version failure, and}\\ \text{the other coincident version success}}} \qquad - \qquad \underbrace{\langle P_1, P_1\rangle\langle P_2, P_2\rangle.}_{\substack{\text{A product of probabilities,}\\ \text{where each probability concerns}\\ \text{the exclusive failure of a version}}}$$

In this factored form it is clear that the sign of the covariance is determined by the sizes of two terms: a product of probabilities where the versions have identical behaviour (that is, probability of "both versions fail" and probability of "both versions succeed"), and a product of probabilities where the versions have dissimilar behaviour (that is, for each version, the probability of the version failing exclusively). Therefore, the versions fail independently *if and only if*

$$\langle P, P\rangle\langle Y, Y\rangle = \langle P_1, P_1\rangle\langle P_2, P_2\rangle.$$

However, it is unobvious how a development process may be constrained in such a way as to ensure such a relationship for the pair of versions produced. Additionally, the versions

[7]If we recall that:

$$P\begin{pmatrix}\pi_1 \text{ fails on a}\\ \text{random demand } X\end{pmatrix} = pfd_1 = \langle V_1, \bar{P}\rangle = \langle P_1 + P, \bar{P}\rangle = \langle P_1, \bar{P}\rangle + \langle P, \bar{P}\rangle,$$

$$P\begin{pmatrix}\pi_2 \text{ fails on a}\\ \text{random demand } X\end{pmatrix} = pfd_2 = \langle V_2, \bar{P}\rangle = \langle P_2 + P, \bar{P}\rangle = \langle P_2, \bar{P}\rangle + \langle P, \bar{P}\rangle = q$$

then the covariance between the versions $V_1$ and $V_2$ can be written as

$$\begin{aligned}\langle V_1, V_2 - q\bar{P}\rangle &= \langle P_1 + P, P_2 + P - \left(\langle P_2, \bar{P}\rangle + \langle P, \bar{P}\rangle\right)\bar{P}\rangle\\ &= \langle P, P\rangle\left(1 - \langle P_1, P_1\rangle - \langle P_2, P_2\rangle - \langle P, P\rangle\right) - \langle P_1, P_1\rangle\langle P_2, P_2\rangle\\ &= \langle P, P\rangle\langle Y, Y\rangle - \langle P_1, P_1\rangle\langle P_2, P_2\rangle\end{aligned}$$

where $\langle Y, Y\rangle = \big(1 - \langle P_1, P_1\rangle - \langle P_2, P_2\rangle - \langle P_2, P_2\rangle - \langle P, P\rangle\big)$ is the probability that both versions succeed in operation.

exhibit negative failure correlation *if and only if*

$$\big\langle P, P \big\rangle \big\langle Y, Y \big\rangle < \big\langle P_1, P_1 \big\rangle \big\langle P_2, P_2 \big\rangle,$$

and positive failure correlation *if and only if*

$$\big\langle P, P \big\rangle \big\langle Y, Y \big\rangle > \big\langle P_1, P_1 \big\rangle \big\langle P_2, P_2 \big\rangle.$$

Now, if the development process is such that the versions created tend to be very reliable, then the probability of joint success $\big\langle Y, Y \big\rangle$ is very large and the probability of coincident failure $\big\langle P, P \big\rangle$ is very small $\big(\big\langle P, P \big\rangle \ll 1\,,\, \big\langle Y, Y \big\rangle\big)$. So, in order for the versions to be positively correlated, it is sufficient that one version have a probability of failing exclusively that is equal to the probability of coincident failure. While we do not expect equality in practice, it might be the case that the sizes of these probabilities are comparable. Experimental studies of diversity have not focused on estimating the relative sizes of these probabilities[8] and, for development processes that result in very reliable systems, failures (whether coincident or not) are extremely rare. This presents difficulties for making such estimates. However, if these probabilities were comparable then, due to the relatively large probability of coincident success $\big\langle Y, Y \big\rangle$, positive failure correlation can be expected. This might explain the lack of negatively correlated version pairs that are not orthogonal in [10], and the lack of negative failure correlation for the pairs of versions created to solve the "$3n + 1$" and "*factovisors*" programming challenges reported in [57]. To round off this section we make two observations:

- for a scenario with 2 demands (this approximates a situation where the demand space can be partitioned into 2 large, meaningful subsets where versions fail identically over the subset) it is impossible for non–orthogonal versions to be negatively correlated. In 2–dimensions, for a pair of versions that have some coincident failure, we have $\big\langle P, P \big\rangle \neq 0$ and $\big\langle P_i, P_i \big\rangle = 0$ for some $i = 1, 2$. This implies that the rhs of Eq. (B.4) cannot be negative.
- the factored form of covariance – the rhs of Eq. (B.4) – is derived for a pair of imperfect versions. However, this form is still true if, instead of a pair of versions, we considered a population of versions (in this case the vectors would represent difficulty functions). That is, the covariance in the LM Eq. (B.3) can be factored similarly. Consequently, the observations made for the case of a particular pair of versions carry over to the case where version pairs are averaged over. The difference being that considerations on average are weaker than considerations for a particular pair of versions: negative correlation, on average, does not imply negative correlation for particular version pairs.

[8]However, it is the case that the studies might give indications to general patterns, if any. For instance, half of the faults found in the Knight and Leveson experiment involved 2 or more of their (mostly very reliable) programs.

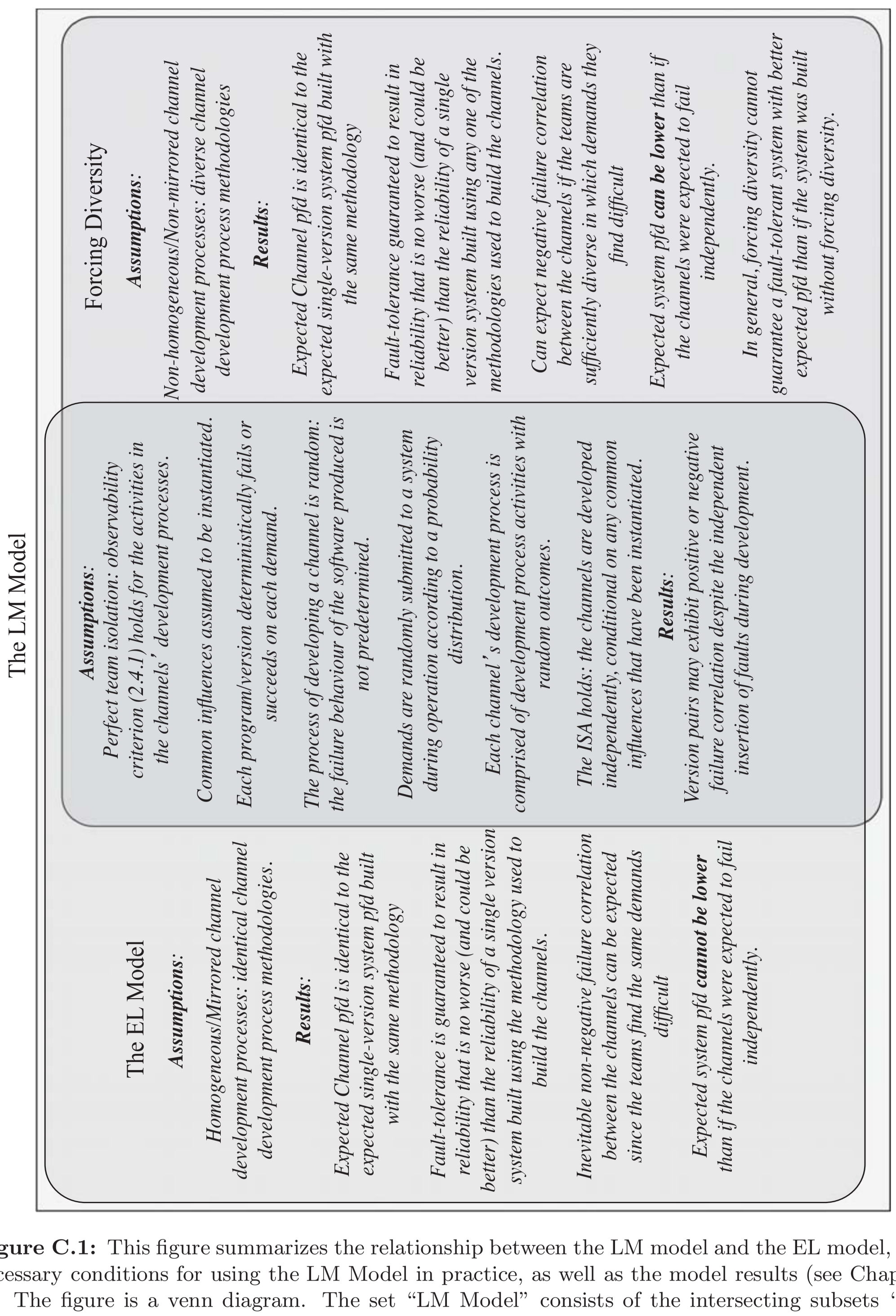
# Appendix C

# The LM Model: Interpretations and Practical Considerations

The LM model may be applied in a variety of situations. In this chapter we further discuss the definition of the LM model, considering the various mathematical constructs that make up the LM model, and showing different ways in which the model may be applied in practice (see Chapter 2 for a detailed introduction to the LM model). In so doing we aim to demonstrate the wide applicability of the LM model.

## C.1 A Characterization of Isolated Teams

Given a system development process (consisting of a pair of channel development processes) how would one determine, in practice, if the development teams thereof are perfectly isolated? Intuitively, achieving perfectly isolated development teams in practice should imply that each team's respective development process is indistinguishable from a single–version system development process. In this regard, when introducing the LM model in Chapter 2 ,a necessary requirement for perfectly isolated development teams was observability criterion 2.4.1. This was the requirement that *the outcomes of activities in a given channel development process be such that from them a participant in the process cannot confirm or refute the existence of another channel development process.* In practice, any software development process that aims to achieve the independent development of multiple, functionally-equivalent software versions must satisfy this requirement. Note, however, that this is not a sufficient condition for the channels to be developed probabilistically independently: despite this property holding further justification needs to be given for why the probability of developing an arbitrary pair of versions factors into a product of the probabilities of developing each of the versions. It is worth noting that criterion 2.4.1 is a property of the outcomes of the process, and not the state of mind of the development team members: it does not mean that one development team is not aware of the other team's existence. For instance, the development teams may be separated in time with one development process beginning only after the other



**The EL Model**

***Assumptions*:**

*Homogeneous/Mirrored channel development processes: identical channel development process methodologies.*

***Results*:**

*Expected Channel pfd is identical to the expected single-version system pfd built with the same methodology*

*Fault-tolerance is guaranteed to result in reliability that is no worse (and could be better) than the reliability of a single version system built using the methodology used to build the channels.*

*Inevitable non-negative failure correlation between the channels can be expected since the teams find the same demands difficult*

*Expected system pfd* ***cannot be lower*** *than if the channels were expected to fail independently.*

***Assumptions*:**

*Perfect team isolation: observability criterion (2.4.1) holds for the activities in the channels' development processes.*

*Common influences assumed to be instantiated.*

*Each program/version deterministically fails or succeeds on each demand.*

*The process of developing a channel is random: the failure behaviour of the software produced is not predetermined.*

*Demands are randomly submitted to a system during operation according to a probability distribution.*

*Each channel's development process is comprised of development process activities with random outcomes.*

*The ISA holds: the channels are developed independently, conditional on any common influences that have been instantiated.*

***Results*:**

*Version pairs may exhibit positive or negative failure correlation despite the independent insertion of faults during development.*

**Forcing Diversity**

***Assumptions*:**

*Non-homogeneous/Non-mirrored channel development processes: diverse channel development process methodologies*

***Results*:**

*Expected Channel pfd is identical to the expected single-version system pfd built with the same methodology*

*Fault-tolerance guaranteed to result in reliability that is no worse (and could be better) than the reliability of a single version system built using any one of the methodologies used to build the channels.*

*Can expect negative failure correlation between the channels if the teams are sufficiently diverse in which demands they find difficult*

*Expected system pfd* ***can be lower*** *than if the channels were expected to fail independently.*

*In general, forcing diversity cannot guarantee a fault-tolerant system with better expected pfd than if the system was built without forcing diversity.*

**Figure C.1:** This figure summarizes the relationship between the LM model and the EL model, the necessary conditions for using the LM Model in practice, as well as the model results (see Chapter 2). The figure is a venn diagram. The set "LM Model" consists of the intersecting subsets "EL Model" and "Forcing Diversity" (these sets intersect in some of their assumptions).

process has finished (e.g. functionally equivalent COTS software with development processes separated by a year). The team members in the latter process may be aware that the former process took place, and yet the outcomes of the activities in the latter process are such that they do not reference (even if indirectly) the previous process.

The criterion is also necessary for the development of a single version system, using a given methodology[1], to be probabilistically indistinguishable from the development of a 1–out–of–2 system's channel built using the same methodology. This is because the criterion ensures that the sample spaces in each of these scenarios is identical; examination of the outcomes cannot be used to distinguish between these two development scenarios. The further requirement that the same methodology be used in both scenarios ensures that the probability distributions are identical. This will be useful when discussing the result that, under the LM model assumptions, "on average, using a 1–out–of–2 system architecture ensures a system reliability that is no worse, and may be better, than the system reliability resulting from employing a related single–version system architecture" (see Section 2.5 for a more precise statement of this result). Note that if the criterion does not hold then the sample spaces are not identical, and therefore the probability distributions cannot be identical in the usual meaningful way.

## C.2 Conditionally Independent Version Sampling Distributions

The version sampling distributions in the LM model are conditionally independent, conditional on some common event during the system development process or during system operation. This is exemplified in Fig. C.2. In this diagram there are no random variables depicted as occurring during the system development process in such a way as to lie on a path connecting the channel development processes. However, the purpose of the diagram is to depict relationships between sources of uncertainty. Therefore, if a random variable has its value fixed for the period over which the model is valid this random variable is not depicted in the graph. Typical examples of such random variables include an unchanging system specification, the education and experience of the development team members, the organisation funding the development process and the industry "state of the art". Each of these potential sources of uncertainty might be unchanging over a significant period of time. As a consequence a diagram such as Fig. C.2 may not depict them. However, the values of such random variables can affect both channel development processes, thereby inducing conditional independence between the channel development processes. So, for the pair of versions $\pi_1$ and $\pi_2$ the selection/development of $\pi_1$ for one channel is conditionally independent of the selection/development of $\pi_2$ for the second channel, conditional on common influences that describe the "state of the world" shared by the channel developments, but not depicted in the graph.

While in general these conditional influences can be subject to variation and uncertainty

[1]For the definition of methodology used in this thesis please see Section 2.1)

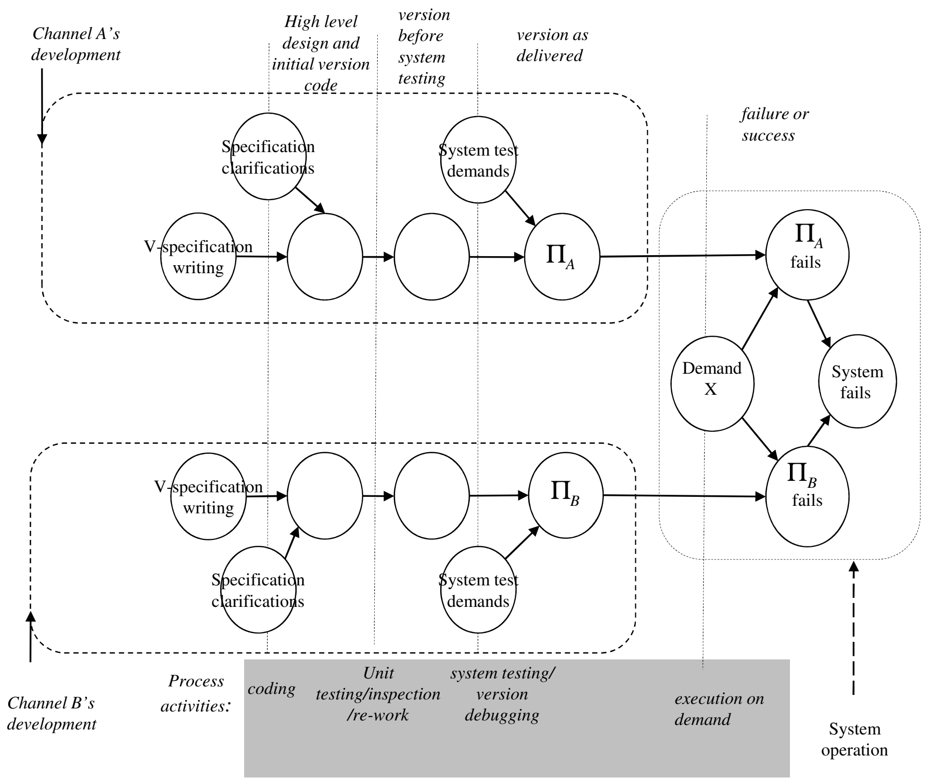


**Figure C.2:** An example of a development process that may be modelled via the LM model. Such a diagram does not distinguish between a scenario where there are no common influences during system development, and a scenario where all of the common influences are instantiated (that is, they have values that remain fixed throughout the development process.)

there are two view points where such uncertainty disappears:

1. before the system development process has begun the values of such influences are decided upon, and intended to be fixed for the duration of the system development process (e.g. a fixed budget for the project, and personnel). Consequently, one may apply the LM model with fixed values for these influences before the system development process has begun[2];
2. after the system development process has completed the values of these influences have been observed, and are known with certainty to have been unchanging;

Note that even if these common influences change during development the LM model may still be used, conditional on the "latest" values of the influences[3] and the satisfying of observability criterion 2.4.1.

## C.3 Relationships Between The EL and LM Models

The EL model has been described as a particular case of the LM model: it is the LM model in the situation where the channel development processes utilise identical methodologies. Consequently, the teams develop their respective versions "in the same way". They are equally likely to make the same mistakes during development. The mathematical characterisation of this is that the channel development processes have identical version sampling distributions. However, it is possible to describe the LM model as resulting from a sampling with replacement from a distribution whose mean is given by the EL model. To see this recall from Chapter 2 that a typical version sampling distribution is conditional on the outcomes of the activities in a channel's development process (for example, $\mathrm{P}_{\Pi}(\pi|d(n-1),\ldots,d1)\big)$. In this sense the version sampling distribution can be viewed as a measurable function of random variables thus making it a random variable (i.e. $\mathrm{P}_{\Pi}(\pi|D(n-1),\ldots,D1)$). We refer to this as a ***version sampling random variable***. This means that if $d^{'}(n-1) \in \Omega_{d_{n-1}},\ldots,d^{'}1 \in \Omega_{d_1}$ is a set of development process activity outcomes (and, therefore, may be viewed as a development process methodology) and $d(n-1) \in \Omega_{d_{n-1}},\ldots,d1 \in \Omega_{d_1}$ is a different set/methodology then

$$\mathrm{P}_{\Pi}(\pi|d(n-1),\ldots,d1) \neq \mathrm{P}_{\Pi}(\pi|d^{'}(n-1),\ldots,d^{'}1)$$

in general. This suggests that there is variation in the version sampling random variable across different values of the development activities' outcomes. Each instantiation of the version sampling random variable defines a conditional distribution for modelling a channel development process with the given activity outcomes. A pair of instantiations would be a

[2]Whether this model is still valid after the development process has completed is a related, but separate matter. Certainly, it is not unusual for personnel or budgetary changes to be made during the course of system development, and for such changes to have undesired effects [61]. Nevertheless, the point here is that when the LM model is instantiated it is valid to treat these possible sources of uncertainty as known and fixed since this would reflect the intended conditions under which system development will occur.

[3]"Latest" is in quotes because the intention here is not to suggest that time is necessarily the relevant factor for deciding which values of influences to use. Time is used here because it is easy to think in terms of a model with a memoryless property of the distant past: only the most recent values of random variables matter in determining the stochastic law to use for the system's further evolution. However, in general, as long as the relevant values of the influences can be identified the version sampling distributions are conditioned on these values.

diverse pair of channel development processes, in general. Such a pair of sampling distributions define a system development process with diverse channel development processes. The probability that a version $\pi$ is created for one channel is not necessarily the same as the probability the version is created for the other channel. Note that the mean of the distribution of the random variable $\mathrm{P}_{\Pi}(\pi|D(n-1), \ldots, D1)$ is obtained by averaging the version sampling distribution over all the possible values of the development process activities resulting in the version sampling distribution $\mathrm{P}_{\Pi}(\pi)$. So, if there is "complete uncertainty" concerning the outcomes of activities in a channel development process the version sampling distribution to use is the one given by the average. Therefore, in a system development process where each channel's development process has "complete uncertainty" the sampling distributions for each channel will be identical: the average sampling distribution $\mathrm{P}_{\Pi}(\pi)$. This is an EL model situation. This illustrates a possible relationship between LM models (being "fine–grained" entities on the one hand) and EL models (being "aggregate" entities on the other hand), and can be characterized as follows: the more uncertainty one has about the independent channel development processes the less diverse the channel development processes appear to be.

## C.4 Modelling Non–linear Software Development Frameworks

In developing the LM model the development process activities were linear, with later activities directly depending only on the outcomes of the activity immediately prior. For instance, in each channel development process depicted in Fig. C.2 the activity "*unit testing and code inspection*" is directly dependent only on the outcome of the activity "*coding*". However, such a linear relationship between the activities is not necessary for the LM model to be applied. This is because even if the activities within a channel development process have non-linear dependence relationships, as long as the outcomes of the activities in the process conform to observability criterion 2.4.1, the version sampling distributions can be conditionally independent. This means that various kinds of software development frameworks (e.g. enhanced waterfall models, the Spiral model, or Agile methods [30, 31, 32]) may be modelled by LM, including software development processes involving cyclical relationships between different activities (e.g. errors discovered in a prototype version may expose errors in the specification resulting in a specification update). The models can describe what happens over each cycle in a cyclical development framework, or what ultimately happens at "the end" of the development process (in those scenarios where it makes sense to speak of the development process ending).

## C.5 When Forcing Diversity Cannot Worsen Reliability

We demonstrated in Chapter 2 (Section 2.5) two senses in which forcing diversity was shown to result in the best reliability, compared with when diversity is not forced. This is true when a manager of a development process is:

1. "indifferent" between which of a set of available methodologies to use when diversity is not forced. Consider a situation where the manager does not have sufficient evidence to determine which methodology results in the best expected system pfd. Even though the manager might be willing to accept that there might exist an ordering between the methodologies – in the sense that use of some methodologies results in better expected system pfds than others – the manager can be agnostic about what that order actually is. So, the discrete probability distribution that models the managers propensity of choosing a given methodology is such that the methodologies are equally likely to be chosen;
2. "indifferent" between the expected system *pfd*s resulting from not forcing diversity. Here, the manager has sufficient evidence to indicate that the numerical value of the expected system *pfd* is the same, irrespective of which methodology is employed in system development.

Both of these results are generalised in Chapter 5 and summarized in Fig. 6.14 on page 186.

## C.6 Cost Considerations

Cost considerations complicate decisions about whether to employ a fault–tolerant architecture and whether to force diversity. For instance, the result just outlined for the virtue of fault–tolerance under the LM model was demonstrated by comparing the reliability resulting from a 1–out–of–2 system development with the reliability of "*a necessarily cheaper*" channel development process. This, however, may be an incomplete consideration. That is, it might be the case that the reliability gain from employing a 1–out–of–2 architecture does not justify the possible extra expense. Generally, there is no simple way in which the LM model can be used to compare the reliability when fault–tolerance is not used with the reliability when it is used. This is because cost can be considered as part of the methodology for a channel development: it affects the way a version is developed and it can change from one development process to another. Consequently, unless explicit assumptions are made about how changes in cost change version sampling distributions, there is no general way in which version sampling distributions may be compared on a cost basis. Similar issues arise when considering whether to force diversity, or not, in the context of the LM model. The use of extra methodologies in forcing diversity may imply development costs cannot decrease, but may increase. Again, a comparison that involves changing costs requires extra assumptions about the relationship between changes in methodologies and the consequences of these changes for version sampling distributions. So, again general results seem unlikely. Certainly, if forcing diversity is too expensive to implement then one may explore how much process diversity can be implemented, and whether this limited process diversity should be pursued. In this regard it seems likely that applying Theorem 5.2.2 (on page 135) to subsets of the set of N channels would still produce arguments for forcing diversity, other things being equal. Experimental evaluation of the relationship between cost and reliability gains from employing fault–tolerant architectures could prove useful in indicating trends; work has been carried out in this regard [60, 62].

## C.7 Definitions of Terms and Concepts in The LM Model

The application of the LM model to practical situations requires care in defining and using the model constructs. Notwithstanding, the model benefits from a significant amount of flexibility so that many different systems can, and have been modelled using the formalism (see [54] for a general overview. Also, see [18] for a discussion of its application to security, [55] for an application to computer–aided detection in a medical environment and [58] for a discussion of its application to database management systems). In this section we explore some of the flexibility in defining LM model concepts for different practical scenarios. The point is to demonstrate the freedom in applying the modelling framework to different scenarios.

### C.7.1 Relationship Between The Score Function and Correctness

For the score function to be defined in practice it is necessary for some notion of correctness to have already been defined. Here we are appealing to the idea that, ultimately, it is possible to determine whether a program has responded correctly, or not, to a demand received from the environment, even if such enlightenment is gained long after the fact. That is, for every practical scenario of interest we postulate the existence of an oracle such that by it's use it is always possible to determine when a given version fails on a given demand. Ideally, such an oracle should be intimately related to the requirements specification for the system. However, there may be errors contained in this documentation. In such cases the LM model is still applicable since the notion of correctness used to define the score function is not necessarily obtained from the requirements and specification. Hence, a program that implements the erroneous specification will be determined by the score function to fail in operation.

### C.7.2 Relationship between Demand Space, Program Space and Score Function

There are a number of different ways in which we may define the triplet of the demand space $\mathcal{X}$, the space of programs $\mathcal{P}$ and the score function $\omega(\pi, x)$. This is useful since it allows the LM model to be used to analyse possibly different systems in different ways, and thereby answer different questions related to system reliability. In what follows we explore various examples of such definitions. Our aim is to point out the care that should be taken when defining $\mathcal{X}$, $\mathcal{P}$ and $\omega(\pi, x)$ since these quantities are necessarily related, and cannot all be defined independently of each other[4]. However, within this constraint there are various possibilities for the definitions. For illustration consider the following scenarios, each with

[4]The reason for this is because we are defining a function, $\omega(\pi, x)$. So, for $\omega(\pi, x)$ to be well defined its domain, the cartesian product of the sets $\mathcal{X}$ and $\mathcal{P}$, must be well defined. In practice, this means that it is impossible for the definition of $\omega(\pi, x)$ to be independent of $\mathcal{X}$ and $\mathcal{P}$.

corresponding suggested definitions for these three entities. The definitions associated with each scenario are non–unique and alternatives are certainly possible.

1. Consider a system that is intended to contain a yet to be built software component, where this software component should operate in a given environment. Treating the software as a black box let us suppose that in order for an observer to determine whether the software has failed it is necessary and sufficient to observe the environmental input to, and the corresponding output of, the software. In this sense the internal state of the software is unimportant in determining system failure. Consequently, we may define $\mathcal{X}$ as the set of all possible inputs to the system from the environment. $\mathcal{P}$ is defined as the set of all programs that may possibly be developed. So, for $\pi \in \mathcal{P}, x \in \mathcal{X}$ we define the score function

$$\omega(\pi, x) = \begin{cases} 1, & \text{if } \pi \text{ can be executed in the system and in operation fails on } x \\ 0, & \text{otherwise.} \end{cases}$$

   Note, it is reasonable to expect that programs are created to run on a specified target software platform (for instance, source code may be compiled to create operating system specific executables). Consequently, programs created for one platform may not execute on another platform. $\mathcal{P}$, as the set of all programs that may be developed, can contain programs that will not be executable on the platform of choice for the system under consideration. Strictly speaking, the act of submitting an input to such a program from the environment does not make sense since such a program cannot be executed as part of the system under consideration. However, the score function as defined above treats system success and the inability to execute the program $\pi$ in the system as equivalent: they are both indicated by the value "0" of the score function. Clearly, these two events are significantly different. So, the version sampling distributions (such as $\mathrm{P}_{\Pi}(\cdot)$) may be defined with the requirement that versions that cannot be executed in the system have zero probability of being created. That is, programs that can be executed are developed, "almost surely"[5]. In essence, this indirectly transforms expectations of the score function into sums of probabilities of either success or failure events exclusively. Alternatively, $\mathcal{P}$ may be defined to be the set of all programs that can be executed in the system. Upon doing this the score function definition may be given as

$$\omega(\pi, x) = \begin{cases} 1, & \text{if } \pi \text{ fails on } x \text{ in operation} \\ 0, & \text{if } \pi \text{ succeeds on } x \text{ in operation} \end{cases}$$

2. If the internal program state is important (in the sense of example 1 above) in determining whether a system fails or succeeds then $\mathcal{X}$ may be defined as the set of all ordered pairs consisting of a possible internal program state and a possible input to the system from the environment. $\mathcal{P}$ is the set of all programs that may possibly be developed, and for

[5]Events whose complements have zero probability of occurrence are said to occur "almost surely".

$\pi \in \mathcal{P}, x \in \mathcal{X}$ we define the score function

$$\omega(\pi, x) = \begin{cases} 1, & \text{if } \pi \text{ can be executed in the system and in operation fails on } x \\ 0, & \text{otherwise.} \end{cases}$$

3. Consider the development of a system channel such that there is a fixed computer memory size that the program to be developed must fit in (for instance, 1 GB). This considerably limits the space of possible programs from the space of all possible programs. Further, suppose that the sensors for the system being built ultimately provide inputs with a digital resolution of the environment so that there is a finite, possibly very large number of binary inputs to the system (for example, a resolution of 64 bits). Then $\mathcal{P}$ may be defined as the set of all programs that may fit into the fixed amount of memory, $\mathcal{X}$ may be defined as the set of all ordered pairs of possible program states and binary inputs to the system that describe the environment, and for $\pi \in \mathcal{P}, x \in \mathcal{X}$ we define the score function

$$\omega(\pi, x) = \begin{cases} 1, & \text{if } \pi \text{ can be executed in the system and in operation fails on } x \\ 0, & \text{otherwise.} \end{cases}$$

   Again, $\mathrm{P}_{\Pi}(\cdot)$ is defined appropriately to make the score function usefully detect success.

4. Given that whatever program is developed must run on a predetermined target (Hardware and Software) platform, $\mathcal{P}$ can be defined as the set of all programs that can fit in some fixed amount of memory, be deployed and are executable on the target platform. Then, the set of demands can be defined as the set of all ordered pairs of possible program states and binary inputs to the system that describe the environment, and for $\pi \in \mathcal{P}, x \in \mathcal{X}$ we define the score function

$$\omega(\pi, x) = \begin{cases} 1, & \text{if } \pi \text{ fails on } x \\ 0, & \text{otherwise.} \end{cases}$$

   In this case the definition of the score function is equivalent to a useful and complete definition of failure and success events. Consequently, the distribution $\mathrm{P}_{\Pi}(\cdot)$ does not have to be defined in such a way as to assign zero probability to the development of certain programs.

5. What about if a subset of the inputs are such that failure to correctly handle these inputs results in significantly worse consequences compared with failing to handle other inputs? Then we may consider these inputs as being critical in this sense, and restrict our analysis to studying the behaviour of software over this input subset. So, we define $\mathcal{X}$ as the set of all ordered pairs of system–state and critical environmental input. $\mathcal{P}$ and $\omega(\pi, x)$ may be defined in the same way as example 4 above.

6. Further still, given partial knowledge of a program's failure behaviour, suppose tests of the program on some subset of the input space show that the program does not fail on the subset. Therefore, this program could potentially be any program from $\mathcal{P}$ that does not fail on the same subset of demands[6]. Thus, we may define $\mathcal{P}$ as the set of all programs

[6]Here, we are assuming that confidence in the results of testing is not an issue, and that no extra knowledge

which have the property that they do not fail on this subset of the demand space. In such a scenario the failure behaviour of such programs on the complement of this subset of demands is of interest. $\mathcal{X}$ and the score function may be defined in the same way as example 5 above.

### C.7.3 The (In)divisibility of Programs

We have illustrated different ways of defining $\mathcal{P}$. Also, the definition of a program for the purposes of modelling has some flexibility. Each definition of program we might be interested in derives from the usual definition of a program, but abstracts away from this definition by concentrating on a property of interest. For instance, as was hinted at previously, a program could be a binary sequence that fits in some fixed amount of memory. Alternatively, all programs that have identical failure sets define an *equivalence class* and, consequently, they may be considered to be "the same" program. Further yet, programs that have the same behaviour in terms of control flow are considered to be the same program despite not having identical binary signatures. It is also useful to note that the modelled notion of a program may be viewed as either being atomic (each member of $\mathcal{P}$ is a single entity) or not. Atomicity is acceptable despite the possibility that in practice there may be a sense in which one program is "contained within" another program. The point is if the programs are intended to be functionally equivalent, forming the channels of a 1–out–of–2 system, then modularity is unimportant to answer the question "do a pair of programs fail coincidentally?". We have not come across any practical scenario where application of the models has necessarily required a non–atomic notion of programs.

### C.7.4 Subjective vs Objective Probabilities

Another aspect of modelling that is flexible is the "nature" or interpretation of the probabilities. To a large extent this thesis has presented probabilities, like system *pfd*, as *objective* probabilities: the values of these probabilities are the same for all observers. However, this does not need to be the only applicable viewpoint as, alternatively, the probabilities may be *subjective*: possibly changing from one observer to another and dependent on an observer's state of knowledge about the environment in which an experiment is taking place. As an example of this consider the result that under indifference between methodologies forcing diversity guarantees reliability that is no worse, and may be better, than if diversity is not forced. This was first stated in Chapter 2 (see Eq. (2.19) on page 48) and generalised in Chapter 5 (see Theorem 5.2.2 on page 135). Each of these results can be viewed as statements involving either *objective* or *subjective* probabilities. Under a viewpoint of objectivity the probabilities have fixed, possibly unknown, values. The results therefore are statements that are true whatever the fixed values of the *pfd*s are. To use the results the values of the expected *pfd*s do not need to be known; one need only verify that a given scenario satisfies the conditions under which the results hold. Alternatively, if viewed as subjective entities the expected *pfd* values in these results are the values an observer/assessor has evidence

concerning the behaviour of the program on other subsets of the demand space is available.

to support a belief in. In this sense the results act as a consistency check for the observers beliefs: if the values do not obey the inequalities then they are inconsistent, and the observer will need to modify her beliefs.

### C.7.5 The Definition of Program and System Failure

Use of the LM model requires that the failure behaviour of the versions in isolation be sufficient to determine the failure behaviour of the 1–out–of–2 system that is comprised of the versions. The score function, $\omega(\pi, x)$, states when a given program $\pi$ fails on a given demand $x$. To determine when a 1–out–of–2 system consisting of a pair of versions, $\pi_1$ and $\pi_2$ say, fails on a given demand $x$ it is sufficient to multiply the respective values of the score function, $\omega(\pi_1, x)\omega(\pi_2, x)$. There is an unstated assumption here. We assume that the act of combining the programs into a system does not introduce, or remove, the failure behaviour of the individual programs. To this end only the consequences of design faults are the main focus of these conceptual models. However, further care should be taken in what the definitions of channel and system failure are, and how these are related. This is because the score function is not necessarily a direct statement about the outputs of the channels. Instead, it is a statement about a consistent classification of the channel and system outputs into failure and success categories, where coincident channel failure is sufficient for system failure.

As an example of the kinds of issues that may arise, consider an application of fault–tolerance in detecting channel failure using back–to–back, channel–output comparisons. That is, comparing the numerical outputs of a system's constituent channels, and thereby looking for differences between these values which would indicate that at least one of the channels has failed. For the purposes of giving a simple example suppose that differences in the channel outputs due to the use of finite approximation mathematics do not occur[7]. In effect this means that we do not have a problem with false–positives (like the protection–system scenario we considered when introducing the LM model in Chapter 2 ). So, whenever the versions both succeed their outputs will be identical. Then the LM model can be viewed as a "pessimistic"[8] demonstration of the extent to which coincident failure undermines the failure detection capability of fault–tolerance. Pessimistic because coincident failure may involve different wrong outputs from the channels which would be detected as a failure occurrence in practice, but the LM model as presented here ignores[9].

[7]A dramatic example of such phenomena is the so–called *consistent comparison problem* detailed in [63].

[8]The quotes indicate that the pessimism here can be argued to be optimism, since false positives due to phenomena like the *consistent comparison problem* are assumed to not be an issue.

[9]The use of diverse, coincident failure as a means of failure detection may or may not be practical. Certainly, this has been argued as a possible cost–effective means of failure detection, and is intimately linked with the ethos of fault–tolerant approaches like *N–version programming* [14, 8]. This approach might be adequate in a scenario where upon detecting differences in channel output transfer of control is given to some backup–system, such as in the A310 flap/slat control system where failure detection causes a backup–system to take over for the continued safe flight and landing of the aircraft [10]. There are, however, issues which need to be considered with the approach including the *consistent comparison problem* (which implies differences of channel output may not necessarily be due to failure) and the added possible impact on system availability; even if failure is detected there may be a need to carry out further analysis offline thus reducing system availability. In the NASA sponsored "*four universities*" experiment [11] "continued service" was a priority and consequently coincident failure was sufficient for system failure despite the possibility of the

### C.7.6 The Definition of Development Teams

There is also flexibility in the number of development teams that the model caters for. Thus far, in the model's development and use, it has been useful to model in terms of a single development team per channel development process. However, this is not a necessary requirement: a channel's development process may involve multiple teams, with varied responsibilities, interacting with one another within the process. As long as the ISA holds the intra–team interactions within a given channel development process can be as complex as possible. Also, the models do not preclude the possibility or the effects of teams controlling the channel development processes from "outside", such as the coordinating team (C-team) in the development of *N–version software*. Furthermore, the number of "actual" teams that constitute "a single team" in each process need not be the same and does not change the model's applicability.

channels failing with different outputs.

# Bibliography

[1] P. Traverse, "Airbus and A.T.R. System Architecture and Specification," in *Software diversity in computerized control systems* (U. Voges, ed.), pp. 95–104, New York, NY, USA: Springer–Verlag New York, Inc., 1988.

[2] P. Regan and S. Hamilton, "Nasa's mission reliable," *Computer*, vol. 37, no. 1, pp. 59–68, 2004.

[3] G. Hagelin, "Ericsson safety system for railway control," in *Software diversity in computerized control systems* (U. Voges, ed.), pp. 11–22, New York, NY, USA: Springer–Verlag New York, Inc., 1988.

[4] U. Voges, "Use of diversity in experimental reactor safety systems," in *Software diversity in computerized control systems* (U. Voges, ed.), pp. 29–50, New York, NY, USA: Springer–Verlag New York, Inc., 1988.

[5] P. Bishop, "The PODS Diversity Experiment," in *Software diversity in computerized control systems* (U. Voges, ed.), pp. 51–84, New York, NY, USA: Springer–Verlag New York, Inc., 1988.

[6] A. Avizienis and L. Chen, "On the implementation of n–version programming for software fault–tolerance during program execution," in *COMPSAC77, Computer Software and Applications Conference*, pp. 149–155, November 1977.

[7] A. Avizienis and L. Chen, "N–version programming: A fault–tolerance approach to reliability of software operation," in *Digest of papers: 8th annual International Symposium on Fault–tolerant Computing*, pp. 3–9, 1978.

[8] A. Avizienis, "The methodology of n-version programming," in *Software Fault Tolerance* (M. Lyu, ed.), pp. 23–46, John Wiley and Sons, 1995.

[9] B. Randell, "System structure for software fault tolerance," in *Proceedings of the international conference on Reliable software*, (New York, NY, USA), pp. 437–449, ACM, 1975.

[10] J. C. Knight and N. G. Leveson, "An experimental evaluation of the assumption of independence in multi-version programming," *IEEE Transactions on Software Engineering*, vol. SE-12, pp. 96–109, 1986.

[11] D. E. Eckhardt, A. K. Caglayan, J. C. Knight, L. D. Lee, D. F. McAllister, M. A. Vouk, and J. J. P. Kelly, "An experimental evaluation of software redundancy as a strategy for improving reliability," *IEEE Trans. Softw. Eng.*, vol. 17, no. 7, pp. 692–702, 1991.

[12] D. E. Eckhardt and L. D. Lee, "A theoretical basis for the analysis of multiversion software subject to coincident errors," *IEEE Transactions on Software Engineering*, vol. SE-11, pp. 1511–1517, 1985.

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

[14] M. Lyu and Y. He, "Improving the n-version programming process through the evolution of a design paradigm," *IEEE Transactions on Reliability*, vol. R-42, pp. 179–189, 1993.

[15] B. Littlewood and L. Strigini, "A discussion of practices for enhancing diversity in software designs," DISPO project technical report LS-DI-TR-04, Centre for Software Reliability, City University, 2000.

[16] B. Littlewood, P. Popov, L. Strigini, and N. Shryane, "Modelling the effects of combining diverse software fault removal techniques," *IEEE Transactions on Software Engineering*, vol. SE-26, pp. 1157–1167, 2000.

[17] D. Bosio, B. Littlewood, M. J. Newby, and L. Strigini, "Advantages of open source processes for reliability: clarifying the issues." http://www.csr.city.ac.uk/people/lorenzo.strigini/ls.papers/, 2002.

[18] B. Littlewood and L. Strigini, "Redundancy and diversity in security," in *ESORICS 2004, 9th European Symposium on Research in Computer Security* (P. Ryan and P. Samarati, eds.), Lecture Notes in Computer Science, (Sophia Antipolis, France), pp. 423–438, Springer–Verlag, 2004.

[19] L. Strigini, A. A. Povyakalo, and E. Alberdi, "Human-machine diversity in the use of computerised advisory systems: a case study," in *DSN 2003, International Conference on Dependable Systems and Networks*, (San Francisco, U.S.A.), pp. 249–258, 2003.

[20] M. Lyu, *A design paradigm for multi-version software.* PhD thesis, Los Angeles, CA, USA, 1988.

[21] W. Feller, *An Introduction to Probability Theory and Its Applications, Volume 1.* New York: Wiley, January 1968.

[22] W. Feller, *An Introduction to Probability Theory and Its Applications, Volume 2.* New York: Wiley, January 1968.

[23] P. R. Halmos, *Measure Theory.* The University series in Higher Mathematics, D. Vam Nostrand Company Inc., New York, 1950.

[24] H. Kestelman, *Modern Theories of Integration.* Dover Books on Advanced Mathematics, Dover Publications Inc, New York, 1960.

[25] I. Pesin, *Classical and Modern Integration theories.* Probability and Mathematical Statistics, Academic Press, New York, 1970.

[26] A. N. Kolmogorov, *Foundations of the Theory of Probability.* New York: Chelsea Publishing Company, 1956.

[27] M. R. Spiegel, *Schaum's Outline of Theory and Problems of Real Variables; Lebesgue Measure and Integration With Applications to Fourier Series.* New York: McGraw-Hill Publishing Company, 1969.

[28] M. C. Paulk, C. V. Weber, B. Curtis, and M. B. Chrissis, *The Capability Maturity Model: Guidelines for Improving the Software Process.* SEI series in software emgineering: Software Engineering Institute, Carnegie Mellon University, Addison Wesley Longman, Inc., 11 ed., 1995.

[29] J. S. Collofello, "Introduction to software Verification and Validation," vol. SEI-CM-13-1.1, Software Engineering Institute, Carnegie Mellon University, Addison Wesley Longman, Inc., 1988.

[30] McConnell, Steve, *Professional Software Development: Shorter Schedules, Higher Quality Products, More Successful Projects, Enhanced Careers.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[31] McConnell, Steve, *Rapid Development: Taming Wild Software Schedules.* Microsoft Press, 1 ed., July 1996.

[32] I. Sommerville, *Software Engineering (7th Edition) (International Computer Science Series).* Addison Wesley, May 2004.

[33] P. T. Popov and L. Strigini, "Conceptual models for the reliability of diverse systems - new results," in *28th International Symposium on Fault-Tolerant Computing (FTCS-28)*, (Munich, Germany), pp. 80–89, IEEE Computer Society Press, 1998.

[34] B. Littlewood, P. Popov, and L. Strigini, "Modelling software design diversity - a review," *ACM Computing Surveys*, vol. 33, pp. 177–208, 2001.

[35] S. Lauritzen, *Graphical Models*, vol. 17 of *Oxford Statistical Science Series.* Clarendon Press, Oxford, 1996.

[36] P.-J. Courtois, B. Littlewood, L. Strigini, D. Wright, N. Fenton, and M. Neil, "Bayesian belief networks for safety assessment of computer-based systems," in *System Performance Evaluation: Methodologies and Applications* (E. Gelenbe, ed.), pp. 349–363, CRC Press, 2000.

[37] D. Heckerman, A. Mamdani, and M. P. Wellman, "Real-world applications of bayesian networks," *Communications of the ACM, Special Issue*, vol. 38, pp. 24–26, March 1995.

[38] P. Popov and B. Littlewood, "The effect of testing on the reliability of fault-tolerant software," in *DSN 2004, International Conference on Dependable Systems and Networks*, (Florence, Italy), pp. 265–274, IEEE Computer Society, 2004.

[39] A. Einstein, J. Stachel, and A. Engel, *The Berlin Years: Writings, 1914-1917. (English translation supplement)*, vol. Volume 6. Princeton University Press, Princeton, N.J. :, 1987.

[40] R. A. Sharipov, *A Course of Linear Algebra and Multidimensional Geometry.* Bashkir State University, February 1996.

[41] G. Strang, *Linear Algebra and Its Applications.* Brooks Cole, February 1988.

[42] D. C. Lay, *Linear Algebra and Its Applications.* Addison Wesley, second ed., 1997.

[43] S. Lipshutz, *Theory and Problems of Linear Algebra.* Mc-Graw-Hill Book Company, 1968.

[44] P. R. Halmos, *Finite-dimensional vector spaces.* Springer–Verlag, 1974.

[45] K. Salako, "Bounds on the reliability of fault-tolerant software built by forcing diversity," in *SAFECOMP '07, 26th International Conference on Computer Safety, Reliability and Security* (F. Saglietti and N. Oster, eds.), vol. 4680/2007 of *Lecture Notes in Computer Science*, pp. 411–416, Springer Berlin / Heidelberg, 2007.

[46] G. H. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities.* Cambridge, UK: Cambridge University Press, 1934.

[47] A. Cauchy, "Cours d'analyse de l'Ecole Royale Polytechnique," *Analyse Algebrique*, p. 457ff, 1821.

[48] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* New York: Dover, ninth Dover printing ed., 1964.

[49] Yeh, Y. C. (Bob), "Design Considerations in Boeing 777 Fly-By-Wire Computers," in *3rd IEEE High-Assurance Systems Engineering Symposium (HASE)*, (Washington, DC, USA), pp. 64–73, IEEE Computer Society Press, 1998.

[50] M. R. Lyu, *Software Fault Tolerance.* New York, NY, USA: John Wiley & Sons, Inc., 1995.

[51] P. Popov, L. Strigini, and A. Romanovsky, "Choosing effective methods for design diversity – how to progress from intuition to science," in *SAFECOMP '99, 18th International Conference on Computer Safety, Reliability and Security* (M. Felici, K. Kanoun, and A. Pasquini, eds.), Lecture Notes in Computer Science, pp. 272–285, Springer, 1999.

[52] B. Littlewood, P. Popov, and L. Strigini, "N-version design versus one good version," in *Proc. International Conference On Dependable Systems And Networks (FTCS-30 And DCCA-8*, pp. 42–43, 2000.

[53] P. Popov and L. Strigini, "The Reliability of Diverse Systems: a Contribution using Modelling of the Fault Creation Process," in *DSN 2001 – International Conference on Dependable Systems and Networks, 2001*, pp. 5–14, IEEE Computer Society Press, 2001.

[54] B. Littlewood, P. Popov, and L. Strigini, "Modeling software design diversity: a review," *ACM Comput. Surv.*, vol. 33, no. 2, pp. 177–208, 2001.

[55] Strigini, L. and Povyakalo, A. and Alberdi, E., "Human–machine diversity in the use of computerised advisory systems: a case study," in *DSN 2003 – IEEE International Conference on Dependable Systems and Networks*, pp. 249–258, IEEE Computer Society Press, 2003.

[56] A. Avizienis, M. Lyu, and W. Schutz, "In search of effective diversity: A six language study of fault tolerant flight control software," in *Proceedngs of the 18th International Symposium on Fault–Tolerant Computing*, pp. 15–22, June 1988.

[57] van der Meulen, M.J.P. and Revilla, M., "The effectiveness of Choice of Programming Language as a Diversity Seeking Decision," in *5th European Dependable Computing Conference (EDDC-5)*, Lecture Notes on Computer Science, pp. 199–209, Springer, 2005.

[58] I. Gashi, P. Popov, and L. Strigini, "Fault diversity among off-the-shelf SQL database servers," in *DSN 2004 – International Conference on Dependable Systems and Networks*, pp. 389–398, IEEE Computer Society Press, 2004.

[59] I. Gashi, P. Popov, V. Stankovic, and L. Strigini, "On Designing Dependable Services with Diverse Off-The-Shelf SQL Servers," in *Architecting Dependable Systems II* (de Lemos, R. and Gacek, C. and Romanovsky, A., ed.), vol. 3069 of *Lecture Notes in Computer Science*, pp. 191–214, Springer–Verlag, 2004.

[60] M. Kersken and F. Saglietti, eds., *Software Fault Tolerance: Achievement and assessment strategies.* Research Reports ESPRIT, Springer–Verlag, 1992.

[61] F. P. Brooks, Jr., *The mythical man-month (anniversary ed.).* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[62] K. Kanoun, "Real-world design diversity: A case study on cost," *IEEE Softw.*, vol. 18, pp. 29–33, July 2001.

[63] S. S. Brilliant, J. C. Knight, and N. G. Leveson, "The consistent comparison problem in n-version software," *IEEE Trans. Softw. Eng.*, vol. 15, no. 11, pp. 1481–1485, 1989.