
Proceedings Paper:
Walkinshaw, N. and Shepperd, M. (2020) Reasoning about uncertainty in empirical results. In: EASE '20: Proceedings of the Evaluation and Assessment in Software Engineering. EASE '20: Evaluation and Assessment in Software Engineering, 15-17 Apr 2020, Trondheim, Norway. Association for Computing Machinery (ACM) , pp. 140-149. ISBN 9781450377317

https://doi.org/10.1145/3383219.3383234

Reasoning about Uncertainty in Empirical Results

Neil Walkinshaw
n.walkinshaw@sheffield.ac.uk
The University of Sheffield
Sheffield

Martin Shepperd
martin.shepperd@brunel.ac.uk
Brunel University London
Uxbridge

ABSTRACT

Conclusions that are drawn from experiments are subject to varying degrees of uncertainty. For example, they might rely on small data sets, employ statistical techniques that make assumptions that are hard to verify, or there may be unknown confounding factors. In this paper we propose an alternative but complementary mechanism to *explicitly* incorporate these various sources of uncertainty into reasoning about empirical findings, by applying Subjective Logic. To do this we show how typical traditional results can be encoded as "subjective opinions" – the building blocks of Subjective Logic. We demonstrate the value of the approach by using Subjective Logic to aggregate empirical results from two large published studies that explore the relationship between programming languages and defects or failures.

CCS CONCEPTS

• **General and reference** → **Empirical studies**; • **Computing methodologies** → *Uncertainty quantification.*

ACM Reference Format:
Neil Walkinshaw and Martin Shepperd. 2020. Reasoning about Uncertainty in Empirical Results. In *Proceedings of EASE '20: Evaluation and Assessment in Software Engineering*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The task of analysing and communicating empirical data inevitably requires the ability to reason about uncertainty. There may be experimental factors that are beyond our control. Results might be dispersed and not point to a specific outcome. Data may be subject to measurement errors or sampling biases.

This uncertainty is generally impossible to eradicate. Good practice dictates that findings should be accompanied by appropriate caveats and metrics to support a fully informed interpretation. These might include extensive threats to validity, power statistics, significance statistics, confidence intervals, probabilistic models and openly-available data and materials to enable replications.

Within the Software Engineering community there have been several recent laudable initiatives that have sought to improve the trustworthiness of empirical results. This has been spurred by numerous negative findings, both from broader scientific community [17] as well as the Software Engineering community itself [12, 19], that highlight major problems with the reliance on traditional 'frequentist' statistics. For example Jørgensen *et al.* [19] suggest that up to 40% of results that are deemed to be "statistically significant" could be incorrect. To address this apparent crisis, Software Engineering conferences are increasingly supporting replicability by encouraging artefact submissions, the Open Data initiative is being increasingly promoted, as is the use of Bayesian techniques to explicitly reason about uncertainty in empirical results [12, 14].

Despite this increasing armoury of statistical tools, current approaches can be difficult to apply, particularly when multiple studies are involved. An overall "uncertainty", whether conveyed as a probability distribution, p-value, or confidence-interval, is inherently tied to a specific experimental setting. When there are multiple experiments, each will invariably have its own sources of uncertainty, which can be difficult to reconcile when the results of the experiment are combined. This becomes especially problematic when (for example):

- The data from the studies are measured using different techniques and are associated with different measurement errors.
- The studies use different empirical paradigms (e.g., surveys and quasi-experiments).
- The studies sample non-randomly from disjoint or partially-overlapping populations.

The broader challenge of "fusing" together sources of uncertain data is well-established. In the context of statistics, work on using uncertainty as an explicit component to reason about epistemic beliefs dates back to the work of Dempster and Shafer [33]. This line of work has, over the years, given rise to Subjective Logic [21] - a framework for reasoning about complex phenomena that are subject to uncertainty. Such frameworks have been successfully applied to reason about a wide variety of decision problems, spanning fields such as Sensor Fusion [23], Law [22], Argumentation Theory [3], safety critical systems [8, 25], and vehicular communications systems [7].

In this paper we show how Subjective Logic can provide a useful framework within empirical software engineering to handle the problems of uncertainty that arise in empirical studies. Most importantly, it offers a degree of flexibility, enabling the researcher to combine findings from studies that might be diverse in nature, whilst explicitly factoring-in their respective underlying uncertainties. We do not suggest that Subjective Logic should replace other reasoning mechanisms such as meta-analysis but we do believe it is complementary.

The rest of this paper is structured as follows. In Section 2 we provide an overview of existing approaches, and introduce a small motivating example, where we present the results from a multi-site study comparing test-driven development to iterative test-last

EASE '20, April 2020, Trondheim, Norway

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

development [34]. In Section 3 we provide an introduction to Subjective Logic. In Section 4 we show how Subjective Logic can be used to examine experimental outcomes, illustrating some of the key steps with respect to the motivating example presented in Section 2. In Section 5 we use Subjective Logic to combine data from two (loosely) related published studies on the fault / failure proneness of programming languages [26, 30]. Finally, in Section 6 we close with conclusions and avenues for future work.

2 BACKGROUND

We start with an overview of existing approaches to aggregate empirical results. This is followed by an introduction to our motivating example, which will also be used to illustrate our application of Subjective Logic to software engineering.

2.1 The Challenge of Aggregating Empirical Results

Software Engineering experiments commonly comprise the collection of response data (in a controlled setting) to evaluate different treatments [36]. The analysis typically involves the application of traditional statistical techniques, such as testing for statistical significance or establishing effect-sizes with associated confidence intervals [9, 12]. Using practitioners is challenging, because developer time is at a premium [10]. Moreover projects and settings can vary significantly in terms of scale and complexity from one domain to the other. As a consequence, findings are subject to a considerable degree of intrinsic uncertainty and need to be treated with some circumspection.

There are numerous approaches that can be used to analyse or aggregate empirical results. The choice of approach critically depends on the nature of the studies under analysis. In their review of candidate approaches [31], Santos and Juristo refer to three commonly used approaches: Narrative Synthesis, Aggregated Data, and Individual Participant Data (they also discuss the aggregation of p-values, which is problematic and not widely used in Software Engineering, so we do not cover it here). We do however include a further approach that is particularly relevant to Subjective Logic - Bayesian Analysis. We briefly review these four approaches here.

Narrative Synthesis [28] is used to provide a textual summary of empirical results, with the aim of forming an overall conclusion. It is widely used because it is straightforward to apply, but it does not provide quantitative results (e.g., joint effect-sizes or p-values) and can also lead to ambiguities [31]. In addition, results are not easy to reproduce.

Aggregated Data (AD) or 'meta-analysis of effect-sizes' is increasingly used in Software Engineering [31] (we will be providing an example in Figure 1). AD is popular because it can accommodate experiments with different designs and response variable scales, and can handle heterogeneity [31]. Moderating factors (e.g., programmer experience in developer studies) can be handled by performing a meta-regression with each moderating factor [32] (i.e., to fit a line between scores given for programmer experience and the effect-size). This approach to handling moderating factors can be problematic, in the sense that the core results of the AD (effect sizes and confidence intervals) are presented separately from the meta-regression results. If there are many moderating factors, this can make the outcomes of the experiment as a whole difficult to interpret.

Individual Participant Data mega-trial (IPD) refers to the process of pooling the raw data from multiple experiments, and re-analysing it as if the pooled data were obtained from a single, large experiment. Santos and Juristo note that this can produce biased results if there are differences in the design of the studies — even subtle ones — or different numbers of experimental units. One further obvious obstacle is that it depends upon the availability of the raw data, which is not the case for any of the other techniques reviewed here. If the raw data is available, however, it can be very helpful for deriving uncertainty values.

Bayesian analysis refers to a family of techniques that take advantage of Bayes' Theorem, the statement that the posterior probability is proportional to the likelihood times the prior. If any two of these are known, it is possible to derive the third. This provides a powerful framework, within which it is possible to examine the relationship between potential causes and effects, whilst incorporating assumptions about prior probabilities. Furia *et al.* [12] have shown how Bayesian analysis can improve upon traditional "frequentist" counterparts (we shall be re-visiting the same data they used for our own case study in this paper).

One weakness of Bayesian analyses is the requirement for prior probabilities, which can be difficult to obtain [13]. In the absence of a prior, Bayesian approaches are forced to resort to using "uninformative" priors such as the uniform prior. In other words, the probabilities of two propositions A and $\overline{A}$ might be deemed to be equal (0.5 each). The problem is that term 'uninformative' is in fact a misnomer; to indicate that two propositions are equally likely is not equivalent to the statement that their respective probabilities are in fact unknown. This apportioning of arbitrary prior probabilities can end up leading to skewed results [24].

2.2 Running Example

As a running example we refer to a multi-site experiment carried out by Tosun *et al.* [34] (Santos and Juristo subsequently used this to motivate their review of meta-analysis / experiment aggregation techniques [32]). The experiment investigated the effect of test-driven development (TDD) on software quality. For this, participants were split into two groups, where one group applied TDD and the other followed the Iterative-Test Last (ITL) process. The response variable is the "functional correctness" of the resulting software, measured as the percentage of passing test cases. Some key data-points and summary statistics[1] are shown in Table 1. They consist of three small-scale experiments at an industrial partner (F-Secure), prefixed with 'FS' in the table, and one larger student-based study at Universidad Politecnica de Madrid (denoted 'UPV' in the table).

The task of determining "the outcome" for any of the individual experiments is challenging. The usual approach is to undertake a meta-analysis [2, 15]. Figure 1 shows the forest plot of a random

[1]The summary statistics we have computed here are slightly larger than those published by Santos *et al.* [32] who use a different random effects model to ours, because we were unable to trace the correlation-between-groups data that would have been required to parameterise the Q-statistic estimator that they incorporated. Note, also the power calculations are based upon the *post hoc* effect sizes which are very likely upwardly biased [18].

Table 1: Key statistics for the multi-site testing experiments of Tosun *et al.* [34]

Experiment	Experience Scores				Results (per group)				Summary Results		
	Programming	Java	Unit	JUnit	Treatment	N	Mean	SD	Hedge's g	CI(95%)	Power
F-Secure H	3.67	2.33	2.17	2.17	ITL	6	30.71	36.58	0.25	[-0.89,1.39]	0.071
					TDD	6	40.23	33.43			
F-Secure K	2.91	1.82	1.64	1.27	ITL	11	22.17	20.44	0.44	[-0.40,1.29]	0.179
					TDD	11	35.42	35.40			
F-Secure O	3.29	2.71	2.71	2	ITL	7	16.05	20.81	1.85	[0.6,3.11]	0.93
					TDD	7	68.97	31.53			
UPV	2.36	1.88	1.04	1	ITL	31	33.38	39.79	1.34	[0.78,1.91]	0.999
					TDD	29	77.16	21.04			

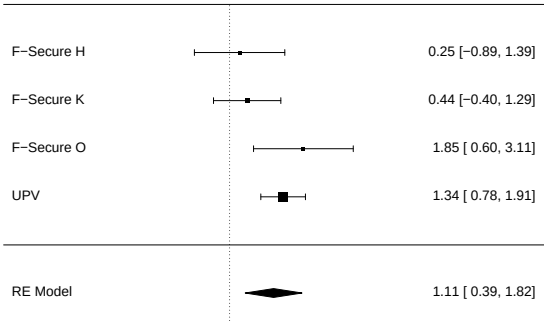


Figure 1: Forest plot of random-effects model for the multi-site testing experiments of Tosun *et al.* [34]

effects model. Some of the studies (F-Secure H and K) have small numbers of participants, and produce results with large standard deviations, which explains their low power (this is conveyed by the size of the centre point for each experiment. Note that that the pooled result is shown by the diamond and denoted "RE [random effects] model". Aside from this statistical uncertainty conveyed by the confidence intervals in the forest plot, there is also the uncertainty from potential moderators such that the appropriateness of the experimental design and analyses, and the experience of participants. When trying to aggregate the various results (e.g., via meta-analysis), the analyst also has to accommodate the heterogeneity of the experiments. As highlighted by Santos and Juristo [32], the results do not appear to be consistent, so the analyst has to determine whether this is because of differences in the experimental setting — for instance, the numbers of participants and their difference in experience — and has to adjust for this potential bias.

Ultimately, empirical results are difficult to interpret because they are beset by different degrees and types of uncertainty that can be difficult to capture. Currently, the task of interpreting and fusing-together uncertainties from different statistics is largely left to the intuition of the analyst. In this paper we try to make this process more transparent and explicit with the help of Subjective Logic.

3 BELIEF MODELLING, UNCERTAINTY, AND SUBJECTIVE LOGIC

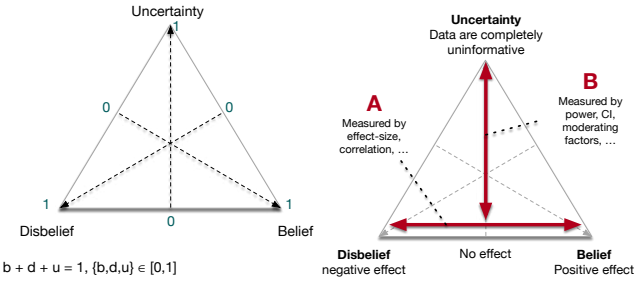
Over the past 50 years (spurred by the emergence of AI in the 60s and 70s) numerous probabilistic frameworks have been developed to reason about uncertain phenomena. These (in simplistic terms) seek to '*... combine the capacity of probability theory to handle likelihood with the capacity of binary logic to make inference from argument structures.*' [21]. These enable a modeller to link propositions ("beliefs") with probabilities. Popular approaches include Fuzzy Logic [37], Dempster Shafer theory [33], and Bayesian analysis techniques such as Bayesian Networks [27]. To motivate our choice of probabilistic logic, it is first necessary to provide a more detailed definition of 'uncertainty'.

Definition 3.1. By 'uncertainty', we refer to the situation where a lack of information or knowledge influences our ability to produce a definitive description or quantification of some proposition or belief. This lack of information can come in various forms [6], such as the inability to derive a firm result because of the intrinsic randomness or variability of some variable (*aleatoric* uncertainty), or because of the absolute absence or lack of information or data (*epistemic* uncertainty).

These sources of uncertainty (both types feature in our running example) are difficult to characterise and measure. It is impossible, for example, to be certain that we have considered all of the possible confounding factors in the TDD experiment, especially since we were not involved in designing or running the actual experiments). This leads us to *second-order uncertainty* [13] - our trust (or lack thereof) in our assessment of the various uncertainties.

Dempster Shafer theory [33] is particularly useful because it provides us with a basis for accommodating these different levels of uncertainty. In Dempster Shafer theory (as is the case in most conventional probabilistic logics), a proposition about some phenomenon (a "belief") is characterised as a probability (e.g., the belief that application of some testing methodology improves software correctness). However, instead of forcing us to pick a 'hard' probability (which is not possible if we consider the evidence from the above study), we can associate this with an explicit measure of *uncertainty* (e.g., we might consider the heterogeneity between studies, weighing up the expertise versus power, etc.).

In his work on Subjective Logic [21], which is derived from Dempster Shafer theory, Jøsang provides a helpful formalism within which to model and reason about beliefs. The building-blocks to



(a) Basic dimensions. (b) Mapping from experiments.

Figure 2: Barycentric triangles.

model basic beliefs are referred to as *subjective opinions*. In this paper we focus on *binomial* opinions, where the opinion revolves around the truth or falsehood of some x (as opposed to a multinomial opinion where a belief can have multiple dimensions – though these can also be readily represented in Subjective Logic [21]).

Definition 3.2. For a (binomial) subjective opinion, let $\mathbb{X} = \{x, \overline{x}\}$ be a binary domain with binomial random variable $X \in \mathbb{X}$. A subjective opinion about the truth of x is represented as a quadruplet: $\omega_x = (b_x, d_x, u_x, a_x)$:

- b_x: Belief mass in support of x being true.
- d_x: Disbelief mass in support of x being false.
- u_x: Uncertainty mass representing the 'vacuity of evidence'.
- a_x: Prior probability of x without any evidence. For the sake of simplicity, we will assume the convention that without any prior evidence, the prior probability $a_x = 0.5$.

For an opinion to be valid, $b_x + d_x + u_x = 1$.

There are several special types of opinions:

- $b_x = 1$ or $d_x = 1$: absolute opinion - equivalent to Boolean true or false,
- $u_x = 0$: dogmatic opinion – a traditional probability,
- $u_x = 1$: a vacuous opinion with no belief or disbelief.

A subjective opinion can be visualised as an equilateral barycentric triangle [20], where each of the vertices represents the maximum for belief, disbelief and uncertainty respectively. This coordinate system is illustrated in Figure 2(a).

Definition 3.3. Given an opinion ω_x, the *projected probability* $P(x)$ [21] can be defined as $P(x) = b_x + a_x u_x$.

Definition 3.4. The *beta-distribution* refers to a family of distributions that are a continuous version of the binomial distribution, which make them appropriate for modelling probabilities [8]. Its probability density function is defined by two 'shape' parameters α and β, and is described by:

$$f(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}, 0 < x < 1$$

where $\alpha > 0, \beta > 0$ and $B(\alpha, \beta)$ is the beta function.

Jøsang has shown how subjective opinions can be mapped to the α and β parameters [21], making it possible to interpret a single subjective opinion 'coordinate' as continuous distribution, where the 'density' represents the probability. A distribution with a well-defined peak indicates that the probability is highly concentrated around a particular point, whereas a flatter distribution indicates a higher uncertainty. Given some subjective opinion $\omega_x = (b_x, d_x, u_x, a_x)$ (where $W = 2$ by default): $\alpha = \frac{b_x * W}{u_x} + a_x * W$, and $\beta = \frac{d_x * W}{u_x} + (1 - a_x) * W$.

Subjective opinions can be combined (or separated) with various operators [21]. In this work we will use "belief fusion" operators. Their purpose is to 'fuse' evidence from different sources about a particular phenomenon, in order to *"produce an opinion that better reflects the collection of different opinions, or that is closer to the ground truth than each opinion in isolation"* [21]. For a group of subjective opinions, there are many different ways in which they could be combined, and choosing the wrong operator can yield counter-intuitive results (this was the basis for Zadeh's criticism of Dempster Shafer theory [38] which, at the time, had only a single Belief Fusion operator).

In this work we will be fusing opinions that represent multiple experimental results. To accommodate this we apply a Weighted Belief Fusion operator, originally proposed by Jøsang [21]. This operator produces a fused opinion that attributes weight to its constituent opinions based on their confidence; opinions with less uncertainty are given more weight. The version we use here is a binomial version of the multi-source (i.e., associative) adaptation of Jøsang's original binary operator by van der Heijden *et al.* [35], which allows us to accommodate more than two opinions.

Definition 3.5. The Weighted Belief Fusion for multiple sources: Let $\mathbb{A}$ be a finite set of actors and let $\omega_X^A = (b_X^A, d_X^A, u_X^A, a_X^A)$ denote the opinion held by $A \in \mathbb{A}$ over X. The weighted belief fusion of these opinions $\omega_X^{\hat{\diamond}\mathbb{A}} = (b_X^{\hat{\diamond}\mathbb{A}}, d_X^{\hat{\diamond}\mathbb{A}}, u_X^{\hat{\diamond}\mathbb{A}}, a_X^{\hat{\diamond}\mathbb{A}})$ is defined as below[2]:

$$b_X^{\hat{\diamond}\mathbb{A}} = \frac{\Sigma_{A\in\mathbb{A}} b_X^A(x)(1-u_X^A) \prod_{A'\in\mathbb{A}, A'\neq A} u_X^{A'}}{(\Sigma_{A\in\mathbb{A}} \prod_{A'\neq A} u_X^A) - |\mathbb{A}| \cdot \prod_{A\in\mathbb{A}} u_X^A}$$

$$u_X^{\hat{\diamond}\mathbb{A}} = \frac{(|\mathbb{A}| - \Sigma_{A\in\mathbb{A}} u_X^A) \cdot \prod_{A\in\mathbb{A}} u_X^A}{(\Sigma_{A\in\mathbb{A}} \prod_{A'\neq A} u_X^A) - |\mathbb{A}| \cdot \prod_{A\in\mathbb{A}} u_X^A}$$

$$d_X^{\hat{\diamond}\mathbb{A}} = 1 - (b_X^{\hat{\diamond}A} + u_X^{\hat{\diamond}A})$$

$$a_X^{\hat{\diamond}\mathbb{A}} = \frac{\Sigma_{A\in\mathbb{A}} a_X^A(x)(1-u_X^A)}{|\mathbb{A}| - \Sigma_{A\in\mathbb{A}} u_X^A}$$

4 ANALYSING EXPERIMENTAL RESULTS WITH SUBJECTIVE LOGIC

Let us recall our running example from Section 2.2. For every site of the multi-site study, the results (Table 1) indicate some positive effect of TDD versus ITL. However, each result is subject to various sources of uncertainty - wide confidence intervals, low power, and other moderating factors such as the experience of the candidates. It is important that any communication of these results and conclusions captures this associated uncertainty. The challenge we address in this paper is to develop a process by which to express the magnitude and extent of this uncertainty in a precise manner.

We believe that Subjective Logic presents a helpful framework within which to capture and reason about this uncertainty. Instead of interpreting data-points and their associated statistics at

[2]For space reasons we restrict ourselves to the situation where $(\forall A \in \mathbb{A} : u_X^A \neq 0) \wedge (\exists A \in \mathbb{A} : u_X^A \neq 1)$ - the other cases are also defined by van der Heijden *et al.* [35].

face-value, casting them as Subjective Opinions enables us to associate them with a degree of uncertainty. The Subjective Logic then enables us to aggregate and compare results in such a way that uncertainty becomes a primary factor. This section presents an approach to derive subjective opinions from an experiment.

In order to reason about experimental data in terms of Subjective Logic it is necessary to (1) decide which experimental outcomes to represent as subjective-opinions, and (2) to populate these opinions with appropriate belief, disbelief and uncertainty values. Given the diverse range of empirical settings and analyses, we endeavour to present an approach that is flexible and non-prescriptive. To offer a more concrete impression of how this might be applied in practice, we illustrate our approach with reference to TDD versus ITL study from Section 2.2.

4.1 Decomposing Experiments into Subjective Opinions

Every subjective opinion represents a belief in some proposition, with an associated measure of uncertainty. There are two types of subjective opinion; those that are computed as the result of some operation (e.g., belief-fusion), and those that are obtained directly from data (or human input). In order to apply Subjective Logic in an experimental setting, we start by identifying a subjective opinion for each research question.

It is important to highlight that this application of Subjective Logic is currently restricted to research questions that can be phrased or interpreted as 'yes or no' questions. The subjective opinions we refer to in this paper are binomial[3]. The point is that, in an empirical context, the answer to such questions is often uncertain, and subject to doubt (which is what motivates our use of Subjective Logic).

Each high level subjective opinion (corresponding to a research question) is obtained by combining ('fusing') several low-level subjective opinions, which represent the "information sources" from which the answer is to be derived. The granularity of these low-level subjective opinions is flexible. The experimenter might wish to wrap several data-analyses into a single subjective opinion, or might wish to create a more granular set of opinions where each represents a separate information source. This flexibility means that the approach is straightforward to extend to the scenario discussed in Section 4.2.1, where we are seeking to aggregate the results from *multiple* experiments. For such cases (as we shall demonstrate) the subjective opinions that are computed for each individual experiment can themselves be fused together to provide an aggregate subjective opinion.

4.2 Forming Subjective Opinions from Experimental Data

For each low-level source of information that feeds into an answer to a research question (e.g., the outcome of a statistical analysis), it is necessary to formulate a subjective opinion. In our setting, the meaning of an individual subjective opinion is illustrated in Figure 2(b): A high belief value should support a positive answer to the high-level research question. A high disbelief should support a negative answer. A high degree of uncertainty should reduce the extent to which the answer to the overall research question is supported in either direction.

The question of how to define the values of belief, disbelief, and uncertainty is highly dependent on the nature of the sources of information. They may be possible to derive entirely from statistical data; effect-sizes can modulate belief and disbelief, and measures such as Confidence Intervals or statistical power can modulate the uncertainty. However in some settings, for example for qualitative studies or for situations where the analyst is aware of threats to the validity of the data that do not manifest themselves in the statistics, it will be necessary to provide the subjective-opinion values by hand, ideally by consensus with other researchers.

Regardless of whether the opinion values are derived automatically or by hand, there is a specific two-phase process that can be adopted. For some source of information x we start by deriving a measure of uncertainty u_x about that source (between 0 and 1). For example, if the source of information is a statistic, we could map the confidence interval to a value between 0 (there is no interval at all) and 1 (the confidence interval is too large for the statistic to convey any useful information). The overriding constraint for subjective opinions is that the $b_x + d_x + u_x = 1$. Knowing u_x provides us with a fixed probability (or belief) mass $(1 - u_x)$ that can be split between b_x and d_x.

For this split, we start with some value e that represents the source of information in question (as an example, let it be some effect size statistic). A high value of e should correspond to a high belief and a low disbelief, and a low value should correspond to a low belief and a high disbelief. To map this to b_x and d_x we first scale e such that $0 \leq e \leq 1$. From this, b_x can be calculated as $b_x = e * (1 - u)$, and d_x can be calculated as $d_x = (1 - e) * (1 - u)$. In other words, if $e = 1$, b_x is at its highest possible value and d_x at its lowest, and vice versa if $e = 0$.

4.2.1 Illustration of Running Example. We illustrate the approach by applying it to the running example for the TDD study. For this the top-level subjective opinion represents our belief in the hypothesis that TDD outperforms ITL. We wish to aggregate the results for all of the experiments, so we capture the outcomes for each experiment as a single subjective opinion.

We start by, for each low-level subjective opinion, computing an uncertainty value. There are several possible approaches by which to derive an uncertainty value. One could provide a value based on intuition, e.g., by scaling a Likert-scale assessment to the interval between 0 and 1. Alternatively, one could use the statistical data in a more direct manner. For this example we have used a simple formula, provided in Appendix A. For the F-Secure H example, this gives us a value of 0.43 (leaving us with a remaining belief mass of 0.57 to divide between belief and disbelief.

To determine how this belief mass is divided up, we produce our value e from the confidence interval around the effect size, by computing its extent as a proportion of the maximum extent: $e = 1 - \frac{ci_{max} - ci_{min}}{(2.7 - -2.7)}$. A positive effect (which we limit at 2.7 [11]) indicates a strong affirmative answer to the research question. A negative effect size (which we limit at -2.7) indicates a negative answer. Given these limits, we calculate $e = \frac{\text{effect} - (-2.7)}{(2.7 - -2.7)}$. For our

[3]Subjective Logic does offer other types of opinions - Multinomial and Hyper opinions [21], which could in principle offer the basis for capturing results for more complex research questions, and this is an avenue we intend to explore in future work.

Table 2: Subjective opinions for all experiments in running example

Experiment	Belief	Disbelief	Uncertainty
F-Secure H	0.31	0.26	0.43
F-Secure K	0.32	0.24	0.45
F-Secure O	0.24	0.05	0.71
UPV	0.20	0.07	0.71

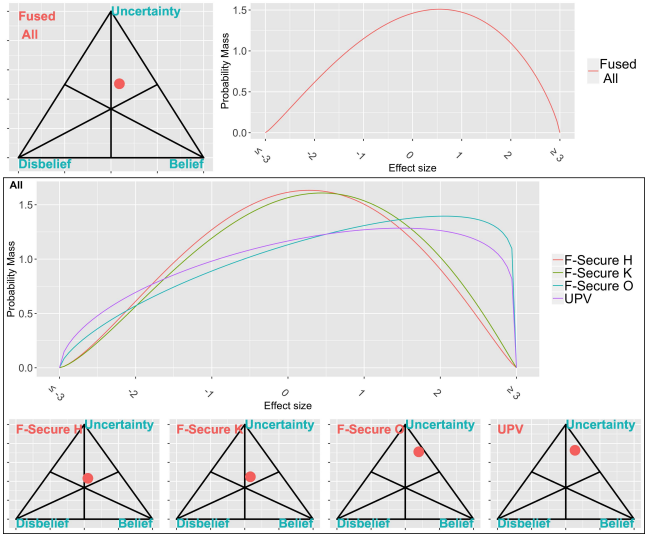


Figure 3: Fusion of (subjective opinions of) results from four experiments into a single fused subjective opinion.

F-Secure H study, the effect size is 0.25, which gives us an e value of 0.55.

Having computed e, we obtain b and d:

- $b = (0.55) * (1 - 0.43) = 0.31$
- $d = (1 - 0.55) * (1 - 0.43) = 0.26$

This results in a subjective opinion of ($b = 0.34, d = 0.28, u = 0.38$). The full set of subjective opinions for all of the experiments are shown in Table 2. The visualisations of the barycentric triangles and the corresponding beta distributions are shown in the bottom box (labelled 'All') in Figure 3. NB the beta distribution [0,1] is rescaled to [-2.7, 2.7] since this is our range of admissible effect sizes. These show a clear difference between the F-Secure H and K experiments, which have a moderate effect size but less uncertainty, and the F-Secure O and UPV experiments, which have a higher effect size but also more uncertainty. The higher uncertainty is reflected in the spread the distributions; higher certainty leads to a more discernible peak.

4.3 Fusing Experimental Results

When there are multiple results, either within a single experimental setting or from multiple replications of the same experiment, it is often helpful to be able to aggregate them [32]. For this we use the Weighted Belief Fusion operator [35] (see Definition 3.5), which attributes a greater weighting to opinions with less uncertainty: for a set of experiments $\mathbb{E}$ on some phenomenon X, where each experiment $E \in \mathbb{E}$ produces a subjective opinion ω_X^E, we compute the fusion $\omega_X^{\hat{\diamond}\mathbb{E}}$.

Although we choose this operator here, there are numerous alternative operators that could be considered, depending on the analyst's goals when it comes to combining or aggregating evidence. For example, the Averaging Fusion operator combines opinions without weighting them, or the Cumulative Fusion operator combines opinions in such a way that every additional piece of evidence will only ever increase the fused belief level [3, 21].

4.3.1 Illustration on Running Example. To illustrate the fusion, we combine the subjective opinions for all four experiments shown in Table 2. Here, the set of experiments $\mathbb{E} = \{\omega_{fsH}, \omega_{fsK}, \omega_{fsO}, UPV\}$, and X represents the belief that TDD outperforms ITL. The Weighted Belief Fusion $\omega_X^{\hat{\diamond}\mathbb{E}}$ gives the following results: $b_X^{\hat{\diamond}\mathbb{E}} = 0.29, d_X^{\hat{\diamond}\mathbb{E}} = 0.20, u_X^{\hat{\diamond}\mathbb{E}} = 0.51$.

This fused opinion and the corresponding beta distribution are shown in the top of Figure 3. The projected probability (Definition 3.3) for the fused opinion is 0.55, albeit with high level of uncertainty, which leads to the broad arc of the beta distribution, without a distinctive peak. It is worth highlighting that this subjective opinion (and corresponding beta distribution) are all we need to interpret the results. All of the data-points that convey information about the various sources of uncertainty - moderators, confidence intervals, etc., are incorporated into the uncertainty value.

5 CASE STUDY: A META ANALYSIS OF THE RELATIONSHIP BETWEEN PROGRAMMING LANGUAGES AND DEFECTS

For our case-study we re-examine data from two studies on the defect-proneness of different programming languages. We use data from a study by Nanz and Furia, using the Rosetta Code archive to compare failure rates (and other characteristics) of different program languages [26]. We supplement this with the results of a different study on programming language defects by Ray *et al.* [29] that was referred to by Nanz and Furia. The study by Ray *et al.* does not record execution failures, but instead records defects as bug-fixes mined from GitHub repositories.

This presents an interesting aggregation challenge. The studies use differing methodologies and collect different types of data from different sources. Whereas we access the raw data in the Nanz and Furia study, we only access the summary data in the Ray study. Both sources of data are subject to very different types of uncertainty.

For our analysis, we have made the full code and dataset available[4]. This not only contains the code used to implement the fusion operator and to visualise the results, but also provides a record of the procedures used to extract the belief, disbelief, and uncertainty values from the data.

5.1 Subjective Logic Analysis

For our analysis we start with the Nanz and Furia study, and create a subjective opinion for failure-proneness for each language. For

[4]https://figshare.com/s/5b26abc456f34664f415

Table 3: Table with summary data and computed subjective opinions for each language

Language	Failures	Executions	Timeouts	Implemented	Effect	CI lower	CI upper	Belief	Disbelief	Uncertainty
C	51	442	0.11	0.81	0.35	0.30	0.39	0.30	0.56	0.13
Python	215	751	0.10	0.92	0.56	0.53	0.60	0.52	0.40	0.08
Ruby	71	563	0.08	0.87	0.36	0.32	0.41	0.33	0.57	0.10
Fused weak								0.39	0.5	0.1
C sharp	18	307	0.20	0.58	0.25	0.19	0.30	0.19	0.57	0.24
F Sharp	24	236	0.09	0.46	0.33	0.26	0.39	0.24	0.50	0.25
Go	8	427	0.09	0.80	0.14	0.09	0.19	0.12	0.75	0.13
Haskell	25	426	0.12	0.68	0.25	0.20	0.29	0.20	0.62	0.18
Java	80	339	0.11	0.63	0.51	0.45	0.56	0.41	0.40	0.19
Fused strong								0.22	0.59	0.18

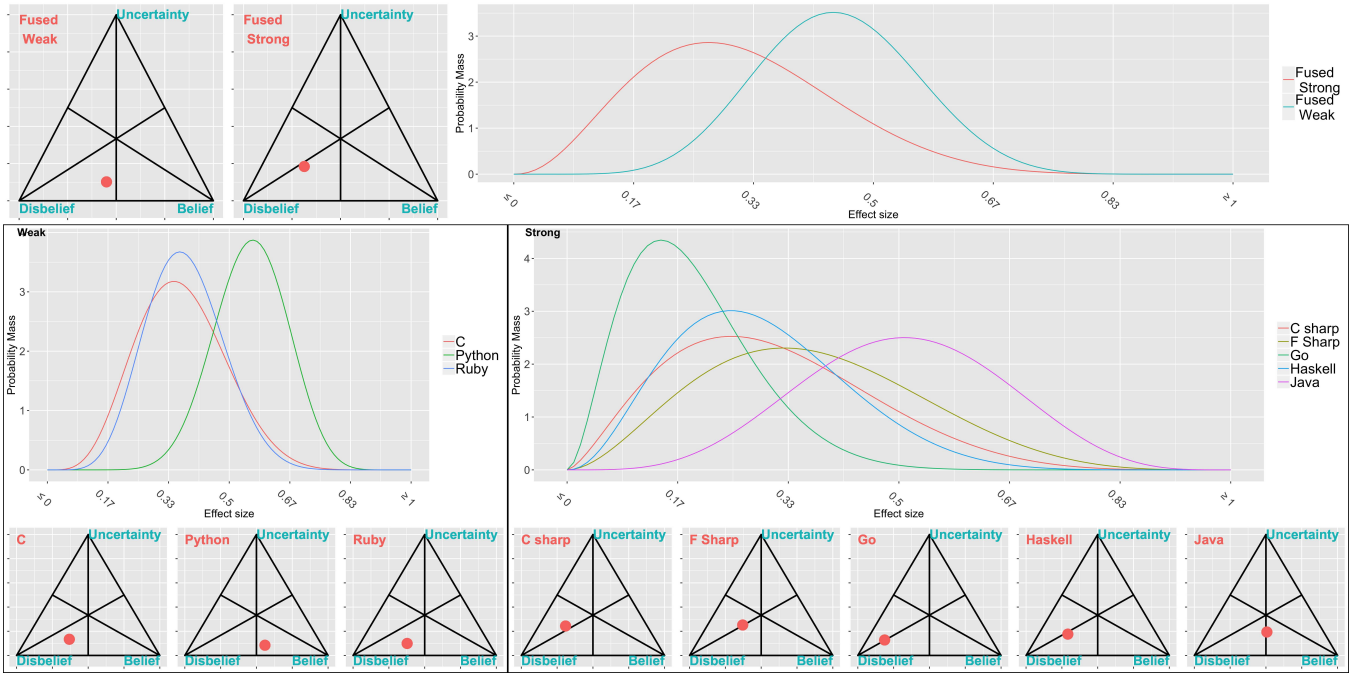


Figure 4: Visualisation of subjective opinions for failure data from study by Nanz and Furia, with fusion into strongly and weakly-typed languages.

this we use the size of the confidence intervals, the numbers of missing implementations (missing implementations correspond to an absence of data), and the number of timeouts (a timeout may not necessarily amount to a failure) to form an uncertainty score. The function that computes this from the data is in the data and code-pack for this paper. The results are shown in Table 3, and the corresponding barycentric triangles and Beta distributions are visualised in the bottom half of Figure 4. From this we note that, for all of the weakly-typed languages, the uncertainty scores are quite low, whereas for all of the strongly-typed languages apart from Go, there is a much higher degree of uncertainty (mainly because of lower availability of implementations and a higher number of time-outs).

We now repeat this process for an earlier study by Ray *et al.* [29]. This analysis spanned a selection of 728 GitHub projects, used commit messages to identify bug-fixes, and used these to derive statistics about the prevalence of defects, differentiated by the language in which they were written. For this analysis, the authors fitted a Negative Binomial Regression (NBR) model [16] to the data, such that each language was associated with a coefficient (in the range [-1,1]) and a p-value. Since the original study was carried out, Ray *et al.* followed-it up with a re-analysis of the data [30]. The study has since also been replicated in a more in-depth manner by Berger *et al.* [1].

Clearly there are fundamental differences between Nanz and Furia's, and the study by Ray *et al.*. The former counts defects in terms of observed execution failures, whereas the latter counts

Table 4: Summary results from NBR produced by Ray*et al.*

Label	Coeff.	Std.err.	Proj.	Bel.	Dis.	Unc.
C	0.11	0.04	220	0.36	0.23	0.40
C++	0.18	0.04	149	0.45	0.21	0.34
Clojure	-0.30	0.05	60	0.14	0.54	0.32
CoffeeScript	0.06	0.05	92	0.29	0.23	0.48
Erlang	-0.03	0.05	51	0.24	0.27	0.49
JavaScript	0.03	0.03	432	0.28	0.25	0.47
Objective-C	0.15	0.05	93	0.37	0.20	0.42
Perl	-0.12	0.08	106	0.21	0.34	0.45
PHP	0.10	0.05	109	0.33	0.22	0.46
Python	0.08	0.04	286	0.33	0.24	0.43
Ruby	-0.13	0.05	188	0.23	0.38	0.39
TypeScript	0.15	0.10	14	0.33	0.18	0.49
Fused weak				0.30	0.27	0.42
C sharp	-0.02	0.05	77	0.25	0.27	0.49
Go	-0.11	0.06	54	0.21	0.32	0.47
Haskell	-0.26	0.06	55	0.15	0.47	0.38
Java	-0.06	0.04	141	0.23	0.30	0.47
Scala	-0.24	0.05	55	0.16	0.45	0.39
Fused strong				0.25	0.35	0.40

defects in terms of recorded source-code fixes. The former concentrated on reasonably small, atomic programming tasks, whereas the latter concentrated on a wide range of systems of varying sizes. Nevertheless, their aims are broadly the same; to determine the "defect-proneness" of projects, and to differentiate between these in terms of the choice (or type) of language.

For the study by Ray *et al.* we only access their published statistics, not the raw data. This includes, for each language, a coefficient from the regression model indicating the relative prevalence of defects, a standard error, a statistical significance[5], and the number of projects examined for each language. The results are shown in Table 4[6].

Figure 5 provides an overview of the resulting opinions, fused in to strongly / weakly typed groups. One thing to note is that, since we only have access to summary statistics, the uncertainty is higher than for the Nanz and Furia study. The fused opinions at the top do however agree with Nanz and Furia (the projected probability for the fused strongly-typed opinions is 0.51 whereas the probability for weakly-typed opinions is 0.45). The difference is however even more marginal than with Nanz and Furia (0.06).

In both studies defects are represented differently (execution failures in Nanz and Furia's study versus code repairs in Ray *et al.*'s). In one we derive our values from raw data (proportions of executions that fail for each language) and in the other we obtain our effect sizes from the published summary statistic. Both studies are subject to different sources of uncertainty. By encoding both sets of results as subjective opinions, it becomes possible to analyse them alongside each other (accepting that the different effect-size scales are mapped to a single generic 'probability scale' [0, 1].

There two apparent options for fusing the results: (1) To fuse the individual subjective opinions from both experiments into fused opinions for strong and weak languages respectively, or (2) to fuse the fused opinions for strong languages from both experiments with each other, and to do the same for weak languages. We choose the latter because it enforces an equal weighting for each experiment (but different settings might suit alternative fusion arrangements). The results of the fusion are shown in Figure 6.

As would be expected, the fused opinions corroborate the findings from both studies that strongly typed languages tend to lead to fewer defects (i.e., execution failures or code fixes) than weakly-typed languages. However, the fused results also indicate that there is a slightly higher degree of uncertainty surrounding the results for strongly typed languages.

5.2 Discussion

The main strength of Subjective Logic is the flexibility offered by the use of subjective opinions. Once encoded as subjective opinions, the origin or type of the underlying data no longer matters, and they can be fused or compared against each other as required, whilst explicitly factoring in the associated (un-)certainties. In our analysis of the two fault studies, for example, we base our findings on completely different sources of information, subject to very different sources of uncertainty, yet still the results are summarised in a manner that makes them easy to interpret and compare.

A caveat is that the process of defining the uncertainty, belief, and disbelief values is inherently subjective. Different analysts might identify different sources of uncertainty, or interpret the underlying statistical analyses differently. As a consequence, it is important that the use of subjective logic to analyse empirical results is accompanied by a transparent description (or even source code) that is used to encode the subjective beliefs. For our case study, the results from both experiments were encoded into subjective opinions using fixed formulae (written in R), which are available as part of the data and source code accompanying the paper.

Throughout this paper, we have only used van der Heijden's Weighted Belief Fusion operator [35] (Definition 3.5). This is intended for situations where we wish to 'average' the underlying beliefs (whilst weighting them in terms of their confidence so that opinions with lower uncertainty are given more weight). This makes sense for our case study. From Figure 6, it is clear that the Nanz results offer (according to our interpretation) greater certainty than the Ray results, and should therefore be weighted accordingly during fusion.

It is important to emphasise that we are not restricted to this form of fusion in Subjective Logic. There are many alternative fusion operators [3, 21], which are better suited to different scenarios. It is also possible to use Addition or Subtraction operators [21] if a particular information source can only have a positive or respectively negative effect on the overall assessment.

[5]Note that Subjective Logic is agnostic about the value and problems associated with null hypothesis significance testing [5]. In this example we simply reason that a 'significant' p value has some evidential value against the null hypothesis.

[6]In their replication, Berger *et al.* [1] produced results that differed, in many cases lowering the effect and significance of the findings. We include the original results here because Berger's results have not yet (to our knowledge) been peer-reviewed, and Ray's results were what Nanz *et al.* were referring to. Of course, Berger's newer results could be incorporated in a similar fashion.

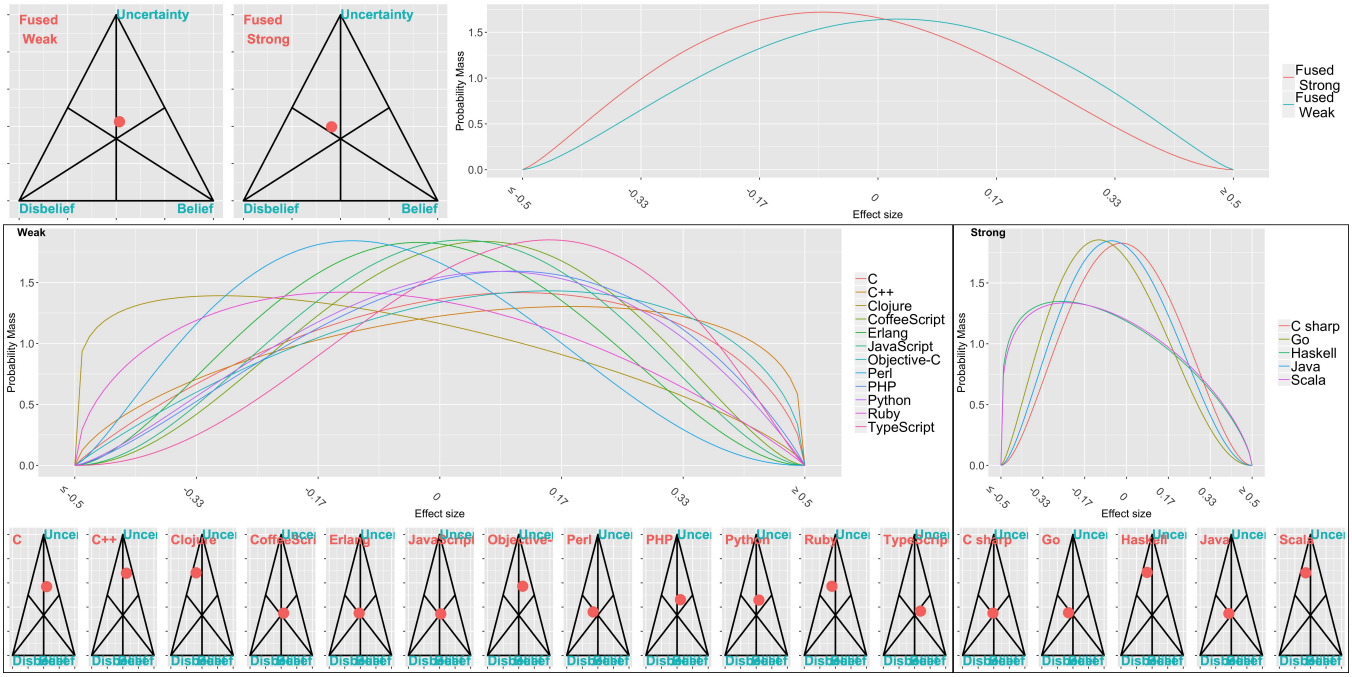


Figure 5: Visualisation of subjective opinions for study by Ray *et al.*

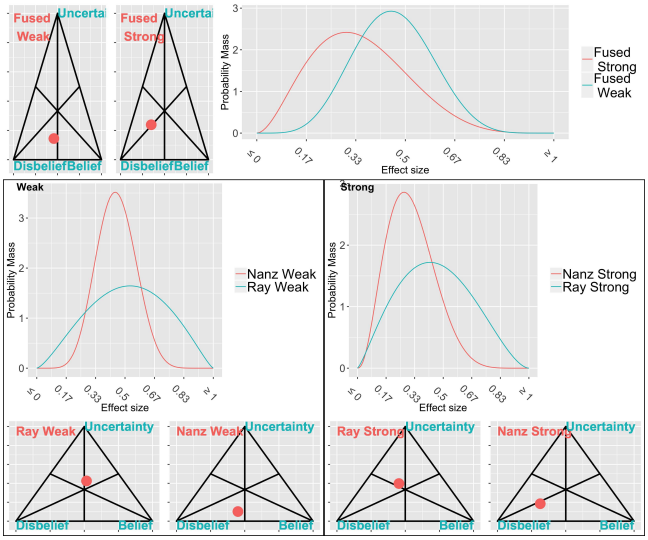


Figure 6: Fusion of all results from defect studies by Nanz and Furia and Ray *et al.*

6 CONCLUSIONS AND FUTURE WORK

As is the case with most disciplines, the task of deriving valid insights and inferences from empirical data (or groups of studies) can be challenging. Conclusions are invariably subject to caveats and threats to validity. Data can be missing, noisy, or misleading. Ultimately, any results from an empirical study are inherently *uncertain*, and any conclusions to be drawn from these results need to explicitly take this uncertainty into account.

We have shown how Subjective Logic can be used to explicitly incorporate this uncertainty into the process of reasoning about empirical data. This has been demonstrated with respect to a small running-example and a larger study of the relationship between programming languages and run-time failures. We believe this transparency in reasoning contributes to research progress. Additionally, we have shown how Subjective Logic can also be used at a Meta-analysis level to fuse together different results from multiple studies and multiple types of study.

Of course Subjective Logic is not a panacea to all research ills. Skill, judgement and insight are required. Researcher bias could creep in. However, these problems can be mitigated by the explicit nature of the reasoning. Other researchers are free to adopt different stances and reach different conclusions. We make progress through meaningful dialogue.

So far, our use of Subjective Logic has focused on the expression of results as subjective opinions and the use of fusion operators to combine empirical results. However, the extension of approaches such as Bayesian Networks to incorporate subjective opinions [21] makes it possible to apply other forms of reasoning – for example, working *backwards* from "aggregate" probabilities to identify the individual roles of different potential causes. Our future work will investigate how these more advanced Subjective Logic techniques can be used to reason about the various uncertainties that arise within Software Engineering empirical studies.

REFERENCES

[1] Emery Berger, Celeste Hollenbeck, Petr Maj, Olga Vitek, and Jan Vitek. 2019. On the Impact of Programming Languages on Code Quality. *arXiv preprint arXiv:1901.10220* (2019).

[2] Michael Borenstein, Larry V Hedges, Julian PT Higgins, and Hannah R Rothstein. 2011. *Introduction to meta-analysis.* John Wiley & Sons.

[3] Federico Cerutti, Lance M Kaplan, Timothy J Norman, Nir Oren, and Alice Toniolo. 2015. Subjective logic operators in trust assessment: an empirical study. *Information Systems Frontiers* 17, 4 (2015), 743–762.

[4] Jacob Cohen. 1988. Statistical power analysis for the behavioral sciences 2nd edn.

[5] David Colquhoun. 2014. An investigation of the false discovery rate and the misinterpretation of p-values. *Royal Society open science* 1, 3 (2014), 140216.

[6] Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? Does it matter? *Structural Safety* 31, 2 (2009), 105–112.

[7] Stefan Dietzel, Rens van der Heijden, Hendrik Decke, and Frank Kargl. 2014. A flexible, subjective logic-based framework for misbehavior detection in V2V networks. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014.* IEEE, 1–6.

[8] Lian Duan, Sanjai Rayadurgam, Mats Heimdahl, Oleg Sokolsky, and Insup Lee. 2016. Representation of confidence in assurance cases using the beta distribution. In *2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE).* IEEE, 86–93.

[9] Paul D Ellis. 2010. *The essential guide to effect sizes: Statistical power, meta-analysis, and the interpretation of research results.* Cambridge University Press.

[10] Robert Feldt, Thomas Zimmermann, Gunnar R Bergersen, Davide Falessi, Andreas Jedlitschka, Natalia Juristo, Jürgen Münch, Markku Oivo, Per Runeson, Martin Shepperd, et al. 2018. Four commentaries on the use of students and professionals in empirical software engineering experiments. *Empirical Software Engineering* 23, 6 (2018), 3801–3820.

[11] Christopher J Ferguson. 2009. An effect size primer: A guide for clinicians and researchers. *Professional Psychology: Research and Practice* 40, 5 (2009), 532.

[12] Carlo A Furia, Robert Feldt, and Richard Torkar. 2018. Bayesian Data Analysis in Empirical Software Engineering Research. *arXiv preprint arXiv:1811.05422* (2018).

[13] Peter Gärdenfors and Nils-Eric Sahlin. 1982. Unreliable probabilities, risk taking, and decision making. *Synthese* 53 (1982), 361–386.

[14] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. 2013. *Bayesian data analysis.* Chapman and Hall/CRC.

[15] Larry Hedges and Ingram Olkin. 1985. *Statistical Methods for Meta-Analysis.* Academic Press.

[16] Joseph M Hilbe. 2011. *Negative binomial regression.* Cambridge University Press.

[17] John PA Ioannidis. 2005. Why most published research findings are false. *PLoS medicine* 2, 8 (2005), e124.

[18] John PA Ioannidis. 2008. Why most discovered true associations are inflated. *Epidemiology* (2008), 640–648.

[19] Magne Jørgensen, Tore Dybå, Knut Liestøl, and Dag IK Sjøberg. 2016. Incorrect results in software engineering experiments: How to improve research practices. *Journal of Systems and Software* 116 (2016), 133–145.

[20] Audun Jøsang. 2001. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 9, 03 (2001), 279–311.

[21] Audun Jøsang. 2016. *Subjective logic.* Springer.

[22] Audun Jøsang and Viggo A Bondi. 2000. Legal reasoning with subjective logic. *Artificial Intelligence and Law* 8, 4 (2000), 289–315.

[23] Audun Jøsang and Robin Hankin. 2012. Interpretation and fusion of hyper opinions in subjective logic. In *2012 15th International Conference on Information Fusion.* IEEE, 1225–1232.

[24] Paul C Lambert, Alex J Sutton, Paul R Burton, Keith R Abrams, and David R Jones. 2005. How vague is vague? A simulation study of the impact of the use of vague prior distributions in MCMC using WinBUGS. *Statistics in medicine* 24, 15 (2005), 2401–2428.

[25] Sunil Nair, Neil Walkinshaw, Tim Kelly, and Jose Luis de la Vara. 2015. An evidential reasoning approach for assessing confidence in safety evidence. In *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE).* IEEE, 541–552.

[26] Sebastian Nanz and Carlo A Furia. 2015. A comparative study of programming languages in rosetta code. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 778–788.

[27] Judea Pearl. 2014. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Elsevier.

[28] Jennie Popay, Helen Roberts, Amanda Sowden, Mark Petticrew, Lisa Arai, Mark Rodgers, Nicky Britten, Katrina Roen, and Steven Duffy. 2006. Guidance on the conduct of narrative synthesis in systematic reviews. *A product from the ESRC methods programme Version* 1 (2006), b92.

[29] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. 2014. A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering.* ACM, 155–165.

[30] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. 2017. A large scale study of programming languages and code quality in github. *Commun. ACM* 60 (2017). Issue 10.

[31] Adrian Santos, Omar S Gómez, and Natalia Juristo. 2018. Analyzing Families of Experiments in SE: a Systematic Mapping Study. *IEEE Transactions on Software Engineering* (2018).

[32] Adrian Santos and Natalia Juristo. 2018. Comparing techniques for aggregating interrelated replications in software engineering. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement.* ACM, 8.

[33] Glenn Shafer. 1976. *A mathematical theory of evidence.* Vol. 42. Princeton university press.

[34] Ayse Tosun, Oscar Dieste, Davide Fucci, Sira Vegas, Burak Turhan, Hakan Erdogmus, Adrian Santos, Markku Oivo, Kimmo Toro, Janne Jarvinen, et al. 2017. An industry experiment on the effects of test-driven development on external quality and productivity. *Empirical Software Engineering* 22, 6 (2017), 2763–2805.

[35] Rens W Van Der Heijden, Henning Kopp, and Frank Kargl. 2018. Multi-source fusion operations in subjective logic. In *2018 21st International Conference on Information Fusion (FUSION).* IEEE.

[36] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering.* Springer Science & Business Media.

[37] Lotfi A Zadeh. 1965. Fuzzy sets. *Information and control* 8, 3 (1965), 338–353.

[38] Lotfi A Zadeh. 1984. Review of a mathematical theory of evidence. *AI magazine* 5, 3 (1984), 81.

A UNCERTAINTY FORMULA FOR THE RUNNING EXAMPLE

For this the summary data from the experiments (Table 1) provides us with the following factors:

- A set of expertise-ratings where each value is in the range $[0, 4]$ and is the average ordinal score submitted by participants to rate their ability to program, their experience with Java, their experience with unit testing, and their experience with JUnit.
- A confidence interval for the effect-size.
- The statistical power of the result [4].

We would consider the results to be at its maximum if the programmer's expertise is uniformly high (4), the CI is zero, and there is a statistical power of 1. To compute the uncertainty we first map each of these measures to a score between 0 and 1. $expertise = \frac{\text{prog+java+unit+}j\text{unit}}{(4+4+4+4)}$ is calculated as the sum of actual expertise scores divided by the maximum possible scores. $ci = 1 - \frac{ci_{\text{upper}} - ci_{\text{lower}}}{5.4}$ measures the CI as a proportion of 5.4 - this limit is based on Ferguson's heuristic [11] that a Hedges' g (which the CI is pertaining to) of 2.7 amounts to a "strong effect", so a confidence interval that spans this in both directions (+2.7 and -2.7) would amount to 5.4. We take pow to just be the unadulterated power value. If we apply this to the first experiment (F-Secure H) in Table 1, the result would be computed as follows:

- $expertise = \frac{3.67+2.33+2.17+2.17}{4+4+4+4} = \frac{10.34}{16} = 0.646$
- $ci = \frac{1.39-(-0.89)}{2.7-(-2.7)} = 0.42$
- $pow = 0.071$

Averaging these results in an uncertainty value of 0.38.