Enhancing Reasoning Approaches to Diagnose Functional and Non-Functional Errors

†‡**Nuno Cardoso** and †‡**Rui Abreu**
†Department of Informatics Engineering
Faculty of Engineering of University of Porto
Porto, Portugal
‡HASLab / INESC Tec
Campus de Gualtar
Braga, Portugal
nunopcardoso@gmail.com, rui@computer.org

Abstract

Most approaches to automatic software diagnosis abstract the system under analysis in terms of component activity and correct/incorrect behaviour (colectivelly known as spectra). While this binary error abstraction has been shown to be capable of diagnosing functional errors, when diagnosing non-functional errors it yields suboptimal accuracy. The main reason for this limitation is related to the lack of mechanisms for encoding error symptoms (such as performance degradation) in such a binary schema. In this paper, we propose a novel approach to diagnose both functional and non-functional errors by incorporating into the classic, bayesian reasoning approaches to error diagnosis concepts from the fuzzy logic domain. The empirical evaluation on 27000 synthetic scenarios demonstrates that the proposed fuzzy logic-based approach considerably improves the diagnostic accuracy (20% on average, with 99% statistical significance) when compared to the classic, state-of-the-art approach.

1 Introduction

Reasoning approaches to automated error diagnosis work by abstracting the run-time behavior of the system under analysis in terms of two general concepts: components and transactions. A component is an element of the system that, for diagnostic purposes, is considered to be atomic[1], whereas a transaction is a set of component activations that (1) share a common goal, and (2) the correctness of the output can be verified. Using such abstraction, existent reasoning-based diagnostic algorithms use a näive Bayes classifier to detect the patterns in the components' activity that most resemble the error behavior of the system [1; 2].

A limitation of such approaches is related to the assumption that any transaction behavior can be categorized in terms of correct/incorrect [1; 2; 3; 4]. While a binary error abstraction works well when diagnosing functional errors (i.e., the output value differs from the expected value), such abstraction is unable to accurately represent non-functional errors (e.g., performance degradation errors).

[1]In a software environment, a component can be for instance a statement, a function, a class, or a service.

The presence of non-functional errors in a system implies that the distinction between correct and incorrect states is often *fuzzy*, existing instead a gradual transition between such states. In such scenarios, it is often the case that a system does not break down recognizably but rather deteriorates over time [5]. Using a binary abstraction to the system correctness implies that the perceived deterioration of the system (i.e., the error symptoms) is completely overlooked by the diagnostic algorithm and, as a consequence, the diagnostic quality is negatively affected.

The challenge of solving this limitation is thereby twofold. First, it is necessary to define an appropriate method for both detecting and abstracting non-functional errors and the associated error symptoms. Second, it is necessary to integrate the additional knowledge in the diagnostic process. To overcome this limitation we propose an extension of existing error detection mechanisms (e.g., [3]) to detect/encode non-functional errors using fuzzy logic. Furthermore, we generalize over the classical reasoning-based diagnostic framework to take advantage of additional information encoded by the improved abstraction.

Results on 27000 synthetic test scenarios showed that our generalization improved the classical reasoning approach in 65% of the cases and achieved at least equal performance in 94% of the cases. The overall relative improvement in the diagnostic quality was of 20% on average, with a 99% confidence interval.

This paper makes the following contributions:

- We discuss the limitations imposed by the classical binary error abstraction;
- We propose a generalization of the classical reasoning-based diagnostic framework aimed at improving its accuracy when diagnosing non-functional errors;
- We compare the accuracies of the classical and our novel approaches using a simulation-based setup, which has been shown to be able to generate realistic scenarios.

This paper is organized as follows. In Section 2 we introduce the relevant details of the classical reasoning-based diagnostic framework. In Section 3 we motivate and present our approach. In Section 4 we discuss the results of the benchmark. Finally, in Sections 5 and 6, we present the related work and draw some conclusions about the paper.

2 Reasoning-based Diagnosis

In this section we introduce concepts and definitions used throughout the paper, as well as the reasoning-based ap-

proach to diagnosis.

Definition 1 (Diagnosing System)**.** *A diagnostic system DS is defined as the triple $DS = (SD, COMPS, OBS)$, where:*

- *SD is a propositional theory describing the behavior of the system*
- *$COMPS = \{c_1, \ldots, c_M\}$ is a set of components in SD*
- *OBS is a set of observable variables in SD*

Definition 2 (h-literal)**.** *An h-literal, h_j or $\neg h_j$ for $c_j \in COMPS$, denotes the component's health. Under an observation term obs over variables in OBS, a component is considered healthy if it performed nominally and unhealthy otherwise.*

Definition 3 (Diagnostic Candidate)**.** *Let $S_{\neg h} \subseteq COMPS$ be a set of unhealthy components and $S_h = COMPS \setminus S_{\neg h}$ the corresponding set of healthy components. We define $d(S_{\neg h})$ to be the conjunction*

$$(\bigwedge_{m \in S_{\neg h}} \neg h_m) \wedge (\bigwedge_{m \in S_h} h_m) \tag{1}$$

Given an observation term obs over variables in OBS, a diagnostic candidate for DS is a conjunction $d(S_{\neg h})$ such that $SD \wedge obs \wedge d_(S_{\neg h})$ is consistent.

In the remainder we refer to $d(S_{\neg h})$ simply as d, which we identify with the set $S_{\neg h}$ of indices of the negative literals.

A problem with the above definition is that the hypothesis that all the system is unhealthy always holds. To apply the concept of a diagnosis candidate to real systems with success, one must refine the definition so that the candidate contains the minimum number of components while still restoring consistency to $SD \wedge obs \wedge d$. Even though the minimality constraint is not strictly required, it is useful in most real-world problems to decrease the computational overhead of the diagnostic process.

Definition 4 (Minimal Diagnostic Candidate)**.** *A candidate d is minimal iff $\nexists d' : d' \subset d$ such that d' is a diagnostic candidate.*

Definition 5 (Diagnostic Report)**.** *A diagnosis $D = (d_1, \ldots, d_k, \ldots, d_K)$ is an ordered set of K diagnostic candidates, such that*

$$\forall d_k \in D : Pr(d_k|obs) \geq Pr(d_{k+1}|obs) \tag{2}$$

The calculation of a diagnostic report can be broadly divided in two sub-problems: diagnostic candidate generation and ranking.

The candidate generation problem is typically solved by using search algorithms [6; 7; 8; 9; 10] to produce candidates that, heuristically, have a higher chance of being correct.

In the remainder of this section we describe the relevant aspects of the classical reasoning-based diagnostic approach to address the ranking problem [1; 2]. We assume that a set of run-time observations have been collected using the so-called hit spectra abstraction [11].

Definition 6 (Hit Spectra)**.** *The hit spectra encodes the activity of each $c_j \in COMPS$ in transaction i in terms of hit/not hit as well as the outcome of each transaction in terms of pass/fail. Formally, let A_i be a set containing the components involved in transaction i. $A = \{A_1, \ldots, A_i, \ldots, A_N\}$ represents thereby the collection of the components' activity in each of the N transactions of the system. Additionally, let e denote the error vector, defined as*

$$e_i = \begin{cases} 0, & \textit{if transaction i performed nominally} \\ 1, & \textit{otherwise} \end{cases} \tag{3}$$

The hit spectra is composed of the pair (A, e).

Under a set of observations (A, e), the posterior probabilities are calculated according to naïve[2] Bayes rule as

$$\Pr(d \mid A, e) = \Pr(d) \cdot \prod_{i \in 1..N} \frac{\Pr(A_i, e_i \mid d)}{\Pr(A_i)} \tag{4}$$

The denominator $\Pr(A_i)$ is a normalizing term that is identical for all $d \in D$ and needs not to be calculated for ranking purposes as it does not alter the rank order.

$\Pr(d)$ estimates the probability that a candidate, without further evidence, is responsible for the system's malfunction. To define $\Pr(d)$, let p_j[3] denote the prior probability that a component c_j is at fault. Assuming that components fail independently, the prior probability for a particular candidate $d \in D$ is given by

$$\Pr(d) = \prod_{j \in d} p_j \cdot \prod_{j \in COMPS \setminus d} (1 - p_j) \tag{5}$$

By using equal values for all p_j it follows that the larger the candidate the smaller its *a priori* probability is.

To bias the prior probability taking run-time information (i.e., observations) into account, $\Pr(A_i, e_i \mid d)$ (referred to as likelihood) is defined as

$$\Pr(A_i, e_i \mid d) = \begin{cases} \mathrm{G}(d, A_i) & \text{if } e_i = 0 \\ 1 - \mathrm{G}(d, A_i) & \text{otherwise} \end{cases} \tag{6}$$

$\mathrm{G}(d, A_i)$ (referred to as transaction goodness) is used to account for the fact that components may fail intermittently, estimating the probability of nominal system behavior under an activation pattern A_i and a diagnostic candidate d.

Let g_j (referred to as component goodness) denote the probability that a component c_j performs nominally. Considering that all components must perform nominally to observe a nominal system behavior, $\mathrm{G}(d, A_i)$ is defined as

$$\mathrm{G}(d, A_i) = \prod_{j \in (d \cap A_i)} g_j \tag{7}$$

In scenarios where the values for g_j are not otherwise available, those values can be estimated by maximizing $\Pr(A, e \mid d)$ (Maximum Likelihood Estimation (MLE) for naive Bayes classifier) under parameters $\{g_j \mid j \in d \wedge 0 \leq g_j \leq 1\}$ [1].

3 Approach

In this section we discuss how non-functional errors (also referred to as fuzzy errors) can be more accurately detected/represented and how the diagnostic framework presented in Section 2 can be enhanced to more accurately diagnose such kind of errors.

[2] In order to maintain the problem tractable, conditional independence is assumed throughout the process.

[3] The value of p_j is application dependent. In the context of development-time fault localization it is often approximated as $p_j = 1/1000$, i.e., 1 fault for each 1000 lines of code [12].

Fuzzy Error Detection

The first challenge in diagnosing fuzzy errors is related to their detection. Existent approaches to error detection (e.g., [3]) make use of first-order logic descriptions of the correct behavior of the system (weak-fault models) to assign transactions to one of two possible sets: the pass set and the fail set (P and F respectively, where $F = \overline{P}$). A consequence of such fault models is the *crisp* distinction between correct and incorrect system states. While this crisp logic description enables an accurate representation of functional errors, it is unable to accurately represent a large variety of non-functional errors. Take for instance a type of non-functional error that, informally, can be described by the statement "The system is slow". Even though we can easily relate the slowness of the system to an appropriate metric (e.g., response time), it is not easy to define a crisp boundary in this same metric to distinguish acceptable and slow transactions. By setting a crisp boundary at, for instance, 1 second, a response time of 0.9999 seconds would be considered to be correct whereas a marginally superior response time would be considered incorrect. Also, a response time of 0.9999 seconds would result in the same type of error information (pass) as a smaller response time even though the larger response time may represent an error symptom.

To overcome the expressiveness limitation of crisp logic error detection mechanisms, we propose the generalization of such mechanisms using fuzzy logic [13]. Fuzzy logic extends the notion of binary set membership by introducing the concept of membership functions, denoted μ_A (membership function for set A), that map a particular domain on the real continuous interval $[0, 1]$, where the endpoints of 0 and 1 conform to no membership and full membership, respectively. In the context of error detection, the concept of fuzzy membership enables the representation of 3 types of system states: correct ($\mu_F(x) = 0$), incorrect ($\mu_F(x) = 1$) and degraded ($0 < \mu_F(x) < 1$). Since $F = \overline{P}$, and as a consequence of the new fuzzy error model, a degraded transaction exhibits both correct and incorrect behaviors simultaneously, however with different degrees.

As an example, consider the crisp fail set containing all response times (rt) above 1 second. This same set could be represented in terms of a membership function as

$$\mu_F(rt) = \begin{cases} 0 & , \ rt \leq 1 \\ 1 & , \ rt > 1 \end{cases} \tag{8}$$

To achieve the goal of representing non-functional errors (and implicitly the degraded state), consider that all response times below 0.5 seconds could be considered correct and all times above 1 second incorrect. Furthermore, consider that the amount of degradation follows a linear pattern between those two thresholds. The fuzzy fail set representing this particular type of error could be defined as

$$\mu_{\widetilde{F}}(rt) = \begin{cases} 0 & , \ rt < 0.5 \\ 2 \cdot rt - 1 & , \ 0.5 \leq rt \leq 1 \\ 1 & , \ rt > 1 \end{cases} \tag{9}$$

Both membership functions are presented in Figure 1.

To conclude the illustration of the fuzzy error detection process, consider the spectra presented in Table 1, which also contains the run-times for each transaction (marked in Figure 1). From this spectra we can see that, in particular for t_2, the crisp error vector neglected an error symptom

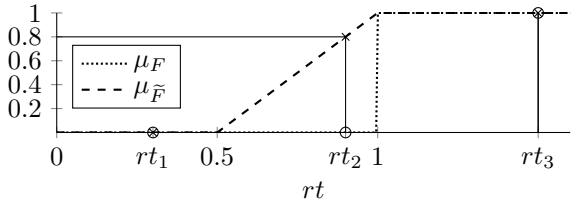

Figure 1: Crisp vs. fuzzy sets

i	rt_i	A_i	e_i	
			$\mu_F(rt_i)$	$\mu_{\widetilde{F}}(rt_i)$
1	0.3	$\{c_2\}$	0	0
2	0.9	$\{c_1\}$	0	0.8
3	1.5	$\{c_1, c_2\}$	1	1

Table 1: Fuzzy error hit spectra example

whereas the fuzzy error vector categorized that same transaction as being 80% degraded.

Fuzzy Error Diagnosis

Using fuzzy logic to detect errors, it is possible to assert that a particular transaction is 80% degraded (i.e., $\mu_{\widetilde{F}} = 0.8$ and consequently $\mu_{\widetilde{P}} = 0.2$). The remaining challenge consists in integrating this additional knowledge in the diagnostic process.

As an example consider again the spectra depicted in Table 1. Using the approach explained in Section 2 (i.e., using $e = \mu_F$), it follows that the candidates $d_1 = \{c_1\}$ and $d_2 = \{c_2\}$ are ranked equally. However, intuitively we would expect d_1 to be ranked ahead of d_2 since transaction t_2, in which component c_1 was involved, shows error symptoms whereas t_1 doesn't.

To solve this limitation we make use of the concept of probability of a fuzzy event [14]. The probability of a fuzzy event is defined as

$$\Pr(\alpha) = \sum_{x \in \Omega} \mu_x(\alpha) \cdot \Pr(x) \tag{10}$$

where α is an arbitrary event, and Ω is a set representing all the possible outcomes of α. Mapping this definition to the problem at hands, we generalize Equation 6 as

$$\Pr(A_i, e_i \mid d) = \overbrace{e_i \cdot (1 - \mathrm{G}(d, A_i))}^{x=F} + \overbrace{(1 - e_i) \cdot \mathrm{G}(d, A_i)}^{x=P} \tag{11}$$

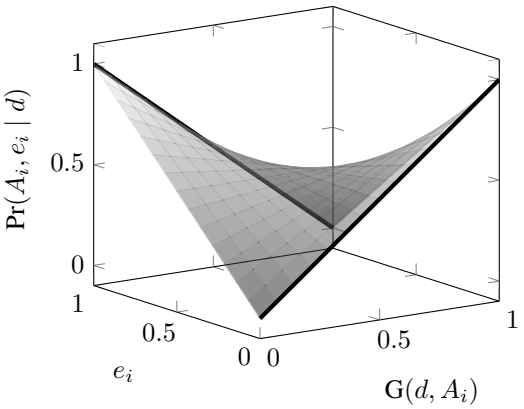

Figure 2: Likelihood function plot

where the first part of the equation ($x = F$) accounts for the incorrect behavior and the second part ($x = P$) for the correct behavior. In contrast to Equation 6, this generalization is valid for fuzzy error values (i.e., e = $\mu_{\widetilde{F}}$). Figure 2 shows the plot of the Equation 11 with respect to e_i and G(d, A_i). For comparison, we also plot Equation 6 with thick black lines.

Using the above generalization, the probabilities of the two candidates[4] are calculated as follows

$$\Pr(A, e \mid d_1) = \underbrace{(0.8 \cdot (1 - g_1) + (1 - 0.8) \cdot g_1)}_{t_2} \times \underbrace{(1 \cdot (1 - g_1) + (1 - 1) \cdot g_1)}_{t_3} \quad (12)$$

$$\Pr(A, e \mid d_2) = \underbrace{(0 \cdot (1 - g_2) + (1 - 0) \cdot g_2)}_{t_1} \times \underbrace{(1 \cdot (1 - g_2) + (1 - 1) \cdot g_2)}_{t_3} \quad (13)$$

By performing an MLE for both functions it follows that $\Pr(A, e \mid d_1)$ is maximized for $g_1 = 0$ and $\Pr(A, e \mid d_2)$ for $g_2 = 0.5$. Applying the maximizing values to both expressions, it follows that $\Pr(d_1 \mid A, e) \approx 8 \times 10^{-4}$ and $\Pr(d_2 \mid A, e) \approx 2.5 \times 10^{-4}$. Such probabilities entail the ranking $\langle d_1, d_2 \rangle$, which breaks the ambiguity between d_1 and d_2, thus improving the diagnostic accuracy.

4 Benchmark

In this section we describe our benchmark approach and discuss results.

Simulator

Performing benchmarks on real applications requires extensive adaptation and is therefore a very time consuming endeavor. Furthermore, the use of a limited set of applications limits the generalization of the conclusions taken from such observations.

To overcome such issues we make use of a simulator as proposed in [4] [5]. Such simulator provides functions to describe and execute a probabilistic model of an arbitrary system, thereby gathering the required spectra. The authors showed that the benchmark results for both real and synthetic data are comparable.

Concretely, the probabilistic model is created by defining a number of components, which are identified by their respective numeric IDs and a list of links to other components. Whenever a component is activated all the links belonging to that particular component are sequentially activated. A link is an abstraction to the components' interaction that contains a set of component IDs with their respective call probabilities. With the activation of a link, the current component and link list position are pushed onto a call stack and a component is randomly selected to continue the execution. At the end of the component's execution, an element is popped from the call stack, returning the control to the caller component. Using this model, a transaction can be generated by pushing a component marked as an entry point onto the call stack.

To emulate the error behavior, components may be injected with faults which are parameterized over 3 variables (p_c, p_d, and p_i) corresponding to the probabilities of correct, degraded, and incorrect behavior, respectively. During the simulation, whenever a faulty component is activated, the outcome of such activation (in terms of correct, degraded, or incorrect) is randomly determined using such probabilities.

To determine the transaction's fuzzy error value, we apply the following rules:

$\mu_{\widetilde{F}} = 1$, if at least one component performed erroneously;

$\mu_{\widetilde{F}} = 0$, if all components performed correctly;

$\mu_{\widetilde{F}} = rand(0, 1)$, otherwise[6].

Setup

To generate the spectra required for our benchmark we undergo a two-step process. In the first stage we randomly generate a set of system models while in the second we use such models to generate the required spectra.

We generate system models that comply with a N-tier service architecture. The system generation is parameterized by setting the minimum and maximum values for two different intervals: number of tiers (nt), and number of components per tier (nc). Systems are created by randomly selecting the number of tiers ($T \in nt$) as well as generating a list with the number of components in each tier ($C : C_i \in nc, 0 \leq i < T$). All components in each tier are connected to all the components of the next tier with equal probability. To exhibit erroneous behavior, a number of faults (nf) is randomly injected (in terms of position) in the systems.

For our benchmark setup, we generated 100 systems for each value $nf \in [2, 4]$, totaling 300 systems. The generation of such systems was done with both the number of levels and components per level comprised in the interval $[3, 10]$. The injected faults had 90% and 10% probabilities of degraded, and erroneous behavior, respectively.

The spectra generation is parameterized with a single variable ne, representing the number of errors at the end of which the simulation stops. For each generated system, we ran 10 simulations for each value $ne \in [1, 9]$, totaling 90 spectra per system. Overall, our benchmark is composed of $300 \times 90 = 27000$ test cases.

Metrics

The wasted effort metric evaluates how many components need to be inspected before all faulty components are found [15]. To calculate this metric one must undergo an iterative process. Starting with the first candidate, all of the candidate's components are inspected to determine whether or not that particular component was responsible for the erroneous behavior. Depending on the result of such inspection two outcomes may occur. On the one hand, if the component is found to be faulty, that particular component is removed from all other candidates in the ranking. On the other hand, if the component is found to be healthy, all candidates in the ranking containing that particular component are removed. This process is repeated until all faulty components

[4] The candidates for the fuzzy approach were calculated by setting a threshold for $\mu_{\widetilde{F}}$ to discretize transactions in terms of pass/fail. In this paper we use the threshold $\mu_{\widetilde{F}} = 1$.

[5] https://github.com/SERG-Delft/sfl-simulator

[6] This happens if no component performed erroneously, but at least one exhibited a degraded performance.

are found. In the case of the last inspected candidate being tied with other candidates, it is assumed that, on average, half of the healthy components are examined.

During this iterative process, we keep track of two counters: inspected components (I) and faulty components (C). Using these two counters, the wasted effort metric is calculated as

$$W = I - C \tag{14}$$

	d	Rank	I	C
1	$\{c_1, c_4\}$	1	2	1
2	~~$\{c_2, c_3, c_4\}$~~	2		
3	~~$\{c_3, c_4, c_5\}$~~	3		
4	~~$\{c_1, c_2\}$~~	4	4	2
5	$\{c_3, c_5\}$	4		

Table 2: Example diagnostic report

As an example consider the diagnostic report presented in Table 2 for which the correct diagnostic candidate is $d = \{c_1, c_2\}$. In order to calculate the wasted effort, we start by examining c_1 and c_4 finding that c_1 is faulty while c_4 is healthy. Due to c_4 being healthy, candidates d_2 and d_3 are not examined. Examining d_4 we observe that the only unexplored component (c_2) is faulty. Additionally, we see that both system's faults were discovered. However, as d_5 is tied with d_4, we must inspect half of the healthy components. The wasted effort of this diagnosis is therefore $W = 4 - 2 = 2$, meaning that 2 healthy components (c_4 and c_3/c_5) were examined in the process of finding the root cause of the system errors.

A normalized version of the wasted effort is called diagnostic quality and is defined as

$$Q = 1 - W/(M - C) \tag{15}$$

where M is the number of system components. The diagnostic quality value is contained between zero and one and estimates the fraction of system's healthy components that need to be examined before all faulty components are found.

In this paper we refine the diagnostic quality metric to take into account the fact that, for a specific spectra, not all components of the systems can be at fault. As an example consider a system with 1000 components with a spectra consisting of a single failing transaction activating 2 components. Assuming the diagnostic algorithm only proposes plausible[7] candidates, the quality is contained in the interval between 1 and $\frac{999}{1000}$. Instead of calculating the diagnostic quality using the M components of the system, we use M_s, the number of "suspicious" components to calculate the new metric, which shall be referred to as "fair quality" (Q_f). A component is said to be suspicious if it was activated in a failing transaction. A consequence of using Q_f is that the diagnostic qualities of all possible permutations of the ranking always have a lower bound quality of 0.

Results

In this section, we compare the performance of the crisp diagnostic approach, presented in Section 2, with our fuzzy approach for the generated spectra.

In Figure 3, we compare the average Q_f for each test scenario. From the analysis of the plot we can see that the crisp approach is always (on average) outperformed by the fuzzy approach. This is due to the fact that the fuzzy approach is able to successfully take advantage of the extra fuzzy error information to break the ties in the ranking (as shown in the example from Table 1) that occur when dealing with small numbers of erroneous transactions.

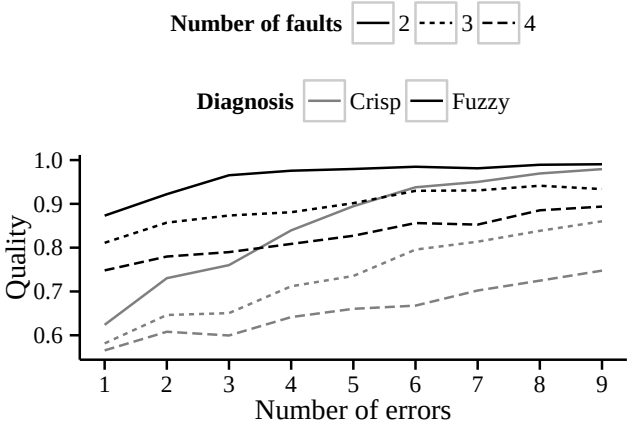

Figure 3: Benchmark Averages

A more detailed analysis of the data (Figure 4) shows that our approach outperformed the crisp approach in 65% of the test cases. Moreover, in 94% of the cases our approach was at least as accurate as the classical approach. In the remaining 6% of the test cases the accuracy loss was due to (1) lack of observations, and (2) marginal variations in the posteriori probability, still enough to make the relative ranking change. The overall average improvement of quality introduced by our algorithm was of $\Delta Q_f = 0.153$, representing a relative improvement of 21%. By performing a paired one-tailed T-test, we can ascertain that our approach introduced a relative improvement of 20%, with a 99% confidence interval.

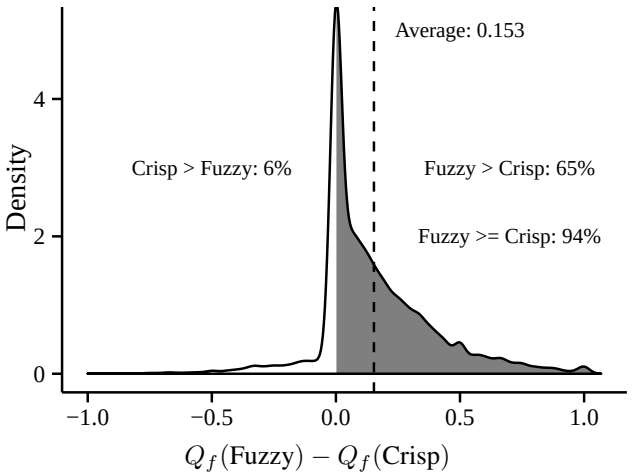

Figure 4: Quality improvement density plot

In Figure 5, we present a set of boxplots[8] comparing the

[7] By plausible we mean that all the candidate's components were at least activated once in an erroneous transaction.

[8] For each test scenario, the box corresponds to 2^{nd} and 3^{rd} quartiles (i.e., 50% of the cases), the vertical lines correspond to the 1^{st} and 4^{th} quartiles, and the small dashes correspond to test cases categorized as outliers. A test case is considered to be an outlier if its distance from the box is greater that $1.5 * IQR$ (interquartile range, i.e., the height of the box).

quality distributions of both approaches for each test scenario. From the analysis of the plots we can see that not only the fuzzy approach has a better performance than the crisp approach, but also that the fuzzy approach distribution is much more skewed towards better quality results than the crisp approach. Additionally, we can see that the fuzzy approach exhibits a higher consistency (i.e., smaller interquartile range) than the crisp approach.

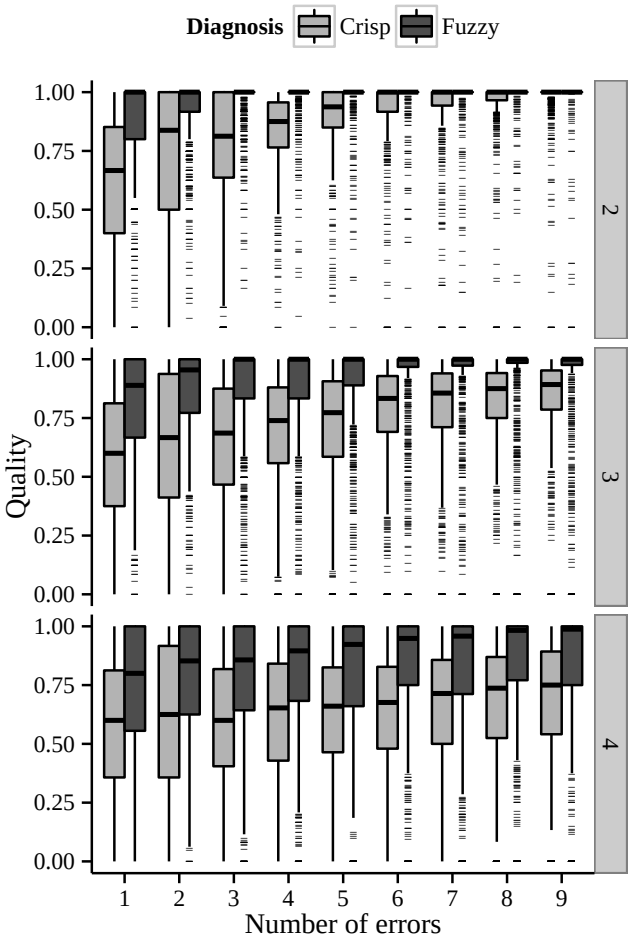

Figure 5: Benchmark Boxplots

A final remark is that, with the increase of erroneous transactions, it appears that the crisp approach quality seems to converge towards to the same average quality as the fuzzy approach. This happens due to the fact that the information introduced by the occurrence of errors eventually compensates the limitations imposed by the crisp error abstraction.

5 Related Work

In addition to the approach presented in Section 2, there is a wide set of different approaches to error diagnosis.

In the scope of lightweight fault localization techniques, we can highlight examples such as Pinpoint [16], and Tarantula [17], Ochiai [18]. While extremely efficient, such approaches do not consider multiple faults.

In the scope of run-time diagnosis, Kahuna [19] is an approach that aims at diagnosing performance problems in Map-Reduce systems through the usage of peer similarity. In [20], the authors apply the same concept to the diagnosis of failures in distributed file systems, such as PVFS or Lustre.

In the scope of candidate generation, [6], [7], [8], [9; 21], [22], and [10] propose different approaches to compute the minimal diagnostic candidates.

In the scope of candidate ranking, [1; 23; 24], [2], and [25] propose different methods for estimating the g_j parameters.

As opposed to our fuzzy approach, such diagnostic approaches are not able to appropriately encode and effectively diagnose non-functional errors.

6 Conclusions

We presented a generalization to the classical reasoning-based diagnostic approach that not only guarantees equal diagnostic quality when diagnosing functional errors but also improves the diagnostic quality when diagnosing non-functional errors.

The conducted synthetic benchmark showed that, for our setup with 27000 test cases, our approach improved the diagnostic quality in 65% of the cases and performed at least as good as the classical approach in 94% of the test cases. On average, the relative improvement introduced by our approach was of 20%, with a 99% confidence interval.

Future work includes the extension of existent error detection frameworks to include the fuzzy error abstraction proposed in this paper. The existence of such a framework would enable a real-world validation of the proposed approach.

Acknowledgements

We would like thank Lígia Massena, André Silva, and Alexandre Perez for the useful discussions about this work. This material is based upon work supported by the National Science Foundation under Grant No. CNS 1116848, by the scholarship number SFRH/BD/79368/2011 from Fundação para a Ciência e Tecnologia (FCT), and by the ERDF through the Programme COMPETE, the Portuguese Government through FCT - Foundation for Science and Technology, project reference FCOMP-01-0124-FEDER-020484.

References

[1] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. A new bayesian approach to multiple intermittent fault diagnosis. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, pages 653–658, 2009.

[2] Johan de Kleer. Diagnosing multiple persistent and intermittent faults. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, pages 733–738, 2009.

[3] Paulo Casanova, David Garlan, Bradley Schmerl, and Rui Abreu. Diagnosing architectural run-time failures. In *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS'13, pages 103–112, 2013.

[4] Cuiting Chen, Hans-Gerhard Gross, and Andy Zaidman. Improving service diagnosis through increased monitoring granularity. In *Proceedings of the 7th International Conference on Software Security and Reliability*, SERE'13, 2013.

[5] Debanjan Ghosh, Raj Sharman, H. Raghav Rao, and Shambhu Upadhyaya. Self-healing systems - survey and synthesis. *Decis. Support Syst.*, 42(4):2164–2185, 2007.

[6] Johan de Kleer and Brian C. Williams. Readings in model-based diagnosis. In *Readings in model-based diagnosis*, chapter Diagnosing multiple faults, pages 100–117. 1992.

[7] Franz Wotawa. A variant of Reiter's hitting-set algorithm. *Information Processing Letters*, 79(1):45–51, 2001.

[8] Alexander Feldman, Gregory Provan, and Arjan J. C. van Gemund. Computing minimal diagnoses by greedy stochastic search. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, pages 911–918, 2008.

[9] Rui Abreu and Arjan J. C. van Gemund. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In *Proceedings of the 8th Symposium on Abstraction, Reformulation, and Approximation*, SARA'09, 2009.

[10] Nuno Cardoso and Rui Abreu. MHS^2: A map-reduce heuristic-driven minimal hitting set search algorithm. In *Proceedings of the International Conference on Multicore Software Engineering, Performance, and Tools*, MUSEPAT'13, 2013.

[11] Mary Jean Harrold, Gregg Rothermel, Rui Wu, and Liu Yi. An empirical investigation of program spectra. In *Proceedings of the 1998 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering*, PASTE'98, pages 83–90, 1998.

[12] John Carey, Neil Gross, Marcia Stepanek, and Otis Port. Software hell. In *Business Week*, pages 391–411, 1999.

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

[14] Lotfi Asker Zadeh. Probability measures of fuzzy events. *Journal of mathematical analysis and applications*, 23(2):421–427, 1968.

[15] Friedrich Steimann, Marcus Frenkel, and Rui Abreu. Threats to the validity and value of empirical assessments of the accuracy of coverage-based fault locators. In *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, ISSTA'2013, pages 314–324, 2013.

[16] Mike Y. Chen, Emre Kiciman, Eugene Fratkin, Armando Fox, O Fox, and Eric Brewer. Pinpoint: Problem determination in large, dynamic internet services. In *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, DSN 2002, pages 595–604, 2002.

[17] James A. Jones and Mary Jean Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, ASE '05, pages 273–282, 2005.

[18] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. On the accuracy of spectrum-based fault localization. In *Testing: Academic and Industrial Conference Practice and Research Techniques*, TAICPART'07, pages 89–98, 2007.

[19] Jiaqi Tan, Xinghao Pan, Eugene Marinelli, Soila Kavulya, Rajeev Gandhi, and Priya Narasimhan. Kahuna: Problem diagnosis for mapreduce-based cloud computing environments. In *Proceedings of the 12th Network Operations and Management Symposium*, NOMS, pages 112–119, 2010.

[20] Michael P. Kasick, Jiaqi Tan, Rajeev Gandhi, and Priya Narasimhan. Black-box problem diagnosis in parallel file systems. In *Proceedings of the 8th Conference on File and Storage Technologies*, FAST, pages 43–56, 2010.

[21] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. Diagnosing multiple intermittent failures using maximum likelihood estimation. *Artificial Intelligence*, 174(18):1481–1497, 2010.

[22] Roni Tzvi Stern, Meir Kalech, Alexander Feldman, and Gregory M Provan. Exploring the duality in conflict-directed model-based diagnosis. In *Proceedings of the 26st National Conference on Artificial Intelligence*, AAAI'12, pages 548–555, 2012.

[23] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. A dynamic modeling approach to software multiple-fault localization. In *Proceedings of the 19th International Workshop on Principles of Diagnosis*, DX'08, pages 7–14, 2008.

[24] Rui Abreu, Wolfgang Mayer, Markus Stumptner, and Arjan J. C. van Gemund. Refining spectrum-based fault localization rankings. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC'09, pages 409–414, 2009.

[25] Nuno Cardoso and Rui Abreu. A kernel density estimate-based approach to component goodness modeling. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, AAAI'13, 2013.