Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

# Leveraging Qualitative Reasoning to Improve SFL

**Alexandre Perez**[1,2], **Rui Abreu**[3,4]
[1] University of Porto, Portugal
[2] HASLab, INESC-TEC
[3] IST, University of Lisbon, Portugal
[4] INESC-ID
alexandre.perez@fe.up.pt, rui@computer.org

## Abstract

Spectrum-based fault localization (SFL) correlates a system's components with observed failures. By reasoning about coverage, SFL allows for a lightweight way of pinpointing faults. This abstraction comes at the cost of missing certain faults, such as errors of omission, and failing to provide enough contextual information to explain why components are considered suspicious. We propose an approach, named Q-SFL, that leverages qualitative reasoning to augment the information made available to SFL techniques. It qualitatively partitions system components, and treats each qualitative state as a new SFL component to be used when diagnosing. Our empirical evaluation shows that augmenting SFL with qualitative components can improve diagnostic accuracy in 54% of the considered real-world subjects.

*In Memoriam:* Danny Bobrow.

## 1 Introduction

SFL [Jones *et al.*, 2002] was shown to be a lightweight, yet effective, technique for locating faults in a system. It consists of keeping a record of which components are involved in each system execution and subsequently ranking those components according to their similarity to failing executions. The intuition being that a faulty component is very likely to be involved in failing executions and not as likely to be covered in nominal ones. Over the years, many extensions were proposed to improve SFL's applicability and effectiveness, such as exploring which similarity coefficients yield better fault localization results [Lucia *et al.*, 2014] and handling multiple or intermittent faults [Abreu *et al.*, 2009b].

Despite the developments and achievements in SFL research, we are unable to find many accounts of successful transitions of this technology into the industry at large. We argue that this is motivated largely by the issues raised by Parnin and Orso in their 2011 user study of automated debugging techniques [Parnin and Orso, 2011; Ang *et al.*, 2017]. Namely, the authors found that there is significant interest drop-off after users inspect a small number of components from the ranked list of potential faults. This issue is exacerbated as the scale of the system increases. Another issue pointed out by Parnin and Orso is the fact that many SFL studies assume *perfect fault understanding* — that is, these studies expect that once users inspect a faulty component, they will correctly identify it as such —, which does not always hold in practice [Gouveia *et al.*, 2013].

This paper proposes an approach that inspects the state of system components, with the intent of augmenting reports generated by SFL techniques and hence providing more diagnostic information. Recording the state of individual components in each execution quickly becomes intractable, even for a lightweight approach as SFL. Therefore, we leverage Qualitative Reasoning (QR), which provides a way of describing a set of values by their discrete, behavioral qualities, to enable the reasoning about a system's behavior without exact quantitative information [Forbus, 1997; Williams and de Kleer, 1991]. Precise numerical quantities are avoided and replaced by qualitative descriptions — such as, for instance: high, low, zero, increasing or decreasing.

We apply QR to the SFL analysis, in an approach named Q-SFL, enabling the introduction of quantitative landmarks that will partition the domains of relevant components into a set of qualitative descriptions, and insert new SFL components for each of these descriptions. As behavioral qualities are now considered as components, their involvement in system executions is therefore recorded and ranked according to their similarity to observed failures, *enriching* the SFL report as a result. This can have benefits in *fault comprehension* — because qualitative properties are considered besides merely recording involvement — and even improve diagnostic report accuracy — whenever a qualitative state is more correlated with failing behavior than its enclosing system component.

We perform an empirical evaluation of Q-SFL with real-world faults from the Defects4J [Just *et al.*, 2014] catalog of faulty software programs. Results show that Q-SFL has the potential to improve the accuracy of SFL reports —with 54% of considered subjects exhibiting a lower effort to diagnose faults. Although the results are promising, we discuss several matters that need further research —namely, uncovering a landmarking strategy that exhibits consistently better results and studying to what extent *fault comprehension* is improved.

This paper's contributions are:

- An approach, named Q-SFL, inspired by qualitative rea-

soning (QR) research, to augment program spectra used by SFL techniques by partitioning system components into a set of qualitative states which are treated as SFL components.

- Empirical evidence that QR-enhanced spectra *can* reduce the effort to diagnose real software bugs in 54% of considered subjects.

## 2 Background

This section briefly summarizes the concepts upon which our approach is based on.

### 2.1 Spectrum-based Fault Localization (SFL)

In SFL, the following is given:

- A finite set $\mathcal{C} = \{c_1, ..., c_M\}$ of $M$ system components;
- A finite set $\mathcal{T} = \{t_1, ..., t_N\}$ of $N$ transactions, which can be seen as records of a system execution;
- An error vector $e = \{e_1, ..., e_N\}$ of transaction outcomes, where $e_i = 1$ if $t_i$ failed, $e_i = 0$ otherwise;
- A $M \times N$ activity matrix $\mathcal{A}$, where $\mathcal{A}_{ij} = 1$ if component $c_j$ is involved in transaction $t_i$, $\mathcal{A}_{ij} = 0$ otherwise.

The goal of SFL is to pinpoint which (sets of) components are more likely to have caused the system to fail. Earlier approaches to SFL measure the similarity between a component's involvement in transactions and the error vector [Jones *et al.*, 2002; Lucia *et al.*, 2014]. Later on, spectrum-based reasoning (SR) was introduced [Abreu *et al.*, 2009b], leveraging a Bayesian reasoning framework to diagnose, even when multiple, intermittent faults are present. The two main steps of SR are candidate generation and candidate ranking:

**Candidate Generation**

The first step in SR is to generate a set $\mathcal{D} = \{d_1, ..., d_k\}$ of diagnosis candidates. A diagnosis candidate $d_k \subseteq C$ is valid if every failed transaction involved at least one component from $d_k$. Candidate $d_k$ is *minimal* if no other valid candidate is contained in $d_k$. We are only interested in minimal candidates, as they can subsume all others. Heuristic approaches to finding these *minimal hitting sets* include STACCATO [Abreu and van Gemund, 2009], SAFARI [Feldman *et al.*, 2008] and MHS$^2$ [Cardoso and Abreu, 2013b].

**Candidate Ranking**

For each candidate $d_k$, their fault probability is calculated using the Naïve Bayes rule[1] [Abreu *et al.*, 2009a]

$$\Pr(d_k \mid (\mathcal{A}, e)) = \Pr(d_k) \cdot \prod_{i \in 1..N} \frac{\Pr((\mathcal{A}_i, e_i) \mid d_k)}{\Pr(\mathcal{A}_i)} \quad (1)$$

Let $\mathcal{A}_i$ be short for $\{\mathcal{A}_{ij} | 1 \in 1..M\}$, representing component activity in $i^{\text{th}}$ transaction. $\Pr(\mathcal{A}_i)$ is a normalizing term that is identical for all candidates. Let $p_l$ denote the prior probability[2] that a component $c_l$ is at fault. The prior probability for a candidate $d_k$ is given by

$$\Pr(d_k) = \prod_{l \in d_k} p_l \cdot \prod_{l \in C \setminus d_k} (1 - p_l) \quad (2)$$

$\Pr(\mathcal{A}_i, e_i \mid d_k)$ is used to bias the prior probability taking observations into account. Let $g_l$ (referred to as component goodness) denote the probability that $c_l$ performs nominally

$$\Pr((\mathcal{A}_i, e_i) \mid d_k) = \begin{cases} \prod_{l \in (d_k \cap \mathcal{A}_i)} g_l & \text{if } e_i = 0 \\ 1 - \prod_{l \in (d_k \cap \mathcal{A}_i)} g_l & \text{otherwise} \end{cases} \quad (3)$$

In cases where values for $g_l$ are not available, they can be estimated by maximizing $\Pr((\mathcal{A}, e) \mid d_k)$ under parameters $\{g_l \mid l \in d_k\}$ [Abreu *et al.*, 2009a].

In order to measure the accuracy of SFL approaches, the cost of diagnosis ($C_d$) metric is often used [Parnin and Orso, 2011]. It measures the number of candidates to be inspected until the real fault is reached, assuming candidates are inspected by descending order of probability. A $C_d$ of 0 indicates an ideal diagnosis where the real fault is at the top of the ranked list of candidates.

### 2.2 Qualitative Reasoning (QR)

QR creates a discrete representation of the continuous world [Forbus, 1997; Williams and de Kleer, 1991; de Kleer, 1977], enabling the reasoning of space, time, and quantity with merely a small amount of information. It is motivated by the fact that humans are able to draw conclusions about the physical world around them with limited information, without the need of solving complex differential equations.

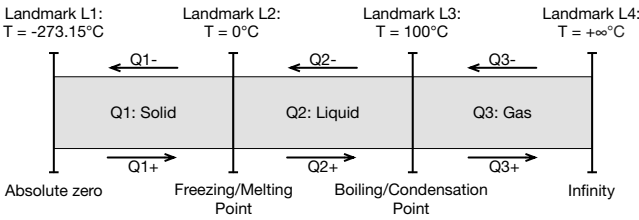


Figure 1: Example of a possible qualitative discretization of water temperature.

Figure 1 provides an example of a potential discretization of the water temperature into three qualitative values: Q1, Q2 and Q3. Our representation *resolution* — the granularity of the information detail — coincides with that of the three physical states of matter that water can assume: solid, liquid, and gas. Note that the established resolution will ultimately define the granularity of the conclusions one can draw from QR. To define the *qualitative states*, one needs to establish *landmarks*. Landmarks are constant quantitative values that establish a point of comparison [Kuipers, 1986]. In this example, we know that if the water is in the liquid state (Q2), then its temperature is somewhere between landmark L2 — corresponding to 0°C, the freezing point of water — and landmark L3 — 100°C, its boiling point. Similarly,

[1] Probabilities are calculated assuming conditional independence throughout the process.

[2] In the case of software diagnosis, one can approximate $p_l$ as $1/1000$, *i.e.*, 1 fault for each 1000 lines of code [Carey *et al.*, 1999].

we can derive that ice (Q1) temperature assumes a value between the absolute zero (L1) and the melting point (L2); and that water vapor (Q3) ranges between the condensation point (L3) and positive infinity (L4).

QR also supports the representation of *derivatives* between two quantities. They are usually represented with '+' and '-' signs, denoting value increases and decreases, respectively. This enables the use of sign algebra to reason about *direct influence* and *proportionality* between two qualitative values [Kuipers, 1986]. Derivatives also enable *envisionments*. An *envisionment* establishes a set of transitions between qualitative states [de Kleer, 1977], essentially modeling the abstracted world. A possible transition in our example's envisionment is the following: given that we observe Q2+ — that is, we observe that the liquid water's temperature is rising —, then we know that the only possible following states are Q2 (continues in the liquid state) and Q3 (condensates into vapor), but never Q1 (freezes into ice).

Summarily, with a QR framework, we establish a way to (i) represent quantities through discrete states, (ii) provide a way to compare values between these states, (iii) enable derivations and sign algebra, and (iv) model envisionments detailing possible transitions between states. With such a framework, we can model, plan, simulate and reason about a multitude of intricate problems in an abstract way.

## 3 Approach

This section motivates the need to augment standard spectrum-based fault localization approaches with more contextual information, and details our Q-SFL approach to achieve it by qualitatively partitioning SFL components.

### 3.1 Limitations of Spectrum-based Analyses

SFL faces several issues preventing it from widespread adoption and use. Not the least of which is the lack of contextual information, essential for understanding *why* diagnostic candidates are considered suspicious. This has been pointed out in the software fault localization literature [Parnin and Orso, 2011]. As SFL reasons about failures at the spectrum level, it only has access to whether a component was involved or not in each system transaction. While this enables a flexible, lightweight analysis, the necessary abstraction can impose a tradeoff both in accuracy and comprehension. Although there have been efforts to incorporate more data into the diagnostic process — by modeling component behavior that considers the system's state and previous diagnoses [Cardoso and Abreu, 2013a]; or by leveraging prediction models trained from issue tracking data [Elmishali *et al.*, 2016] — they were focused on *conditioning* the fault probability of existing diagnostic candidates, increasing accuracy but not necessarily increasing the ability to *comprehend* the diagnostic report.

Aside from *comprehension*, and because SFL only reasons about component involvement in failing transactions, omission errors — such as bound checks — also become difficult to diagnose [Xu *et al.*, 2011]. The abstract nature of the spectra that is fed into current SFL frameworks also leads to the formation of *ambiguity groups* and facilitates the occurrence of *coincidental correctness*. An ambiguity group is a group of components with identical involvement in all transactions [Stenbakken *et al.*, 1989]. Since they exhibit the same coverage pattern, no component within an ambiguity group can be uniquely identified as the root cause of failure, potentially hindering accuracy. Coincidental correctness refers to the event when no failure is detected, even though a fault has been executed [Richardson and Thompson, 1993]. Depending on the component granularity selected for the analysis, coincidental correctness can happen at a frequent rate. In particular, when two tests share the same coverage path, but produce different outcomes, it becomes significantly harder to distinguish them without further contextual information. Coincidental correctness can potentially lead to exonerating real faults as they are observed to behave nominally.

| $A$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ |
|---|---|---|---|---|
| $c_1$ | 1 | 1 | 1 | 1 |
| $c_2$ | 0 | 0 | 1 | 1 |
| $c_3$ | 1 | 1 | 1 | 0 |
| $c_4$ | 0 | 0 | 1 | 0 |
| $e$ | 1 | 1 | 0 | 0 |

(a) Regular spectrum.
$\Pr(\{c_1\} \mid (\mathcal{A}, e)) = 0.30$
$\Pr(\{c_3\} \mid (\mathcal{A}, e)) = \mathbf{0.70}$

| $A'$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ |
|---|---|---|---|---|
| $c_1$ | 1 | 1 | 1 | 1 |
| $c_1'$ | 0 | 0 | 1 | 1 |
| $c_1''$ | 1 | 1 | 0 | 0 |
| $c_2$ | 0 | 0 | 1 | 1 |
| $c_3$ | 1 | 1 | 1 | 0 |
| $c_3'$ | 1 | 0 | 0 | 0 |
| $c_3''$ | 0 | 1 | 1 | 0 |
| $c_4$ | 0 | 0 | 1 | 0 |
| $e'$ | 1 | 1 | 0 | 0 |

(b) QR-augmented spectrum.
$\Pr(\{c_1\} \mid (\mathcal{A}', e')) = 0.05$
$\Pr(\{c_1''\} \mid (\mathcal{A}', e')) = \mathbf{0.82}$
$\Pr(\{c_3\} \mid (\mathcal{A}', e')) = 0.12$
$\Pr(\{c_3', c_3''\} \mid (\mathcal{A}', e')) = 0.01$

Figure 2: Example of coverage partitioning via QR.

### 3.2 Q-SFL

We argue that all of the issues described above can be at least attenuated if we supplement the SFL framework with more contextual information about the system under analysis to perform the diagnosis. Our Q-SFL approach consists of partitioning several SFL components into multiple, meaningful, qualitatively distinct *subcomponents*, to be used in the fault localization. We leverage the QR concept of domain partitioning to inspect existing components during each system execution and assign them a qualitative state. Each of these qualitative states is then considered as a separate SFL component whose involvement per transaction is recorded and fed into the SFL framework for diagnosis. Note that we use software diagnosis to help describe Q-SFL and, later on, to evaluate it. Despite this, the Q-SFL concept applies to other SFL use cases where one can inspect the state of components, as is the case of electronic circuit diagnosis, and others.

Figure 2 depicts how QR can be employed to enhance program spectra. Figure 2a shows a 4 transaction by 4 component spectrum, along with resulting diagnostic scores from applying reasoning-based SFL. Candidate generation, following the methodology described in Section 2.1 yields two candidate diagnoses — components $c_1$ and $c_3$ can independently explain the observed failures as both cover all failing test cases. For this example, suppose that $c_1$ is the faulty component. Since $c_1$ is involved in more passing transactions, the

SFL framework will assign it a lower fault probability than $c_3$. To improve the accuracy of the SFL framework, one needs more contextual information about component executions.

We envision three different types of *landmarking strategies* that can be employed to define qualitative state boundaries: (i) *manual landmarking*, where the system's developers manually define what are the possible qualitative states for a given component; (ii) *static landmarking*, where landmarks depend on the *type* of a component; and (iii) *dynamic landmarking*, where a component's value is inspected at runtime, and partitioned into a set of categories. Examples of dynamic strategies will be presented in Section 4.

Figure 2b depicts the QR-augmented spectrum, where components representing qualitative partitions of both $c_1$ and $c_3$ are added to the original spectrum. An example of such partitioning using *static landmarking*: if $c_1$ represents a software procedure that contains a numeric parameter $i$, we can create two qualitative components $c_1'$ and $c_1''$ that represent invocations of $c_1$ with $i \geq 0$ and $i < 0$, respectively. This is a sign-based static partitioning strategy. Note that the original components $c_1$ and $c_3$ are not removed from the QR-augmented spectrum, as partitions may not provide further fault isolation. Since the original components remain in the spectra, they subsume unsuccessful partitionings which are uncorrelated with the failure, effectively yielding no increase in diagnostic effort.

If we are to diagnose the new spectrum from Figure 2b, component $c_1''$ is now the top-ranked diagnostic candidate. This QR-augmented spectrum avoids spurious inspections of component $c_3$, and provides additional *contextual information* about the fault, namely that $i < 0$ is often observed in failing transactions.

By landmarking data units associated with SFL components so that they are assigned a qualitative state at runtime, we are providing more context to the diagnostic process, and in some cases, consequently reducing the diagnostic effort. Such partitioning is also of crucial importance towards minimizing the impact and frequency of *ambiguity grouping* and *coincidental correctness*, as new, distinct components are added to the system's spectrum.

# 4 Evaluation

To evaluate our approach, we compare the cost of diagnosing a collection of faulty software programs using regular spectra against using QR-augmented spectra.

## 4.1 Methodology

We have sourced experimental subjects from the Defects4J[3] (D4J) database. D4J is a catalog of 395 real, reproducible software bugs from 6 open-source projects — namely JFreeChart, Google Closure compiler, Apache Commons Lang, Apache Commons Math, Mockito, and Joda-Time. For each bug, a developer-written, fault-revealing test suite is made available.

We run the fault-revealing test suite of each buggy D4J subject, gathering method-level coverage and test outcomes, to construct its spectrum. Besides coverage, we also record the value of all *primitive-type* arguments and return values for every method call. This enables us to experiment with different qualitative partitioning strategies in an *offline* manner.

Using the recorded argument and return value data, we create multiple (automated) partitioning models resulting in several Q-SFL variants. A *static partitioning* variant using automated sign partitioning based on the variable's type, as described in Section 3.2, was considered. For *dynamic partitioning*, several clustering and classification algorithms[4] were considered: k-NN, linear classification, logistic regression, decision trees, random forest, and x-means clustering Test outcomes are used as the class labels in the case of supervised models. Note that we are not using the aforementioned models for *prediction*, but rather as a partitioning scheme based on observed values. Hence, we do not break our data into *training* and *test* sets, as is customary in *prediction* scenarios. Because we use automated, domain independent partitioning, only primitive types are considered in the evaluation. All scripts used to run this experiment, as well as the gathered data, are available at https://github.com/aperez/q-sfl-experiments.

To be able to effectively compare a QR-enhanced spectrum against its respective original spectrum, we first reduce the Q-SFL diagnostic report to *method* components. This reduction is done by considering the highest fault probability of any subcomponent belonging to each method, to effectively be able to compare method-level diagnostic effort between the two approaches. A change in diagnostic effort is measured using

$$\Delta C_d = C_d(\text{Original}) - C_d(\text{QR-Enhanced}) \qquad (4)$$

where $C_d$ is the cost of diagnosis, as explained in Section 2.1. A positive $\Delta C_d$ means that the faulty component has risen in the ranking reported by SFL techniques when QR is used, yielding a lowered cost of diagnosing.

## 4.2 Results and Discussion

We were able to automatically partition the faulty method in 167 D4J subjects. The remaining D4J subjects were discarded because (i) the faulty method does not contain parameters nor does it return a value; because (ii) the faulty method only contains non-primitive, non-null, complex-typed parameters, which cannot be handled by the set of partitioning strategies described in Section 4.1; or because (iii) the aforementioned partitioning strategies were unable to create qualitative states whose coverage differs from their enclosing method[5].

Our first research question is

[3]Defects4J 1.1.0 is available at https://github.com/rjust/defects4j (accessed May 2018).

[4]We chose popular classification algorithms [Han *et al.*, 2011] implemented in the `Scikit-learn` package. X-means, as implemented in the `pyclustering` package, was selected as it can automatically decide the optimal number of clusters to use [Pelleg and Moore, 2000].

[5]That is, subjects where all qualitative components either are never active or have exactly the same activation pattern as the enclosing component are discarded since these subjects will invariably produce a $\Delta C_d$ of 0.

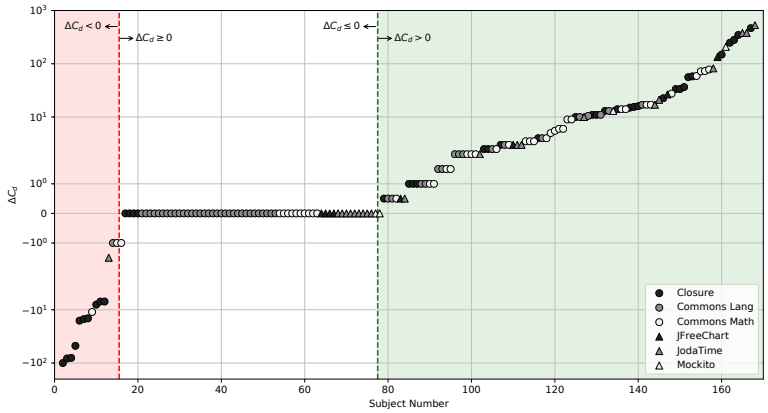


Figure 3: Difference in $C_d$ between original and QR-enhanced spectra per subject.

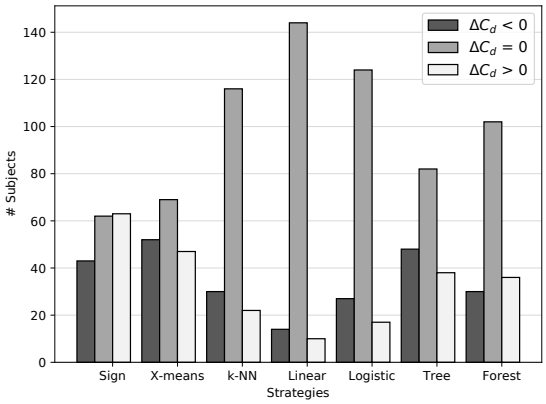


Figure 4: Breakdown of diagnostic performance per partitioning strategy.

**RQ1:** Does augmenting spectra with qualitative components improve their diagnosability?

In **RQ1**, we are concerned with finding if *there exist* qualitative partitionings able to improve the fault localization ranking to the extent that faulty components are inspected earlier — thus decreasing wasted effort in a debugging task. Hence, for each D4J subject, we choose as the landmarking strategy to consider in the evaluation the one that is able to create the largest set of distinct, non-ambiguous qualitative components out of the faulty method(s). The breakdown of selected partitioning strategies per subject is as follows: Sign partitioning: 102 (61% of subjects); X-means: 25 (15%); k-NN: 8 (5%); Linear Regression: 1 (1%); Logistic Regression: 4 (2%); Decision Tree: 11 (7%); Random Forest: 16 (10%). Our sign-partitioning default strategy was used to qualitatively enhance the majority of considered subjects, while other strategies such as linear classification and logistic regression were rarely selected. We believe the reason that supervised learning approaches — which were fed test case outcomes as the target class label — only exhibited superior performance in 40 subjects (24%) is due to the fact that the number of failing tests in test suites is often much smaller than the amount of passing tests, weakening the resulting partitioning model.

Figure 3 shows a scatter plot with the $\Delta C_d$ of all subjects under analysis. Shown in a red background are the 15 subjects (9%) with a negative $\Delta C_d$ — meaning that the report has suffered a decrease in accuracy after augmenting the spectra. The majority of these subjects belong to the Closure project. The 62 subjects (37%) with $\Delta C_d = 0$, where the faulty component has remained in the same position of the ranking, are shown in a white background. Lastly, 90 subjects (54%) that exhibited a positive $\Delta C_d$ — cases where QR-enhanced spectra improved diagnosability — are shown in green. All in all, Q-SFL is at least as effective as the original approach in 92% of scenarios.

Section 4.2 presents statistics computed to assess whether the observed metrics yield statistically significant results. QR-enhanced spectra exhibits an overall lower effort to diagnose when compared to the original spectra, with less variance. To assess significance, we first performed the Shapiro-Wilk test for normality of effort data in both the original spectra case and QR case [Shapiro and Wilk, 1965]. With 99% confidence, the test's results tell us that the distributions are not normal. Given that $C_d$ is not normally distributed and that each observation is paired, we use the non-parametrical statistical hypothesis test Wilcoxon signed-rank [Wilcoxon, 1945]. Our null-hypothesis is that the median difference between the two observations (*i.e.*, $\Delta C_d$) is zero. The fifth row from Section 4.2 shows the resulting $Z$ statistic and $p$-value of Wilcoxon's test. With 99% confidence, we refute the null-hypothesis.

| | **Original Spectra** | **QR-enhanced Spectra** |
|---|---|---|
| **Mean $C_d$** | 60.28 | 37.56 |
| **Median $C_d$** | 6.00 | 2.50 |
| **$C_d$ Variance** | $2.10\times10^{4}$ | $1.56\times10^{4}$ |
| **Shapiro-Wilk** | $W = 0.46$, $p$-value $= 2.20\times10^{-22}$ | $W = 0.32$, $p$-value $= 1.10\times10^{-24}$ |
| **Wilcoxon Signed-rank** | $Z = 5.45$, $p$-value $= 5.10\times10^{-10}$ | |

Table 1: Statistical tests.

**RQ1? Yes**, augmenting faulty spectra with new components resulting from qualitative landmarking of method parameter (and method return) values yields a *statistically significant* improved diagnostic report.

To be able to answer **RQ1**, we have selected for each subject the strategy with the highest number of qualitative partitions targeting the faulty method, as we were only concerned with the *existence* of *a* partitioning strategy that would improve diagnosability. However, in practice, it is not realistic

to know *a-priori* what the faulty method is[6]. Hence, our second research question is

> **RQ2:** Is there a particular automated landmarking strategy that consistently shows improved diagnosability?

Figure 4 shows a breakdown of the number of subjects that fall into the $\Delta C_d < 0$, $\Delta C_d = 0$ and $\Delta C_d > 0$ categories for every partitioning strategy considered in this evaluation. This bar plot tells us that no single strategy achieves the same number of positive $\Delta C_d$ scenarios as the partition cardinality selection scheme employed to answer **RQ1** and to produce Figure 3. Furthermore, strategies that were often picked by that criterion (namely, sign partitioning and X-means strategies) also show an increased number of negative $\Delta C_d$ scenarios when compared to others. This leads us to conclude that no single strategy (out of the ones that were analyzed) is able to *consistently* show improved diagnoses.

**RQ2? No**, at least for the automated landmarking strategies considered in the evaluation, there is no evidence that a single automated strategy can consistently outperform the original spectra. However, since Q-SFL can improve diagnosability, as per the answer to **RQ1**, we presume that *manual* or more complex, context-aware, automated *white-box* strategies — which can perform static and dynamic source code analysis — are much more suited to outperform the original spectra due to more *effective* and more *informed* partitioning.

## 5 Related Work

There have been previous forays into enhancing the diagnostic report of automated fault localization techniques to either improve their accuracy or comprehension of the failing component.

SFL approaches to debugging typically present their report to users as a list of suspicious components that is sorted according to the likelihood of being faulty. Jones *et al.* proposed a *visual* way of depicting the results of a similarity-based software SFL diagnosis, color-coding each component according to their suspiciousness score [Jones *et al.*, 2002]. Campos *et al.* expanded on the visual concept by leveraging tree-based visualizations that innately exploit the tree-like structure of Java code, naturally aggregating neighboring components and aiding exploration of suspicious code regions [Campos *et al.*, 2012]. Another approach to improve the comprehension of faults was proposed by Ko and Myers, called Whyline, which allowed the users to obtain evidence about the program's execution before forming an explanation of the cause by providing the ability to ask "why did" and "why didn't" questions about program output [Ko and Myers, 2004].

De Souza and Chaim proposed an extension to SFL to improve comprehension. It uses *integration coverage* data, by way of capturing method invocation pairs, to guide the fault localization process. By calculating the fault likelihood of component pairs, the authors were able to generate *roadmaps* for component investigation, guiding users through likely faulty paths and increasing the amount of contextual cues [de Souza and Chaim, 2013].

Advancements in bug prediction [D'Ambros *et al.*, 2010] enabled its use within automated fault localization processes. Wang and Lo proposed an ensemble approach to fault localization that exploits information from versioning systems, bug tracking repositories and structured information retrieval from the source code [Wang and Lo, 2014]. Cardoso and Abreu relied on kernel density estimation models of component behavior and previous diagnoses to better estimate the component goodness parameter in spectrum-based reasoning [Cardoso and Abreu, 2013a]. Elmishali *et al.* also modified the traditional spectrum-based reasoning framework by leveraging a fault prediction model trained with historical information from the project's versioning system and bug tracker to compute the *prior probability distribution* of diagnostic candidates [Elmishali *et al.*, 2016].

Augmenting fault-localization via slicing has also been proposed. Mao *et al.* proposed the use of *dynamic backward slices* — comprised of statements that directly or indirectly effect the computation of the output value through data- or control-dependency chains — as components in similarity-based SFL [Mao *et al.*, 2014]. Hofer and Wotawa proposed an approach that leverages a model-based slicing-hitting-set-computation — which computes the dynamic slices of all faulty variables in all failed test cases, derives minimal diagnostic candidates from the slices and computes fault probabilities for each statement based on number of the diagnoses that contain it [Hofer and Wotawa, 2012].

## 6 Conclusion

This paper proposes a new approach to spectrum-based fault localization that leverages qualitative reasoning (QR). The Q-SFL approach splits software components into a set of qualitative states through the creation of qualitative landmarks that partition a component's domain. These qualitative states are then considered as SFL components to be ranked using traditional fault-localization methodologies. Since we treat these qualitative states as components, our diagnostic reports not only recommend likely fault locations, but also provide an insight on what *behaviors* the faulty components assume when failures are detected, facilitating the comprehension of the fault.

We evaluate the approach on subjects from the Defects4J catalog of real faults from medium and large-sized open source software projects. Results show that spectra which were *augmented* using qualitative partitioning of method parameters shows a (statistically significant) improvement in the diagnostic accuracy in 54% of scenarios. However, we also found no evidence of automated partitioning strategies that were consistently better than the original spectra, meaning that more intricate, context-aware partitioning strategies will likely be necessary for practical applications of the approach.

This work lays the first stone in a series of efforts to more deeply integrate reasoning-based AI approaches into spectrum-based fault localization. It paves the way for further efforts by the fault localization research community, namely

[6] Although some effort has been put forth to hierarchically debug programs using SFL [Perez *et al.*, 2014].

by:

1. Improving automated landmarking by expanding its application to complex non-primitive objects and by exploring *ensembles* of multiple strategies.
2. Conducting a systematic user study investigating the extent that qualitative domain partitioning aids *fault understanding*.

## Acknowledgments

This material is based upon work supported by the scholarship number SFRH/BD/95339/2013 from Fundação para a Ciência e Tecnologia (FCT).

## References

[Abreu and van Gemund, 2009] Rui Abreu and Arjan J. C. van Gemund. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In *SARA'09*, 2009.

[Abreu *et al.*, 2009a] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. A new bayesian approach to multiple intermittent fault diagnosis. In *IJCAI'09*, pages 653–658, 2009.

[Abreu *et al.*, 2009b] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. Spectrum-based multiple fault localization. In *ASE'09*, pages 88–99, 2009.

[Ang *et al.*, 2017] Aaron Ang, Alexandre Perez, Arie van Deursen, and Rui Abreu. Revisiting the practical use of automated software fault localization techniques. In *IWPD'17*, pages 175–182, 2017.

[Campos *et al.*, 2012] José Campos, André Riboira, Alexandre Perez, and Rui Abreu. GZoltar: an eclipse plug-in for testing and debugging. In *ASE'12*, pages 378–381, 2012.

[Cardoso and Abreu, 2013a] Nuno Cardoso and Rui Abreu. A kernel density estimate-based approach to component goodness modeling. In *AAAI'13*, 2013.

[Cardoso and Abreu, 2013b] Nuno Cardoso and Rui Abreu. MHS2: A map-reduce heuristic-driven minimal hitting set search algorithm. In *MUSEPAT'13*, pages 25–36, 2013.

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

[D'Ambros *et al.*, 2010] Marco D'Ambros, Michele Lanza, and Romain Robbes. An extensive comparison of bug prediction approaches. In *MSR'10*, pages 31–41, 2010.

[de Kleer, 1977] Johan de Kleer. Multiple representations of knowledge in a mechanics problem-solver. In *IJCAI'77*, pages 299–304, 1977.

[Elmishali *et al.*, 2016] Amir Elmishali, Roni Stern, and Meir Kalech. Data-augmented software diagnosis. In *AAAI'16*, pages 4003–4009, 2016.

[Feldman *et al.*, 2008] Alexander Feldman, Gregory M. Provan, and Arjan J. C. van Gemund. Computing minimal diagnoses by greedy stochastic search. In *AAAI'08*, pages 911–918, 2008.

[Forbus, 1997] Kenneth D. Forbus. Qualitative reasoning. In *The Computer Science and Engineering Handbook*, pages 715–733. 1997.

[Gouveia *et al.*, 2013] Carlos Gouveia, José Campos, and Rui Abreu. Using HTML5 visualizations in software fault localization. In *VISSOFT'13*, pages 1–10, 2013.

[Han *et al.*, 2011] Jiawei Han, Jian Pei, and Micheline Kamber. *Data Mining: Concepts and Techniques*. Elsevier, 2011.

[Hofer and Wotawa, 2012] Birgit Hofer and Franz Wotawa. Spectrum enhanced dynamic slicing for better fault localization. In *ECAI'12*, pages 420–425, 2012.

[Jones *et al.*, 2002] James A. Jones, Mary Jean Harrold, and John T. Stasko. Visualization of test information to assist fault localization. In *ICSE'02*, pages 467–477, 2002.

[Just *et al.*, 2014] René Just, Darioush Jalali, and Michael D. Ernst. Defects4j: a database of existing faults to enable controlled testing studies for java programs. In *ISSTA'14*, pages 437–440, 2014.

[Ko and Myers, 2004] Andrew Jensen Ko and Brad A. Myers. Designing the whyline: a debugging interface for asking questions about program behavior. In *CHI'04*, pages 151–158, 2004.

[Kuipers, 1986] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29(3):289–338, 1986.

[Lucia *et al.*, 2014] Lucia, David Lo, Lingxiao Jiang, Ferdian Thung, and Aditya Budi. Extended comprehensive study of association measures for fault localization. *Journal of Software: Evolution and Process*, 26(2):172–219, 2014.

[Mao *et al.*, 2014] Xiaoguang Mao, Yan Lei, Ziying Dai, Yuhua Qi, and Chengsong Wang. Slice-based statistical fault localization. *Journal of Systems and Software*, 89:51–62, 2014.

[Parnin and Orso, 2011] Chris Parnin and Alessandro Orso. Are automated debugging techniques actually helping programmers? In *ISSTA'11*, pages 199–209, 2011.

[Pelleg and Moore, 2000] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML'00*, pages 727–734, 2000.

[Perez *et al.*, 2014] Alexandre Perez, Rui Abreu, and André Riboira. A dynamic code coverage approach to maximize fault localization efficiency. *Journal of Systems and Software*, 90:18–28, 2014.

[Richardson and Thompson, 1993] Debra J. Richardson and Margaret C. Thompson. An analysis of test data selection criteria using the RELAY model of fault detection. *IEEE Transactions on Software Engineering*, 19(6):533–553, 1993.

[Shapiro and Wilk, 1965] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611, 1965.

[de Souza and Chaim, 2013] Higor Amario de Souza and Marcos Lordello Chaim. Adding context to fault localization with integration coverage. In *ASE'13*, pages 628–633, 2013.

[Stenbakken *et al.*, 1989] G. N. Stenbakken, T. M. Souders, and G. W. Stewart. Ambiguity groups and testability. *IEEE Transactions on Instrumentation and Measurement*, 38(5):941–947, 1989.

[Wang and Lo, 2014] Shaowei Wang and David Lo. Version history, similar report, and structure: putting them together for improved bug localization. In *ICPC'14*, pages 53–63, 2014.

[Wilcoxon, 1945] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[Williams and de Kleer, 1991] Brian C. Williams and Johan de Kleer. Qualitative reasoning about physical systems: A return to roots. *Artificial Intelligence*, 51(1-3):1–9, 1991.

[Xu *et al.*, 2011] Xiaofeng Xu, Vidroha Debroy, W. Eric Wong, and Donghui Guo. Ties within fault localization rankings: Exposing and addressing the problem. *International Journal of Software Engineering and Knowledge Engineering*, 21(6):803–827, 2011.