Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

A Kernel Density Estimate-Based Approach to Component Goodness Modeling

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

Intermittent fault localization approaches account for the fact that faulty components may fail intermittently by considering a parameter (known as goodness) that quantifies the probability that faulty components may still exhibit correct behavior. Current, state-of-the-art approaches (1) assume that this goodness probability is context independent and (2) do not provide means for integrating past diagnosis experience in the diagnostic mechanism. In this paper, we present a novel approach, coined Non-linear Feedback-based Goodness Estimate (NFGE), that uses kernel density estimations (KDE) to address such limitations. We evaluated the approach with both synthetic and real data, yielding lower estimation errors, thus increasing the diagnosis performance.

1 Introduction

Previous approaches to (automatic) fault localization often assumed faults to be persistent (Abreu, Zoeteweij, and Van Gemund 2007; Chen et al. 2002; Jones and Harrold 2005). This is a fair assumption in scenarios where the understanding of the system and its model are very precise as, in theory, almost all systems are deterministic[1].

In practice, due to the inherent complexity of data modeling and analysis, most models are unable to distinguish all possible states the system can be in and, as a consequence, most faults appear to exhibit intermittent behavior (i.e., non-determinism). Concretely, an *intermittent fault* is a fault that, for equal system observations is not consistently triggered. This apparent intermittency was shown to greatly hamper the diagnosis power of persistent fault models (Abreu, Zoeteweij, and Van Gemund 2009).

To circumvent such limitation, the intermittent fault model was proposed in (De Kleer 2006) and later applied in the scope of automatic software debugging in (Abreu, Zoeteweij, and Van Gemund 2009). A key concept to the intermittency framework is the concept of *goodness*.

Definition 1 (Goodness). *The goodness of a (faulty) component, g_j, is the probability that a component j, under a set of observable system variables, exhibits nominal behavior.*

[1]At least outside the Quantum Mechanics world.

Intermittent fault modeling was proved to have a better diagnostic accuracy than persistent fault in several domains (Abreu and Van Gemund 2010; De Kleer 2009; Kuhn et al. 2010). In addition to the steps already required by the persistent fault framework (defined in Section 2), and due to the fact that these values are typically not available, the intermittency model raises the challenge of accurately estimating the goodness probability, g_j.

The great limitations of existent approaches relate to the fact that (1) g_j is estimated as being constant and independent from the system state and (2) such approaches do not provide any mechanism for improving the diagnostic performance given the outcome of previous diagnostics. In this paper we propose an approach that relies on a statistic non-parametric technique called Kernel Density Estimate (KDE) aimed at overcoming such limitations.

Results on synthetic data showed that our approach provides better goodness estimates than its constant counterparts and an overall low estimation error. Results on a real application showed that the accuracy of g_j plays a crucial role in the accuracy of the diagnosis system. The conducted real application tests proved that our approach was not only able to use the system's state to overcome the presented limitations of existent approaches but also that the accuracy of g_j was higher than for existent approaches.

The paper makes the following contributions:

- We provide an overview of the state of the art approach to automatic software fault localization.
- We present a new approach for modeling g_j as a non-linear function using a set of feedback observations. The approach is called Non-linear Feedback-based Goodness Estimate (NFGE).
- We provide an empirical evaluation of our approach using both synthetic and real data.

The remainder of this paper is organized as follows. In Section 2 we introduce the concepts and definitions used throughout this paper as well as an overview of the current fault localization approach. In Section 3 we motivate and present our approach. In Section 4 we test our modeling approach with synthetic data and we evaluate the effects in the diagnosis of a real application with injected faults. In Section 5 we present some related work. Finally, in Section 6 we draw some conclusions about the paper.

2 Preliminaries

In this section we introduce concepts and definitions used throughout the paper.

Definition 2 (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 3 (h-literal). *Under an observation term obs over variables in OBS, a component is considered healthy if it performed nominally and unhealthy otherwise. An h-literal,* h_j *for* $c_j \in COMPS$*, denotes the component's health.*

Definition 4 (h-clause). *An h-clause is a disjunction of h-literals such that no pair of h-literals is complimentary.*

Definition 5 (Diagnosis Candidate). *Let* S_N *and* S_P *be two sets of components' indices, faulty and healthy respectively, such that* $COMPS = \{c_m \mid m \in S_N \cup S_P\} \wedge S_N \cap S_P = \emptyset$*. We define* $d(S_N, S_P)$ *to be the conjunction*

$$(\bigwedge_{m \in S_N} \neg h_m) \wedge (\bigwedge_{m \in S_P} h_m) \tag{1}$$

A diagnosis candidate for DS, given an observation term obs over variables in OBS, is $d(S_N, S_P)$ *such that*

$$SD \wedge obs \wedge d(S_N, S_P) \not\models \bot \tag{2}$$

In the remainder we refer to $d(S_N, S_P)$ *simply as d, which we identify with the set* S_N *of indices of the negative literals.*

Definition 6 (Diagnosis Report). *A diagnosis* $D = (d_1, \ldots, d_k, \ldots, d_K)$ *is an ordered set of K diagnosis candidates, for which*

$$\forall d_k \in D : SD \wedge obs \wedge d_k \not\models \bot \tag{3}$$

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

The candidate generation problem is normally solved by using search algorithms (Abreu and Van Gemund 2009; De Kleer and Williams 1992; Feldman, Provan, and Van Gemund 2008) that produce candidates that (1) are consistent with the observations and (2) heuristically, have a higher chance of being correct.

In the remainder of this section we describe the relevant aspects of the Bayesian approach to address ranking problem. We assume that a set of observations have been collected, either by static modeling or by dynamic profiling, both known as a *spectra* (Harrold et al. 1998).

The far most commonly used type of spectra is called hit spectra (Harrold et al. 1998). Existent approaches (Abreu, Zoeteweij, and Van Gemund 2008b; 2008a; 2009; De Kleer 2009) use the hit spectra abstraction as input to the Bayesian probability update process.

Definition 7 (Hit Spectra). *The hit spectra is a particular type of spectra that encodes involvement of each* $c_j \in COMPS$ *in transaction* i *in terms of hit/not hit. Formally, let* N *denote the total number of transactions and* M *denote the cardinality of* $COMPS$*. Let* A *denote the* $N \times M$ activity matrix *of the system, defined as*

$$A_{ij} = \begin{cases} 1, & \textit{if component j was involved in transaction } i \\ 0, & \textit{otherwise} \end{cases} \tag{4}$$

Let e *denote the* error vector*, defined as*

$$e_i = \begin{cases} 1, & \textit{if transaction i failed} \\ 0, & \textit{otherwise} \end{cases} \tag{5}$$

In the scope of software diagnosis, for instance, a transaction is a sequence of component activity (e.g., statements, functions, services, etc.) that results in a particular return value.

Let p_j[2] denote the prior probability that a component c_j is at fault. Assuming components **fail independently**, the prior probability for a particular candidate $d \in D$ is given by

$$\Pr(d) = \prod_{j \in d} p_j \cdot \prod_{j \in COMPS \setminus d} (1 - p_j) \tag{6}$$

For a set of observations, obs[3], the posterior probabilities are calculated according to Bayes rule as

$$\Pr(d \mid obs) = \Pr(d) \cdot \prod_{obs_i \in obs} \frac{\Pr(obs_i \mid d)}{\Pr(obs_i)} \tag{7}$$

The denominator $\Pr(obs_i)$ is a normalizing term that is identical for all $d \in D$ and thus needs not to be computed directly. $\Pr(obs_i \mid d)$ is defined as

$$\Pr(obs_i \mid d) = \begin{cases} 0, & \text{if } obs_i, e_i, \text{ and } d \text{ are not consistent} \\ \varepsilon, & \text{otherwise} \end{cases} \tag{8}$$

ε (Abreu, Zoeteweij, and Van Gemund 2008b; De Kleer 2009) can be defined as

$$\varepsilon = \begin{cases} 1 - \prod_{j \in (d \cap obs_i)} g_j & \text{if } e_i = 1 \\ \prod_{j \in (d \cap obs_i)} g_j & \text{otherwise} \end{cases} \tag{9}$$

In (Abreu, Zoeteweij, and Van Gemund 2009), the authors address this problem by maximizing $\Pr(obs \mid d)$ (Maximum Likelihood Estimation (MLE) for naive Bayes classifier) under parameters $\{g_j \mid j \in d\}$ for the above epsilon policy.

3 Approach

To motivate our approach and illustrate the limitations of existent approaches, consider an instance of the *Znn.com* case study (Cheng, Garlan, and Schmerl 2009), composed of a load balancer (LB) and a set of databases ($DB_1, \ldots, DB_x$) with respective hard drives ($HD_1, \ldots, HD_x$). The workload of such system consists in delivering static content to a set of clients. In this scenario each transfer represents a transaction and the success/failure of each transaction could

[2]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 (Carey et al. 1999).

[3]Each row A_{i*} of the activity matrix encodes (in a binary form) the set $obs_i \in obs$.

T	LB	DB_1	HD_1	DB_2	HD_2	Error
1	1	1	1	0	0	1
2	1	0	0	1	1	1

Figure 1: Hit spectra example

be verified by computing a hash over the delivered content and checking it against a previously calculated hash.

The first limitation of existent approaches is related with the high level of abstraction enforced by the usage of hit spectra as it does not provide any information about the state of the system during each component's execution. Additionally, it abstracts the number of times each component was used and, consequently, the sequence in which they were used in each transaction. As a consequence, hit spectra-based approaches have difficulties in distinguishing pairs of components for which the activity is similar (i.e., columns present low entropy), such as for instance the databases and respective hard drives in Fig. 1. The occurrence of low entropy spectra is specially incident in environments where the nature of transactions is not easily controllable (e.g., runtime environments).

The second limitation of existent approaches relates to the fact that g_j is estimated as being constant with respect to a set of observations. Consider that the effective (normally unknown) goodness for the hard drives gradually decreases over time, as depicted in Fig. 2c. Due to the fact that the slope of the curve is low and the time is monotonic, g_j can, most of the times, be successfully approximated by a constant with small errors. However, if the observations spanned over a long period of time or the slope of the goodness curve was higher, a constant goodness function would fail to accurately model the actual goodness of the hard drive, entailing large average errors. Given the multiplicative nature of the goodness value usage, even small errors can have a serious impact in the diagnosis report ranking.

Finally, hit spectra-based approaches have limited possibilities of incorporating existent knowledge about the system in the diagnosis. As the state of the system is completely abstracted in the hit spectra, it would be impossible to distinguish a new hard drive from an old hard drive. As a consequence, even if the goodness curve was available for all components in the system, the algorithm would not be able to use it. Furthermore, in the event of all hard drives being failing (e.g., Fig. 1), existent approaches would assign a higher likelihood to the candidate containing only the load balancer as it has a lower cardinality than the correct candidate.

In the remainder of this section we present our approach, called Non-linear Feedback-based Goodness Estimate (NFGE), aimed at modelling the components' goodness as a non-linear function of a set of observable variables, $OBS_j \subseteq OBS$, referred to as $g_j(st)$. Additionally we present an improved version of ε that is used to plug $g_j(st)$ into the Bayesian framework to compute the posterior candidate probabilities $\Pr(d \mid obs)$ given a set of state spectra observations.

Definition 8 (State spectra). *State spectra is a particular type of spectra that encodes the value of variables OBS_j for each execution of $c_j \in COMPS$. Formally, let N denote the total number of processes and M denote the cardinality of $COMPS$. Let A denote the $N \times M$* activity matrix *of the system, defined as*

$$A_{ij} = (st_1, ..., st_k, ..., st_K) \tag{10}$$

Each element of A_{ij}, st_k, encodes the value of variables OBS_j for the kth activation of component j in process i. Let e follow the definition presented in Def. 7.

Our approach can be divided into two independent stages: the modeling and diagnosis stages.

Modeling $g_j(st)$

Let an abstract data type, henceforward referred to as feedback spectra, be the data interface to our modeling approach.

Definition 9 (Feedback Spectra). *The feedback spectra is a particular type of spectra that encodes the result of a set of diagnoses. Concretely, let M denote the cardinality of $COMPS$. FB_{ej} consists of a $2 \times M$ matrix defined as*

$$FB_{ej} = \{st_1, ..., st_k, ..., st_K\} \tag{11}$$

FB_{0j} and FB_{1j} are the pass and fail feedback observations sets respectively, for component c_j. Each element of FB_{ej} encodes the value of variables OBS_j.

In the following we assume that a mechanism for collecting feedback spectra exists. Additionally, we assume an atomic step of modeling that precedes all diagnosis.

Our approach to model $g_j(st)$ consists of a probabilistic model that is derived from the feedback spectra. Concretely, we estimate the pass and fail density functions, parametrized over variables in OBS_j[4], from which we trivially calculate $g_j(st)$.

The estimation of the pass/fail densities consist of a Kernel Density Estimate (KDE), $\hat{f}(st)$, and is described as

$$\hat{f}(st) = \frac{1}{bw} \sum_{c \in C} K\left(\frac{st - c}{bw}\right) \tag{12}$$

where C is a set of instantiations of OBS_j, $bw > 0$ is the bandwidth, a smoothing parameter, and $K(\cdot)$ is a kernel function[5]. A key aspect of KDE is the selection of the bandwidth parameter bw. In our approach, we estimate bw by using the Silverman's "rule of thumb" (Silverman 1986) defined as

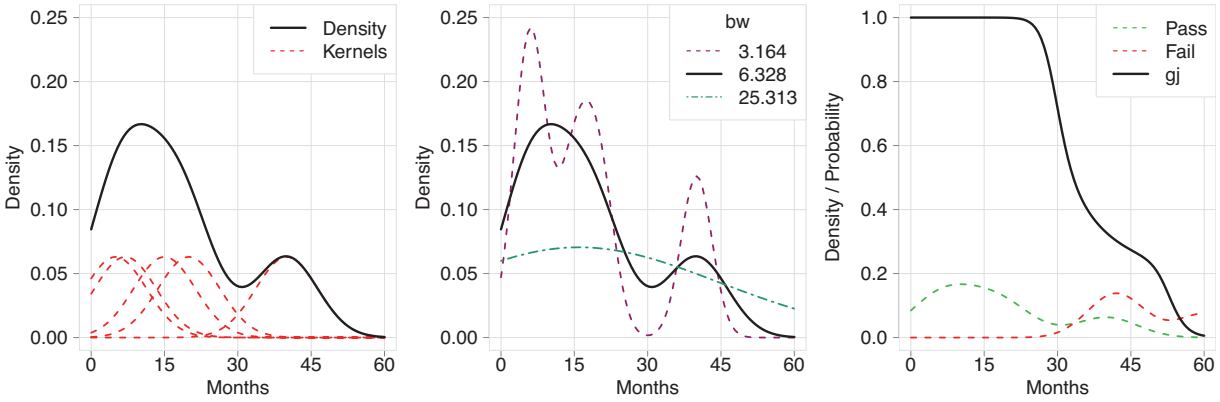
$$0.9 \times min\left(\sigma, \frac{R}{1.34}\right) \times |C|^{(-0.2)} \tag{13}$$

where R is the interquartile range of sample C. Regarding $K(\cdot)$, even though several options exist, in our approach, we use the Gaussian kernel. Additionally, and without loss of generality, we will assume OBS_j contains only one variable.

As an example, consider the modeling process of $g_j(st)$ for a component c_j (e.g., the hard drives from the previous example) given 5 pass and 3 failed feedback observations with values $FB_{0j} = \{5, 7, 15, 20, 40\}$ and $FB_{1j} = \{40, 44, 60\}$, respectively.

[4]Currently, the variables are arbitrary and must be manually selected on a per-component basis.

[5]A kernel is a symmetric but not necessarily positive function that integrates to one.



(a) Density estimation and underlying kernels

(b) Impact of bw value

(c) Goodness vs. pass/fail KDEs

Figure 2: KDE visual intuition

The first step in modeling $g_j(st)$ is the estimation of the density function for the nominal executions depicted in Fig. 2a, formally defined in Eq. 12, with parameters $C = \{5, 7, 15, 20, 40\}$ and $bw = 6.328$. Note that for each value in the horizontal axis, the KDE value corresponds to the summation of all underlying kernels at the same point. From Eq. 12 we can see that C determines each kernel's offset and bw the dispersion of the density. In particular, when using the Gaussian kernel, C_i corresponds to its mean and bw to its standard deviation.

Fig. 2b provides a visual intuition on the effect of the parameter bw in the estimate. A sensible selection of bw is crucial in order to yield good results as using a small bw value will reflect sampling artifacts whereas a large bw value will smooth some behavioral trends.

The second and final step is the derivation of $g_j(st)$ from the pass/fail KDEs. We will assume that the previous step was repeated for the fail executions yielding the densities depicted in Fig. 2c. $g_j(st)$, depicted in black, is defined as

$$g_j(st) = \frac{\hat{f}_{\text{pass}}(st)}{\hat{f}_{\text{pass}}(st) + \hat{f}_{\text{fail}}(st)} \tag{14}$$

Ranking using $g_j(st)$

In order apply $g_j(st)$ to the Bayesian framework we define a new policy for ε. For a given process i and a set of state spectra observations, the epsilon policy is given by

$$\varepsilon = \begin{cases} 1 - \prod_{j \in d \wedge st_k \in A_{ij}} g_j(st_k) & \text{if } e_i = 1 \\ \prod_{j \in d \wedge st_k \in A_{ij}} g_j(st_k) & \text{otherwise} \end{cases} \tag{15}$$

In contrast to existent ε policies, our policy does not only take into account the state of c_j but also the number of times it was executed.

Complexity analysis

The time complexity of calculating the bandwidth parameter, bw, for a set of F feedback observations is $O(F \cdot log(F))$ (due to the calculation of the interquartile range). Consequently, the overall complexity of the modelling phase, for M components, would be $O(2 \cdot M \cdot F \cdot log(F))$. In practice, as the number of feedback observations needed to build an accurate model is small (< 50), we observe a maximum complexity of $O(2 \cdot f \cdot log(f) \cdot M)$, where f is a cutoff constant for the number of feedback observations.

Regarding the diagnosis phase, we can unfold it in simpler steps. For each component (M), transaction (T) and observation (S) we calculate the associated goodness ($O(M \cdot T \cdot S)$). The goodness estimation consists in calculating the result of Eq. 12 ($O(F) \approx O(f)$). Globally, the time complexity of this step is $O(f \cdot M \cdot T \cdot S)$.

4 Results

To assess the performance of our approach we conducted two studies. The first study is aimed at evaluating the prediction error of the modelling approach. At this stage, we use synthetic goodness models in order to be able to test a wider set of goodness patterns. After establishing the prediction error of our approach, the second study aims at exploring, in a real application, the cases where existent approaches tend to fail.

Prediction Error Study

To assess the prediction error of our approach we generated a set of 20000 random synthetic goodness models (M). With the purpose of having different learning and observation generation processes, we modified Eq. 12 such that each underlying kernel has an individual bw_i. Formally, the synthetic goodness models have the underlying pass and fail distributions defined as

$$\hat{f}(st) = \sum_{i=1}^{|C|} \frac{K(\frac{st - C_i}{bw_i})}{bw_i} \tag{16}$$

Additionally, in our synthetic data setup, two types of models can be distinguished. The first set of models use the Gaussian kernel as their building block. This kind of models

are intended to mimic the behavior of components that exhibit smooth transitions between any two points in the feature space (i.e., the domain of the observable variables). This can be the case of component ageing in which the goodness normally decreases gradually with time (e.g., Fig. 2c).

The second type of models use the Box kernel, i.e., a rectangular-shaped kernel centered in C_i with bw_i width and $\frac{1}{bw_i}$ height. This set of models is intended to emulate components that exhibit abrupt transitions in their goodness functions. Also, as the original kernel differs from the learning kernel, the process of generation and learning becomes substantially different and enables drawing more meaningful conclusions.

Finally, the models generated range from simple patterns such as for instance the one depicted in Fig. 2c to more complex patterns with up to 20 supporting kernels for both the pass and fail densities.

To generate the feedback spectra, for each model we randomly selected a set of 200 values, F, in its feature space. For each of them we determine whether the component performed nominally in a Bernoulli trial process parameterized with $M_l(F_k)$. For each $M_l \in M$, we trained estimators with gradually increasing amounts of feedback spectra.

For each trained model, the prediction error is obtained by comparing the predicted goodness and the original goodness for 1000 evenly distributed samples of the feature space. The results are summarized by means of 4 metrics, as depicted in Fig. 3: absolute mean error, standard deviation, positive mean error and negative mean error. The first two plots depict the average results for NFGE in both the Gaussian and Box cases, respectively. The last plot depicts the average results for a constant estimator defined as

$$g_j(st) = \frac{|FB_{0j}|}{|FB_{0j}| + |FB_{1j}|} \quad (17)$$

As discussed, current approaches are not able to incorporate feedback information, rendering a direct comparison with NFGE impossible. Despite, we provide the results for the constant estimator with the goal of establishing a ground of comparison to the hypothetical performance of existent algorithms if they were able to incorporate such information.

From the analysis of the results, we can see a clear improvement introduced by NFGE over the baseline results in all the observed metrics. As expected, the constant estimation presents a large average error of 44%. Additionally, the constant estimator does not scale with the size of the available feedback data.

In contrast, NFGE, in its best case scenario, was able to perform at 10% of average error. Globally, NFGE presented an average of 21% and 26% of prediction error for the smooth and non-smooth cases, respectively. The reason for the 5% difference relates to the fact that the gaussian-based KDE is not able, with a limited set of observations, to tightly fit the original box-based model.

Additionally a fairly quick convergence is observed: with 42 observations 90% of the maximum performance is obtained for both the Gaussian model/Gaussian estimator case and the Box model/Gaussian estimator case.

Finally, the results showed that when the amount of available data is small (< 11 observations), the constant estimate, on average, outperforms NFGE.

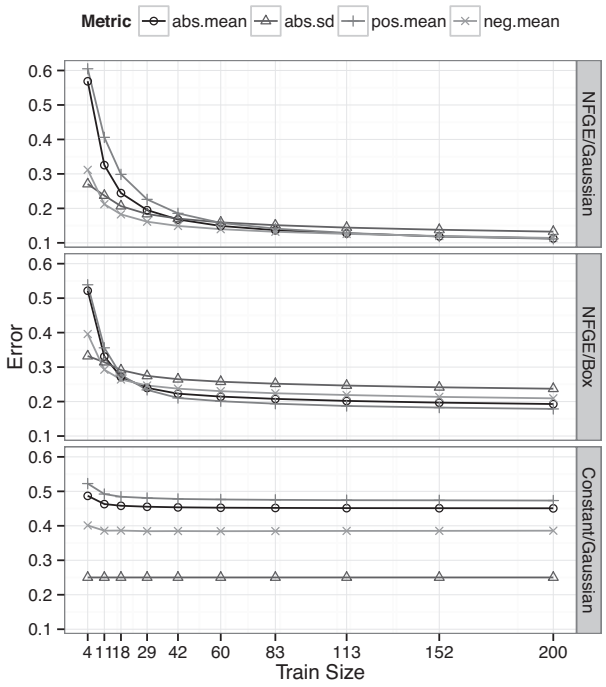


Figure 3: Prediction errors

Diagnosis Study

In this section we evaluate the diagnosis performance of our approach in scenarios where existent approaches tend to fail. This study is mainly intended to serve as a proof of concept.

The selected application for this study was a simple http server, webfs[6], which was instrumented to generate state spectra. Additionally, we injected code to emulate the behavior of ageing faults (e.g., memory leaks, hard drive wear, etc.). The injected faults are parametrizable over 3 variables: min, max, and $total$. For each execution of each injected fault a counter is incremented with a random value greater than min and lesser than max. Whenever a counter reaches the value of its associated $total$ variable, the fault is triggered and the transaction fails. The counters may either be shared among a set of faults or, unless stated otherwise, independent.

To collect the feedback spectra required for generating the goodness models, we did a prior test run where the faults were targeted individually. For each injected fault execution, we recorded the number of previous invocations and process age, i.e., the time since the http server start. For each fault we created two univariate goodness models using 20 feedback observations: one as a function of the number of invocations and the other as a function of the process age. As we only created non-linear goodness estimates for the components with injected faults, for the remainder components we

[6]Available at: *http://linux.bytesex.org/misc/webfs.html*

downgraded the state spectra into hit spectra and used the MLE approach (as presented in Section 2).

In a first scenario we only activated one fault. At this stage we tried to isolate the faulty component from a 5 element diagnosis report. The results showed that the MLE approach was only able to exonerate 1 out of 5 candidates. This was due to the fact that all transactions shared the same "bootstrap" sequence and the fault was injected in such sequence. As such components had the same activity pattern, i.e., equal columns in the hit spectra, and it remained impossible to distinguish them. In contrast, NFGE was able to clearly isolate the real faulty component. When comparing the candidates' probabilities for both approaches, NFGE calculated a probability of 99.9999% for the actual diagnosis whereas MLE calculated a probability $\approx 25\%$ for both the correct diagnosis and the 3 remaining components. This great difference in the likelihood magnitudes is in accordance with the results obtained in the synthetic experiments.

In a second scenario, we enabled two faulty components for which we had the corresponding goodness models. In this setup we generated a set of transactions that would eventually trigger one fault but not the other. Additionally, the two components were always activated in succession, generating low entropy hit spectra (as in Fig. 1). The goal of this test was to confirm that NFGE is both able to both indict and exonerate components depending on the execution context.

In this scenario, the MLE ranked equally the real error source and the component that should be exonerated. In contrast, NFGE was again able to both indict and exonerate components correctly.

Even though the previous setups produced positive results, it is important to note that the selection of the modeling variables is a crucial aspect. As previously stated we observed two distinct variables: the number of component executions and the process age. Even though both variables are able to model the goodness pattern of the injected faults, the degree to which each variable is able to cope with new scenarios may vary.

From this test setup it is easy to understand that even though correlated, the process age is not the cause of the fault activation: if a component is "old" but was never activated, the corresponding counter was never incremented. The process age variable indirectly encodes some aspects of the application workload. If a age based model was constructed for a specific workload, i.e., activations/second, its ability to produce positive results in a different workload may be limited. On the other hand it is clear that the component activation count is much more independent from the application's workload.

In the previous setups we used independent counters. If on the other hand we had a global counter and still used independent activation counters, the same variable would also implicitly encode workload patterns. Furthermore, if the modeling workload is substantially different from the diagnosis workload it could happen that the components that caused the errors would be exonerated.

5 Related Work

In addition to the previously present approach, there is a wide set of different approaches to fault localization.

In the scope of lightweight fault localization techniques, we can highlight examples such as Ochiai (Abreu, Zoeteweij, and Van Gemund 2007), Pinpoint (Chen et al. 2002) and Tarantula (Jones and Harrold 2005). While extremely efficient, such approaches do not consider multiple faults.

In the scope of diagnostic candidate generation, (De Kleer and Williams 1992) and (Abreu and Van Gemund 2009; 2010) compute the minimal diagnostic candidates that are able to explain the observed failures. Particularly, Staccato (Abreu and Van Gemund 2009) uses the Ochiai similarity coefficient as an heuristic to drive the candidate generation.

NFGE was inspired by reasoning approaches to automatic fault localization, such as (De Kleer 2009) and (Abreu, Zoeteweij, and Van Gemund 2009). These approaches rank components based solely on hit spectra (i.e, no feedback loop) which has limited encoding capacity. Also such techniques do not have the ability to improve their performance given information on prior diagnosis. While successful at development time, such approaches have limited application in run-time environments.

6 Conclusions

In this paper we presented a novel approach to fault localization that is able to absorb past diagnosis experience in order to improve its future diagnosis performance. Additionally, our approach, by using KDEs, is able to model the components' goodness as a non-linear function of a set of observable system variables.

A major improvement over hit spectra-based techniques is the possibility of using the value of the system variables to distinguish components that would otherwise be indistinguishable (due to high entropy spectra). Also, this enables the exoneration of faulty components in cases where the fault was not probably activated.

The conducted synthetic experiments produced promising results which were reproduced in the case study. However, from the real experiments we were able to clearly see some of the limitations of our approach.

Currently, the goodness modeling stage is still essentially manual. Work in devising tools for automatically determining the modeling variables of each component would have a deep impact on the usability of our technique. Also, it should be possible to extract information from previous diagnosis in order to automatically generate the feedback spectra.

Acknowledgements

We would like to thank Lígia Massena, André Silva, David Garlan, Bradley Schemerl, Paulo Casanova and Alexandre Perez for the useful feedback on previous versions of this paper. This material is based upon work supported by the National Science Foundation under Grant No. CNS 1116848, and by the scholarship number SFRH/BD/79368/2011 from Fundação para a Ciência e Tecnologia (FCT).

References

Abreu, R., and Van Gemund, A. 2009. 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.

Abreu, R., and Van Gemund, A. 2010. Diagnosing multiple intermittent failures using maximum likelihood estimation. *Artificial Intelligence* 174(18):1481–1497.

Abreu, R.; Zoeteweij, P.; and Van Gemund, A. 2007. On the accuracy of spectrum-based fault localization. In *Testing: Academic and Industrial Conference Practice and Research Techniques*, TAICPART'07, 89–98.

Abreu, R.; Zoeteweij, P.; and Van Gemund, A. 2008a. A dynamic modeling approach to software multiple-fault localization. In *Proceedings of the 19th International Workshop on Principles of Diagnosis*, DX'08, 7–14.

Abreu, R.; Zoeteweij, P.; and Van Gemund, A. 2008b. An observation-based model for fault localization. In *Proceedings of the 6th Workshop on Dynamic Analysis*, WODA'08, 64–70.

Abreu, R.; Zoeteweij, P.; and Van Gemund, A. 2009. A new bayesian approach to multiple intermittent fault diagnosis. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, 653–658.

Carey, J.; Gross, N.; Stepanek, M.; and Port, O. 1999. Software hell. In *Business Week*, 391–411.

Chen, M. Y.; Kiciman, E.; Fratkin, E.; Fox, A.; Fox, O.; and Brewer, E. 2002. Pinpoint: Problem determination in large, dynamic internet services. In *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, DSN 2002, 595–604.

Cheng, S.-W.; Garlan, D.; and Schmerl, B. R. 2009. Evaluating the effectiveness of the rainbow self-adaptive system. In *Proceedings of Workshop on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS'09, 132–141.

Feldman, A.; Provan, G.; and Van Gemund, A. 2008. Computing minimal diagnoses by greedy stochastic search. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, 911–918.

Harrold, M. J.; Rothermel, G.; Wu, R.; and Yi, L. 1998. 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, 83–90.

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

De Kleer, J., and Williams, B. C. 1992. Readings in model-based diagnosis. In *Readings in model-based diagnosis*. chapter Diagnosing multiple faults, 100–117.

De Kleer, J. 2006. Getting the probabilities right for measurement selection. In *17th International Workshop on Principles of Diagnosis*, DX'06, 141–146.

De Kleer, J. 2009. Diagnosing multiple persistent and intermittent faults. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, 733–738.

Kuhn, L.; Price, B.; Do, M.; Liu, J.; Zhou, R.; Schmidt, T.; and De Kleer, J. 2010. Pervasive diagnosis. *Transactions on Systems, Man Cybernetics - Part A* 40(5):1562–1595.

Silverman, B. W. 1986. *Density estimation for statistics and data analysis*. Chapman and Hall.