Submitted for publication

Diagnosing Unobserved Components in Self-Adaptive Systems

Paulo Casanova
Carnegie Mellon University
Pittsburgh, PA, USA
paulo.casanova@cs.cmu.edu

David Garlan
Carnegie Mellon University
Pittsburgh, PA, USA
garlan@cs.cmu.edu

Bradley Schmerl
Carnegie Mellon University
Pittsburgh, PA, USA
schmerl@cs.cmu.edu

Rui Abreu
Universidade do Porto
Porto, Portugal
rui@computer.org

ABSTRACT

Availability is an increasingly important quality for today's software-based systems and it has been successfully addressed by the use of closed-loop control systems in self-adaptive systems. Probes are inserted into a running system to obtain information and the information is fed to a controller that, through provided interfaces, acts on the system to alter its behavior. When a failure is detected, pinpointing the source of the failure is a critical step for a repair action. However, information obtained from a running system is commonly incomplete due to probing costs or unavailability of probes. In this paper we address the problem of fault localization in the presence of incomplete system monitoring. We may not be able to directly observe a component but we may be able to infer its health state. We provide formal criteria to determine when health states of unobservable components can be inferred and establish formal theoretical bounds for accuracy when using any spectrum-based fault localization algorithm.

Categories and Subject Descriptors

D.2.5 [**Testing and Debugging**]: Diagnostic aids,Monitors; D.2.4 [**Software**]: Program Verification—*Reliability,Correctness proofs*

General Terms

Algorithms, Reliability, Theory

Keywords

Self-adaptive systems; Diagnostics; Monitoring.

1. INTRODUCTION

An increasingly important quality for today's software-based systems is high availability. While high availability used to be confined to certain technology outliers (the phone system, the electrical grid, etc.) our increasing reliance on software systems has made high availability a requirement for many systems. This trend has led to an interest in improving system resilience by endowing systems with the ability to automatically cope with faults, attacks, changes in resource availability, and shifting system requirements.

One particularly successful approach to improving system resilience is to adopt a closed loop control paradigm: a system is monitored through a set of "probes" to determine whether the system is working within an appropriate behavioral envelope. If not, adaptation mechanisms are selected to improve that behavior, adapting the system through a run-time interface that the system provides [20, 23, 31, 32]. Adding such a control loop turns a system into a self-adaptive system.

Within such a control loop, a critical step is the ability to detect problems and pinpoint their source – sometimes referred to as *fault detection and localization*. Fault detection and localization is in general a hard problem for a variety of reasons. First, there may be many logical explanations for an observed problem. Second, faults may be intermittent or occur only under certain specific circumstances (such as when two particular components are communicating). Third, algorithms for localizing faults must find the right balance between speed of diagnosis and accuracy, qualities that are often in conflict. Fourth, the information available to the control layer through probing mechanisms may be incomplete. Prior research by the authors and others has shown how to address the first three problems [9, 7, 26, 27, 28]. However, the third remains an open area.

Incompleteness in monitoring information is particularly problematic and, unfortunately, arises commonly. It is *problematic* because if we are observing a particular part of the system it may be hard to determine directly whether observed problems are caused by elements within that part of the system. It is *common* because, in general, system probes provide only partial coverage of system behavior. Partial coverage arises for two reasons. First, there are typically costs associated with probing, such as degraded system performance, increased system complexity, and deployment cost. Second, parts of the system may be outside our control to monitor. This occurs, for instance, if those parts are

managed by another organization, the available technology does not exist to detect what is going on, or monitoring that part of the system may adversely affect important quality attributes.

In this paper we address the problem of fault localization in the presence of incomplete system monitoring. The key observation is that even though we may not be able to *directly* observe a component of the system, we may be able to indirectly *infer* its health state through a collection of observations about its behavior in the context of other system components that we can monitor. Specifically, given some knowledge about the possible system behaviors, we may be able to infer whether a unobservable component was used or not and infer its health as if it had been observed. Given this observation, we provide formal criteria for determining whether such inference is possible. We exemplify the usage of our criteria in the context of a well-known algorithm for fault localization, called Barinel [4]. Such formal criteria establish:

(1) theoretical maximum bounds for accuracy of diagnosis using probabilistic reasoning methods,

(2) theoretical minimum bounds for the case of single-fault systems, and

(3) an algorithm that computes the maximum possible accuracy of fault localization on a system.

Having the ability to determine formally when indirect observation is adequate to localize faults is an important capability. First, it allows us to reason about the quality of our monitoring infrastructure with respect to its abilities to determine problems. Second, it extends the reach of fault localization algorithms showing how we can use inference to localize faults even in the absence of direct observation. Third, when it is not possible to make such inferences, it helps provide a basis for deciding whether (and where) to dynamically adapt the probing infrastructure (through dynamic probe placement) to focus on "hidden" parts of the system if we suspect that problems are occurring in that region.

The rest of the paper is structured as follows: In Section 2 we present an example which we will use throughout the paper to illustrate fault localization and how our results apply. In Section 3 we present the general principles of spectrum-based fault localization and in Section 4 we present our results. Section 5 shows our algorithm working together with a specific fault localization algorithm and in Section 6 we show how probe placement affects diagnosis, going through an example with several possible probe placements and checking the accuracy of each one. Finally, we present related work in Section 7 and the conclusions and future work in Section 8.

2. MOTIVATION

In this section, we provide an example which we use throughout the paper to illustrate the application of our techniques. The system is a simulator of a semiconductor manufacturing control system used by Samsung Electronics,[1] built according to the specifications provided by Samsung Electronics. [8] In a semiconductor factory, semiconductors are manufactured in lots that go through several hundreds of processing stages before they can be shipped or integrated in other products. The simulated system's responsibility is to track the semiconductor lots through the various processing stages, deciding on which pieces of equipment to use and when. It effectively runs the factory.

[1] `http://www.samsung.com/`

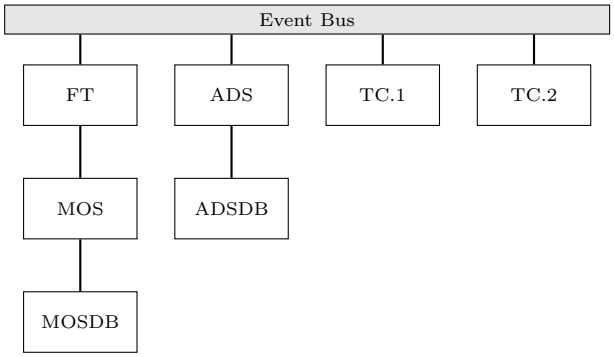

Figure 1: A partial view of an industrial semiconductor manufactoring control system.

The architecture of the system in illustrated in Figure 1. The system contains an event bus, which mediates the interaction of several components that control the production of semiconductors in Samsung's fabrication factories.

There are 7 components in this system and one event bus. The Manufacturing Operating System [MOS] controls the manufacturing process, keeping information about the lots produced, including the processing stage of each one. The MOS is connected to the event bus through a Fault-Tolerance mechanism [FT]. Several other instances of the MOS are connected to the event bus and FT provides failover capabilities. Because we will not be considering MOS faults that can be transparently handled by the FT, the other instances are not present in the diagram. The MOS stores its information in a database, the MOSDB, which must run at high throughput rates in order to keep up with the volume of thousands of requests per second.

When a product lot needs to go through another stage of processing, several pieces of equipment may be available to perform the task. The Automatic Dispatch System [ADS] keeps track of the equipment schedule and decides which one is best suited to perform the task. The ADS keeps its information in another high performance database, the ADSDB. Tool Controllers (TC) connect to the physical factory equipment themselves. There are two such controllers in this example.

The actual system at Samsung Electronics is, of course, much larger with many more component instances, but, as they do not fundamentally change the paradigm of fault-related issues, it is sufficient to work with a simplified version of the system. (For more details on this system and our approach to fault localization, including scalability results, see [8].)

The components in Figure 1 communicate with each other by exchanging events through an event bus. There are two important interactions that occur when a lot is processed by a piece of equipment: a track-in [TKIN], generated when the lot is about to be processed; and a track-out [TKOUT], when a lot exits a piece equipment after processing. TKIN and TKOUT are complex computations performed by the factory systems. For simplicity, we consider here a small but representative subset of the computations involved in a

Table 1: Computations performed by the system.

Name	Components	Description
C1: MOS request dispatch	MOS, ADS, ADSDB	The MOS requests equipment from the ADS to process a lot.
C2: ADS process query	ADS, ADSDB	The ADS computes the best piece of equipment to process a lot.
C3: MOS update dispatch	FT, MOS, MOSDB	The MOS receives information about the best equipment and updates routing information.
C4: MOS TC update	MOS, MOSDB, TC.1, TC.2	The MOS informs the factory equipment to prepare to receive the lot.
C5: MOS reschedule	MOS, MOSDB, ADS, ADSDB	The MOS needs to reschedule a lot and queries the ADS for confirmation.

TKIN. Table 1 contains a description of the computations.

In this system, we are allowed to probe all components except the databases. Due to the high volume of data processed, the databases are often the bottleneck and Samsung is wary of the potential overhead that probing them might incur. So, both MOSDB and ADSDB are not observable.

With the system in Figure 1 and the list of computations in Table 1 we can see that some failure patterns can uniquely identify faulty components and some cannot. For example, if only C3 fails, the faulty component has to be FT (because if MOS or MOSDB were faulty, C4 would also fail). However, if only C4 fails, we cannot tell whether it was TC.1 or TC.2. Interestingly, if we rule out multiple failures, if *both* C3 and C4 fail then the fault has to be located in MOSDB, an unobservable component. In the next section we will discuss a formal model that allows us to determine which components can and cannot be diagnosed as faulty upon observation.

3. FAULT LOCALIZATION

In this paper we address the problem of fault localization when we have incomplete system monitoring. Previous work on fault localization assumes that information is complete. We build upon work in spectrum-based fault localization (SFL), which adopts a reasoning-based approach to fault localization founded on probability theory. The main principles underlying the technique rely on model-based diagnosis (MBD) [15, 17, 18, 25, 34, 21], which uses *logical reasoning* to find faults and rank them using statistical techniques.

The key insight of spectrum-based fault localization is that we can infer which components of a system are faulty by examining the components that participated in computations together with a judgement of whether the computations succeeded or failed. The list of components that participated in a computation is termed a *spectrum*.

In traditional SFL, a component is a program element (e.g., functions, classes, statements) and the computations are test cases. A suite of test cases is run with the code instrumented to keep track of which components were exercised by each test case. The resulting spectra, together with the success/failure output of the test case, is fed into an algorithm that computes and ranks *fault candidates*. A fault candidate is a set of components that, if all are faulty, could explain the observed failures. Several algorithms have been proposed, which differ in *how* they compute the fault candidates and how they rank them. Two examples are Tarantula [22] or Barinel [4].

Though SFL has mostly been used with test cases and program elements, more recently we have successfully used these algorithms at run time [9]. Here, the components are architectural elements of the system and the computations are the observed behaviors in a system that are classified by an oracle producing spectra and a success evaluations just like in the traditional case.

Table 2: Spectra in the example system (unobservable components are placed in parenthesis).

Computation	FT	MOS	(MOSDB)	ADS	(ADSDB)	TC.1	TC.2
C1	-	X	(-)	X	(X)	-	-
C2	-	-	(-)	X	(X)	-	-
C3	X	X	(X)	-	(-)	-	-
C4	-	X	(X)	-	(-)	X	X
C5	-	X	(X)	X	(X)	-	-

A Formal Model for SFL

Formally discussing diagnosis and accuracy requires having a formal model of the system and what is meant by faulty behavior. In this section we introduce a formal probabilistic model of a system used by spectrum-based fault localization algorithms, which we build on in subsequent sections.

Let σ be a system comprising several components that, interacting with each other, perform computations. In our example, σ is the system described in Figure 1. Let $comps_\sigma$ be the non-empty set of all components in system σ. In our example, our components are the 7 components: FT, MOS, MOSDB, ADS, ADSDB, TC.1 and TC.2.

Each computation that the system can perform, c, is drawn from $(2^{comps_\sigma} \setminus \{\emptyset\}) \times \{\top, \bot\}$: the first element in the pair is the non-empty set of components that contributed to the computation and the second element is either true ($\top$) or false ($\bot$) depending on whether the computation succeeded or failed, respectively. We refer to the first element as the *spectrum* of the computation ($\text{spec}(c)$) and to the second element as the *failure evaluation* of the computation ($\text{feval}(c)$).

A system σ can only produce spectra from a predefined set, $all_\sigma \subseteq (2^{comps_\sigma} \setminus \{\emptyset\})$. This arises from the structure and function of the system. For example, in Figure 1, ADSDB cannot communicate directly with MOSDB, and so a spectrum containing only these components is not possible. Table 2 contains the spectra that the sample system given in Section 2 can generate.

A *behavior* of a system defines the probability of a spectrum being generated. If B is a behavior of system σ, then B is a random variable drawing values from all_σ. To simplify

our notation, we call $p_B(s) \mathrel{\widehat{=}} P(B = s)$.

4. THE ACCURACY THEOREMS

In this section, we extend the model in Section 3 and show how diagnosis accuracy can be computed given that only some components are observable.

Diagnosis accuracy is defined as a partition of the system components into groups that have two characteristics:

- If two components are in the same partition, then a fault in one of them is indistinguishable from a fault in the other. This means an SFL algorithm will not be able to tell the difference between their health. This is the *theoretical maximum bound* for diagnosis accuracy.
- If two components are *not* in the same partition, then a fault in one of them is distinguishable from a fault in the other. This means an SFL algorithm will be able to tell the difference between their health. This is the *theoretical minimum bound* for diagnosis accuracy. The minimum bound guarantee only holds for single-fault systems as multiple faults in several components across different groups may be indistinguishable from a failure in a single group.

Computing the diagnosis accuracy enables reasoning about whether there are enough probes in the system, whether they are in the right place, and whether enough behaviors have been observed for diagnosis.

For example, consider the computations in Table 1. If only C4 fails, we can infer that MOSDB is *not* faulty despite the fact that we cannot observe it. Otherwise we would see C3 and C5 failing too. We know it has to be *either* TC.1 or TC.2. However, because there are no computations in which only TC.1 or TC.2 appear, we cannot tell whether the problem is in TC.1 or TC.2. In the example in Table 1, the MOSDB is in a different accuracy group from TC.1 and TC.2. TC.1 and TC.2 are in the same accuracy group. Table 5 contains all accuracy groups of the system in Section 2.

Reasoning about unobservable components requires *a priori* knowledge of the possible spectra in a system. In the previous paragraph, our reasoning required us to refer to Table 1 which represents our *a priori* knowledge in this case. We argue that this *a priori* knowledge is generally available: System designers know the system's architecture and what paths through the architecture are exercised. Even if some components or connectors cannot be probed, system designers know that they exist and what they do. Still, when this *a priori* knowledge is unavailable, analysis of accuracy of the observable components can still be done, even though diagnosing unobservable components is not possible, as we will discuss in Section 4.5.

Note that computation of diagnosis accuracy does not itself compute or rank fault candidates. Such rankings would be handed as a second step, as is typical in SFL, and could be handled by any number of algorithms (some are referenced in Section 7). Indeed, diagnostic accuracy is compatible with *any* spectrum-based fault localization algorithm that conforms to the model presented in Sections 3, 4.1 and 4.2.

4.1 A Probabilistic Behavior Model

In system σ, a computation c fails ($\text{feval}(c) = \bot$) if and only if any of the components in its spectrum fail. A component that does not cause the spectrum to fail is a *healthy* component. The *health* of a component is a value in $[0, 1]$ that defines the probability that the component will not fail if exercised.

A health state (or *state*, for short) H of system σ is an assignment of *health* values, $h_H(i)$ to each component $i \in comps_\sigma$. The probability of a computation c succeeding is given by $\prod_{i \in \text{spec}(c)} h_H(i)$. (This model assumes that failures of the components are independent. Handling correlated faults can be done by introducing correlation components as described in [7].)

For a system σ, behavior B and state H, we define an *instance* of the system, $\mathcal{I}_{\sigma,B,H}$, which is a random variable drawing values from $(2^{comps_\sigma} \setminus \{\emptyset\}) \times \{\top, \bot\}$. The probability of the system generating a correct computation with spectrum x is given by $P(\mathcal{I}_{\sigma,B,H} = c \wedge \text{spec}(c) = x \wedge \text{feval}(c) = \top) = p_B(x) \prod_{i \in x} h_H(i)$. The probability of the system generating an incorrect computation with spectrum x is $P(\mathcal{I}_{\sigma,B,H} = c \wedge \text{spec}(c) = x \wedge \text{feval}(c) = \bot) = p_B(x)(1 - \prod_{i \in x} h_H(i))$.

These probabilities state how likely are we to observe successes and failures in different spectra given a set of healthly components, and forms the basis for accuracy: as we will see later, if two different health states yield the exact same probability of success for all possible spectra, then we cannot distinguish one from the other by observation of the spectra.

4.2 Observability and Distinguishability

The previous section presented a general probabilistic behavior model for systems. In this section, we will extend the model to account for the (un)observability of components.

In general, not all components in a system may be observable. In our example, the databases cannot be observed. Unobservability can happen, as described in Section 1, for a variety of reasons. However, these components may still fail and we want to be able to pinpoint them as the source of a failure in such cases. If we do not account for failures in unobservable components, we may diagnose the incorrect components, triggering incorrect repair actions.

To compensate for the lack of observability of some components, we need to have some information about *when* such components can potentially be used. This is less information than observing the components directly. For example, computations C1 and C5 in Table 2 use the same visible components. In either case we only know that the MOS and ADS were involved in the computation. So, if we observe a computation using the MOS and ADS and neither the FT, TC.1 or TC.2, we know that ADSDB was involved and *maybe* MOSDB was involved, depending on whether we are observing C1 or C5.

To account for unobservability, we divide the components of a system σ into two groups, the group of *observable* components, obs_σ, and the group of *unobservable* components, $nobs_\sigma$. This division must form a *partition* of $comps_\sigma$: the two sets are disjoint ($obs_\sigma \cap nobs_\sigma = \emptyset$), and complete ($obs_\sigma \cup nobs_\sigma = comps_\sigma$).

We assume there are no spectra in which only unobservable components take part as SFL algorithms assume there are no empty spectra in the system and, therefore, there is no reason to handle this degenerate case.

For each spectrum x of system σ, we define its *projected spectrum*, x', which is the subset of x that contains only observable components: $x' \mathrel{\widehat{=}} x \cap obs_\sigma$. The projected spectrum x' is what we observe when x happens in the system.

Table 3: Projected spectra corresponding to the example of Table 2.

Projected	FT	MOS	ADS	TC.1	TC.2
P1 (C1,C5)	-	X	X	-	-
P2 (C2)	-	-	X	-	-
P3 (C3)	X	X	-	-	-
P4 (C4)	-	X	-	X	X

Table 3 contains the projected spectra corresponding to the example in Table 2.

Because projected spectra are subsets of the spectra, it may happen that two spectra, $x_1, x_2 \in all_\sigma$ are such that $x_1 \neq x_2 \wedge x_1' = x_2'$. In this case, we say that spectra x_1 and x_2 are *indistinguishable*. What we observe when x_1 happens is the same as when x_2 happens. Given a projected spectra y', we define the *reverse projection of* y', $\text{rev}_\sigma(y')$, the set of spectra x such that $x \in all_\sigma$ and $x' = y'$. Naturally, $x \in \text{rev}_\sigma(x')$. In our example, the projected spectra of both C1 and C5 is P1. The reverse projection of P1 is C1 and C5. The reverse projection tells us which spectra could have been responsible for the observation.

Given a system instance $\mathcal{I}_{\sigma,B,H}$, the probability that we observe a projected spectra y' and a failure evaluation of $\top$ is given by $\mathfrak{h}_H(y) \mathrel{\widehat{=}} \sum_{x \in \text{rev}_\sigma(y')} p_B(x) \prod_{i \in x} h_H(i)$. This equation just states that the probability of observing a successful projected spectra y' is to observe any successful spectra x such that $x' = y'$.

We define the set of all *projected spectra* of a system σ, $allproj_\sigma$, as the set with the projections of all spectra in all_σ. $allproj_\sigma = \bigcup_{x \in all_\sigma} x'$. The projected spectra define the *observable behavior* of the system. The probability of a projected spectra y' being observed is $p_B(y') = \sum_{x \in \text{rev}_\sigma(y')} p_B(x)$.

Two systems, σ with behavior B_σ and state H_σ, and ϕ, with behavior B_ϕ and state H_ϕ, such that (1) $allproj_\sigma = allproj_\phi$, (2) $\forall\, y' : allproj_\sigma \bullet p_{B_\sigma}(y') = p_{B_\phi}(y')$ and (3) $\forall\, y' : allproj_\sigma \bullet \mathfrak{h}_{H_\sigma}(y') = \mathfrak{h}_{H_\phi}(y')$, are *indistinguishable* by observation.

If two systems σ and ϕ are indistinguishable by observation then, without any additional information other than observation of projected spectra, it is not possible to know whether observations are being produced by σ or ϕ as they will generate the same observations with the same probability distribution.

4.3 The Accuracy Groups

For a component i of a system σ, we define its *participating projection*, $\text{partproj}_\sigma(i)$ as being the set of projected spectra of σ that contain in its reverse projection at least one spectra containing i. The participating projection of a component is the set of all projected spectra that the component may influence. If the component i is observable, then the projected spectra is the set of all computations in which i is involved. Observing one of those spectra implies i was used. For example, P1 is in the participating projection of the MOS; the MOS was used if we observe P1.

However, if the component is not observable, then observing one of the spectra in $\text{partproj}_\sigma(i)$ means component i *may* have been used in the computation. We may not be able to tell for sure as there may be computations x_1 and x_2 such that i participates in x_1 but not x_2 and $x_1' = x_2'$. For example, P1 is in the participating projection of the MOSDB; the MOSDB *may* have been used if we observe P1. However, P2 is not in its participating projection so observing P2 means the MOSDB was *not* used. Table 4 contains the participating projections of the system whose projected spectra are in Table 3.

Table 4: Participating projection of all components in the example of tables 2 and 3.

Component	Participating projection
FT	P3
MOS	P1, P3, P4
ADS	P1, P2
MOSDB	P1, P3, P4
ADSDB	P1, P2
TC.1	P4
TC.2	P4

Table 5: Accuracy groups corresponding to the components in 4.

Group	Participating projection
FT	P3
MOS, MOSDB	P1, P3, P4
ADS, ADSDB	P1, P2
TC.1, TC.2	P4

For a system σ, we define its *accuracy partition*, acc_σ, as a partition of all components in $comps_\sigma$ (including those in $nobs_\sigma$) into *accuracy groups*. Two components i and j belong to the same accuracy group if and only if they have the same partipating projection. Two components belong to the same accuracy group if and only if their health can influence the exact same set of visible behaviors. Table 5 contains the accuracy groups corresponding to the participating projections of Table 4.

4.4 The Accuracy Theorems

To prove the claims about the theoretical maximum and minimum bounds of accuracy we need to prove some preliminary results. First we show through Theorem 1 that the probability of projected spectra being observed *can* always be computed. Then we show through Theorem 2 that it is not possible to compute the relative probability of undistinguishable spectra.

Then we prove the theorems that establish the maximum bounds for diagnosis accuracy: the Strong Accuarcy Theorem, Theorem 3, and the Weak Accuracy Theorem, Theorem 4. The former is more restrictive than the latter but the latter is universally applicable. The last theorem presented, Theorem 5 provides the minimum bounds for diagnosis accuracy.

4.4.1 Initial Theorems

THEOREM 1. *Let $\mathcal{I}_{\sigma,B,H}$ be an instance of a known system σ with an unknown behavior B and unknown health H. Observation of the system allows eventually determining the value of $p_B(y')$ (with an arbitrary low error margin) for all projected spectra y'.*

This theorem states that by observing a system, eventually we will determine the probabilities with which each projected spectra will occur.

PROOF. Since, by definition, all projected spectra are distinguishable by observation and $p_B(y')$ is the probability

that y' is generated, then, by the law of large numbers, the statistical observation of outcomes of the projected spectra will converge to their probabilities. □

THEOREM 2. *Let $\mathcal{I}_{\sigma,B,H}$ be an instance of a known system σ with unknown behavior B and unknown health H. Observation of the system does not allow determining $p_B(x \mid x')$ where x is a spectra except when there is only x whose projected spectra is x'.*

This theorem states that just by observing a system we cannot determine the probabilities of the original spectra ocurring . We will, by Theorem 1, know the probabilities that *projected spectra* will occur, but will not be able to know how these probabilities decompose in the different spectra that comprise $\text{rev}_\sigma(y')$.

PROOF. If $x_1 \neq x_2 \wedge y = x_1' = x_2'$, then $p_B(y) = p_B(x_1) + p_B(x_2)$ (assuming x_1 and x_2 are the only spectra that project to y – extending this to several x_i is trivial). Therefore, any combination of values of the probabilities of x_1 and x_2 that sum to the same value will yield the same observations.

Because $p_B(x_1) = p_B(x_1 \mid y)p_B(y)$, we know that $p_B(x_1 \mid y) + p_B(x_2 \mid y) = 1$. However, because x_1 and x_2 are indistinguishable by observation, all distributions of 1 over $p_B(x_i \mid y)$ will yield the same result and are, therefore, indistinguishable. □

4.4.2 The Strong Accuracy Theorem

THEOREM 3 (STRONG ACCURACY THEOREM). *Let an instance of a known system σ in an unknown state H_1 be $\mathcal{I}_{\sigma,B,H_1}$. Let i and j be two observable components of σ such that i and j belong to the same accuracy group and $h_{H_1}(i) \neq h_{H_1}(j)$. Then, the instance $\mathcal{I}_{\sigma,B,H_2}$ where H_2 is the same as H_1 except that the healths of i and j are reversed, is indistinguishable from observation from $\mathcal{I}_{\sigma,B,H_1}$.*

Theorem 3 states an important and intuitive result: if two components are always used together, we can never distinguish the health of one from the health of the other. This means trying to rank the two components as fault candidates is useless as any distibution of blame for the failures among them is arbitrary. An example of this is presented in Section 6 where we show how these results can be used to reduce the search space for fault candidates.

PROOF. The probability that a certain projected spectra y' succeeds in $\mathcal{I}_{\sigma,B_1,H_1}$, $S(y')$, is given by:

$$\begin{aligned} S(y') &\mathrel{\widehat{=}} \textstyle\sum_{x \in \text{rev}_\sigma(y')} p_{B1}(x) \prod_{i \in x} h_{H_1}(i) \\ &= p_{B_1}(y') \textstyle\sum_{x \in \text{rev}_\sigma(y')} p_{B_1}(x \mid y') \prod_{i \in x} h_{H_1}(i) \end{aligned} \quad (1)$$

Because all observable components appear in all spectra that project to the same projected spectra, the previous equation can be rewritten as:

$$\begin{aligned} &S(y') = A_o \times A_n \text{, where} \\ &A_o = p_{B_1}(y') \textstyle\prod_{i \in y} h_{H_1}(i) \\ &A_n = \textstyle\sum_{x \in \text{rev}_\sigma(y')} p_{B_1}(x \mid y') \prod_{i \in x \setminus y'} h_{H_1}(i) \end{aligned}$$

If i and j are both observable then their health will appear in A_o and not in A_n. Because $A_o \propto h_{H_1}(i)h_{H_1}(j)$, the success probability of each spectra is the same in both cases. □

4.4.3 The Weak Accuracy Theorem

THEOREM 4 (WEAK ACCURACY THEOREM). *Let σ be a system with components i and j in the same accuracy group. Let H_1 and H_2 be two states of the system such that $h_{H_1}(k) = h_{H_2}(k)$ for all components $k \notin \{i, j\}$. Let $h_{H_1}(i) \neq h_{H_1}(j)$ and let the health of components i and j be reversed in H_2. Then, for every behavior B_1, there is at least one behavior B_2 such that $\mathcal{I}_{\sigma,B_1,H_1}$ is indistinguishable by observation from $\mathcal{I}_{\sigma,B_2,H_2}$.*

Theorem 4 states that a system with observations that have components i and j in the same accuracy group, regardless of whether the compoenents are observable or not, is indistinguishable from a system in which the healths of i and j are reversed. This theorem states that, associated with each health state, there may be different behavior ($B_2 \neq B_1$). However, the *observable* behavior is not changed, otherwise the systems would not be indistinguishable. Because this is a weaker condition that the one in Theorem 3, which says that the behavior for both health states may be *exactly* the same, this theorem is named "weak".

PROOF. This theorem essentially states that, for every B_1, there is a B_2 such that, for all $y' \in allproj_\sigma$, Equation 1 holds.

For all $y' \in allproj_\sigma$, $p_{B_1}(y') = p_{B_2}(y')$ otherwise, by Theorem 1 the behaviors would be distinguishable. This leaves choosing a B_2 such that

$$\begin{aligned} &\textstyle\sum_{x \in \text{rev}_\sigma(y')} p_{B2}(x \mid y') \prod_{k \in x \setminus y'} h_{H_2}(k) \\ &= \textstyle\sum_{x \in \text{rev}_\sigma(y')} p_{B1}(x \mid y') \prod_{k \in x \setminus y'} h_{H_1}(k) \end{aligned} \quad (2)$$

For notation simplicity, let $h_i \mathrel{\widehat{=}} h_{H_1}(i) = h_{H_2}(j)$ and $h_j \mathrel{\widehat{=}} h_{H_1}(j) = h_{H_2}(i)$. It is important to note that either both components i and j appear in Equation 2 or none appear. Otherwise they would not be in the same accuracy group. Equation 2 can be rewritten as:

$$\begin{aligned} &\alpha_2 C + \beta_2 Dh_i + \gamma_2 Eh_j + \delta_2 Fh_i h_j \\ &= \alpha_1 C + \beta_1 Dh_i + \gamma_1 Eh_j + \delta_1 Fh_i h_j \end{aligned} \quad (3)$$

The terms C, D, E, and F depend only on the healths in H_1 and H_2 but not on h_i nor on h_j so they are equal in both sides. The terms α_k, β_k, γ_k and δ_k depend only on B_1 and B_2. By Theorem 2, as long as the values in α_2, β_2, γ_2 and δ_2 add up to 1, we can choose whatever values we want as it will not affect the visible behavior.

Because $\alpha_k + \beta_k + \gamma_k + \delta_k = 1$, we can see these as weights for the other terms. We can choose $\alpha_2 = \alpha_1$ and $\delta_2 = \delta_1$ eliminating these terms from the equation. The remaining equation is:

$$\beta_2 Dh_j + \gamma_2 Eh_i = \beta_1 Dh_i + \gamma_1 Eh_j$$

In this equation, β_1 and γ_1 are fixed (they are determined by $B1$) and we need to find out if there is a β_2 and a γ_2 (that define $B2$) that can solve this equation. It may happen that either D or E (or both) are zero but in either case solving the equation is trivial.

Because β_2 and γ_2 are weighting Dh_j and Eh_i, this equation has no solution if and only if $Dh_j < \beta_1 Dh_i + \gamma_1 Eh_j$ and if $Eh_i < \beta_1 Dh_i + \gamma_1 Eh_j$. Solving these two inequations for h_i and h_j yields:

$$h_j < \frac{\beta_1}{1-\gamma_1 \frac{E}{D}} h_i$$
$$h_i < \frac{\gamma_1}{1-\beta_1 \frac{D}{E}} h_j$$

Replacing h_i on the top equation yields $h_j < \frac{\beta_1}{1-\gamma_1 \frac{E}{D}} \frac{\gamma_1}{1-\beta_1 \frac{D}{E}} h_j$. With some algebraic manipulation this leads to $1 + \beta_1 \frac{D}{E} + \gamma_1 \frac{E}{D} < 0$ which is impossible as β_1, γ_1, D and E are all drawn from $[0, 1]$.

This means that there is *always* a β_2 and γ_2 that make the equation $\beta_1 Dh_i + \gamma_1 Eh_j = \beta_2 Dh_j + \gamma_2 Eh_i$ true, proving the theorem. □

4.4.4 Minimum Accuracy Theorem

A *single fault state* of a system is a state in which a single component, either observable or not, has health < 1.

THEOREM 5 (MINIMUM ACCURACY THEOREM). *Let σ be a system, B_1 its behavior and H_1 its single fault state. Let i be the faulty component. If component i is used in B_1, then let H_2 be an arbitrary single-fault state such that there is a B_2 that makes $\mathcal{I}_{\sigma,B_1,H_1}$ indistinguishable from $\mathcal{I}_{\sigma,B_2,H_2}$. H_2 will have a single fault in the same accuracy group.*

The minimum accuracy theorem, Theorem 5, states that we can *always* identify the accuracy group that contains the faulty component in a single fault state. It states that any two single-fault states that generate the same observable behavior will necessarily have faults in the same accuracy group. Together with the weak accuracy theorem, Theorem 4, it states that in a single fault system, it is *always and only* possible to identify the accuracy group of the component that has failed.

PROOF. Let H_2 be a single fault state with faulty component j which is not in the same accuracy group of i.

Because i and j are not in the same accuracy group, then $\text{partproj}_\sigma(i) \neq \text{partproj}_\sigma(j)$, otherwise they would be in the same accuracy group. Then there exists either a spectrum in $\text{partproj}_\sigma(i)$ which does not exist in $\text{partproj}_\sigma(j)$ or the other way around. Lets consider both cases separately.

- If there is a projected spectrum $y \in \text{partproj}_\sigma(i)$ such that $y \notin \text{partproj}_\sigma(j)$, then $p_{B_1}(y) > 0$, meaning that eventually a spectrum y is observed with a failure evaluation $\perp$. Since $y \notin \text{partproj}_\sigma(j)$, then this behavior cannot be reproduced with a single fault in j.
- If there is a projected spectrum $y \in \text{partproj}_\sigma(j)$ such that $y \notin \text{partproj}_\sigma(i)$, then $p_{B_2}(y) > 0$. This means that, in the single fault scenario with H_2, y will eventually be observed with a failure evaluation of $\perp$ which cannot be reproduced with a single fault in i.

□

4.5 A Priori and A Posteriori Analyses

Accuracy analysis, as described so far, uses *a priori* knowledge of the system to determine accuracy groups. If this knowledge does not exist, it is not possible to pinpoint non-observable components because we have no information about them.

However, the strong accuracy theorem (Theorem 3) and the minimum accuracy theorem (Theorem 5) are still useful as they may significantly reduce the search space for solutions.

Without a model of all_σ, the accuracy groups cannot be computed. However, the law of large numbers guarantees that, as the number of observations increase, our estimates of $p_B(y)$ converge to the actual values and we will eventually identify all possible projected spectra.

This means that, *a posteriori* we can compute the accuracy groups using an *estimate* of all_σ, which is the set of observed projected spectra. For example, even if the data in Table 1 were unavailable, we would observe the MOS being used with the ADS performing computations, the ADS by itself, the FT with the MOS and the MOS with TC.1 and TC.2, effectively allowing us to reconstruct Table 3. Naturally we would not be able to observe MOSDB and ADSDB and, consequently, Table 1 is not reconstructable from observation. Hence the limitation the *a posteriori* analysis.

4.6 Implementation Algorithm

The computations described in the previous sections lead to an algorithm that computes the accuracy groups of a system σ given its spectra, all_σ, and the list of observable (obs_σ) and non-observable ($nobs_\sigma$) components. The *pizzicato* algorithm presented below performs the same computations we've done in the previous sections. It first computes PP, the participating projections. This corresponds to what was done in Table 4. The transformation done for Table 3 is performed implicitly. Then it computes AG, the accuracy groups, corresponding to what was done in Table 5.

```
function PIZZICATO(all_σ, obs_σ, nobs_σ)
    PP ← ∅
    for all s ∈ all_σ do
        s' = s ∩ obs_σ
        for all i ∈ s do
            if ¬(∃(a, b) : PP | a = i) then
                PP(i) ← ∅
            end if
            PP(i) ← PP(i) ∪ {s'}
        end for
    end for
    AG_t ← ∅
    for all (i, pp) ∈ PP do
        if ¬(∃(a, b) : AG_t | a = pp then
            AG1(pp) = ∅
        end if
        AG_t(pp) ← AG_t(pp) ∪ {i}
    end for
    AG ← ∅
    for all (pp, ag) ∈ AG_t do
        AG ← ag
    end for
    return AG
end function
```

The *pizzicato* algorithm has linear complexity in the number of components and the number of spectra. Its complexity is $O(M \times N)$ where $M = \#comps_\sigma$ and $N = \#all_\sigma$ making it a fast-performing algorithm even for very large systems.

5. RELEVANCE FOR FAULT LOCALIZATION

The accuracy theorems, described above, establish accuracy limits on fault localization. However, they do not define *how* to perform localization.

Table 6: Example spectra

FT	MOS	ADS	TC.1	TC.2	R1	R2
-	X	X	-	-	$\top$	$\bot$
-	-	X	-	-	$\top$	$\top$
-	-	X	-	-	$\top$	$\top$
-	-	X	-	-	$\top$	$\top$
X	X	-	-	-	$\top$	$\top$
-	X	-	X	X	$\bot$	$\top$
-	X	-	X	X	$\top$	$\top$

Table 7: Results of running example of Table 6 through Barinel [4] without *pizzicato* preprocessing

Scenario	Candidate	Rank
R1	TC.1	$\approx 41\%$
R1	TC.2	$\approx 41\%$
R1	MOS	$\approx 17\%$
R2	MOS	50%
R2	ADS	50%

Several algorithms exist that perform statistical fault localization, e.g., Tarantula [22] or Barinel [4]. We argue that the results presented in Section 4, plus the *pizzicato* algorithm defined in Section 4.6, *complement* existing fault localization algorithms by: (1) reducing the size of the search space for faults and increasing the accuracy of the output and (2) allowing *a priori* detection of indistinguishable faults, enabling decisions on probe placement and test generation. In the following sections, we explore these two enhancements.

5.1 Reducing Search Space and Increasing Accuracy

The first enhancement is the reduction of the search space by removing indistinguishable components. This is done by replacing spectra of components with spectra of accuracy groups before fault localization. Because several fault localization algorithms do not perform this analysis, they have to process larger sets and produce worse outputs. As shown by Theorem 4, it is not possible to pinpoint specific components in accuracy groups so either fault localization algorithms will not pinpoint the correct components or they will with less confidence.

Consider, for example, the set of spectra in Table 6 (only observable components considered) which could be observed from the system described in Section 2. Results in column R1 were obtained when TC.1 is non-healthy and results in column R2 were obtained when MOS was unhealthy. Both runs are single-fault scenarios which we ran through Barinel [4], an algorithm which is optimal for this case (see later in this section for some more information on Barinel). Table 7 contains the results when running Barinel directly on the data in Table 6 and Table 8 contains the results when running Barinel after processing Table 6 through *pizzicato*.

These results illustrate that, in this case, with *pizzicato* preprocessing, the confidence on the first scenario is much greater because there are fewer components to "split" the rank. In fact, if the number of samples is increased, the confidence without *pizzicato* preprocessing will stabilize assigning blame to TC.1 and TC.2 at 50%/50%, while with *pizzicato* it will increase to 100%.

The second scenario illustrates this in comparison with the previous scenario. Here we also have a 50%/50% split but this is due to the lack of observations. Since the only scenario we have a failure in is the first, in which both MOS and ADS were used together, both have the same probability.

Essentially, preprocessing the output with *pizzicato* allows the diagnostic system to distinguish when more information will eventually pinpoint the source of failure (Scenario 2) or when it is not useful to wait for more information as failures are indistinguishable. This is especially useful when using a measure such as entropy [30] to decide whether to observe more spectra or produce a diagnosis based on the current set of spectra, as we have shown in [7].

Table 8: Results of running example of Table 6 through Barinel [4] *with pizzicato* preprocessing

Scenario	Candidate	Rank
R1	(TC.1,TC.2)	$\approx 71\%$
R1	MOS	$\approx 29\%$
R2	MOS	50%
R2	ADS	50%

Some Details on Barinel

Barinel is described in more detail in [2]. Here we provide some basic description of the algorithm so that the values in Table 7 and Table 8 can be more easily understood.

Let D denote a set of fault candidates expressing a *diagnosis*. For instance, $D = \langle\{c_1, c_3, c_4\}\rangle$ indicates that components c_1, c_3, and c_4 are *simultaneously* at fault, and no other. Barinel sorts the faulty candidates in D by the probability that each candidate explains the fault. As with most spectrum-based reasoning, it is comprised of two phases: candidate generation and candidate ranking.

The problem of finding fault candidates can be defined in terms of the widely-known minimal hitting set (MHS) problem [15]. The precise computation of MHS is highly demanding [19], restricting its direct usage for diagnosis. However, in practice, previous research has found that precise computation of D is not necessary [1]. *Staccato* is a low-cost heuristic for computing a relevant set of multiple-fault candidates.

Candidate Ranking.

The candidate generation phase may result in an extensive list of diagnosis candidates. As not all candidates have the same probability of being the true fault explanation, techniques have been devised to assign a probability to each diagnosis candidate d_k. Each candidate d_k is a subset of the system components that, when at fault, can explain the faulty behavior. The probability of a diagnosis candidate being the true fault explanation, $\Pr(d_k \mid obs)$, given a number of observations obs, is computed using Bayesian probability updates. An observation obs is a tuple (a_{i*}, e_i).

Executing each test case t_i from the test suite T, the probability of each candidate is updated following Bayes' rule

$$\Pr(d_k \mid obs) = \Pr(d_k) \cdot \prod_{obs_i \in obs} \frac{\Pr(obs_i \mid d_k)}{\Pr(obs_i)}$$

$\Pr(d_k)$ is the *a priori* probability of the candidate (i.e., the probability before any test is executed), defined as $\Pr(d_k) = p^{|d_k|} \cdot (1-p)^{M-|d_k|}$ where p is the a priori probability of a component being faulty. The prior probability given no observations is an approximation to 1 fault for every 1000LOC, $1/1000 = 0.001$ [6].

$\Pr(obs_i \mid d_k)$ represents the conditional probability of the observed outcome e_i produced by a test t_i (obs_i), assuming that candidate d_k is the actual diagnosis

$$\Pr(obs_i \mid d_k) = \begin{cases} 0 & \text{if } e_i \wedge d_k \text{ are inconsistent;} \\ 1 & \text{if } e_i \text{ is unique to } d_k; \\ \varepsilon & \text{otherwise.} \end{cases}$$

where ε is defined as

$$\varepsilon = \begin{cases} \prod_{j \in d_k \wedge a_{ij}=1} h_j & \text{if } e_i = 0 \\ 1 - \prod_{j \in d_k \wedge a_{ij}=1} h_j & \text{if } e_i = 1 \end{cases}$$

and a_{ij} represents the coverage of the component j when the test i is executed. As this information is typically not available, the values for $h_j \in [0, 1]$ are determined by maximizing $\Pr(obs \mid d_k)$ using maximum likelihood estimation. To solve the maximization problem, a simple gradient ascent procedure [5] (bounded within the domain $0 < h_j < 1$) is applied.

$\Pr(obs_i)$ represents the probability of the observed outcome, independently of which diagnostic explanation is the correct one and thus needs not be computed directly. The value of $\Pr(obs_i)$ is a normalizing factor given by $\Pr(obs_i) = \sum_{d_k \in D} \Pr(obs_i \mid d_k) \cdot \Pr(d_k)$.

The *Barinel* algorithm is used to compute the probabilities of each diagnosis candidate d_k.

5.2 Enabling Probe Placement and Test Generation

When using the accuracy analysis we can detect which components are indistinguishable. This indistinguishability may not be fundamental: it may only be due to lack of probing. In the example of Section 2, components TC.1 and TC.2 are indistinguishable because they are both invoked from the MOS in the same computation.

However, placing additional probes may allow splitting the communication with TC.1 and TC.2, effectively splitting the spectrum in two: one with MOS and TC.1 and another with MOS and TC.2.

This decision can be made before the system is run or dynamically. At run time, as part of a self-adaptive system, it is possible to decide *not* to place the probes in the bus connections (as this will slow the system down) and accept the inaccurate diagnosis. However, after a problem has been diagnosed in the accuracy group, probes could be deployed dynamically to obtain a more accurate diagnosis.

6. APPLICATION EXAMPLE

In the previous sections we presented theoretical results on diagnosis accuracy and exemplified them on the system presented in Section 2. In this section we present a second example, based on a different computation paradigm, in order to illustrate how observability affects diagnosis accuracy. We chose a simple, well-known, example to allow for an intuitive confirmation of the theoretical results obtained.

The system to which we apply the example is a standard web system in which several clients access a load balancer which dispatches requests to one web server in a web server farm. Some requests will not require database access and some will. This system is similar to the one used in our previous work [9, 7] and its architecture is depicted in Figure 2.

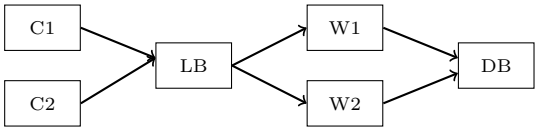

Figure 2: Example web system.

Table 9: Spectra of computations performed in the system of Figure 2.

Compt.	C1	C1-LB	C2	C2-LB	LB	LB-W1	W1	LB-W2	W2	W1-DB	W2-DB	DB
SR1W1	X	X	-	-	X	X	X	-	-	-	-	-
SR2W1	-	-	X	X	X	X	X	-	-	-	-	-
SR1W2	X	X	-	-	X	-	-	X	X	-	-	-
SR2W2	-	-	X	X	X	-	-	X	X	-	-	-
DR1W1	X	X	-	-	X	X	X	-	-	X	-	X
DR2W1	-	-	X	X	X	X	X	-	-	X	-	X
DR1W2	X	X	-	-	X	-	-	X	X	-	X	X
DR2W2	-	-	X	X	X	-	-	X	X	-	X	X
P1	-	-	-	-	X	X	X	-	-	-	-	-
P2	-	-	-	-	X	-	-	X	X	-	-	-

Table 9 contains the spectra of all computations performed in the system of Figure 2. We represent not only the components but also the connectors between them. SR1xx and SR2xx correspond to the static web requests made by clients, which do not require the database to be used. Static web requests are usually HTML pages or images. DR1xx and DR2xx correspond to dynamic web requests made by clients. In these requests, the web server needs to fetch data from the database. The Px requests are pings made by the load balancer to check whether the web servers are still available.

One first scenario, σ_1 is the "external" view of the system, when only the clients are observable. In this case, the accuracy groups become: $\{C1, C1\text{-}LB\}$, $\{C2, C2\text{-}LB\}$, $\{LB, LB\text{-}W1, W1, LB\text{-}W2, W2, W1\text{-}DB, W2\text{-}DB, DB\}$

This result means that, for σ_1, we cannot distinguish between a failure in the client and its connection to the load balancer, we can distinguish between problems in different clients and we can identify problems from the load balancer onwards. We cannot pinpoint any specific component from the load balancer onwards, which makes sense because we will know whether the system is responding correctly or not but not which piece is responsible for the failure.

Another scenario, σ_2 is a view of the system administrator when only the load balancer is being observed. Here, LB is observable and no other component or connector is. This scenario will yield a single accuracy group with all components inside. While it appears uninteresting, this approach is commonly used in practice, where load balancers are observed to look for failures. When a failure is identified it will be necessary to deploy more probes to localize a problem.

A more interesting scenario is when we observe W1 and W2 (the results are the same if (1) we observe LB-W1 and LB-W2 instead and also do if (2) LB is observed). Here the accuracy groups become: $\{W1, LB\text{-}W1, W1\text{-}DB\}$, $\{W2, LB\text{-}W2, W2\text{-}D$
$\{C1, C1\text{-}LB, C2, C2\text{-}LB, LB, DB\}$

In this scenario we can pinpoint faults in the web servers (or their connections) or in everything. However, if we add C1-LB and C2-LB to the observable components, meaning,

in practical terms, that we probe the input of the load balancer, the accuracy groups become: {C1, C1-LB, }, {C2, C2-LB}, {W1, LB-W1, W1-DB}, {W2, LB-W2, W2-DB}, {LB, DB}

In this scenario we can now distinguish failures in clients and web servers although we cannot tell the difference between failures in the load balancer and the database. While this may seem unintuitive, it is an expected result: a scenario in which the load balancer fails half the time is equal to a scenario in which the database always fails and half of the requests are dynamic.

One aspect of this example that may appear unclear is that we always assume that all computations terminate normally, meaning that the load balancer always returns a reply to a request made by a client. Handling non-responsiveness requires adding new spectra with only the involved components. For example, to handle non-responsiveness of the load balancer would require a computations that include C1,C1-LB and LB and another one that includes C2,C2-LB and LB. Addition of these computations would make the LB identifiable as a fault in its own failure group because now, if both C1 and C2 have a non-responsive failure from LB, the problem must lay in the common component, LB.

7. RELATED WORK

Recent work has already been done in the area related to diagnosis accuracy. In [24] probe placement and its impact on diagnosis is directly discussed. The concept of *ambiguity group* is introduced (equivalent, more or less to our *accuracy group*) and ways to introduce probing are proposed. However, only sequential computations are considered, e.g., computations in which the components of the group are exercised sequentially. Thus our work generalizes that proposed in [24] to any computation model. [10] addresses how to improve SFL accuracy in tightly coupled systems by splitting components into smaller components, although probing itself is not discussed as all components are assumed to be observable.

More generally, in the area of self-adaptive systems (SAS), a significant amount of research has been done. Several multi-goal optimization frameworks for self-adaptive systems exist that could benefit from diagnosis accuracy to drive system adaptation. Rainbow [12, 20] uses software architecture as a backbone for SAS and Zanshin [31] uses goal-oriented requirements. Several surveys and roadmaps exist that refer some of the many approachs to SAS: [11, 29, 16]. However, none of this work has directly tackled the problem of diagnosis for unobservable components.

Spectrum-based fault localization, the core algorithms that pinpoint the accuracy groups at fault, also have a variety of approaches, including algorithmic approaches (e.g., [22, 2]), simularity coefficient calculation (e.g., [14] and those listed in [3]), and the use of neural networks [33]. Our work complements these approaches by reducing the size of the candidates that need to be considered by these approaches, and increasing the accuracy of their output.

Furthermore, diagnosis accuracy enables reasoning about diagnosis uncertainty and has the potential to be used by AI planners to decide where to place probes. A discussion of planning under uncertainty can be found in the Cassandra Planner work [13].

8. CONCLUSIONS AND FUTURE WORK

This paper provides a definition of diagnosis accuracy in the presence of unobservable components, and an algorithm to compute it. Diagnosis accuracy is defined as a partition of the system's components into groups: components in the same group cannot be individually identified as faulty. We provided a theoretical proof that these groups establish a maximum accuracy bound and that they are also the minimum accuracy bound in the case of a single-fault scenario.

These results allow a diagnostic system to define which probes to have on a system and identify what they are capable of pinpointing in the event of a failure. Further, they provide a basis for dynamically controlling probing on a self-adaptive system.

We plan to continue future work to extend these results along three dimensions. First, this work can be used to extend current state of the art for self-adaptive systems by including formal reasoning about dynamic probe placement and dynamic testing. Dynamic probe placement controls when probes can be placed in a system in order to achieve an optimum balance between probing cost and diagnosis accuracy. Dynamic testing can be used to produce additional computations and improve diagnosis. For example, if we are unsure whether the MOSDB or the MOS are faulty, we can explicitly exercise just one of them.

Second, the theoretical results in this paper can be improved under some conditions. In the approach taken by this paper, we have assumed that components are either observable or non-observable, but, especially in the presence of dynamic probe placement, components may be observable at some times but not others. This may also occur if probes cannot always deterimine whether a component participated in a computation. For example, a file system probe may be able to detect when a component was used only if the component uses the file system. With the current approach, we need to disregard knowledge of the observations of these components and consider them as non-observable, throwing away useful information that could help us to provide a more accurate diagnosis. We believe it is possible to extend the theory to consider these sporadic observations.

Third, although in this paper we explored the application of diagnosis accuracy in the context of self-adaptive systems, the results presented here are potentially much more general and may also apply to design time. Design-time code instrumentation is akin to probe placement at run time and our approach would allow (1) determining what needs to be instrumented, and (2) when there are enough test cases to distingush bugs in different parts of the code. Additionally, *pizzicato*'s reduction of the search space for faults may improve the performance of existing SFL algorithms.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant CNS 1116848, the Army Research Office under Award No. W911NF-09-1-0273, the Office of Naval Research under Grant N000141310401 and Samsung Electronics. Furthermore we would like to thank Jungsik Ahn, of Samsung Electronics, who helped us define the simulator described in Section 2.

9. REFERENCES

[1] R. Abreu and A. J. C. van Gemund. A low-cost approximate minimal hitting set algorithm and its

application to model-based diagnosis. In V. Bulitko and J. C. Beck, editors, *Proceedings of the 8th Symposium on Abstraction, Reformulation and Approximation (SARA'09)*, Lake Arrowhead, California, USA, 8 – 10 July 2009. AAAI Press.

[2] R. Abreu and A. J. C. van Gemund. Diagnosing multiple intermittent failures using maximum likelihood estimation. *Artif. Intell.*, 174(18):1481–1497, 2010.

[3] R. Abreu, P. Zoeteweij, and A. J. C. V. Gemund. An Evaluation of Similarity Coefficients for Software Fault Localization. In *Pacific Rim International Symposium on Dependable Computing*, pages 39–46, 2006.

[4] R. Abreu, P. Zoeteweij, and A. J. C. v. Gemund. Spectrum-based multiple fault localization. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, pages 88–99, Washington, DC, USA, 2009. IEEE Computer Society.

[5] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Dover Books on Computer Science Series. Dover Publications, 2003.

[6] J. Carey, N. Gross, M. Stepanek, and O. Port. Software hell. pages 391–411, 1999.

[7] P. Casanova, D. Garlan, B. Schmerl, and R. Abreu. Diagnosing architectural run-time failures. In *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 20-21 May 2013.

[8] P. Casanova, D. Garlan, B. Schmerl, R. Abreu, and J. Ahn. Applying autonomic diagnosis at samsung electronics. Technical Report CMU-ISR-13-111, Carnegie Mellon University, Sep 2013.

[9] P. Casanova, B. R. Schmerl, D. Garlan, and R. Abreu. Architecture-based run-time fault diagnosis. In I. Crnkovic, V. Gruhn, and M. Book, editors, *ECSA*, volume 6903 of *Lecture Notes in Computer Science*, pages 261–277. Springer, 2011.

[10] C. Chen, H.-G. Gross, and A. Zaidman. Improving service diagnosis through increased monitoring granularity. In *Proceedings of the 7th International Conference on Software Security and Reliability (SERE)*, 18-29 June 2013.

[11] B. H. Cheng et al. In B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, and J. Magee, editors, *Software Engineering for Self-Adaptive Systems*, chapter Software Engineering for Self-Adaptive Systems: A Research Roadmap, pages 1–26. Springer-Verlag, Berlin, Heidelberg, 2009.

[12] S.-W. Cheng, D. Garlan, and B. Schmerl. Architecture-based self-adaptation in the presence of multiple objectives. In *Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems*, SEAMS '06, pages 2–8, New York, NY, USA, 2006. ACM.

[13] G. Collins and L. Pryor. Planning under uncertainty: Some key issues. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1567–1573, 1995.

[14] A. da Silva Meyer, A. A. F. Farcia, and A. P. de Souza. Comparison of similarity coefficients used for cluster analysis with dominant markers in maize (zea mays l). *Genetics and Molecular Biology*, 27(1):83–91, 2004.

[15] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artif. Intell.*, 32(1):97–130, Apr. 1987.

[16] R. de Lemos et al. Software Engineering for Self-Adaptive Systems: A second Research Roadmap. In R. de Lemos, H. Giese, H. Müller, and M. Shaw, editors, *Software Engineering for Self-Adaptive Systems*, number 10431 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2011. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[17] A. Feldman, G. M. Provan, and A. J. C. van Gemund. Computing minimal diagnoses by greedy stochastic search. In D. Fox and C. P. Gomes, editors, *AAAI*, pages 911–918. AAAI Press, 2008.

[18] A. Feldman and A. van Gemund. A two-step hierarchical algorithm for model-based diagnosis. In *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, AAAI'06, pages 827–833. AAAI Press, 2006.

[19] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[20] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste. Rainbow: architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10):46–54, 2004.

[21] A. Gonzalez-Sanchez, E. Piel, H.-G. Gross, and A. van Gemund. Prioritizing tests for software fault localization. In *Quality Software (QSIC), 2010 10th International Conference on*, pages 42–51, 2010.

[22] J. A. Jones and M. J. 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, New York, NY, USA, 2005. ACM.

[23] J. Kephart and D. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[24] C. Landi, A. van Gemund, and M. Zanella. Test oracle placement in spectrum-based fault localization. In *Proceedings 24th International Workshop on Principles of Diagnosis: DX-2013*, October 1-4 2013. To appear.

[25] W. Mayer and M. Stumptner. Evaluating models for model-based debugging. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, ASE '08, pages 128–137, Washington, DC, USA, 2008. IEEE Computer Society.

[26] E. Piel, A. Gonzalez-Sanchez, H. Gross, and A. J. Van Gemund. Spectrum-based health monitoring for self-adaptive systems. In *Self-Adaptive and Self-Organizing Systems (SASO), 2011 Fifth IEEE International Conference on*, pages 99–108. IEEE, 2011.

[27] É. Piel, A. Gonzalez-Sanchez, H.-G. Gross, and A. J. van Gemund. Online fault localization and health monitoring for software systems. In *Situation Awareness with Systems of Systems*, pages 229–245. Springer, 2013.

[28] É. Piel, A. Gonzalez-Sanchez, H.-G. Gross, A. J. van Gemund, and R. Abreu. Online spectrum-based fault

localization for health monitoring and fault recovery of self-adaptive systems. In *ICAS 2012, The Eighth International Conference on Autonomic and Autonomous Systems*, pages 64–73, 2012.

[29] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1–14:42, May 2009.

[30] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. Univ of Illinois Press, 1949.

[31] V. Souza and J. Mylopoulos. From awareness requirements to adaptive systems: A control-theoretic approach. In *Requirements@Run.Time (RE@RunTime), 2011 2nd International Workshop on*, pages 9–15, 2011.

[32] D. Sykes, D. Corapi, J. Magee, J. Kramer, A. Russo, and K. Inoue. Learning revised models for planning in adaptive systems. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 63–71, Piscataway, NJ, USA, 2013. IEEE Press.

[33] W. E. Wong and Y. Qi. BP neural network-based effective fault localization. *International Journal of Software Engineering and Knowledge Engineering*, 19(04):573–597, 2009.

[34] F. Wotawa, M. Stumptner, and W. Mayer. Model-based debugging or how to diagnose programs automatically. In T. Hendtlass and M. Ali, editors, *Developments in Applied Artificial Intelligence*, volume 2358 of *Lecture Notes in Computer Science*, pages 746–757. Springer Berlin Heidelberg, 2002.