

# From Testing to Diagnosis: An Automated Approach

Franck Fleurey and Yves Le Traon  
IRISA, Campus Universitaire de Beaulieu,  
35042 Rennes Cedex, France  
{ffleurey, yletraon}@irisa.fr

Benoit Baudry  
CEA-List, 91191  
Gif-sur-Yvette Cedex, France  
benoit.baudry@cea.fr

## Abstract

*The need for testing-for-diagnosis strategies has been identified for a long time, but the explicit link from testing to diagnosis is rare. Here, we start with the study of an algorithm for fault localization that consists of cross-checking information collected from test cases execution traces. Analyzing the type of information needed for an efficient localization, we identify the attribute (called Dynamic Basic Block) that restricts the accuracy of a diagnosis algorithm. Based on this attribute, a test criterion is proposed and validated through rigorous case studies: it shows that test cases can be completed to reach a high level of diagnosis accuracy. So, the dilemma between a reduced testing effort (with as few test cases as possible) and the diagnosis accuracy (that needs as much test cases as possible to get more information) is partly solved by selecting only test cases relevant for diagnosis.*

## 1. Introduction

In practice, no clear continuity exists between the testing task and the diagnosis one. While the former aims at generating test data and oracles with a high fault-revealing power, the latter uses all available symptoms (e.g. traces) coming from testing to locate the detected faults. The richer the information coming from testing is, the more precise the diagnosis may be. This need for testing-for-diagnosis strategies is mentioned in the literature [1-3], but the explicit link from testing to diagnosis is rare. The usual assumption states that test cases satisfying a chosen test adequacy criterion are sufficient to perform diagnosis [2]. This assumption is verified neither by specific experiments nor by intuitive considerations. Indeed, reducing the testing effort implies generating a minimal set of test cases for reaching the given criterion. On the other hand, good diagnosis supposes maximizing information coming from testing for a precise cross-checking and fault localization. Both objectives seem contradictory, and, for example, the good diagnosis results obtained by Jones et al. [1] are reached thanks to a large amount of input test data.

In terms of scalability, the question of choosing test data guided by diagnosis criteria is crucial if we consider the application of thousands of input test

cases for a system as unrealistic (e.g., because of the oracle production, the execution time...). In this paper, we deal with OO (sub)-system testing in the sense that the test and diagnosis are not applied at unit level but on a set of interconnected classes. The main objective is to reduce the number of test data, still achieving high diagnosis accuracy.

To reach this objective, we studied existing fault localization techniques and identified an important feature that contributes to the reduction of the diagnosis analysis effort. This feature is called *Dynamic Basic Block* (DBB) and relates the properties of test cases and the adequacy of the diagnosis. A test adequacy criterion is then defined according to this notion of DBB. The experimental validation consists in generating test cases according to this criterion and the quality of these test cases for diagnosis is estimated using mutation analysis.

The idea is to automatically enhance the existing test cases to locate detected faults.

This paper is organized in 4 sections. Section 2 presents the principles of automatic fault localization and details a few related work. Section 3 identifies the notion of dynamic basic block as the key property of the tests to ensure the efficiency diagnosis and defines a corresponding test criterion. Section 4 is dedicated to an experimental validation of this criterion. Finally section 5 presents the conclusions of this work and suggests some possible future work.

## 2. Reducing the diagnosis effort

When an error is detected by a test case, the diagnosis consists in locating the source of this error in the code. Several techniques are presented in the literature to help the programmer locate faults. These techniques mainly consist in using data coming from the test process to select a reduced set of "suspicious" statements. This paper focuses on fault localisation techniques [1-4] that use as an input statement coverage (slices) and verdicts (either pass or fail) for each test case (this set of information is called the *diagnosis matrix*). For example, Figure 1 displays

execution traces for four different test cases (TC1,...,TC4) executed on a program composed of 7 statements, as well as the verdict for the test cases. The grey blocs correspond to statements that are covered by a test case; e.g., test case 1 covers statements 1,2,4,5,6.

TC1	TC2	TC3	TC4
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
P	P	F	P

**Figure 1 - execution traces**

The intuition of the diagnosis techniques is that a failed test case necessarily executes a faulty statement while a passed test case might not. In practice, there are two important issues: a test case may pass even if it executes a faulty statement (*non ideal verdicts*) and there may multiple faults in one execution trace of *multiple faults*.

In [1] the idea is to cross-check all the test case slices in order to associate a suspicion level to each statement of the program under test. The level of suspicion associated to a statement is related to the number of times a statement is included in the trace of a failed test case, as well as the number of times this statement as been covered by a test case. The assumption being that a statement is more likely to be faulty of it is present in many failed test cases traces. Moreover, this suspicion estimation is more reliable if it is computed with a large number of traces. This diagnosis technique has been chosen for the experimental validation of the work presented in this paper. Other diagnosis techniques propose heuristics for fault localisation based on the diagnosis matrix [2-4].

The relevance of the results of a fault localisation algorithm can be estimated by the number of statements one has to examine before finding a fault. Thus, we define the *diagnosis accuracy* as the number of statements to be examined before finding the actual faulty statement and the *relative diagnosis accuracy* as the corresponding percentage of the source code of the program to examine.

### 3. From test to diagnosis

All diagnosis algorithms presented in previous section are based on cross-checking the execution traces of test

cases. As such, intuitively, a strong dependency exists between the test cases and the quality of the diagnosis produced by these algorithms; the diagnosis should be as accurate as the number of test cases is high. Unfortunately, this idea is contradictory with test generation practices which consist of minimizing the number of test cases to reach a given test criterion. This section discusses several test adequacy criteria in order to fit the diagnosis requirements.

#### 3.1 Code coverage based criteria

In order to detect and locate faults anywhere in the program, any diagnosis technique requires each statement to be covered by the tests. So, a first, minimal, test criterion for an accurate diagnosis is *statement coverage*. However, this criterion is known to be quite poor in terms of fault detection and may also be inadequate for diagnosis. As an extreme example, if a single test case covers all the code of the program under test then the statement coverage criterion is satisfied but as the slice of the test case contains all the statements of the program it is helpless for fault localisation. Thus, simple code coverage is necessary but certainly not sufficient to ensure an accurate diagnosis.

To allow an accurate diagnosis, the inputs of the diagnosis algorithm, i.e. the slices of the test cases should allow distinguishing statements from each others, especially the faulty one from each other. Intuitively, to achieve that each statement of the program under test should be covered by several test cases. This leads to the definition of a second test criterion for diagnosis: covering at least N times each statement of the program with different test cases. In the following we call this criterion *N-Coverage* (1-Coverage corresponds to simple statement coverage). The intuition is that the higher is N, the more statements the test cases may be able to distinguish.

However none of these criteria focus on the specificity needed for good fault localization: the algorithm needs test cases that enable distinguishing statements one from the other. The next section introduces an original test adequacy criterion directly based on statement distinction.

#### 3.2 Distinguishing statements

The diagnosis objective being to pinpoint faulty statements in a program, any diagnosis technique requires being able to distinguish the statements of the

program under test and to pinpoint “suspicious” statements. An ideal diagnosis algorithm would be able to distinguish each statements of the program and select single statements as faulty. Unfortunately, since statements are distinguished based on their belonging to an execution trace, it is only possible to distinguish blocks of statements. Indeed, along an execution path, there exist sequential blocks of statements between control points. If a test case covers a statement in such a sequential block, it covers all the statements. We call these statements that can not be distinguished by the execution traces, indistinguishable statements. In the particular context of diagnosis algorithms, statements are indistinguishable if they are covered by strictly the same test cases. The notion of *dynamic basic block* is introduced to explain the link between the test cases and the accuracy of diagnosis.

**Dynamic basic block.** *Let  $P$  be the program under test and  $TS$  a test suite. A dynamic basic block is the set of statements of  $P$  that is covered by strictly the same test cases of  $TS$ . The set of dynamic basic blocks in  $P$ , distinguished by  $TS$ , is denoted  $B(TS)$ .*

For example, **Erreur ! Source du renvoi introuvable.** distinguishes 4 blocks  $\{(1, 2), (3, 7), (4), (5, 6)\}$ .

From this definition, it appears that if one statement is “suspected” by the diagnosis algorithm, every statement in the same DBB is also suspected. Thus, the fault localisation algorithm, instead of being able to individually select suspicious statement can only select suspicious DBB.

In this context, let’s consider an ideal diagnosis algorithm which always selects as suspicious one and only one DBB that actually contains an error. The accuracy of such an algorithm, i.e. the number of statements the tester would have to examine before finding an actual error, only depends on the size of the DBB that contains an error and have been selected by the algorithm. With such an algorithm the diagnosis accuracy is related to the size of DBB that contain faulty statements. An important issue is thus to produce small DBBs. A criterion to select a set of test cases that allow an accurate diagnosis might be to maximize the number of distinguished DBB (in order to minimize their size) the test case. In the following section this criterion is referred as *MaxDBB*.

In practice, concrete diagnosis algorithms are not ideal because of the non ideal-verdicts issue presented in section 2. Yet, the problem of statement distinction is not specific to the ideal algorithm and the *MaxDBB*

criterion may remain relevant with a non-ideal algorithm. Next section presents an experimental investigation of the test-for-diagnosis criteria proposed in this section.

## 4. Experimental validation

To estimate the quality of the criteria described in the previous section a careful experimental procedure is defined. Firstly, sets of test cases for the system under study need to be generated to fit the N-Coverage and *MaxDBB* criteria. The test cases were automatically generated using an AI optimization algorithm defined called bacteriologic algorithm [5].

Secondly a diagnosis algorithm must be applied on a representative set of faulty versions of the system under study to estimate the average diagnosis accuracy using the generated test suites. Concerning the diagnosis algorithm, the experiment presented uses the algorithm the one proposed by Jones et al. in [1]. The faulty versions of the system under study are automatically generated by injecting faults using mutation analysis [6, 7]. Indeed, we claim that a small number of faults (even coming from real data) is not sufficient for a systematic study and replicable results.

Two studies have been completed on two different systems under study. The results presented in the following correspond to the VIRTUALMEETING system. VIRTUALMEETING (VM), is a server that simulates business meetings over network. It is made of 72 classes and 1478 executable lines of code.

Table 1 summarizes the results we obtained for 5 test suites generated according to 5 test criteria from 1-Coverage to 4-Coverage and *MaxDBB*. For each test suite this table presents on one hand the number of test cases, the number of DBB distinguished by the test suites and the average size of the DBB that contains the actual faulty statement. On the other hand, the average and standard deviation of the diagnosis accuracy for the 250 mutants of the VM system is presented.

This study confirms the intuition that using an N-Coverage criterion, the number of distinguished DBB increases and the diagnosis accuracy is better while N increases. In fact, using 1-Coverage, an average of 15 statements has to be examined to find the actual faulty statement while using 4-Coverage only 9 statements have to be examined. Yet, increasing N also significantly increases the number of test cases: from 15 test cases for 1-Coverage to 63 test cases for 4-Coverage.

**Table 1 – Results summary for the virtual meeting system**

Test suite	Criterion	Number of test cases	Code coverage	Number of dynamic basic blocks	Average size of the dynamic basic blocks	Number of statements to examine (Average/std deviation)
TS1	Coverage	15	86,60%	113	7,2	14,95 / 16,86
TS2	2-Coverage	29	86,94%	148	5,53	10,18 / 11,61
TS3	3-Coverage	47	88,56%	177	4,66	9,57 / 10,84
TS4	4-Coverage	63	88,56%	182	4,53	8,95 / 9,82
TS5	MaxDBB	31	86,94%	186	4,41	7,09 / 7,12

The *MaxDBB* criterion seems to fit better the diagnosis requirements: with only 31 test cases, the test suite TS5 generated by the bacteriologic algorithm is able to distinguish 186 DBB and leads to an average diagnosis accuracy of 7 statements. In fact, with half less test cases than in TS4 this test leads to a better diagnosis accuracy.

## 5. Conclusion

This work establishes an explicit connection between testing and diagnostic. In fact, the main contribution is the identification of a “diagnosis adequacy criterion” for test generation which is defined to ensure a satisfactory “fault locating power” with respect to the studied diagnosis techniques. This criterion consists in maximizing the number of distinguished dynamic basic blocs (*MaxDBB*). The benefits of such a criterion are two-fold:

- It allows minimizing the number of test cases required for an accurate diagnosis. It thus conciliates the actual test practices with the diagnosis requirements.
- It provides a way to automatically estimate the quality of a particular test set with respect to the diagnosis requirements.

In practice, this criterion might be used, either to select a sufficient set of test cases w.r.t. to the diagnosis requirement among a set of existing test cases or to optimize the set of existing test cases in order to fit the diagnosis requirements. An advantage of using the number of DBB as a criterion is that it does not require the verdict (and thus an explicit oracle).

The technique is experimentally validated on an OO system made of over a thousand statements. In future work, more experiments should be run on various programs to validate the efficiency of the test criterion and the benefits of the local optimization technique.

## 6. References

1. J.A. Jones, M.J. Harrold, and J. Stasko. *Visualization of Test Information to Assist Fault Localization*. In proceedings of ICSE'02. Orlando, FL, USA, May 2002. pp.467 - 477.
2. H. Agrawal, J. Horgan, S. London, and W. Wong. *Fault Localization using Execution Slices and Dataflow Tests*. In proceedings of ISSRE'95 (Int. Symposium on Software Reliability Engineering). Toulouse, France, October 1995. pp.143 - 151.
3. M. Renieris and S.P. Reiss. *Fault Localization With Nearest Neighbor Queries*. In proceedings of proceedings of the 18th IEEE International Conference on Automated Software Engineering, 2003.
4. M. Khalil, Y. Le Traon, and C. Robach. *Towards an automatic diagnosis for high-level design validation*. In proceedings of International Test Conference. Washington, DC, USA, October 1998. pp.1010 - 1018.
5. B. Baudry, F. Fleurey, J.-M. Jézéquel, and Y. Le Traon, *From Genetic to Bacteriological Algorithms for Mutation-Based Testing*. Software Testing, Verification and Reliability, 2004.
6. R. DeMillo, R. Lipton, and F. Sayward, *Hints on Test Data Selection : Help For The Practicing Programmer*. IEEE Computer, 1978. **11**(4): p. 34 - 41.
7. A.J. Offutt, A. Lee, G. Rothermel, R.H. Untch, and C. Zapf, *An Experimental Determination of Sufficient Mutant Operators*. ACM Transactions on Software Engineering and Methodology, 1996. **5**(2): p. 99 - 118.