Interdisciplinary Survey of Fault Localization Techniques to Aid Software Engineering

Árpád Beszédes

University of Szeged, Department of Software Engineering
Árpád tér 2, H-6720 Szeged, Hungary
beszedes@inf.u-szeged.hu

Abstract: Fault localization (narrowing down the cause of a failure to a small number of suspicious components of the system) is an important concern in many different engineering fields and there have been a large number of algorithmic solutions proposed to aid this activity. In this work, we performed a systematic analysis of related literature, not limiting the search to any specific engineering field, with the aim to find solutions in non-software areas that could be successfully adapted to software fault localization. We found out that few areas have significant literature, in this topic, that are good candidates for adaptation (computer networks, for instance), and that although some classes of methods are less suitable, there are useful ideas in almost all fields that could potentially be reused for software fault localization.

Keywords: faults/defects/failures; fault localization; software fault localization; literature review; method assessment

1 Introduction

Our everyday lives are driven by complex systems; we are directly interacting with some of them, while others support background technologies in diverse industrial areas [1]. These complex systems may be mechanical, electrical, software-driven, or any combination thereof, and are developed and produced by the respective engineering disciplines. These systems are often mission, safety or business critical, and every effort is made to avoid failures in them. Failures can cause damage to the environment, people's health and lives, or the operation of businesses and governments. Hence, failures and the underlying faults are a high priority concern.

Among the many different engineering areas that deal with complex systems, there is one common subtopic, the central theme of this article, *fault localization*. Without loss of generality, fault localization means identifying components (parts, modules, software code parts, etc.) of the system that are responsible for a specific observed failure. Fault localization as a discipline is given a high priority in many fields, especially in the case of highly critical systems.

In this paper, we explore semi-automatic fault localization techniques from various domains, and aim at producing an interdisciplinary analysis of the area. Our goal is specific, though. The background area is software engineering, and our research agenda deals with enhancing existing techniques and providing new approaches in the field of *software fault localization* [2] [3]. To this end, the primary goal of this survey is to provide a systematic analysis of fault localization techniques from non-software domains and discuss their possible adaptation to and implementation in software fault localization.

In any of the mentioned engineering areas, systems tend to be large and complex, and they are often connected to each other, forming even more complex systems-of-systems [4]. This has the implication that, upon occurring failures, it may be very difficult to localize their source (root cause). Hence, various fields have developed algorithmic approaches to automate the fault localization process.

Naturally, each field deals with its peculiarities and many of the techniques are domain-dependent, yet we found out that there are some similarities across disciplines. Furthermore, some of the methods are generic and could be applied, theoretically, to any engineering field and fault localization problem.

Software fault localization is a relatively young area compared to, for instance, aerospace or electronics. Yet, there is already a large literature covering many different subtopics [2] [3]. A lot of research has been performed to design effective fault localization algorithms and propose their use in different phases of the software process, most notably debugging. However, related research suggests that the practical applicability of research results in this area is still limited [5], and further research is needed to achieve more widespread use of automatic software fault localization by practitioners.

It is noticeable that existing software fault localization techniques concentrate around a relatively small number of fundamental approaches with little overlap between them [2]. This motivated the present work: to investigate other engineering fields and find out if they employ techniques that could be adapted to software and hence advance the state-of-the-art in this field.

This paper is a first attempt to investigate the applicability of fault localization methods to software from other fields; we are not aware of any similar research. Our preliminary investigations show that there are promising related approaches, but we also found that in some cases there are barriers to the adoption of such techniques. This is due to fundamental differences in how these systems (software and non-software) are described and handled (for example, if a detailed behavioral model is required). In many other cases, however, the techniques or some underlying ideas could be successfully adapted to software.

The paper is organized as follows. In Section 2, we briefly overview the terms used in the remaining parts of the paper. Section 3 deals with the assessment criteria we used for the analysis of the literature. The assessment results are presented in Section 4, while Section 5 contains their evaluation. Section 6 concludes this work.

2 Background

One of the main difficulties in a cross-disciplinary analysis of a specific topic is the diversity of the used terms. Often, the same concepts are referred to by different terms, and specific terms may have different meanings in different technological areas. In this work, we came across the following areas: software technology, computer networks, electric engineering, aerospace, among others. In the following, we overview the main constituents of a general fault localization approach, and the terminology we will use to describe it.

System and its components. Since this paper deals with many different areas, a system may refer to any complex artifact that performs a specific task [1]. It may either be a mechanical, electrical, chemical, computer software, etc. system that is composed of specific, interacting components. Often, a complex system includes components of different types, e.g. interacting mechanical and electrical, or computer based using hardware and software components. A system is often described using a domain specific *model*, which is then used in the fault localization process.

Fault. Without loss of generality, in this paper, fault refers to a defective component (or a set of defective components) of a system [6]. A fault may be defined at different granularity levels, depending on the domain and fault localization method. A fault may be present due to a design or implementation error made by a human or other external entity, or may be developed during operation by natural wear or physical damage. (This, of course, does not apply to software, for instance.)

Fault identification. This refers to the (systematic or incidental) process of discovering that there is a fault in a system. This process merely proves that there is at least one fault, and does not necessarily shows its exact location and context.

Execution and observation. A fault in a system may be identified by merely analyzing the system's components by automatic or manual means (we call this a *static* approach), or by executing (using) it and observing its behavior. Execution, in a general sense, means using the system in its intended or test environment and usage scenarios, either in its entirety or using only some of its sub-components. A fault identified in such a way will be referred to as using the *dynamic* approach. Execution and observation may mean diverse things in the case of different systems, such as real-time observing a working system in live environment, running software test cases, probing a network with test packages, etc.

Test. An individual test will mean any atomic execution of the system whose behavior can be observed, measured and interpreted. Alternatively, a system may be statically tested by analyzing the components. This, again, can be very diverse in the different domains.

Intended (or expected) behavior. This will refer to a type of execution of the system, which conforms to a set of explicit or implicit behavioral requirements. In other words, it is the behavior when all of the system's components work correctly. Some parts of the intended behavior are defined by a *behavioral model* (documentation, or formal model), while in other cases undesired behavior is documented (such as possible failure modes), or it may even refer to implicit, undocumented, expected behavior.

Failure. Based on the previous, a failure of a system should mean any observed behavior which is different from the intended one [6]. Note, that failure may mean many different things and can be classified according to severity starting from minor glitches, through functional and non-functional issues (for example, performance) to serious malfunctions. (The static fault identification does not require the manifestation of a failure.)

Fault localization. Finally, fault localization refers to any automated or semi-automated process whose goal is to select a sub-component or set of components of a complex system, which are most probably responsible for a set of observed failures or identified (but not yet localized) faults.

In the case of various domains, fault localization may mean different concrete things but a basic approach is to perform a set of tests on the system, observe its behavior and, based on the failures, use an algorithm to narrow down the possible causes to specific sub-components of the system. In this process, a behavior model may or may not be required, and in some cases the tests may be performed statically, as discussed above.

The different fault localization approaches can have various properties that determine its effectiveness and usage efficiency. In this context, *effectiveness* means how successful the method is in localizing the fault (successfulness can, in turn, mean different things but usually refers to how many of them and how precisely the location of the faults are found). *Efficiency*, on the other hand, means any practical property of the method that determines its execution time, complexity, storage requirement, or any other aspect which is important for its usability.

3 Assessment Criteria

The process for identifying the corresponding research reports and their selection was the following. In the first phase, we used general and research oriented search engines and research repositories, which included google, google scholar, ResearchGate, Mendeley, and Scopus. We did not use generic search terms like "fault localization" alone because these produced too much irrelevant results. Instead, we added specific keywords that we expected to be relevant fields for our search: networks, electronics, engineering, operations, systems, etc. We also applied different variations and synonyms to the term, which included localizing faults, failure diagnosis, problem diagnosis, error localization and similar terms.

We then restricted the search results to publicly available full-text scientific publications. We aimed at limiting the results to publications that appeared in peer-reviewed journals or conferences, however there were few exceptions such as doctoral theses and technical reports. The next filtering, we applied was to limit the list to papers that correspond to some of the following categories: software-related, generic algorithms, methods in engineering fields that we expected to be relatively easy to adapt to software-related artifacts. For example, pure mathematical methods, methods used in programming education, or approaches in non-related scientific branches like biomedicine, navigation, linguistics or other, were removed.

In the next phase, we performed a lightweight "snowballing" with the identified papers: considering the referenced works for new candidates. Finally, we consolidated the results by organizing the works by specific research groups or authors and concentrating on a few relevant reports by the same team.

In the next phase, we started the classification of the papers based on the criteria set forth in this section. In this phase, several papers also dropped out because they were difficult to categorize according to the criteria (mainly due to the *fundamental area* category as described below). Also, the criteria had to be modified slightly during this phase.

Fundamental area. The main classification direction was the fundamental area in which the method is applied. To enable easy further processing of the methods, we decided to use a very simple classification in this respect. We have the following categories: *software*, *networking*, *other engineering* and *various/generic*. The description of the methods in Section 4 is organized along these categories.

Since our goal was to identify potential approaches from other areas different from software faults, the methods we include belonging to the *software* category are only the most important, basic approaches, which are provided for reference.

We soon realized that there exists a large amount of publications that deal with fault localization in *computer networks*, hence we established a separate category for this area.

The *other engineering* category includes all methods that belong to a specific engineering field other than software or networks. In the corresponding table in Section 4, we will denote the specific field in question.

Finally, there are some approaches that are not limited to any specific field (although some of them include one or more example applications); in this sense, they are *generic*. We used the same category to denote methods belonging to some other *various* fields.

The other classification criteria we used for each method are the following:

Base method. This refers to the fundamental approach (mathematical model, algorithm) on which the method is based on. Of course, many methods are using complex solutions and it is difficult to categorize them into a single approach, but we managed to classify most of the methods into one of the following: *Machine Learning* including any subfield thereof, *Statistics*, which are based on statistical analysis of the failures, tests, etc., *Entropy*, a special case of statistics which also includes probabilistic approaches. Finally, *Model* refers to model-based approaches that include various types of models such as mathematic structures or engineering descriptions of the systems. In some cases, a combination of the previous was applied in which case we used *Combined*. Finally, if the base method could not be determined or would be very different than the mentioned ones, we used *Other*.

Faults. It is an important property of a method if it relies on an assumption that there is a single fault in the system, or it can handle (or is designed to handle) multiple faults occurring at the same time. Therefore, we use the *Single* and *Multiple* categories for this aspect.

Base Data. The next category we used is the basic type of data the method relies on for performing the fault localization computation. We found that most of the approaches are using either a *Graph* representation of the elements, probes, tests, etc., or they are represented in a *Matrix* format (such as rows containing the probes and columns the elements on which localization is to be performed with test results in the cells). In a number of cases, the base data is much more complex, in which case we used *Complex*. Finally, some approaches use a *Domain specific* data representation.

Behavior model. This category deals with the question if a behavioral model is required to perform fault localization. Such a model describes the expected behavior of the system. In simple cases, the tests (or probes) are providing simple pass/fail answers, but in other cases, a more complex model is needed. We used *Yes* or *No*.

Empirical. This category classifies the methods according to whether they include empirical measurements, and if yes, what kind of. The *Theory* category means that only theory is described, *Simulated* refers to a case when simulation data were used in the experiments, while in the case of *Real*, real data was used.

Data set. If the method included any kind of experiments, this category will provide the amount of data they were executed on. *Example* means that only toy examples were used, *Small* refers to a realistic but small data set, while *Large* includes any real data that can be treated large but is limited to a small number of projects or sets. Finally, *Mass* was used when an automated method was used to collect mass amounts of data from some repositories.

Availability. This category deals with the availability of the underlying information of the method. Namely, if only the *Implementation* or the measurement *Data* are available, *Both* of these or *None*.

For each of the criteria from above, if it cannot be interpreted for a specific method, we will use *N/A* to denote this situation.

4 Methods by Areas

In this section, we present the results of our assessment of fault localization techniques literature. We list the identified papers along with the properties following the categorization presented in the previous section. This section is organized into subsections by the *Fundamental area* category defined above. Each subsection is composed of a table of the same structure: we list the papers with their authors and publication year noted to help easier identification, and make a brief note of the assessment results for each classification aspect. An exception is the *Other Engineering Fields* category, in which case an additional column is used to indicate the specific field.

4.1 Software

Research related to fault localization in computer software is a large and diverse area. It is not the purpose of the present paper to provide a comprehensive overview of this literature, as the goal is to identify method *not related* to software. For an interested reader, we refer to the excellent surveys of Wong *et al.* [2] and Parmar and Patel [3]. Nevertheless, we include several works related to this area (Table 1), which we think are important representatives of the field. These approaches are diverse enough to serve as examples of the main techniques for software fault localization.

The basic goal of any software fault localization approach is to identify the location of software defect(s) in the source code given one or more faulty executions of the system. In software testing, one just shows that there is a defect somewhere in the system, and it is the task of fault localization to identify the exact point of the fault, typically in the source code.

A fundamental approach to software fault localization is to observe the behavior of distinct test cases and, based on their outcomes and their interaction with the system, compute the most suspicious code elements to contain the defects.

Table 1
Software fault localization techniques

Paper	***Base Method***	***Faults***	***Base Data***	***Behav. model***	***Empirical***	***Data set***	***Avai-lability***
Abreu et al., 2007 [7]	Combi-ned	Multiple	Complex	No	Real	N/A	Both
Abreu et al., 2009 [8]	Combi-ned	Multiple	Complex	No	Simulated	Example	Imple-ment
Artzi et al., 2010 [9]	Model	Multiple	Matrix	No	Simulated	Example	None
Christ et al., 2013 [10]	Other	N/A	Domain specific	No	Theory	N/A	Imple-ment
Pearson et al., 2017 [11]	N/A	N/A	N/A	N/A	Real	Large	Data
Ravindranath et al., 2014 [12]	Model	Multiple	Matrix	Yes	Real	N/A	Data
Renieris et al.,2003[13]	Mach. learn	Single	Complex	No	Simulated	Small	Data
Wang et al., 2011 [14]	Mach. learn.	Multiple	Domain specific	No	Simulated	Example	None

4.2 Networking

Fault localization in computer networks is a large and important area as networking technologies are becoming more and more complex as well as the internet itself, and the reliability of computer networks is increasingly important.

In networking, the goal of fault localization is to identify faulty networking elements (“nodes”) such as routers, etc. This is typically done by probing the network with network packages, and based on the responses from the nodes and the routes taken, the faulty nodes are identified.

Table 2 contains the results of our assessment of methods in the computer networking area.

Table 2
Networking fault localization techniques

Paper	***Base Meth.***	***Faults***	***Base Data***	***Behav. model***	***Empirical***	***Data set***	***Avai-lability***
Aghasaryan et al., 1997 [15]	Model	N/A	Complex	N/A	Theory	N/A	None
Aghasaryan et al., 1997 [16]	Model	Multiple	Complex	N/A	Theory	N/A	Imple-ment.

Alekseev et al., 2014 [17]	Model	Multiple	Complex	No	Theory	N/A	None
Brodie et al., 2002 [18]	Model	Multiple	Complex	No	Theory	N/A	None
Chao et al., 1999 [19]	Model	Multiple	Domain specific	No	Theory	N/A	Imple-ment.
Chen et al., 2004 [20]	Mach. learn.	Multiple	Domain specific	Yes	Real	N/A	None
Deng et al., 1993 [21]	Mach. learn.	N/A	Domain specific	No	Theory	Example	Imple-ment.
Fecko et al., 2001 [22]	Com-bined	N/A	Complex	No	Simulated	Example	None
Garshasbi et al., 2013 [23]	Other	Multiple	Matrix	No	Theory	Example	Data
Hood, 1997 [24]	Mach. learn.	Multiple	Domain specific	No	Simulated	Example	None
Kant et al., 2003 [25]	Model	N/A	Complex	No	Theory	N/A	None
Katzela et al., 1995 [26]	Model	Multiple	Domain specific	No	Simulated	Example	None
Kompella et al., 2005 [27]	Model	Multiple	Matrix	No	Simulated	Example	Imple-ment.
Lu et al., 2013 [28]	Model	Multiple	Complex	No	Simulated	Example	Data
Natu et al., 2006 [29]	Other	N/A	Matrix	No	Theory	N/A	Imple-ment.
Natu et al., 2007 [30]	Model	Multiple	Domain specific	No	Theory	N/A	Imple-ment.
Natu et al., 2007 [31]	Statis-tics	Multiple	Matrix	No	Simulated	Example	Both
Rish et al., 2004 [32]	Other	N/A	Complex	No	Real	N/A	Imple-ment.
Steinder et al., 2004[33]	Model	N/A	Complex	Yes	Simulated	Example	Imple-ment.
Steinder et al., 2004[34]	Model	N/A	Complex	Yes	Simulated	Example	Imple-ment.
Tang et al., 2005 [35]	Model	Multiple	Complex	Yes	Simulated	Example	Both
Traczyk, 2004 [36]	N/A	Multiple	Matrix	No	Simulated	Example	None
Wang et al., 2012 [37]	Com-bined	Multiple	Complex	No	Simulated	Example	Both
Zhang et al., 2011 [38]	N/A	N/A	N/A	Yes	Theory	N/A	None

4.3 Other Engineering Fields

This category deals with different engineering fields in which some form of automated fault localization is investigated. Faults are possible and need to be avoided or identified in virtually any automatic system, whether it is mechanical, electrical, logical (software), or even chemical or biological. Some systems are complex and composed of different components of the mentioned types.

Automatic fault localization is used to various degree in these areas, typically based on the criticality of the system. Some areas are particularly notable in this respect, which have a relatively large literature on fault localization. These areas include the aerospace industry (detecting faults in aircraft systems), power electronics (detecting faults and source of outages in electrical networks), electronics (detecting faults in hardware components of computer systems or other electronic devices, most typically in the digital domain). Other areas we encountered include mechanical engineering (detecting faults of rotary machines), oil pipelines (detecting leakage points) and chemistry (detecting faults in chemical plants that implement complex chemical reactions).

We are certain that there may be many other areas that encounter similar issues and have domain-specific solutions to fault localization, but the domains we list in this section illustrate the diversity of approaches used. Interestingly, there are many common basic approaches used in these diverse areas (such as entropy-based and neural networks), which means that they might be good candidates in reusing the methods to software fault localization.

Table 3 contains the results of our assessment of other engineering field methods.

Table 3
Other engineering fault localization techniques

Paper	***Base Method***	***Faults***	***Base Data***	***Behav. model***	***Em-pirical***	***Data set***	***Avail***	***Field***
Adamovits et al., 1993 [39]	Model	Multiple	Domain spec.	Yes	Theory	N/A	None	Aero-space
Balaban et al.,2007 [40]	Model	Multiple	Domain spec.	Yes	Theory	N/A	None	Aero-space
Benbouzid et al., 1999 [41]	N/A	N/A	N/A	N/A	Theory	Example	None	Power electr.
Beschta et al.,1993 [42]	Model	Single	Complex	No	Theory	N/A	None	Power electr.
Digernes, 1980 [43]	Model	Single	Complex	No	Simulated	Example	None	Oil pipeli-nes
Dries, 1990 [44]	Model	Multiple	Domain spec.	N/A	Theory	N/A	Im-plem.	Aero-space
Pálfi et al., 2017 [45]	Other	Multiple	Domain	N/A	Real	Small	None	Power electr.
Poon, 2015 [46]	Model	Multiple	Domain spec.	Yes	Simulated	Example	Data	Power electr.
Peischl et al.,2006 [47]	Model	N/A	Graph	N/A	Simulated	Example	None	Elec-tronics
Tanwani et al.,2011 [48]	Model	N/A	Complex	N/A	Simulated	Example	Im-plem.	Power electr.

Tóth et al., 2013 [49]	Other	Multiple	Domain	No	Simulated	Example	None	Mech. eng.
Venkatasubramania et al.,1990 [50]	Model	Multiple	Domain	Yes	Simulated	Example	None	Che-mistry
Yan et al., 2014 [51]	Other	Multiple	Domain	No	Real	Example	None	Mech. eng.

4.4 Various and Generic Methods

During the assessment of the identified literature, we encountered several works that introduce a fault localization algorithm, which is theoretically application independent. To a certain degree, these generic methods could be applied to any field, including software. Many of these publications are illustrating the use of the approach in a specific field, but it is generally not discussed to what degree is the method generalizable to other areas.

Some methods listed in this category are purely theoretical and advance a certain mathematical subfield, with no obvious practical application. Hence, the applicability of the methods listed in this section should be carefully investigated to any particular field, notably software faults.

Table 4 contains the associated results of our assessment.

Table 4

Various other fields fault localization techniques

Paper	***Base Method***	***Faults***	***Base Data***	***Behav. model***	***Empirical***	***Data set***	***Availa-bility***
Frank, 1996 [52]	N/A	N/A	Complex	Yes	Theory	N/A	None
Gertler, 1991 [53]	Machine learning	Multiple	Matrix	No	Theory	N/A	None
Isermann, 1984 [54]	N/A	N/A	N/A	N/A	Theory	N/A	None
Kleer, 2009 [55]	Entropy	Multiple	Domain specific	Yes	Simulated	Large	None
Kleer et al., 1987 [56]	Entropy	Multiple	Domain specific	Yes	Theory	N/A	None
Lerner et al.,2000[57]	Model	Multiple	Complex	No	Simulated	Example	Implemen-tation
Massoumnia et al., 1986 [58]	Model	Multiple	Complex	No	Theory	N/A	None
Mehra et al.,1971[59]	Statistics	Multiple	Complex	No	Theory	N/A	None
Olivier-Maget et al., 2009 [60]	Combined	Multiple	Complex	No	Theory	N/A	None

Shchekotykhin et al., 2016 [61]	Model	Multiple	Domain specific	N/A	Simulated	Example	Implemen-tation
Tidriri et al, 2016 [62]	N/A	N/A	N/A	Yes	Theory	N/A	None
Varga, 2003 [63]	Statistics	N/A	Domain specific	No	Theory	Example	None

5 Evaluation

The main goal of the paper was to identify potential approaches from non-software domains that can be successfully adapted to software faults and fault localization. Based on the summaries in the previous chapter, it is not easy to pinpoint only a few candidate methods, rather many of them may provide interesting ideas, even if not the complete method is adapted. In particular, we found the following. Figure 1 contains the overview of the various fields we investigated in this article. The arrows from specific areas to software bugs indicate the level of their applicability (dashed lines = moderate, solid lines = probable).

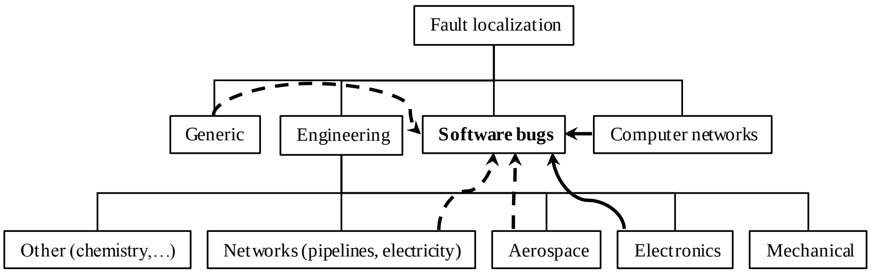

Figure 1
Overview of the investigated fault localization areas and their relation to software faults

5.1 Networking

The most promising techniques for adapting to software faults is the *probing method* in computer networks [36]. A probe is a program that executes on a particular network node and sends commands or transactions to the other elements of the network. Then, the responses are observed and their various properties are measured. From this information, various network issues, bottlenecks and faulty nodes can be estimated. Steinder and Sethi provide a survey of fault localization techniques in computer networks [64].

An interesting property of such network fault localization methods is that an almost direct analogy can be drawn to software fault localization: a network node corresponds to a software component, a probe can be seen as a test case, and the responses from the network can be identified as the dynamic behavior of the system by executing the test cases. Thus, the traditional *spectrum-based fault localization* methods in software [2, 3, 7] may benefit from advances in probe-based networking fault localization.

For instance, the approaches by Brodie *et al.* [18], and Natu *et al.* [29, 30, 31], provide various optimizations to the basic probing approaches, which are good candidates for adaptation to software faults.

Another common element of network fault localization is the use of probabilistic approaches (such as conditional probabilities and Bayes networks) [17, 19, 34, 35], among others, as well as machine learning [20, 21, 24]. These can be probably adapted to software.

5.2 Other Engineering Fields

Overall, the techniques used by other engineering fields are typically not directly applicable to software faults because of the big differences in the domains. Often, reliable behavioral models are the basis for these approaches which is in many cases difficult to obtain with software. The probabilistic approach used often in some areas may, however, be considered to enhance fault localization in software. Indeed, there are already several enhanced methods in software fault localization that employ conditional probabilities and entropies, such as Abreu *et al.*'s method [7] (also see [2] [3]).

In the *aerospace* industry, use of artificial intelligence, in particular, model based reasoning, seems to be prevalent [39, 40, 44]. Although these approaches seem quite advanced, their application to software fault localization may be limited due to the difficulty of producing a reliable model of the software.

The situation is similar with the *power electronics* area [41, 42, 46], these also frequently utilize various models describing the system. However, they seem to be less complex and more similar to computer networks, hence their applicability may be easier.

Some approaches in fault localization in *electronic circuits* may almost directly be applied because the description of the hardware is done in a similar way to computer software source code [47]. However, often simulation is done based on the circuit model, which is more difficult to employ on software. It is interesting to note, that some techniques that we categorized as "Generic methods" (see next section) have their main application in electronic circuits (Kleer *et al.* [55] [56]), which are based on entropy minimization and probabilistic approach (as with many methods in computer networks).

The other areas we investigated also often use simulation and probabilistic approaches [43], or machine learning with neural networks [50], but in these cases a model of the system is required as well. Often, advanced concepts are applied in these areas such as Kalman filters to increase the accuracy of fault estimates.

A notable field is that of machine fault diagnosis in mechanical engineering [49] [51]. This concerns of finding faults in machine elements, most specifically in rotating machinery. This area is only remotely related though, as the methods used are very specific to the field, and include spectral and waveform analysis of vibration signals. Reference [51] provides an overview of the field with specific emphasis on wavelets for fault diagnosis of rotary machines.

5.3 Generic Methods

The common property of most generic methods is that they rely on a behavioral model of the system. Many of these model the system as a process, and hence process analysis approaches are used from control theory [50, 54, 58, 59]. This is often applied to fault tolerant systems. Often, these are called Model-Based Diagnosis techniques, which aim at finding the fault of an observed system based on knowledge about the system's expected behavior [52, 55, 56, 61]. The mentioned entropy based and probabilistic approaches are typically used.

Tidriri *et al.* [62] combine model based approaches with data driven methods (which process a large amount of data from the system's output and are based on training data for a correctly working system). This may be a good candidate to be applied to software fault localization, because in this case often the model is not available but the operational data from software executions is easily obtainable through system logs. This publication refers other related work in this area, which can be useful sources for more information about this set of techniques.

Conclusions

This paper presented the results of our interdisciplinary analysis of fault localization techniques. As this was a preliminary study, our goal was to find related publications in various engineering fields, initially evaluate the proposed methods and assess their usability to our central topic, software fault localization. We found that, among the many different engineering fields, computer networks, aerospace, (power) electronics and some other areas are the most promising to help advance software fault localization.

The detailed analysis results presented in Section 4 could provide a starting point for further analyzing the techniques. Based on the various properties of the method, we provided (fault types, base data, empirical results, etc.), the most promising approaches could be selected for further consideration. Section 5, on the other hand, could be used to pin-point specific topics (with references to the main articles) to be used to enhance software fault localization.

Although we performed a systematic Literature Analysis, we cannot claim any completeness thereof. Based on the identified and here referenced works, further publications could be searched by investigating the references, authors and research groups, etc. Also, scientific venues (conferences, journals) of specific engineering areas could be further analyzed to discover additional results.

Nevertheless, we believe that the survey in its present state is suitable for us to continue our quest for enhancing software fault localization, and for other readers to obtain a wider view of this important and diverse topic.

In future work, we will evaluate the most promising approaches in more detail and eventually implement the findings, for software fault localization.

Acknowledgement

This work was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. The author would like to help the supporting work of Ottó Eötvös and Brúnó Ilovai.

References

[1] Frank Schweitzer (editor-in-chief): Advances in Complex Systems – A Multidisciplinary Journal, World Scientific, ISSN (print): 0219-5259, ISSN (online): 1793-6802

[2] W. E. Wong, R. Gao, Y. Li, R. Abreu and F. Wotawa: A survey of software fault localization, IEEE Transactions on Software Engineering, 42(8): 707-740, August 2016

[3] P. Parmar and M. Patel: Software Fault Localization: A Survey, International Journal of Computer Applications, 154(9):6-13, 2016

[4] Held, J. M.: The Modelling of Systems of Systems, PhD Thesis, University of Sydney, 2008

[5] Tien-Duy B. Le, Ferdian Thung, David Lo: Theory and Practice, Do They Match? A Case with Spectrum-Based Fault Localization, 2013 IEEE International Conference on Software Maintenance, pp. 380-383

[6] P. Kavulya, Soila, Joshi, Kaustubh, Giandomenico, Felicita and Narasimhan, Priya: Failure Diagnosis of Complex Systems, Resilience Assessment and Evaluation of Computing Systems, 2012, pp. 239-261

[7] R. Abreu, P. Zoeteweij and A. J. C. van Gemund: Spectrum-based multiple fault localization, In Proceedings of IEEE/ACM International Conference on Automated Software Engineering, pp. 88-99, November 2009

[8] R. Abreu, W. Mayer, M. Stumptner and A. J. C. van Gemund: Refining spectrum-based fault localization rankings, In Proceedings of the 2009 ACM Symposium on Applied Computing, pp. 409-414, 2009

[9] Sh. Artzi, J. Dolby, F. Tip and M. Pistoia: Practical fault localization for dynamic web applications, Proceedings of the 32nd ACM/IEEE

International Conference on Software Engineering - Volume 1, pp. 265-274, 2010

[10] J. Christ, E. Ermis, M. Shäf and T. Wies: Flow sensitive fault localization, In Proceedings of Verification, Model Checking, and Abstract Interpretation: 14th International Conference, VMCAI 2013, pp. 189-208, January 2013

[11] S. Pearson, J. Campos, R. Just, G. Fraser, R. Abreu, M. D. Ernst, D. Pang and B. Keller: Evaluating and improving fault localization, Proceedings of the 39th International Conference on Software Engineering, pp. 609-620, 2017

[12] L. Ravindranath, S. Nath, J. Padhye and H. Balakrishnan: Automatic and scalable fault detection for mobile applications, Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, pp. 190-203, 2014

[13] M. Renieris and S. P. Reiss: Fault localization with nearest neighbor queries, In Proceedings of 18th IEEE International Conference on Automated Software Engineering, pp. 130-139, 2003

[14] Sh. Wang, D. Lo, L. Jiang, Lucia and H. Ch. Lau: Search-based fault localization, Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering, pp. 556-559, 2011

[15] A. Aghasaryan, E. Fabre and C. Jard: A Petri net approach to fault detection and diagnosis in distributed systems I., Proceedings of the 36th IEEE Conference on Decision and Control, pp. 720-725, December 1997

[16] A. Aghasaryan, E. Fabre and C. Jard: A Petri net approach to fault detection and diagnosis in distributed systems II, Proceedings of the 36th IEEE Conference on Decision and Control, pp. 726-731, December 1997

[17] D. Alekseev and V. Sayenko: Proactive fault detection in computer networks, In Proceedings of 2014 First International Scientific-Practical Conference Problems of Infocommunications Science and Technology, pp. 90-91, 2014

[18] M. Brodie, I. Risha and Sh. Ma: Intelligent probing: A cost-effective approach to fault diagnosis in computer networks, IBM Systems Journal, 41(3):372-385, 2002

[19] C. S. Chao, D. L. Yang and A. C. Liu: An automated fault diagnosis system using hierarchical reasoning and alarm correlation, Proceedings of 1999 IEEE Workshop on Internet Applications (Cat. No.PR00197), pp. 120-127, August 1999

[20] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan and E. Brewer: Failure diagnosis using decision trees, In Proceedings of International Conference on Autonomic Computing, pp. 36-43, May 2004

[21] R. H. Deng, A. A. Lazar and W. Wang: A probabilistic approach to fault diagnosis in linear lightwave networks, IEEE Journal on Selected Areas in Communications, 11(9):1438-1448, December 1993

[22] M. A. Fecko and M. Steinder: Combinatorial designs in multiple faults localization for battlefield networks, In Proceedings of 2001 MILCOM Communications for Network-Centric Operations: Creating the Information Force (Cat. No.01CH37277), pp. 938-942, 2001

[23] M. S. Garshasbi and Sh. Jamali: A new fault detection method using end-to-end data and sequential testing for computer networks, Reliability Engineering & System Safety, 114(1):45-51, June 2013

[24] C. S. Hood: Proactive network fault detection, IEEE Transactions on Reliability, 46(3):333-341, September 1997

[25] L. Kant, A. S. Sethi and M. Steinder: Fault localization and self-healing mechanisms for FCS networks, Proc. 23rd Army Science Conference, January 2003

[26] I. Katzela and M. Schwarz: Schemes for fault identification in communication networks, IEEE/ACM Trans. Netw., 3(6):753-764, December 1995

[27] R. R. Kompella, J. Yates, A. Greenberg and A. C. Snoeren: IP fault localization via risk modeling, Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2, pp. 57-70, 2005

[28] L. Lu, Zh. Xu, W. Wang and Y. Sun: A new fault detection method for computer networks, Reliability Engineering & System Safety, 114(Supplement C):45-51, 2013

[29] M. Natu and A. S. Sethi: Active probing approach for fault localization in computer networks, In Proceedings of 2006 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services, pp. 25-33, April 2006

[30] M. Natu and A. S. Sethi: Efficient probing techniques for fault diagnosis, In Proceedings of Second International Conference on Internet Monitoring and Protection (ICIMP 2007), pp. 20-20, July 2007

[31] M. Natu and A. S. Sethi: Probabilistic fault diagnosis using adaptive probing, In Proceedings of Managing Virtualization of Networks and Services, pp. 38-49, 2007

[32] I. Rish, M. Brodie, N. Odintsova, Sh. Ma and G. Grabarnik: Real-time problem determination in distributed systems using active probing, In Proceedings of 2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507), pp. 133-146, April 2004

[33] M. Steinder and A. S. Sethi: Non-deterministic fault localization in communication systems using belief networks, IEEE/ACM Trans. Netw., 12(5):809-822, October 2004

[34] M. Steinder and A. S. Sethi: Probabilistic fault localization in communication systems using belief networks, IEEE/ACM Transactions on Networking, 12(5):809-822, October 2004

[35] Y. Tang, E. S. Al-Shaer and R. Boutaba: Active integrated fault localization in communication networks, In Proceedings of 2005 9th IFIP/IEEE International Symposium on Integrated Network Management, 2005, IM 2005, pp. 543-556, 2005

[36] W. Traczyk: Probes for fault localization in computer networks, Journal of Telecommunications and Information Technology, 3:23-27, 2004

[37] B. Wang, W. Wei, W. Zeng and K. R. Pattipati: Fault localization using passive end-to-end measurements and sequential testing for wireless sensor networks, IEEE Transactions on Mobile Computing, 11(3):439-452, March 2012

[38] X. Zhang, Z. Zhou, G. Hasker, A. Perrig and V. Gligor: Network fault localization with small TCB, In Proceedings of 2011 19th IEEE International Conference on Network Protocols, pp. 143-154, October 2011

[39] P. J. Adamovits and B. Pagurek: Simulation (model) based fault detection and diagnosis of a spacecraft electrical power system, Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications, pp. 422-428, March 1993

[40] E. Balaban, S. Narasimhan, H. N. Cannon and L. S. Brownston: Model-based fault detection and diagnosis system for NASA Mars subsurface drill prototype, In Proceedings of 2007 IEEE Aerospace Conference, pp. 1-13, March 2007

[41] M. E. H. Benbouzid, M. Vieira and C. Theys: Induction motors' faults detection and localization using stator current advanced signal processing techniques, IEEE Transactions on Power Electronics, 14(1):14-22, January 1999

[42] A. Beschta, O. Dressler, H. Freitag, M. Montag and P. Struß: Model-based approach to fault localization in power transmission networks, Intelligent Systems Engineering, 2:3-14, February 1993

[43] T. Digernes: Real-time failure detection and identification applied to supervision of oil transport in pipelines, Modeling, Identification and Control, 1(1):39-49, 1980

[44] R. W. Dries: Model-based reasoning in the detection of satellite anomalies, MS Thesis, AFIT/GSO/ENG/90D-03, School of Engineering, Air Force Institute of Technology, 1990

[45] Judith Pálfi, Miklós Tompa and Péter Holcsik: Analysis of the Efficiency of the Recloser Function of LV Smart Switchboards, Acta Polytechnica Hungarica, Volume 14, Number 2, 2017, pp. 131-150

[46] J. Poon: Model based fault detection and identification for power electronics systems, Technical Report No. UCB/EECS-2015-238, University of California, Berkeley, 2015

[47] B. Peischl and F. Wotawa: Automated source-level error localization in hardware designs, IEEE Design Test of Computers, 23(1):8-19, January 2006

[48] A. Tanwani, A. D. Domínguez-Garcia and D. Liberzon: An inversion based approach for fault detection and isolation in switching electrical networks, IEEE Transactions on Control Systems Technology, 19(5):1059-1074, September 2011

[49] Lajos Tóth and Tibor Tóth: On Finding Better Wavelet Basis for Bearing Fault Detection, Acta Polytechnica Hungarica, Volume 10, Number 3, 2013, pp. 17-35

[50] V. Venkatasubramanian, R. Vaidyanathan and Y. Yamamoto: Process fault detection and diagnosis using neural networks, Computers & Chemical Engineering, 14(7):699-712, 1990

[51] Ruqiang Yan, Robert X. Gao and Xuefeng Chen: Wavelets for fault diagnosis of rotary machines: A review with applications, Signal Processing, Volume 96, Part A, 2014, pp. 1-15, Elsevier

[52] P. M. Frank: Analytical and Qualitative Model-based Fault Diagnosis – A Survey and Some New Results, European Journal of Control, 2(1):6-28, 1996

[53] J. Gertler: Analytical redundancy methods in fault detection and isolation, IFAC Proceedings Volumes, 24(6):9-21, September 1991

[54] R. Isermann: Process fault detection based on modeling and estimation methods - A survey, Automatica, 20(4):384-404, 1984

[55] Johan de Kleer: Diagnosing Multiple Persistent and Intermittent Faults, In Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 733-738, 2009

[56] Johan de Kleer and Brian C. Williams: Diagnosing multiple faults, Artificial Intelligence, 32(1):97-130, 1987

[57] U. Lerner, R. Parr, D. Koller and G. Biswas: Bayesian fault detection and diagnosis in dynamic systems, In Proc. AAAI, pp. 531-537, 2000

[58] M. A. Massoumnia, G. C. Verghese and A. S. Willsky: Failure detection and identification in linear time-invariant systems, Technology, No. July, 1986

[59] R. K. Mehra and J. Peschon: An innovations approach to fault detection and diagnosis in dynamic systems, Automatica, 7(5):637-640, 1971

[60] N. Olivier-Maget, S. Negny, G. Hétreux and J. M. Le Lann: Fault diagnosis and process monitoring through model-based and case based reasoning, In Proceedings of 19th European Symposium on Computer Aided Process Engineering, pp. 345-350, 2009

[61] K. Shchekotykhin, T. Schmitz and D. Jannach: Efficient sequential model-based fault localization with partial diagnosis, In Proceedings of IJCAI'16 Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pp. 1251-1257

[62] Khaoula Tidriri and Nizar Chatti and Sylvain Verron and Teodor Tiplica: Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges, Annual Reviews in Control, Volume 42, pp. 63-81, 2016

[63] A. Varga: On computing least order fault detection using rational nullspace bases, IFAC Proceedings Volumes, 36(5):227-232, 2003

[64] M. Steinder and A. S. Sethi: A survey of fault localization techniques in computer networks, Science of Computer Programming, 53(2):165-194, 2004