Towards a Diagnosis Framework for the Verification of Critical Systems Design

Vincent Leildé[1], Vincent Ribaud[2], Ciprian Teodorov[1], and Philippe Dhaussy[1]

[1] Lab-STICC, team MOCS, ENSTA-Bretagne, rue François Verny, Brest, France
firstname.lastname@ensta-bretagne.fr,
[2] Lab-STICC, team MOCS, Université de Bretagne Occidentale, Avenue le Gorgeu, Brest, France Vincent.Ribaud@univ-brest.fr

Abstract. For critical systems design, the verification tasks play a crucial role. If abnormalities are detected, a diagnostic process must be started to find and understand the root causes before corrective actions are applied. Detection and diagnosis are notions that overlap in common speech. Detection basically means to identify something as unusual, diagnosis means to investigate its root cause. The meaning of diagnosis is also fuzzy, because diagnosis is either an activity - an investigation - or an output result - the nature or the type of a problem. This paper proposes an organizational framework for structuring diagnoses around three principles: that propositional data (including detection) are the inputs of the diagnostic system; that activities are made of methods and techniques; and that associations specialize that relationships between the two preceding categories.

Keywords: Diagnosis, Verification, Critical systems, Framework

1 Introduction

Critical systems are concerned by *dependability*, i.e. the ability of an entity to perform as and when required [7], that requires the *means* of improving the quality of systems design. This should be realized in three cyclical phases: verification, diagnosis and correction. Verification aims to demonstrate whether a system meets specification properties, also called verification conditions. This may be achieved using various techniques such as static analysis, simulation or model-checking. Model checking is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for that model [8]. If a property is violated, a counter-example is produced as a trace from the initial state to the state in which the error was detected. When a property is falsified, a diagnosis process is triggered with the objective of outlining the root causes that violated the verification conditions. Then from this diagnosis, generally realized through detection, localization and identification tasks, the system is corrected and the verification cycle is repeated.

Several frameworks and approaches are proposed to perform diagnosis, but most of them are restricted to only one kind of diagnosis. Indeed, there are two

reasons of such lack of relevant diagnosis tools. On the one hand, the verification process, in particular by model checking, is not well managed and controlled [27] producing a large amount of heterogeneous interrelated models. It naturally increases the complexity of extracting the essence of errors, in particular locating causes from detailed source-level traces of a failing run [16]. On the other hand, diagnosis is also loosely formalized. As a result, models produced during the design and verification process are not well adapted to diagnosis tasks. Therewith, diagnosis is weakly integrated with other tasks and collaboration between tools and processes is not smoothly achieved.

For the above reasons, understanding and formalizing the diagnosis is intended to foster the definition of diagnosis tools and methodologies, and reduce the set of diagnoses. If *diagnosis* is applicable in different fields (medicine, supervision of industrial processes), frameworks differ, and cannot be fully applied to trace-based diagnosis. We propose an organizational framework for diagnosis systems, based on three concepts: *activity*, *propositional object* and *association*.

2 Background

If model-checking is often dedicated to faults detection, some frameworks also employ it for faults localization. For instance, slicing-based approaches [29] use dependency analysis to retrieve the set of elements which contains the fault. State space reduction [19] aims at reducing the state space size by exploiting the concurrent transitions commutativity. Ball et al. [9] introduced an approach to compare the counter-examples with successful traces and thus isolate faulty state transitions. In [10], the authors propose a Symbolic Model Checking framework for safety analysis diagnosis. These approaches focus on trace processing, and the identification task, i.e. identifying the specific nature of faults, is not considered. Consequently, a semantic gap between design models and traces still holds.

Some approaches allow for a complete diagnostic. For instance in [17], the authors define a framework that combines an abductive model-based diagnosis approach with a Labelled Transition System. This kind of method is also experienced by [6], who associated logic learning with trace-based diagnosis and error correction using positive and negative traces. These approaches are restricted to one diagnosis technique, model-based, that imposes the presence of either a fault or a well-functioning model, which is not always available.

Venkatasubramanian [28] has broadly classified fault diagnosis methods into quantitative model-based methods, qualitative model-based methods, and process history based methods. This classification provides a large spectrum of methods and techniques, but focuses on industrial processes, and put aside important techniques for trace-based diagnosis like interaction-based techniques.

To the best of our knowledge, there are no frameworks for characterizing diagnosis systems, unrestricted to any diagnosis techniques, activities or application domains. Therefore, we focus on understanding diagnosis in order to identify a core set of concepts that can be applied for any diagnoses systems.

3 Conceptual Framework

Our goal is to propose a framework for characterizing diagnosis systems, not restricted to a diagnosis technique or method. We will start from a general definition of diagnosis given by Merriam Webster [2] : "*diagnosis is an investigation or analysis of the cause or nature of a condition, situation, or problem*". This framework is based on three concepts: - *Activity*, a set of mechanisms or tasks used to perform the diagnosis ; - *Propositional object*, tangible or immaterial, produced or consumed by activities ; - *Association* between propositional objects and activities.

3.1 Activities

The foremost part of the diagnosis definition refers to an *activity*, whether *an investigation or an analysis*. Other kinds of activities can be carried out during a diagnosis and an *activity* is itself composed of internal activities.

Main diagnosis activities. According to the literature, diagnosis systems support three main activities, *fault detection*, *isolation*, and *analysis* [28].

Fault detection establishes that a system run raises so-called abnormal event. In the particular case of verification by model-checking, detection is done by model-checking itself. Subsequently after the detection, one requires an evaluation of the magnitude of the abnormal situation, to choose whether a diagnosis is required or not[4].

When required, the ensuing step consists in *isolating* the subset of elements, part of models, that needs to be corrected [14]. *Isolation* is performed through various techniques, such as slicing-based approaches [29], state space reduction techniques [19] or counter-example comparisons [9].

Once suspicious elements are localized, the *analysis* task, associates causes to the observed abnormalities. This is generally a reasoning process [3], either deduction, induction or abduction. Deduction is concerned by deducting knowledge from already learned knowledge, induction identifies general rules from observations, and abductive reasoning discovers causes from facts by elaborating hypothesis. In abductive thought, one can have different answers, and therefore have to decide among alternatives [13]. Each type of reasoning fits with a different situation, abduction produces ideas and concepts to be explained, then induction contributes to the construction of the abductive hypothesis by giving it consistency, finally deduction formulates a predictive explanation from this construction [12]. Nevertheless, we always reason by looking for the fastest way, this is the principle of the cognitive economy [15].

Mechanisms. An activity is performed through a set of mechanisms, gathering tools and methods, that can be organized in various categories, *qualitative or quantitative model based* and *process history based*. We complete this list with a category called *interaction*, relevant in case of trace-based diagnosis.

Model-based methods assume that a model of the system is available, representing its correct (consistency-based) [26] or abnormal behavior (abductive-based)[30]. In consistency-based, the reasoning consists in rejecting a set of assumptions using the correct behaviour, in order to restore consistency with

(abnormal) observations [11]. In the opposite abductive-based reasoning works with causes and effects models, it includes approaches like [6].

Under the *process-history based* methods, only the availability of large amount of historical process data is needed. It allows for knowledge extraction techniques, like data mining or statistical analysis. Regarding model-checking, Liu [23] uses statistical models to remove false positive traces. Besides, probabilities can also be applied, using decision trees or Bayesian networks. Focusing on learning, Neural networks and Case-based reasoning [5] try to reproduce the human way of reasoning. But when a strong expertise is available, one can simply use expert systems [20], gathering problems set, rules and an inference engine.

Interactions mechanisms play an important role in trace-based diagnosis. Interactive tools allows for observing, controlling, understanding and altering the system execution. By storing the execution traces, omniscient debuggers enable back-in-time navigation features, postmortem query processing, trace-analysis and reduction facilities, and execution replay [25]. Besides, a large number of visualization tools exists [18], including a wide range of diagram structures ranging from waveforms, finite state machines, business representations.

3.2 Propositional Objects

Activities handle different kind of information [2], in the one hand "situation or problem", and in the other hand "cause or nature". As information may be tangible or immaterial, and we define any information items as *propositional objects* that are, or represent sets of propositions about real or imaginary things.

Situation or problem represent *observations* about the system. A *situation* is a way in which something is positioned with respect to its surroundings [2]. Regarding model-checking, it comprises design models, properties, exploration graphs or model-checker configurations. A *problem* is a difficulty that has to be resolved or dealt with [2]. *Problems* are revealed by *symptoms*, effects or visible consequences of the passage of the system into an abnormal state. Regarding model-checking it includes counter-examples. As stated by [26], "real world diagnostic settings involve observations, and without observations, have no way determining whether something is wrong and hence whether a diagnosis is called for". Thus *situation* and *problem* are both *observations*, i.e. acts of recognizing and noting a fact or occurrence [2], about the system.

Cause or nature are both explanations or *diagnoses*, i.e. statements or conclusions from diagnosis analysis [2], which are not always observable [21]. A *cause* is a reason for an action or condition [2], and is part of a *causality* phenomenon. A *nature* is a kind or class usually distinguished by essential characteristics [2], and is concerned with classification aspects.

3.3 Associations

Propositional objects and *activities* are linked over *associations*. Following a systemic triangulation, we organize *associations* in three viewpoints, *causality*, concerned with functional aspects, *nature*, concerned with structural aspects, and *evolution*, concerned with historical aspects.

Causality is defined by [24] as a sequence of linked events. Consider for instance a car with flat tires that suddenly slips on a water poll, resulting to an accident. The accident is a succession of related events. Closed to our concerns, a *Fault*, an *Error* and a *Failure* are considered for [7] as causal events, a *fault* may produce an *error*, which may lead to a *failure*.

Nature consists in determining the type, the characteristics or the essence of something, "what the object is". By taking up the example of a car, the owner inspects each tires and finds that some are more damaged than others, and classifies one tire in the category "too flat". The *nature* association itself can be refined in more specific relations, like generalization or specialization.

Evolution represents the historical, that is linked to the evolutionary nature of the system, "what the system was or is becoming". Considering the example above, a man is driving when suddenly an impact happen somewhere closed to the car wheels. He remembers he found one flat tire during his last car inspection, and supposes the tire is scratched.

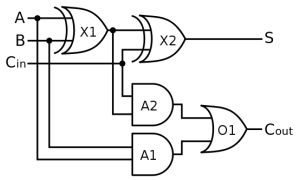

Fig. 1. Full adder

4 Framework by example

A *system* is an organization forming a network especially for serving a common purpose [2]. It emphasizes that a system pursues an objective. We present different kind of diagnosis systems using our framework, each pursuing a different objective. We refer to the classical example of a one bit adder [26], an illustration is given in figure 1 from [1]. A full adder is composed of five gates, A1 and A2 are AND gates, X1 and X2 are XOR gates, and O1 is an OR gate.

(**Analysis** ∨ *Investigation*)	(*Analysis* ∨ **Investigation**)	(**Analysis** ∨ *Investigation*)
of the	of the	of the
(*Cause* ∨ **Nature** ∨ *Evolution*)	(**Cause** ∨ *Nature* ∨ *Evolution*)	(*Cause* ∨ *Nature* ∨ **Evolution**)
of a	of a	of a
(**Situation** ∨ *Problem*)	(*Situation* ∨ **Problem**)	(**Situation** ∨ *Problem*)
⇒ **Pedagogical objective**	⇒ **Curative objective**	⇒ **Prognosis objective**

Table 1. Diagnosis Systems Examples

An analysis of the nature of a situation pursues a *pedagogical* objective. If we are not aware of the purpose of a digital circuit, we might build the truth table which sets out the output values for each combination of input values. The truth table is a diagnosis that helps to understand how the circuit

works (assuming the circuit behavior is normal). The analysis associates outputs (observations) to inputs (facts) and tries to figure out the nature of the circuit. Regarding verification, simulation activity helps to understand the way the system behaves, or ensure it behaves correctly.

An investigation of the cause of a problem pursues a *curative* objective. Consider we expect from the circuit a full adder behavior, and thus one expected property is *P1 : "for the set of entries A=1, B=1 and C=1, the result is S=1 and Cout=1"*. Assume that the XOR gate X1 was inadvertently replaced by an OR gate. Then the output of the circuit conflicts with the property P1, i.e. *S=0 and Cout= 1"*, and we must investigate the cause of the failure. Regarding model-checking, if a violation of functional specifications is discovered by a model-checker. One have to correct the design or model accordingly.

An analysis of the evolution of a situation pursues a *prognosis* objective. Given a set of properties (probably non-exhaustive), running the model-checker over the set without any errors yields an indication that the circuit, as far we know, behaves correctly. The underlying diagnosis is used as a prognosis of circuit major dysfunctions. In software, design patterns, like security patterns [31], are prevention mechanisms. Regarding model-checking of system design, if we consider a set of historical state spaces, one could apply design prognosis by using statistical and probability analysis.

5 Conclusion

In this paper we presented a framework for understanding diagnosis, and we defined some core concepts. Associated propositional objects represent knowledge about the system to be diagnosed, and they are processed by diagnosis activities and their underlying means. We believe that this minimal set of concepts will enable the exploration of the possible and constrained compositions of diagnostic systems, reducing the minimal set of diagnoses. This work paves the way for the construction of an organizing system, an ongoing work [22], for storing system data (propositional objects), interpreting them (association), and diagnosing the critical systems (activities).

References

1. Cburnett gfdl (http://www.gnu.org/copyleft/fdl.html), via wikimedia commons
2. Dictionary and Thesaurus | Merriam-Webster
3. Reasoning and the Logic of Things Charles Sanders Peirce, Kenneth Laine Ketner | Harvard University Press
4. Model-based Fault Diagnosis Techniques. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
5. Aamodt, A., Plaza, E.: Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Commun. 7(1), 39–59 (Mar 1994)
6. Alrajeh, D., Kramer, J., Russo, A., Uchitel, S.: Automated support for diagnosis and repair. Communications of the ACM 58(2), 65–72 (2015)

7. Aviienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. Dependable and Secure Computing, IEEE Transactions on 1(1), 11–33 (2004)
8. Baier, C., Katoen, J.P.: Principles of model checking. The MIT Press, Cambridge, Mass (2008)
9. Ball, T., Naik, M., Rajamani, S.K.: From symptom to cause: localizing errors in counterexample traces. In: ACM SIGPLAN. vol. 38. ACM (2003)
10. Bertoli, P., Bozzano, M., Cimatti, A.: A symbolic model checking framework for safety analysis, diagnosis, and synthesis. In: MoChArt'16. pp. 1–18. Springer (2006)
11. Bourahla, M.: Model-Based Diagnostic Using Model Checking. IEEE (2009)
12. Buccafurri, F., Eiter, T., Gottlob, G., Leone, N.: Enhancing model checking in verification by AI techniques. Artificial Intelligence 112(1), 57–104 (1999)
13. Charniak, E.: Introduction to artificial intelligence. Pearson Education India (1985)
14. Cleve, H., Zeller, A.: Locating causes of program failures. p. 342. ACM Press (2005)
15. Fiske, S.T., Taylor, S.E.: Social Cognition. McGraw-Hill Education, New York, NY (Oct 1991)
16. Groce, A., Visser, W.: What went wrong: Explaining counterexamples. In: Model Checking Software, pp. 121–136. Springer (2003)
17. Gromov, M., Willemse, T.A.: Testing and model-checking techniques for diagnosis. In: Testing of Software and Communicating Systems, pp. 138–154. Springer (2007)
18. Hamou-Lhadj, A., Lethbridge, T.C.: A survey of trace exploration tools and techniques. In: CASCON '04. pp. 42–55. IBM Press (2004)
19. Holzmann, G.J.: The Theory and Practice of A Formal Method: NewCoRe. In: IFIP Congress (1). pp. 35–44 (1994)
20. Ignizio, J.P.: Introduction to expert systems: the development and implementation of rule-based expert systems. McGraw-Hill, New York (1991)
21. International, N.: Nursing Diagnoses 2015-17: Definitions and Classification. Wiley-Blackwell, 10 edition edn. (Aug 2014)
22. Leilde, V., Ribaud, V., Dhaussy, P.: An Organizing System to Perform and Enable Verification and Diagnosis Activities. In: International Conference on Intelligent Data Engineering and Automated Learning. pp. 576–587. Springer (2016)
23. Liu, Y., Xu, C., Cheung, S.: AFChecker: Effective model checking for context-aware adaptive applications. Journal of Systems and Software 86(3), 854–867 (Mar 2013)
24. Mackie, J.L.: The cement of the universe: a study of causation. Clarendon library of logic and philosophy, Clarendon Press, Oxford, 5. dr. edn. (1990), oCLC: 258760915
25. Pothier, G., Tanter, ., Piquer, J.: Scalable omniscient debugging. ACM SIGPLAN Notices 42(10), 535–552 (2007)
26. Reiter, R.: A theory of diagnosis from first principles. Artificial intelligence 32(1), 57–95 (1987)
27. Ruys, T.C., Brinksma, E.: Managing the verification trajectory. International Journal on Software Tools for Technology Transfer (STTT) 4(2), 246–259 (Feb 2003)
28. Venkatasubramanian, V., Rengaswamy, R., Kavuri, S.N.: A review of process fault detection and diagnosis. Computers & Chemical Engineering 27(3) (Mar 2003)
29. Visser, W., Havelund, K., Brat, G., Park, S., Lerda, F.: Model checking programs. Automated Software Engineering 10(2), 203–232 (2003)
30. Wotawa, F., Rodriguez-Roda, I., Comas, J.: Abductive Reasoning in Environmental Decision Support Systems. In: AIAI workshops. pp. 270–279. Citeseer (2009)
31. Yoder, J., Barcalow, J.: Architectural patterns for enabling application security. Urbana 51, 61801 (1998)