A Combined Fault Diagnosis and Test Case Selection Assistant for Automotive End-of-Line Test Systems

Sebastian Abele and Michael Weyrich
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Email: {abele, weyrich}@ias.uni-stuttgart.de

***Abstract*—With growing complexity of premium cars, the end-of-line test systems also increase in complexity. The test systems have to provide more and more functionality like flashing of electronic control units (ECUs) and sensor calibration. Current end-of-line test systems evolved to complex networked IT-systems, which consist of various components and subsystems from different suppliers. Automotive production maintenance engineers are challenged to keep the availability of the test system on a high level to not cause production delays. In a case study with automotive test experts, we considered fault diagnosis and test case selection as two major tasks to maintain a high system availability. The experts combine their knowledge and experience about fault-prone system parts and former faults to optimize fault diagnosis and test case selection for regression testing. To support the experts to manage the growing complexity, we propose a combined fault diagnosis and test case selection assistance system. The combination of both techniques enables synergy effects by supporting the fault diagnosis with test case selection and by considering fault data in regression testing. This paper presents the concept of that combined assistant system and describes a prototypical realization used in an exemplary scenario.**

I. Introduction

Industrial quality assurance systems, e.g. automotive end-of-line test systems, have become an important part of the value-adding network for manufacturers. High customer requirements to the product quality lead to increasingly important quality assurance during the production. Tests are performed during the whole production process, which leads to a deep integration of quality assurance systems into the production lines. This deep integration makes the quality assurance systems to critical systems for the production process. Faults and breakdowns of the quality assurance systems cause delays or breakdowns of the whole production. In the automotive industry, the quality assurance systems evolved to systems bringing the cars into operation by flashing ECUs and calibrating a growing number of sensors and thus participate directly in the value-adding.

To achieve a high productivity, the availability of the quality assurance systems has to be high. By interviewing end-of-line test system maintenance experts of a large automotive manufacturer, it has shown that fault diagnosis and test case selection for regression test after new releases are important tasks to maintain the availability. Both tasks are time consuming with a lot of potential for automation and have the same goal: The impact of faults should be minimized. Therefore most of the faults should be found during the regression test phase without impact to the production process. If a fault occurs during operation, it has to be fixed as fast as possible.

The fault diagnosis is started after a system breakdown or after a series of failed car tests, which indicate a problem in the test system. The maintainer connects to the affected test devices and browses the logfiles and result databases for any entries indicating faults. If he cannot recognize the cause of the fault directly, he performs further tests with the system to isolate and finally diagnose the fault. Therefore he needs to repeatedly execute tests and scan the logging for new information.

New requirements coming from new car features or new production variants lead to periodic releases of new system versions which introduce changed hardware and software. Before the new releases are used at a production line, they have to be tested to ensure that the system still works properly. This test is performed by the same maintenance engineers as the fault diagnosis. Thus they bring together the knowledge about newly introduced features and the experience about fault-prone functions and components from the last operational period. With this information they select the best-suited test cases with the goal to find as many faults as possible before the system is used in a production line.

This workflow reveals two drawbacks: Firstly, the maintainers spend a lot of time with scanning logs to identify fault symptoms, even for already known faults. Secondly, the complexity and the distribution of the test system leads to an insufficient sharing of knowledge and experience which leads to inefficient fault diagnosis and testing. To overcome those drawbacks, we propose an assistance system which combines symptom identification, fault diagnosis and test case selection. The assistance system integrates formalized expert knowledge from fault diagnosis and system testing. Additionally, it uses system monitoring information to identify fault symptoms and interfaces test run histories and fault histories to determine the test importance for every system component. Recommendations for further actions are generated based on this information and the formalized expert knowledge.

This article is structured as following: First, fault diagnosis, test case selection and other maintenance techniques are

described in section II. In section III the concept is presented. Section IV depicts a prototypical realization with an exemplary test system. The transferability to other systems is described in section V. Finally we conclude our work in section VI.

II. Fault Diagnosis, Test Case Selection and Predictive Maintenance

Assistance for fault diagnosis and test case selection are well-known techniques to improve the fault management. Fault diagnosis systems are used to determine the cause of a fault automatically with the goal to minimize the repair time. Test case selection is used to find the most appropriate test cases to test a system, especially after changes and bug fixes. With test case selection, the test effectiveness can be increased. Less faults remain in the system and a shorter time is needed for the test.

A. Fault Diagnosis

Fault diagnosis systems have the goal to find the cause of a fault based on symptoms. The cause is usually not directly observable. The fault is revealed by its impact to system functions. The impact of a fault could be for example a broken hardware component or an error message of a software. To be able to fix the fault, its cause has to be found. In order to find fault symptoms, common fault diagnosis methods monitor sensor signals (signal based methods) [1], compare the system behavior with a modeled one (model based methods) [1] or compare the current system state with failures from the past (knowledge-based, data-driven or fault-based methods) [2], [3]. To find the fault cause based on the identified fault symptoms, classification and inference methods are used. Typical classification methods are e.g. decision trees [4] and support vector machines (SVM) [2]. Typical inference methods are e.g. rule-based inference [5], fuzzy logic [2] or Case-Based Reasoning (CBR) [6]. A more complete overview of diagnosis methods and applications can be found in [1] and [2] or in [7].

The system presented in this paper uses case-based reasoning to find the fault cause and to give a recommendation for further handling. case-based reasoning is a methodology that has been developed to solve problems by adapting similar cases from the past to the current situation. The idea of case-based reasoning is to match a human's approach to problem solving by remembering similar problems from the past. CBR has proven its usefulness for fault diagnosis in numerous applications. CBR has been chosen because it matches the way of thinking in fault cases of the automotive testing experts.

B. Test Case Selection

Test case selection techniques support test engineers in finding the most appropriate test cases for effective regression testing after changes in a system [8]. Due to short time and resources, the number of test cases which can be run in one test cycle is limited [9]. Therefore the test cases covering system changes and modules, which depend on changed modules, are selected. A lot of test case selection techniques arise from software engineering. Changes compared to a previous release can be determined by comparing the two source code versions. Test cases are then selected by maximizing the change coverage. Those techniques are called white-box techniques because they require source code access. If there is no source code information available, e.g. in purchased modules, or if evaluating the source code is not sufficient, e.g. if also the hardware changes, other techniques need to be used. Various techniques that do not depend on source code have been proposed, which evaluate e.g. requirements [10] or historical information [11]. Preceding research of the authors investigated test case prioritization with learning software agents [12]. A more complete overview of test case selection can be found in the survey articles [8] and [13].

C. Condition Monitoring, Active Fault Diagnosis and Predictive Maintenance

Condition Monitoring is the supervision of the condition of a technical system [14]. Often, condition monitoring is used to identify and to diagnose faults. Condition monitoring data is used by the presented assistance system for symptom identification. Usually the condition monitoring is passive and does not interfere with the technical process of the monitored system. In contrast, the active fault diagnosis interferes actively with the technical process in order to get more information about the system condition.

The availability of system condition information allows the predictive maintenance. It summarizes techniques to prevent system breakdowns by forecasting them. According maintenance actions are scheduled to prevent the forecast breakdowns. More information and applications can be found in [15] and [16].

III. Design and Concept of the Combined Assistant System

The combination of fault diagnosis and test case selection allows the usage of diagnosis information for regression testing and to support the fault diagnosis with the selection of test cases. A common scenario for the interaction between fault diagnosis and test case selection is depicted in Fig. 1.

The basic system architecture is shown in Fig. 2. It contains a test system specific and an independent part. The independent part manages the test system from a functional point of view. It integrates the different system variants from the particular assembly lines. The system specific part contains the monitoring adapters to interface the system monitoring of each different supplier.

A. Requirements and Boundary Conditions

To be feasible, the assistance system needs to consider boundary conditions that arise from the workflow and the use cases of the automotive production. The most important one is the independence from particular suppliers. The test system is specified during the engineering of a new assembly line. Based on the specification, the system is delivered by one or various

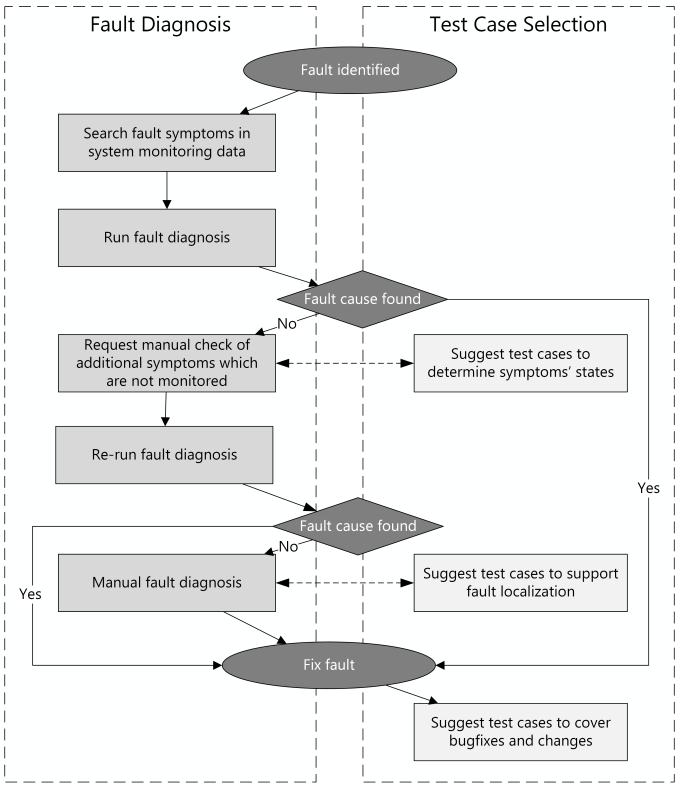

Fig. 1. Scenario for the interaction between fault diagnosis and test case selection

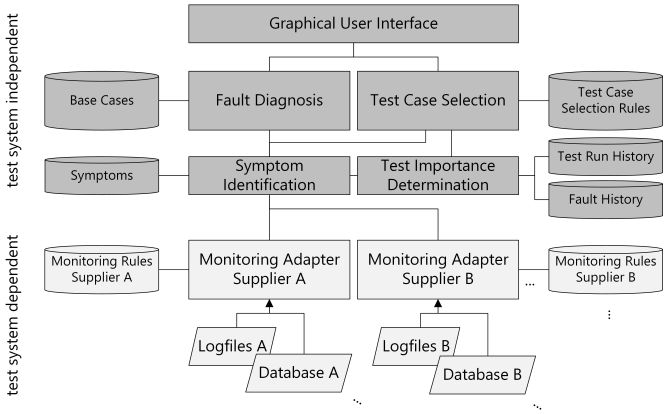

Fig. 2. System architecture of the assistance system

suppliers. The system is managed based on its specification equally for all assembly lines but might be realized differently.

As a second boundary condition, the diagnosis-relevant time span is set to the duration of one car test. Testing cars is the main action in which faults are revealed and in which cause-effect relations can be observed. The symptom identification filters for one relevant car test. This fits to the manual workflow in which the maintainers usually investigate single failed car tests.

B. Design of the Shared Data Model

To fulfill its tasks, the assistance system needs a data model which contains all relevant information for fault diagnosis and test case selection. One of the major advantages of the assistance system is the ability to process a lot of data and to consider many dependencies compared to human experts. Fig. 3 depicts the data model that has been developed for the assistance system. It contains data needed for fault diagnosis and for test case selection.

For the fault diagnosis, information about the system composition and about faults and fault symptoms is needed. The system composition is stored in a common model of components and functions. Particular systems consist of various components and are stored as configuration baselines in the model.

The test case selection requires data about the test process. Test cases and test runs are stored as well as changes in the system. Especially the information about test case coverage and dependencies between functions and components is used to select test cases. This data is usually stored in specialized test management tools. A monitoring adapter for the used test management tool is introduced, which gives the assistant system access to the test data.

C. Symptom Identification by System Monitoring

To enable fault diagnosis, the current system state of the test system has to be known by the diagnosis system. For the fault diagnosis the relevant system state is abstracted to a collection of symptoms which can occur when the test system is faulty. Newly found symptoms of new faults are added subsequently to the symptom model by the maintainers to enable automatic supervision of that symptoms. The first task of the diagnosis system is to identify which of the symptoms are occurred. There are two options for the symptom identification: The user can be asked directly or the system monitoring can be used. The usage of the system monitoring is preferred. However the diagnosis system needs the knowledge about how a symptom's state is contained within the system monitoring data.

The system monitoring is supplier specific whereas the symptom model is used for the whole system. Therefore the diagnosis system uses a supplier specific monitoring adapter for each test system. The adapter contains specific rules for symptom identification (see also Fig. 2). To identify faults, the system-independent symptom identification module requests a symptom's state from the system-specific system monitoring adapter. A request always contains a time-span in which the symptom has to be occurred and the symptoms to search for. The adapter answers for each requested symptom with the state or with a message, that the state could not be identified. There are three reasons why the identification might fail:

- There is no monitoring rule available.
- The relevant time-span is not contained in the monitoring data anymore. This might happen for example when the file size of logfiles is limited.
- The function to which the symptom belongs to has not been executed. The system state is set to *not executed*.

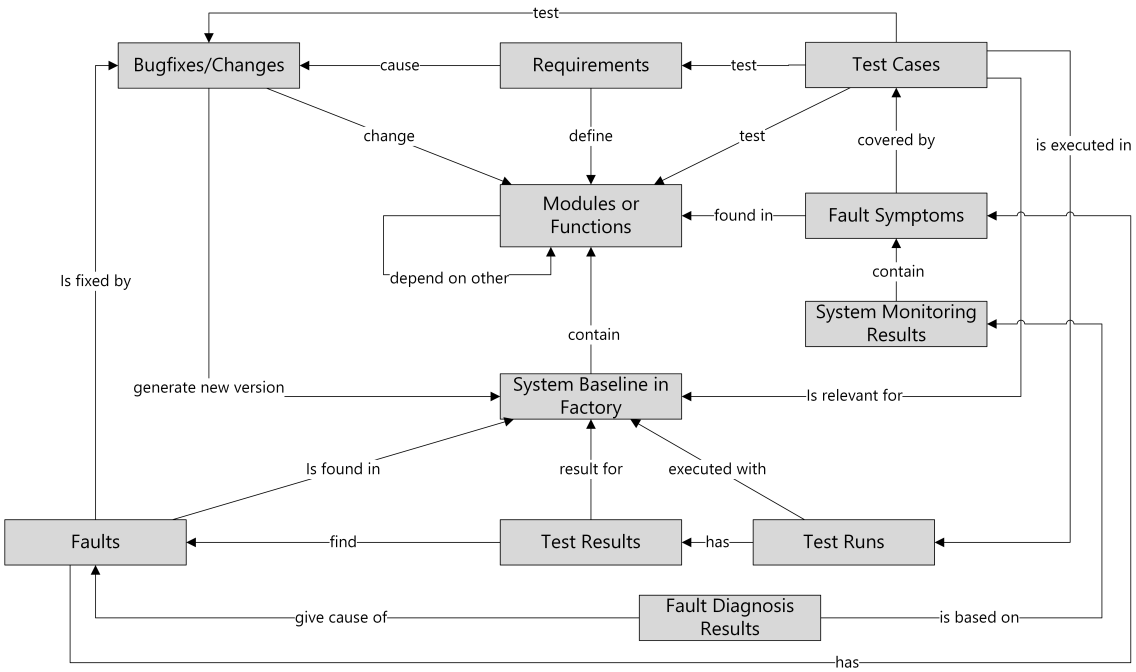

Fig. 3. Shared data model for fault diagnosis and test case selection

The determination of this state requires that the execution of functions is monitored and that the symptoms are linked to that functions.

The monitoring adapter itself uses different data sources to search for symptoms. The most important for the test system are the result logging database and logfiles generated by the test runtime environment which is running on the test devices. For each data source a different kind of rule is used. A rule for the log database contains a SQL statement and a result value identifying that the according symptom is occurred. A rule for the logfile contains a regular expression to identify the message which indicates that the symptom is occurred.

D. Fault Diagnosis with Case-Based Reasoning

Based on the symptoms and their states, the fault diagnosis searches for the fault cause and for a recommendation for further actions. Therefore the CBR compares diagnosis cases with previous solved cases. A diagnosis case consists of the symptoms with their states when a fault occurs and the fault cause together with a recommendation for further action.

To diagnose a fault, a case is constructed from all symptoms of which the state is known. The CBR then searches for similar cases from the past which are called base cases and presents their solution as a candidate for the fault cause of the current case. The similarity is determined by summing up the number of symptoms with the equal state. Symptoms with the state *occurred* are weighted twice . Symptoms that are not contained in the base case are ignored.

The base cases are grouped by the fault causes they represent. There might be various cases describing the same problem. To retrieve the most probable cause for the current problem, the cause with a base case with the highest similarity is chosen. If two base cases with different fault causes have the same similarity value, the fault cause with a base case with the highest unique similarity value is chosen first.

After the CBR has finished, the maintainer gets a list of the most probable fault causes. The maintainer now has the possibility to acknowledge one of the causes as the correct cause for the current diagnosis case. Then the current symptoms are stored as a new base case for this cause. If the correct fault cause could not be found, the maintainer has the possibility to manually enter the correct solution. Therefore he can define new symptoms if he investigated something which is not yet covered by an existing symptom. Then the current case is inserted as a new base case with a new fault cause. By acknowledging or entering correct solutions, the ability of the system to diagnose faults correctly grows over time.

E. Test Case Selection to Support Fault Diagnosis

If the diagnosis cannot find a unique fault cause, more information about the current system state is needed. Test cases are executed to get this information. They provoke further symptoms which finally lead to the correct fault cause. The assistance system should find the best suited test cases for the current diagnosis situation. The decision mechanism depends on whether detailed test case – symptom coverage information is available. This information is usually incomplete since the modeling effort is too high for a complete and up-to-date manual modeling. However, the coverage can be determined automatically when after a test case execution, the symptom identification is executed immediately. All symptoms that were found after a test run are then linked automatically to

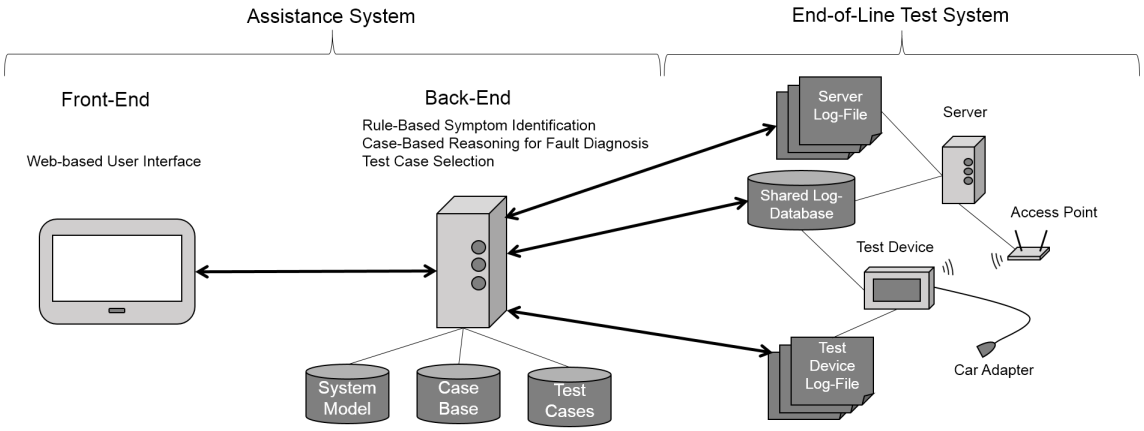

Fig. 4. Protoypical realiziation of the assistance system with an exemplary end-of-line test system

the corresponding test case. In upcoming diagnosis situations this information is then used to find suited test cases. First, symptoms to distinguish two or more non-unique diagnosis results are identified. Therefore symptoms with an unknown state are searched that are contained in the most similar base cases. If they have a different state in the different base cases then they are suited to find the correct solution. This suited symptoms are called relevant symptoms for test case selection. After the relevant symptoms have been identified, the link between them and corresponding test cases is used to select test cases which provoke the symptoms. The number of provoked relevant symptoms is counted for each test case. Then the test cases are ordered by the number of provoked symptoms. After a test case has been executed, the fault diagnosis is started again. If the fault cause could not be found, the test case selection is started again.

If the coverage information between relevant symptoms and test cases is not available or if no relevant symptoms were found, the test case selection uses meta data about system components to calculate a fault probability for each component. If there were already symptoms found which can be linked to a component, test cases are selected that test especially this component and components that are dependent from it. If no symptoms have been found, historical data is used to determine a diagnosis-independent fault probability. This historical data includes the fault-history, the change-history and the test run-history. The number of changes, the number of faults and the number of failed test runs is summed up for each component. Test cases are selected to cover the components with the highest fault probability.

F. Test Case Selection for Regression Test

Changes to solve faults or new functions to fulfill new requirements lead to changes of the test system. These changes are tested in a regression test. To keep the production down-time short, the best test cases to test especially the changed system components are selected. Therefore a release note which describes the changes of the components is read in. In the first step, the assistance system calculates a test importance of each component. Besides the changes themselves the test importance is influenced by changes in dependent components and a user given criticality value. The test importance is calculated as the mean value of the three values *number of changes*, *number of changes in dependent components* and *criticality of the component*.

IV. Prototypical Realization

The assistance system has been realized as a prototype. Its setup is depicted in Fig. 4. It consists of the assistance system itself and an exemplary test system. The assistance system consists of a front-end with the user interface and a back-end that realizes the functions which were described in section III. The database stores the data of the data model from Fig. 3. The front-end provides basic editing functions for this data to give the possibility to enter scenario-related data. A condition monitoring adapter has been realized for logfiles and for a log database. In the exemplary system, historical logfiles or database entries with faults included can be used instead of live condition monitoring data.

Common faults that occur in the communication between the diagnosis device and a ECU in the car have been used to test the assistance system. In the first test, the complete data was available. The test system was modeled in 33 components and 25 functions. 53 symptoms have been defined that might occur during the communication. Then base cases were constructed by asking experts for the steps they would perform to find the fault cause for 34 common faults. Every step was modeled as one of the symptoms. If the symptom is contained in the database or a logfile, a monitoring rule was added.

A common exemplary fault is a loose connection of at least one pin of the car adapter plug. The fault usually leads to a failed car test with a time out or communication error. Fault symptoms are numerous connection losses and reconnects which can be found in the networking log of the test device. Another common fault is the missing of needed files for flashing the ECU which is either caused by a problem in the communication between the server to the test device or

in a wrong device configuration. The cause is distinguished by checking the server log if the test device tried to load the correct data.

After the base cases and the symptom rules had been established, we put the logfiles and database entries containing the real faults on the test system and started the diagnosis process. With this full information given, the diagnosis system was able to identify all symptoms with the corresponding rule. With the overview of occurred symptoms, the experts were already able to give the correct diagnose. The symptom identification itself reduces the fault diagnosis time. The automatic fault diagnosis was able to find 2 out of 3 fault causes correctly. For the last third, the system was not able to distinguish between multiple possibilities. More symptoms would be needed to give a unique fault diagnosis for those faults.

To test the test case selection, we added some test cases with random coverage to the system. Then we executed the test case selection for fault symptom identification and to improve the regression test and reviewed the results by manually investigating the coverage and interdependency links. It showed that the system is able to find the right test cases if some are available. Nevertheless the success depends on the quality of the test cases and the given meta-information.

The ability of the system to handle incomplete data and new faults has to be proven in a long-term case study at a real assembly line. For the exemplary test system with reproduced faults, the system already works well and is suited to support the maintenance engineers.

V. Transferability of the concept

During design of the assistance system, an abstract system model has been used, which consists basically of components, functionalities and fault symptoms. The abstraction is necessary to achieve independence from the realizations of particular suppliers which deliver parts of the system. This abstraction allows for a easy transfer of the assistance system for the usage with other systems. The main task to transfer it is the modeling of the new system, the setup of an initial symptom model and the realization of the new system monitoring adapters.

VI. Conclusion and further work

Fault Diagnosis and Test Case Selection are important tasks to maintain the availability of automotive end-of-line test systems. This article introduced an assistance system that supports the engineers by formalizing their knowledge and experience for both tasks in a shared data model. With that shared data model, information from fault diagnosis is considered automatically in the system test and the test case selection helps in finding and isolating faults for the diagnosis.

The system has been realized prototypically. The experiments with this prototype showed that it is able to diagnose reproduced faults when the necessary knowledge is contained in the data. If it is not, the system is still helpful by providing information about already known symptoms and thus by shortening the manual diagnosis resulting in less maintenance cost and shorter production downtimes. With enough test cases available and full coverage information given, the system is able to find test cases to support symptom identification and to increase the test coverage of the whole test system. Thus we could show that the concept of the combined fault diagnosis and test case selection works. Nevertheless, the assistant system has to prove its function in a long-term evaluation under real conditions, e.g. with incomplete data and fault situations, which are unknown yet.

The next step is that evaluation using a end-of-line test system at a real assembly line. The system has to prove that it can handle unknown situations and is able to support the maintainers even with incomplete data. The test case selection technique has to be investigated for various system releases. It has to be determined if the selected test cases really helped to reduce the fault-proneness of the system.

References

[1] Z. Gao, C. Cecati, and S. X. Ding, "A Survey of Fault Diagnosis and Fault-Tolerant Techniques Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[2] Z. Gao, C. Cecati, and S. Ding, "A Survey of Fault Diagnosis and Fault-Tolerant Techniques Part II: Fault Diagnosis with Knowledge-Based and Hybrid/Active Approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3768–3774, 2015.

[3] M. Bordasch, C. Brand, and P. Göhner, "Fault-based identification and inspection of fault developments to enhance availability in industrial automation systems," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2015, pp. 1–8.

[4] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge and New York: Cambridge University Press, 2012.

[5] H. Liu, A. Gegov, and M. Cocea, *Rule Based Systems for Big Data: A Machine Learning Approach*, ser. Studies in Big Data. Heidelberg: Springer, 2016, vol. 13.

[6] J. Kolodner, *Case-Based Reasoning*. Burlington: Elsevier Science, 2014.

[7] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*, 3rd ed. Berlin and Heidelberg: Springer, 2016.

[8] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[9] C. Malz, N. Jazdi, and P. Göhner, "Prioritization of Test Cases Using Software Agents and Fuzzy Logic," in *IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST)*, 2012, pp. 483–486.

[10] H. Srikanth, L. Williams, and J. Osborne, "System test case prioritization of new and regression test cases," in *International Symposium on Empirical Software Engineering*, 2005, pp. 62–71.

[11] J.-M. Kim and A. Porter, "A history-based test prioritization technique for regression testing in resource constrained environments," in *Proceedings of the 24rd International Conference on Software Engineering (ICSE)*, 2002, pp. 119–129.

[12] S. Abele, M. Bordasch, and P. Göhner, "Qualitätsbasierte Testfallpriorisierung mithilfe von Softwareagenten," in *Entwurf Komplexer Automatisierungssysteme (EKA)*, 2014.

[13] E. Engström, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," *Information and Software Technology*, vol. 52, no. 1, pp. 14–30, 2010.

[14] A. K. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006.

[15] R. K. Mobley, *An Introduction to Predictive Maintenance*, 2nd ed. Amsterdam and New York: Butterworth-Heinemann, 2002.

[16] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine Learning for Predictive Maintenance: A Multiple Classifier Approach," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812–820, 2015.