# Automatisierte Datenauswertung zur Fehlerdiagnose und Absicherungsunterstützung für Qualitätssicherungssysteme

Dipl.-Ing. **Sebastian Abele**,
Prof. Dr.-Ing. **Michael Weyrich**,
Universität Stuttgart, Institut für Automatisierungstechnik und Softwaresysteme
Pfaffenwaldring 47, 70550 Stuttgart

**Kurzfassung**

Systeme zur industriellen Produktion in einer Smart Factory arbeiten mehr und mehr daten- und informationsorientiert. Exemplarisch dafür stehen bereits heute Qualitätssicherungssysteme, deren Funktionsbasis in der Regel durch eine komplexe IT-Infrastruktur mit aufwändiger Datenlogistik gebildet wird. Dieser Beitrag stellt ein Assistenzsystem vor, das das Fehlermanagement von solchen Qualitätssicherungssystemen unterstützt, indem es eine automatisierte Fehlerdiagnose mit einer Unterstützung bei der Absicherung von Neuerungen und Änderungen, zum Beispiel nach einer Fehlerbehebung, kombiniert. Grundlage der Assistenzfunktionen bilden eine Datenintegration und Datenauswertung. In dem entwickelten Verfahren werden unterschiedliche Datenquellen erfasst und mit Systemwissen verknüpft, das von Experten abgefragt wird.

## 1. Einleitung

Die Qualität von hergestellten Produkten wird ein zunehmend wichtiger Faktor im Wettbewerb. Dies führt dazu, dass Qualitätssicherungssysteme ein wichtiger Teil des Wertschöpfungsnetzwerkes von industriellen Produzenten geworden sind. Entlang der gesamten Fertigungslinie finden Qualitätssicherungsmaßnahmen statt. Diese tiefe Integration der Qualitätssicherungssysteme in den Produktionsprozess macht die Qualitätssicherungssysteme zu kritischen Systemen innerhalb der Fertigung. Probleme und Ausfälle in den Qualitätssicherungssystemen wirken sich direkt auf die Produktion aus. Wird ein Fehler im Qualitätssicherungssystem nicht gleich entdeckt, gelangen womöglich fehlerhafte Produkte in den Markt, oder der Ausschuss bzw. der Nacharbeitsaufwand steigt, ohne dass das Produkt selbst fehlerhaft war.

Durch die Befragung von Prüfexperten aus der Automobilindustrie wurden Fehlerdiagnose und Änderungsabsicherung als wichtige Tätigkeitsfelder für eine hohe Verfügbarkeit der Qualitätssicherungssysteme identifiziert. Die Fehlerdiagnose hat das Ziel, die Auswirkungen eines Fehlers zu minimieren, indem die Fehlerursache möglichst schnell gefunden wird, damit sie rasch behoben werden kann. Durch die Änderungsabsicherung soll verhindert werden, dass Fehler durch Änderungen ins System eingebracht oder aufgedeckt werden.

Beide Aufgaben sind sehr zeitaufwändig und setzen ein hohes Maß an vorhandenem Expertenwissen und Erfahrung voraus. Viele Teilschritte der beiden Tätigkeiten werden von den Experten manuell durchgeführt. Entsprechend hoch ist der Zeitaufwand. Viele Teilschritte wiederholen sich dabei häufig. Das Potential, diese Tätigkeiten zu automatisieren und damit zu beschleunigen, ist hoch. Eine typische Tätigkeit ist das Durchsuchen des Ergebnisprotokolls und der System-Logs nach Auffälligkeiten, die auf die Fehlerursache hindeuten.

Bei der Änderungsabsicherung liegt die Herausforderung darin, die kurze Zeit für das Testen optimal zu nutzen. Dazu muss der Testplaner entscheiden, wie der Testaufwand am besten verteilt werden kann indem er beispielweise gezielt Änderungen berücksichtigt, die eingeführt wurden, um einen Fehler zu beheben oder um neue Funktionen umzusetzen. Da für diese Aufgabe, wie auch für die Fehlerdiagnose, ein großes Wissen über das System benötigt wird, sind die gleichen Experten daran beteiligt. Implizit nutzen sie dabei ihr Wissen über das System und über fehleranfällige Teilsysteme, um eine bestmögliche Änderungsabsicherung zu erreichen.

Insbesondere die steigende Komplexität und zunehmende weltweite Verteiltheit der Systeme führen das expertenbasierte Vorgehen an seine Grenzen. Wissen und Erfahrung werden nicht ausreichend dokumentiert und weitergegeben und der Personenkreis, der über das notwendige Wissen verfügt, schrumpft. Um diesen Herausforderungen entgegenzuwirken, erforscht das IAS ein Assistenzsystem, das die beiden Tätigkeitsfelder unterstützt. Eine Hauptfunktion des Assistenzsystems liegt in der automatischen Datenauswertung, auf deren Grundlage dann weiter verfahren werden kann. Für die Fehlerdiagnose werden im Wesentlichen Ergebnis-Protokolle und System-Logs ausgewertet, für die Änderungsabsicherung Release-Notes und Systemmodelle.

## 2. Fehlerdiagnose und Testmanagement

Die Fehlerdiagnose hat zum Ziel, die Ursache eines Fehlers zu finden. Die Fehlerursache ist oftmals nicht direkt ersichtlich. Der Fehler offenbart sich über die Auswirkungen, die er auf das betroffene System und seine Funktionen hat. Solch eine Auswirkung kann beispielsweise eine Software-Fehlermeldung sein. Ausgehend von diesen Auswirkungen wird nach Fehlersymptomen gesucht, die es letztendlich ermöglichen, auf die Fehlerursache zu schließen.

Zur Suche von Fehlersymptomen und zum Rückschluss auf die Fehlerursache wurden zahlreiche Methoden entwickelt. Methoden, die überwachte Systemgrößen beobachten, werden als signalbasierte Verfahren bezeichnet [1]. Verfahren, die das Systemverhalten mit einem Modell vergleichen, werden als modellbasierte Verfahren bezeichnet [1]. Ergänzt werden diese Methoden durch wissensbasierte und fehlerbasierte Verfahren [2]. Dabei werden Fehlerzusammenhänge als Wissen direkt formalisiert oder im Falle der fehlerbasierten Verfahren aus der Fehlerhistorie abgeleitet [3].

Um von identifizierten Fehlersymptomen auf die Fehlerursache zu schließen, kommen Klassifikations- und Inferenzmethoden zum Einsatz. Typische Klassifikationsmethoden sind beispielsweise Entscheidungsbäume [4] und Support Vector Machines [2]. Typische Inferenzmethoden sind zum Beispiel das regelbasierte Schließen [5], Fuzzy-Logik [2] oder das fallbasierte Schließen (engl. Case-Based Reasoning, CBR) [6]. Das in diesem Artikel vorgestellte System verwendet einen regelbasierten Ansatz zur Symptomidentifikation auf Grundlage verschiedener Datenquellen und fallbasiertes Schließen, um von den Symptomen auf die Fehlerursache zu schließen. Beim fallbasierten Schließen wird die momentane Fehlersituation mit Fehlern aus der Vergangenheit verglichen. Dabei wird von der Ähnlichkeit der Fehlerfälle auf die Ähnlichkeit ihrer tatsächlichen Ursachen geschlossen. Damit eignet sich das fallbasierte Schließen, die bisherige Arbeitsweise der Experten abzubilden und leistet damit einen Beitrag zur Formalisierung des Expertenwissens.

Zur Absicherung von Änderungen, insbesondere im Regressionstest der Softwareentwicklung, wurden ebenfalls zahlreiche Methoden entwickelt [7]. Viele von ihnen beschäftigen sich mit der Auswahl und Priorisierung von Testfällen. Diese Methoden haben zum Ziel, beschränkte Zeit und knappe Ressourcen bestmöglich zu nutzen, wenn nicht alle vorhandenen Testfälle ausgeführt werden können [8]. Wichtige Einflussgrößen für die Testfallauswahl und –priorisierung sind Änderungen seit dem letzten Test und

Abhängigkeiten zwischen verschiedenen Modulen. Bei der Softwareentwicklung können diese Änderungen durch einen Vergleich der beiden Quellcodeversionen relativ genau bestimmt werden. Man spricht in diesem Zusammenhang auch von White-Box-Techniken. Im Falle der Qualitätssicherungssysteme gibt es jedoch auch Hardwaremodule oder zugekaufte Module, für die kein Quellcode zur Verfügung steht. In diesem Fall werden andere Parameter herangezogen, wie zum Beispiel Anforderungen [9], oder Historiendaten [10].

### 3. Gemeinsames Datenmodell aus automatisierter Datenauswertung

Der Hauptnutzen des Assistenzsystems liegt darin, dass es automatisiert eine große Menge an heterogenen Daten auswerten kann. Zur Fehlerdiagnose werden beispielsweise Informationen aus dem System Monitoring und Fehlerfälle aus der Historie herangezogen. Zur Absicherungsunterstützung kommen noch Informationen aus der Entwicklung und dem Test des Systems hinzu. Es ergibt sich ein komplexes Datenmodell, das durch den Experten alleine nicht mehr vollständig überblickt werden kann. Das Assistenzsystem hilft ihm dabei, die relevanten Informationen zu filtern, aufzubereiten und daraus Handlungsvorschläge zu generieren.

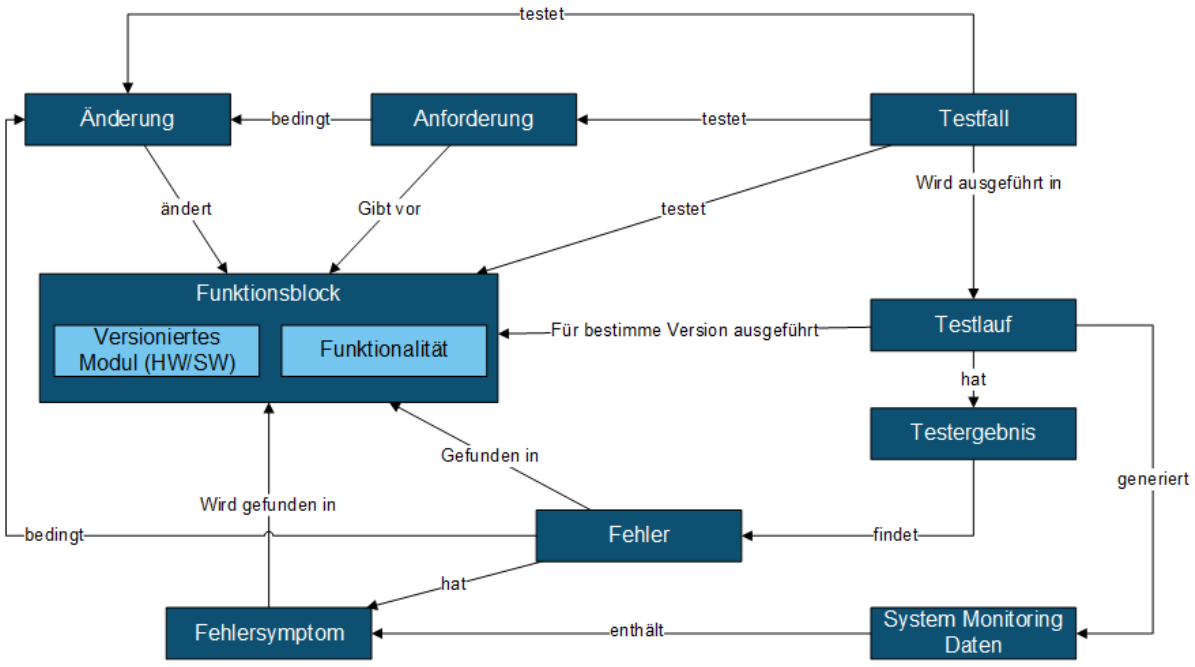


Abbildung 1: Datenmodell des Assistenzsystems

Durch die Verwendung eines einzelnen Assistenzsystems für Fehlerdiagnose und Änderungsabsicherung wird es ermöglicht, ein gemeinsames Datenmodell zu erstellen und

zu pflegen und für beide Assistenzfunktionen zu verwenden. Viele der Informationen, zum Beispiel über den Aufbau des Systems und Abhängigkeiten zwischen den Komponenten, werden von beiden Assistenzfunktionen benötigt. Abbildung 1 zeigt das Datenmodell des Assistenzsystems.

## 4. Assistenzfunktion Fehlerdiagnose

Die Fehlerdiagnose hat zum Ziel, Fehler im überwachten Qualitätssicherungssystem zu identifizieren und zu diagnostizieren. Damit soll verhindert werden, dass ein Fehler die Produktion beeinflusst, zum Beispiel indem Produkte fälschlich als defekt erkennt werden. Die Fehlerdiagnosefunktion ist in Abbildung 2 dargestellt. Sie wurde in zwei Teile aufgeteilt, die separat voneinander ablaufen. Der erste Teil (blau) übernimmt die Symptomidentifikation, der zweite Teil die eigentliche Fehlerdiagnose, indem die Fehlerursache gesucht wird (rot).

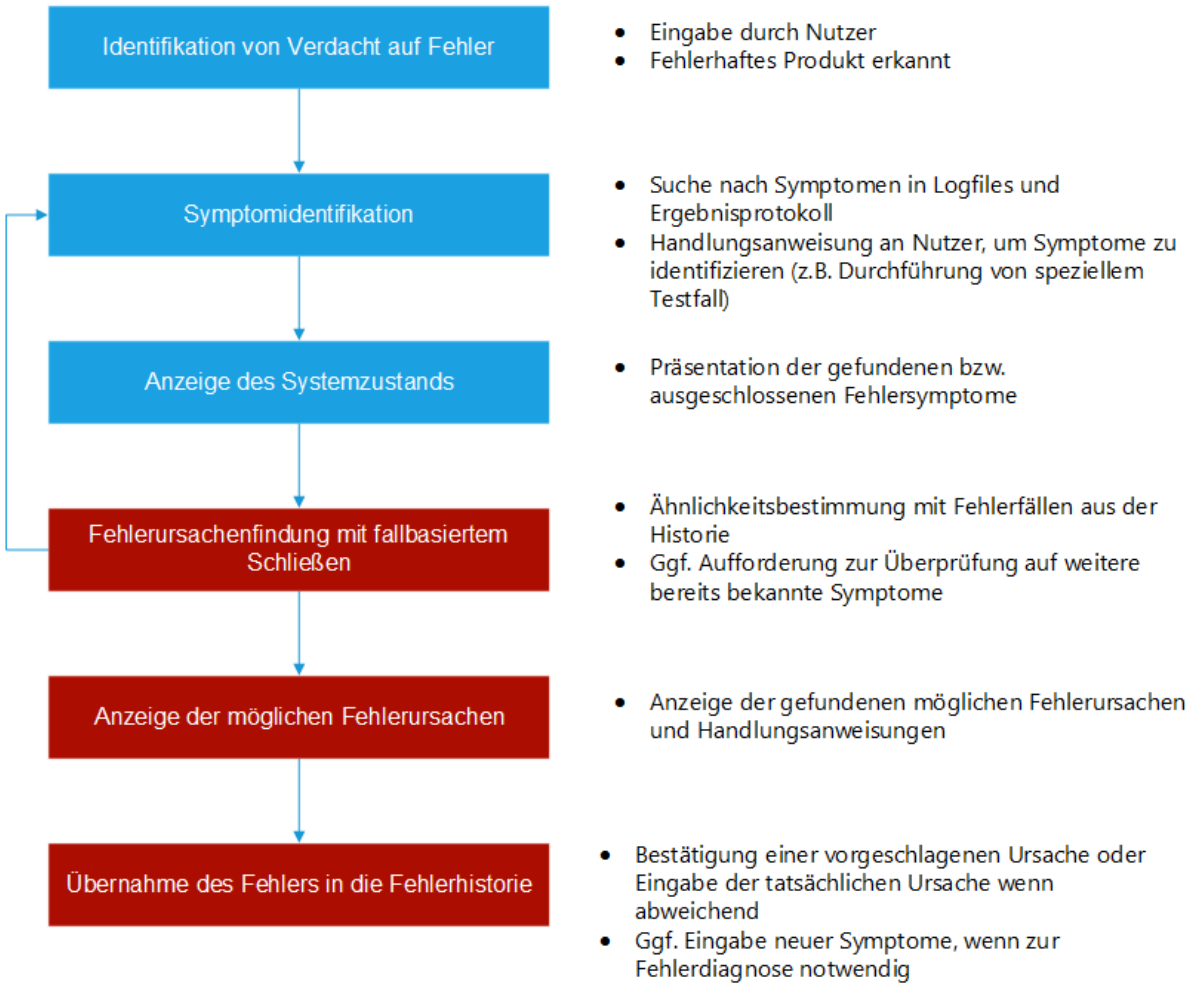


Abbildung 2: Ablauf der Fehlerdiagnose

Ausgangspunkt für die Fehlerdiagnose ist der Verdacht auf einen Fehler. Das Diagnosesystem sucht daraufhin nach Fehlersymptomen. Diese Symptome zusammengefasst ergeben den Systemzustand. Der Systemzustand kann global für das gesamte System oder nur für Teilsysteme, z.B. einzelne Überprüfungsstationen, erfasst werden. Zu Beginn wird eine Ausgangsmenge an Symptomen modelliert. Während das System im Betrieb ist, lernt es neue Symptome durch neu auftretende Fehlerfälle. Die Experten geben ein, welche Schritte zur tatsächlichen Fehlerursachenfindung nötig waren. Daraus werden dann formelle Symptome abgeleitet, die durch das System erfasst werden können.

Für jedes bekannte Fehlersymptom kann eine Auswerteregel hinterlegt werden. Diese Regel beschreibt, wie das Symptom erkannt werden kann. Für die Regel wird zwischen drei Zuständen unterschieden, in die ein Symptom gesetzt werden kann: *aufgetreten*, *nicht aufgetreten* oder *nicht ausgeführt*. Das Symptom wird in den Zustand *nicht ausgeführt* gesetzt, wenn die Funktion, zu der das Symptom gehört, nicht verwendet wurde. Die Bestimmung dieses Zustandes ermöglicht es der Testmanagement-Funktionalität, die Testabdeckung hinsichtlich aufgetretener und überprüfter Symptome zu optimieren.

Wie in Abbildung 3 gezeigt, wurde die Symptomidentifikation modular aufgebaut. Dadurch ist es möglich, für verschiedene Datenquellen unterschiedliche Überwachungsadapter einzusetzen. Für das untersuchte Szenario der Qualitätssicherungssysteme wurden drei Adapter realisiert. Der erste durchsucht Logfiles, der zweite das Ergebnisprotokoll in einer Datenbank und der dritte dient dazu, nicht automatisch überwachbare Symptome vom Nutzer abzufragen. Der Logfiles Adapter kann sich per FTP oder SCP mit einem Teilsystem verbinden und erhält so Zugriff auf Logfiles, die lokal auf den Geräten abgelegt werden. Diese Logfiles werden zunächst auf den relevanten Zeitraum gefiltert. Dieser Zeitraum ist in der Regel die Dauer einer fehlerhaften Prüfung. Innerhalb dieses relevanten Zeitraums wird nun nach Einträgen gesucht, die für ein Fehlersymptom charakteristisch sind. Dies können zum Beispiel Fehlermeldungen, oder auch Grenzwertverletzungen sein. Der Datenbank-Adapter greift auf eine relationale Datenbank zu, die zum Beispiel als Ergebnisprotokoll der durchgeführten Prüfungen dienen kann. Symptome werden anhand von charakteristischen Datenbankeinträgen erkannt. Auch hier wird zunächst der relevante Zeitraum bestimmt und innerhalb des Zeitraums nach den Symptomen gesucht. Dazu werden in der Regel SQL-Statements hinterlegt, mit denen es möglich ist, die drei Symptomzustände *aufgetreten*, *nicht aufgetreten* und *nicht ausgeführt* zu erkennen. Der dritte Adapter zur Nutzerbefragung wird

verwendet wenn keine passende Überwachungsregel hinterlegt ist, der Symptomzustand jedoch für die Diagnose benötigt wird. Der Nutzer wird dann darum gebeten, den Symptomzustand manuell zu überprüfen.

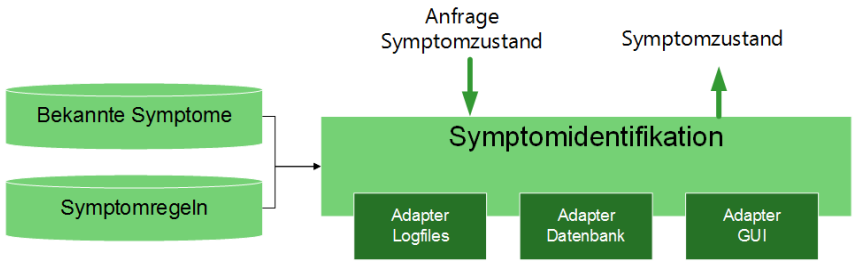


Abbildung 3: Symptomidentifikation

Alle überwachten Teilsysteme des Qualitätssicherungssystems werden mit ihrem erkannten Fehlerstatus auf einem Fehlerdiagnose-Dashboard dargestellt. Für jedes Teilsystem wird angezeigt, ob momentan Fehlersymptome entdeckt wurden. Der Nutzer kann sich die entdeckten Fehlersymptome anzeigen lassen und die eigentliche Fehlerdiagnose starten. Durch die Ansicht der gefundenen Fehlersymptome kann der Experte bereits eine Einschätzung abgeben, was die Fehlerursache sein könnte. Auch weiß er, was bereits überprüft wurde und an welchen Stellen er womöglich weiter suchen muss.

Die automatische Fehlerdiagnose sucht auf Basis des erkannten Systemzustands nach möglichen Fehlerursachen. Dazu verwendet sie den Ansatz des fallbasierten Schließens. Der momentane Systemzustand mit seiner Menge an gefundenen Symptomen wird mit vergangenen Fehlerfällen verglichen. Ähnliche Fehler aus der Vergangenheit haben mit hoher Wahrscheinlichkeit die gleiche oder eine ähnliche Fehlerursache. Die Ähnlichkeit der Fehlerfälle wird anhand der aufgetretenen Symptome bestimmt. Je mehr übereinstimmende Symptome gefunden oder auch ausgeschlossen werden konnten, desto ähnlicher ist ein Fehlerfall.

Die wahrscheinlichsten Fehlerursachen werden dem Nutzer präsentiert. Der Nutzer kann nun bestätigen, dass die gefundene die tatsächliche Fehlerursache war. Der momentane Systemzustand wird dann als Fall in die Fehlerhistorie aufgenommen und in Zukunft für die Fehlerdiagnose mitberücksichtigt. War die Diagnose nicht zutreffend, so kann der Experte die tatsächliche Fehlerursache und den Weg zur Ursachenfindung eingeben. Dazu wird er gefragt, welche Maßnahmen er zusätzlich durchgeführt hat, um auf die Fehlerursache zu

kommen. Aus dieser Beschreibung können dann zusätzliche Symptome abgeleitet und eingepflegt werden, so dass der Fehler in Zukunft identifiziert werden kann.

**5. Absicherungsunterstützung**

Die Funktion zur Absicherungsunterstützung verfolgt zwei Ziele: Zum einen sollen Änderungen und Neuerungen am abgesicherten System möglichst effektiv abgesichert werden. Zum anderen soll die Fehlerdiagnose unterstützt werden, indem gezielt einzelne Systemmodule abgesichert werden, um den Fehler so weiter einzuschränken. Der prinzipielle Ablauf der Absicherungsunterstützung ist in Abbildung 4 dargestellt.

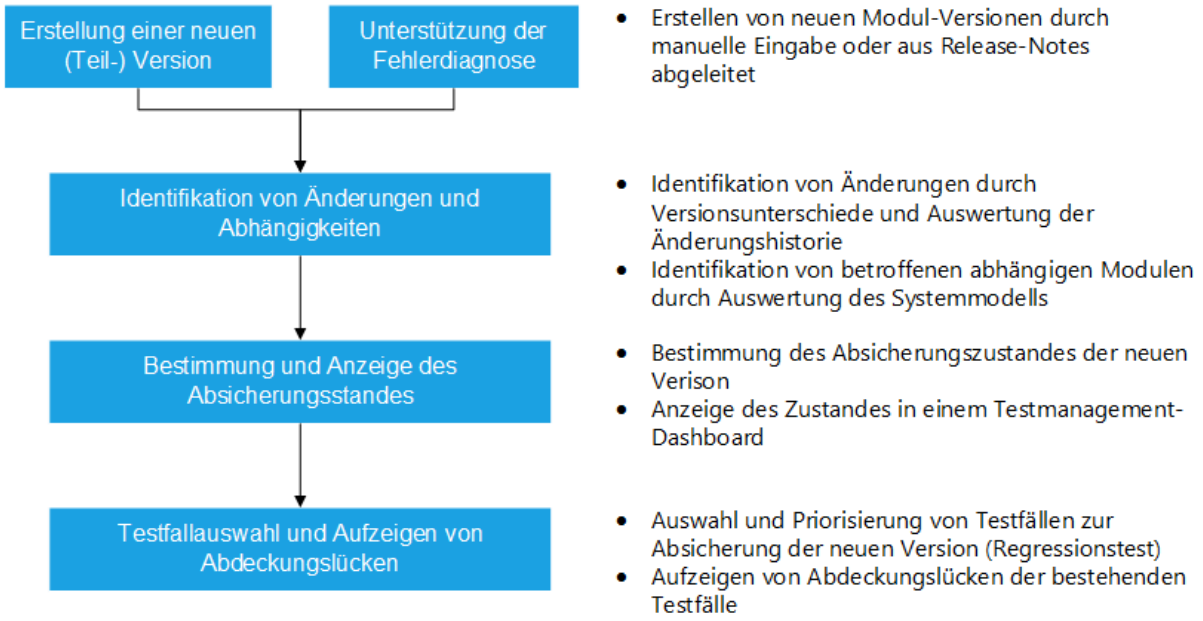


Abbildung 4:Ablauf der Absicherungsunterstützung

Ausgangspunkt für die Absicherungsunterstützung ist entweder eine Änderung am System, die abgesichert werden soll, oder die Notwendigkeit die Fehlerdiagnose durch das Ausführen von zusätzlichen Testfällen zu unterstützen. Bei der Absicherung von Änderungen soll sichergestellt werden, dass das abzusichernde System nach der Änderung weiterhin einwandfrei funktioniert. Dazu sollen speziell solche Testfälle gefunden werden, die die Änderungen und deren Abhängigkeiten testen. Bei der Unterstützung der Fehlerdiagnose soll der Fehler weiter eingeschränkt werden, indem zusätzliche Informationen über das System generiert werden. So sollen zum Beispiel gezielt Testfälle ausgeführt werden, die ein bestimmtes Modul testen, oder ein spezielles Fehlersymptom hervorrufen können.

Wird die Funktion zur Änderungsabsicherung angestoßen, werden zunächst die relevanten Informationen gesammelt. Für die Änderungsabsicherung sind dies Informationen über die Änderungen. Das heißt es wird untersucht, welche Module in welchem Umfang geändert wurden. Diese Information kann entweder vom Nutzer abgefragt oder automatisch aus Release Notes generiert werden. Zur Unterstützung der Fehlerdiagnose übergibt die Diagnosefunktion ein zu testendes Modul oder ein hervorzurufendes Symptom.

Der zweite Schritt besteht darin, den momentanen Absicherungsstand der einzelnen Module bzw. der Funktionalitäten zu bestimmen. Für die Unterstützung der Fehlerdiagnose entfällt dieser Schritt. Dazu wird die Test-, Fehler- und Änderungshistorie ausgewertet. Der Absicherungszustand wird als relativer Wert zwischen den Modulen bestimmt. Je öfter ein Modul seit der letzten Änderung erfolgreich getestet wurde, desto besser wird sein Absicherungszustand bewertet. Wurde eine neue Änderung gemacht, wird der Absicherungszustand als sehr schlecht eingestuft, da noch kein Test für diese Änderung durchgeführt wurde. Der Absicherungsstand wird in einem Absicherungs-Dashboard für jedes Modul und jede Funktionalität dargestellt. Der Absicherungsstand wird anhand einer Farbskala dargestellt, so dass er schnell ersichtlich ist. Mit dieser Information kann der Experte bereits schnell ablesen, an welchen Stellen im System sich eine Änderung auswirkt und an welchen Stellen der Test intensiviert werden sollte. Insbesondere Abhängigkeiten zwischen den Modulen werden aufgelöst und der Absicherungsstand von abhängigen Modulen ebenfalls angepasst.

Ebendiese Abhängigkeiten gilt es auch bei der Testfallauswahl bzw. Testfallpriorisierung zu berücksichtigen. Steht nicht genug Zeit zur Verfügung, um alle Testfälle für eine optimale Absicherung auszuführen, so müssen diejenigen für die verbleibende Zeit ausgewählt werden, die sich am besten zur Absicherung eignen. Dazu werden diejenigen bestimmt, die speziell die Module abdecken, die einen schlechten Absicherungszustand haben. Die Testfallabdeckung wird für jeden Testfall definiert und in der Regel in einem Testmanagement-Werkzeug abgelegt. Von dort werden die Testfallabdeckung und die Ausführungshistorie der Testfälle ausgelesen. Testfälle, die in der Vergangenheit viele Fehler gefunden haben, haben eine höhere Wahrscheinlichkeit auch in Zukunft viele Fehler zu finden.

Um die Testfallpriorität zu berechnen, wird zunächst die Testfallabdeckung mit dem Absicherungsstand gewichtet. Je mehr schlecht abgesicherte Module abgedeckt werden,

desto höher wird die Priorität. Haben zwei Testfälle eine sehr ähnliche Priorität, so wird derjenige bevorzugt, der in der Vergangenheit bereits mehr Fehler gefunden hat.

Werden Testfälle gesucht, um die Fehlerdiagnose zu unterstützen, so ist in der Regel ein relevantes Modul Ausgangspunkt der Testfallauswahl. Dieses Element erhält einen virtuellen maximal schlechten Absicherungsstand. Entsprechend werden dann Testfälle gesucht, die genau dieses Modul, sowie die von ihm abhängigen testet. Als weiteres Kriterium kommt hinzu, ob der Testfall in der Lage ist, ein gesuchtes Symptom hervorzurufen. Ist dieser Zusammenhang bekannt, so kann ein Testfall gesucht werden, der zusätzlich zu einem Modul auch ein bestimmtes Symptom untersucht.

## 7. Anwendung an einem End-of-Line Testsystem der Automobilindustrie

Das Assistenzsystem wurde als Prototyp für ein End-of-Line Testsystem aus der Automobilindustrie realisiert. Es besteht aus dem Assistenzsystem selbst und einem exemplarischen Testsystem, das aus einem Server und mehreren Prüfgeräten, sowie einer Ergebnisdatenbank besteht. Das Assistenzsystem besteht aus einem Front-End mit der Benutzeroberfläche und den beiden Dashboards für Fehlerdiagnose und Absicherungsunterstützung und einem Back-End, in dem die eigentliche Funktionalität realisiert ist sowie einer relationalen Datenbank, die das Datenmodell aus Abbildung 1 speichert. Der prototypische Versuchsaufbau ist in Abbildung 5 dargestellt. Abbildung 6 zeigt die Systemarchitektur des realisierten Assistenzsystems.

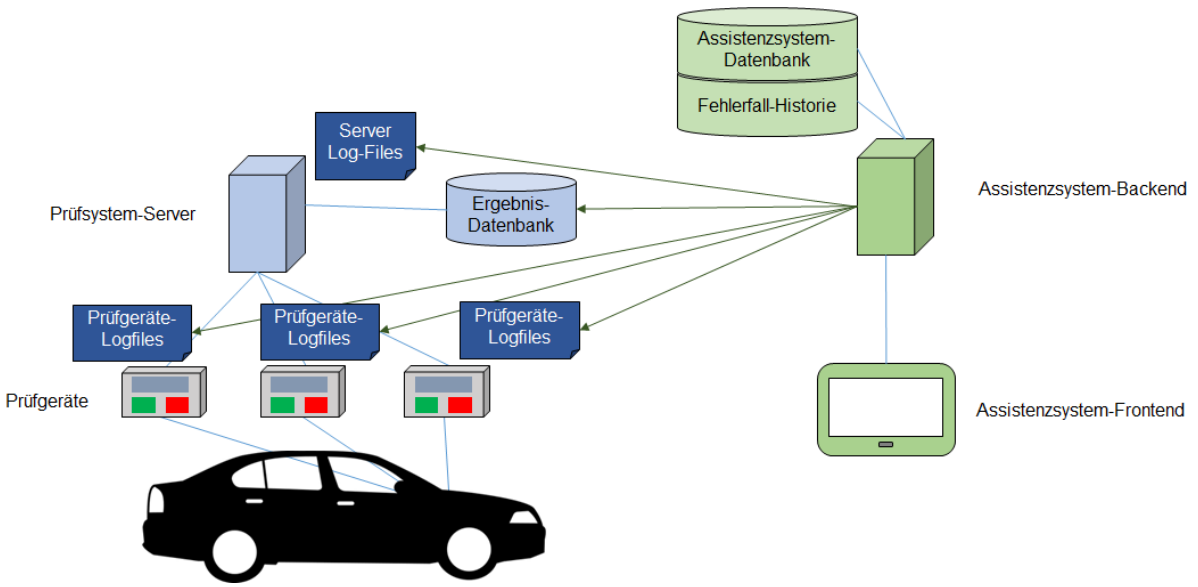


Abbildung 5: Versuchsaufbau mit einem End-of-Line Prüfsystem aus der Automobilindustrie

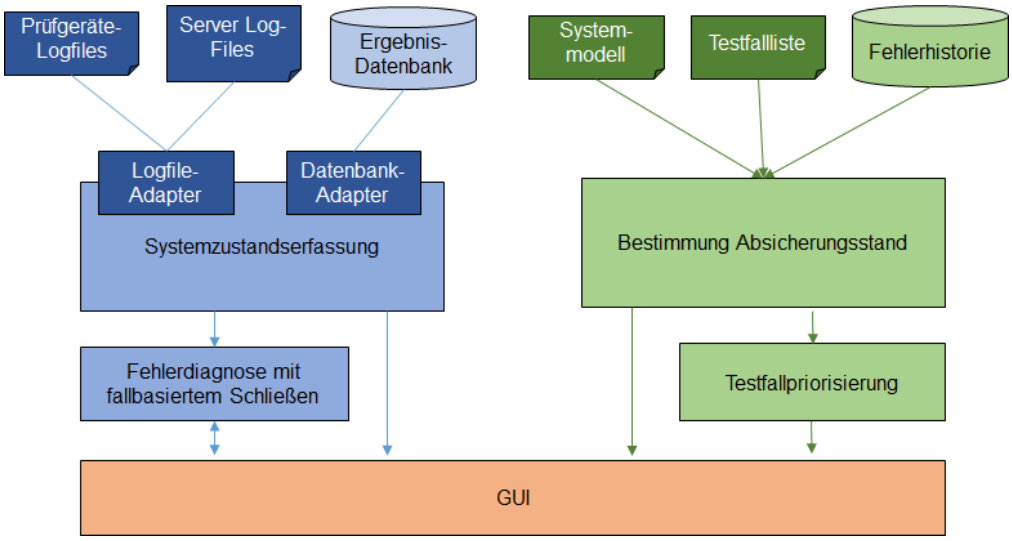


Abbildung 6: Systemarchitektur des prototypischen Assistenzsystems

Mithilfe der Testexperten wurden beide Szenarien, Fehlerdiagnose und Absicherungsunterstützung, exemplarisch erprobt. Dazu wurden beispielhafte Fälle untersucht, Fehlersymptome daraus abgeleitet und entsprechende Überwachungsregeln für die Logfiles und die Ergebnisdatenbank generiert. Mithilfe des Versuchsaufbaus konnten die Fehler dann rekonstruiert und mit dem Assistenzsystem erfolgreich diagnostiziert werden. Das Verhalten für neue, realistische Fehlerfälle und die Erweiterbarkeit durch das Hinzufügen neuer Symptome und Fehlerfälle muss in einer längeren Untersuchung noch gezeigt werden. Für die Absicherungsunterstützungsfunktion wurden Fehler im System gestreut und Änderungen durch die Experten angegeben. Die Auswertung hat gezeigt, dass das System die Daten und hinterlegten Abhängigkeiten korrekt auswertet und den Absicherungszustand wie erwartet setzt. Eine Untersuchung anhand eines neuen System-Release steht auch hier noch aus.

## 8. Zusammenfassung und Ausblick

Um die Verfügbarkeit und Verlässlichkeit von industriellen Qualitätssicherungssystemen zu erhöhen sind Fehlerdiagnose und Absicherung von Systemänderungen wichtige Aufgaben. Die Komplexität dieser Aufgaben steigt mit zunehmendes Komplexität der Qualitätssicherungssysteme und deren weltweite Verteilung und Vernetzung stark an. Um die Experten bei diesen Aufgaben zu unterstützen, wurde ein Assistenzsystem vorgestellt. Ausgehend von einem gemeinsamen Datenmodell zur Auswertung und Integration verschiedener Datenquellen unterstützt das Assistenzsystem beide Aufgaben. Zunächst werden die Daten analysiert, ausgewertet und in einem Dashboard für die jeweilige Funktion

dargestellt. Ausgehend von den Dashboards wird die Fehlerursache gesucht bzw. Testfälle ausgewählt und priorisiert.

Die weitere Arbeit wird sich damit beschaffen, die beiden Assistenzfunktionen noch weiter zu integrieren und zu kombinieren. Dadurch lassen sich weitere Synergien nutzen. Anschließend gilt es, das System anhand eines realen Beispiels über einen längeren Zeitraum zu evaluieren.

**9. Literaturangaben**


[1] Gao, Z., Cecati, C. u. Ding, S. X.: 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 62 (2015) 6, S. 3757–3767

[2] Gao, Z., Cecati, C. u. Ding, S.: 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 62 (2015) 6, S. 3768–3774

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

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

[5] Liu, H., Gegov, A. u. Cocea, M.: Rule Based Systems for Big Data. A Machine Learning Approach. Studies in big data, Bd. 13. Heidelberg: Springer 2016

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

[7] Yoo, S. u. Harman, M.: Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability 22 (2012) 2, S. 67–120

[8] Malz, C.: Agentenbasierte dynamische Testfallpriorisierung. Dissertation, Institut für Automatisierungs- und Softwaretechnik, Universität Stuttgart. IAS-Forschungsberichte, Herzogenrath: Shaker 2013

[9] Srikanth, H., Williams, L. u. Osborne, J.: System test case prioritization of new and regression test cases. International Symposium on Empirical Software Engineering. 2005, S. 62–71

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