

2. Modellbasierte Werkzeuge für Diagnose und Fehleranalyse von Fahrzeugsystemen

*Peter Struss, Ulrich Heller, Andreas Malik und Martin Sachenbacher –
Technische Universität München*

2.1 Einleitung

2.1.1 Die Zielsetzung

Modellbasierte Systeme versprechen automatische oder unterstützende Computer-Problemlösungen in Gebieten, in denen diese Lösungen wesentlich auf prinzipielles Wissen über das Verhalten und möglicherweise das Fehlverhalten komplexer technischer oder natürlicher Systeme zurückgreifen müssen. Diagnose und Fehleranalyse treten als Aufgabenstellungen in unterschiedlichen Ausprägungen während des gesamten Lebenszyklus von Produkten, in unserem Anwendungsbereich Kraftfahrzeug-Subsysteme, auf: Vom Entwurf eines Systems, das auch im Fehlerfall Risiken für Fahrzeuginsassen und Umgebung gering hält, über die Fehlererkennung in On-Board-Systemen und die Bereitstellung von Diagnose- und Reparaturanleitungen für Werkstätten bis zu Prüf- und Diagnoseabläufen auf Werkstatt-Testern. Beispiel für solche Fahrzeug-Subsysteme, die jeweils von einem elektronischen Steuergerät („Electronic Control Unit“, ECU) überwacht und geregelt werden, sind das Anti-Blockier-System (ABS), die Motorregelung oder das Airbag-System. Im INDIA-Projekt wurde in Kooperation zwischen der Robert Bosch GmbH und der Gruppe „Modelbased Systems and Qualitative Reasoning“ (MQM) der Technischen Universität München an Computerunterstützung für drei verschiedene Aufgaben gearbeitet, deren praktischer Hintergrund im folgenden nur kurz charakterisiert werden soll (eine genauere Analyse der Anforderungen ist in den darauffolgenden Kapiteln enthalten):

- *Fehler-Möglichkeits- und Einflußanalyse (FMEA)*

FMEA (engl. „Failure Modes and Effects Analysis“) wird gemeinsam von Entwicklungsingenieuren und Diagnose-Spezialisten in mehreren Sitzungen während der Entwurfsphase eines Subsystems durchgeführt. Ihr Ziel ist, basierend auf dem jeweiligen Stand des Entwurfs, die möglichen Auswirkungen von Komponentenausfällen und -fehlverhalten auf das Verhalten des Fahrzeugs zu analysieren. Schwerpunktmäßig geht es um die Einschätzung der „Kritikalität“, d.h. die Frage, wie schwerwiegend die resultierenden Funktionsstörungen gemäß subjektiver oder objektiver Kriterien sind (z.B. Beeinträchtigung des Fahrkomforts, negative Umwelteinflüsse, Unfallrisiko). Außerdem werden die Auftretenswahrscheinlichkeit der Fehler sowie ihre Entdeckbarkeit abgeschätzt. Auf dieser Grundlage können ggf. Änderungen am Entwurf vorgeschlagen werden. FMEA wird in zunehmendem Maße in verschiedenen Produktionszweigen zu einer gesetzlichen oder auch Kundenanforderung. Da im Fahrzeugbereich Sicherheit und Umweltaspekte betroffen sind, muss die Analyse so vollständig wie möglich sein, d.h. nicht nur sämtliche denkbaren Komponentenfehler umfassen, sondern auch alle ihre möglichen Auswirkungen unter verschiedenen Randbedingungen (also z.B. Fahrsituationen und Wechselwirkungen mit anderen Subsystemen).

- *Werkstattdiagnose*

Ausgangspunkt für die Fehlersuche und -behebung in der Werkstatt ist eine gewisse Menge von Symptomen eines Fehlverhaltens, die entweder als Fahrerbeanstandungen oder als Fehlercodes in den Steuergeräten der verschiedenen Subsysteme eines Fahrzeugs abgelegt sind. Abgesehen von offenkundigen Fällen gilt es, durch eine Folge geeigneter Tests und Messungen die Fehlerursache allmählich so weit einzugrenzen, dass er, gewöhnlich durch Ersetzen einer defekten Komponente, behoben werden kann. Die derzeitigen Fehlercodes stellen in der Regel keine Diagnose im Sinne einer Fehlerlokalisierung dar, sondern ein Symptom, eine von der ECU gemessene Abnormalität (etwa das Ausbleiben eines Signals oder die Überschreitung eines plausiblen Bereichs). Zusätzliche Informationen werden in der Werkstatt entweder aus manuell vorgenommenen Messungen und Beobachtungen (etwa Spannungs- und Widerstandsmessungen in der Elektrik) oder automatischen Messungen eines an die ECU angeschlossenen Werkstatt-Testers gewonnen. Solche Tests erfordern oft vor- und nachbereitende Tätigkeiten mit unterschiedlichem Aufwand (z.B. Demontage von Verkleidungen, Testfahrten, spezielle Ausrüstungen und Geräte), was die Auswahl und Reihenfolge der durchzuführenden Schritte bestimmt.

- *Erzeugung von Fehlersuchanleitungen*

Der Mechaniker in der Reparaturwerkstatt wird u.a. durch Diagnose-Manuale ausgebildet und angeleitet, die von einer zentralen Abteilung des Kundendienstes produziert und (auf Papier, CD-ROM oder künftig über das Internet) verbreitet werden. Bei der Erstellung solcher Manuale („Fehlersuchanleitungen“) führen Ingenieure Informationen verschiedener Art (Tabellen, Abbildungen, textuelle Anleitungen für Prüfabläufe) zusammen, die notwendig oder nützlich für die Durchführung der Diagnose zumindest der herkömmlichen Fehler eines Subsystems sind. Solche Dokumente müssen für jede Variante (oder manchmal Gruppe von Varianten) der verschiedenen Subsysteme produziert werden in Abhängigkeit vom jeweiligen Modell, Baujahr, speziellen Ausfertigungen etc. Vor allem für Zulieferer wie die Robert Bosch GmbH bedeutet dies einen erheblichen Aufwand für die Erstellung und Anpassung einer großen Zahl von spezifischen Fehlersuchanleitungen, wozu noch deren Übersetzung in bis zu 20 verschiedene Sprachen tritt. Der Kern des Dokuments besteht aus einer Menge von Prüfplänen, die die Fehlereingrenzung und -behebung in einzelnen Funktionsgruppen des Subsystems ausgehend entweder von Fahrerbeanstandungen oder abgelegten Fehlercodes beschreiben. Auch diese Prüfpläne müssen natürlich die praktischen Bedingungen und Kosten der Durchführung der notwendigen Tätigkeiten in der Werkstatt reflektieren, was zum Teil durch Überprüfung an Fahrzeugen parallel zur redaktionellen Arbeit geschieht.

Normalerweise stellt keine dieser Aufgaben die entsprechenden Experten vor unlösbare Probleme und erfordert keineswegs Wissen und Kenntnisse extrem spezieller Natur. Dennoch ist ihre Erfüllung oft sehr zeitaufwendig. Dies ist zum einen der erwähnten Vollständigkeit zuzuschreiben, mit der etwa Komponentenfehler und Betriebsbedingungen in der FMEA und in den Prüfplänen abgedeckt werden müssen. Der andere Faktor für den Zeitaufwand ist durch die Vielfalt von Subsystem-Varianten gegeben, die zwar nicht immer völlig neue Analysen und Dokumente verlangen, aber dennoch in aller Regel eine Überprüfung und Anpassung. Diese Situation einer Anwendung eher routinemäßiger, wissensgestützter Tätigkeiten auf eine große Menge von Varianten begründet das starke ökonomische Interesse an Computerunterstützung. Sie ist auch die Basis für eine erfolgreiche Realisierung durch modellbasierte Systeme, weil diese die Möglichkeit bieten, das erforderliche technische Wissen im Rechner in systematischer Form zu repräsentieren und algorithmisch zu nutzen.

2.1.2 Eine Analyse der Aufgabenstellungen

Im folgenden wird versucht, den Kern der jeweiligen Problemlösung und ihre notwendigen Wissensquellen so generell zu charakterisieren, dass sie auf wissensbasierte Algorithmen und Einheiten einer Wissensbasis abgebildet werden können.

- *FMEA-Unterstützung*

Das grundlegende Wissen, das für diese Aufgabe repräsentiert werden muss, betrifft die Wirkungsweise der in einem Subsystem verwendeten Komponenten, d.h. Verhaltensmodelle von Komponenten, die sowohl ihr nominales Verhalten als auch mögliche Fehlverhalten beschreiben. Das zweite offenkundige Element ist eine Darstellung der Struktur des Subsystems (eine „Blaupause“), die die Interaktion der beteiligten Komponenten beschreibt. Der Kern der Aufgabe besteht darin, auf der Grundlage dieser beiden Eingaben die möglichen Effekte von Fehlern der einzelnen Komponenten zu bestimmen, was als modellbasierte Verhaltensvorhersage bezeichnet wird. Die sich anschließende Bewertung stützt sich auf die Wahrscheinlichkeit der angenommenen Fehler und die „Kritikalität“ der Auswirkung. Letzteres erfordert mehr als eine bloße Beschreibung der physikalischen Effekte, nämlich Wissen darüber, ob und in welchem Ausmaß diese Effekte die intendierte Funktionalität des jeweiligen Subsystems innerhalb des Fahrzeugs beeinträchtigen (z.B. ob lediglich Geräusche entstehen, die Motorleistung reduziert ist oder die Gefahr von Feuer gegeben ist).

- *Werkstattdiagnose*

Ziel dieser Aufgabe ist offenkundig, Komponenten zu identifizieren, deren Fehlverhalten die beobachteten störenden Effekte, d.h. Symptome verursacht haben könnten. Wiederum ist also Wissen über Verhalten und evtl. Fehlverhalten der vorhandenen Komponenten grundlegend, ebenso wie Kenntnis der besonderen Struktur des Subsystems und die auf beides gegründete Verhaltensvorhersage, die es gestattet, vom Modell impliziertes Verhalten mit dem tatsächlich beobachteten Verhalten in Beziehung zu setzen. Normalerweise kann die Fehlerursache nicht in einem Schritt ermittelt werden, so dass die Erzeugung geeigneter Tests für die Gewinnung zusätzlicher Information eine weitere Teilaufgabe ist. Deren Natur besteht darin, Aktionen vorzuschlagen, die mit Sicherheit oder zumindest gewisser Wahrscheinlichkeit Unterschiede in den Auswirkungen verschiedener (Fehl-)Verhalten beobachtbar machen. Also erfordert auch dieser Schritt eine Form der Verhaltensvorhersage und darüber hinaus Wissen über die zur Realisierung notwendigen Aktionen und den damit verbundenen Zeit- und Kostenaufwand. Soweit sich die Werkstattdiagnose auf Messwerte stützt, die mit Hilfe eines Werkstatt-Testers aus der ECU ausgelesen werden, stellt sich ferner die Aufgabe,

die Rohsignale im Hinblick auf die Diagnoseaufgabe geeignet zu interpretieren (z.B. die Abweichung gemessener Werte von Sollwerten zu bestimmen).

- *Erzeugung von Fehlersuchanleitungen*

Im Gegensatz zu einem (interaktiven) Werkstattdiagnosesystem, das dynamisch auf sich ändernde Situationen und darin verfügbare Information reagiert, muss eine Fehlersuchanleitung alle möglichen Diagnosesituationen abdecken, die sich während der Fehlersuche aufgrund eines Symptoms ergeben können. Demnach besteht hier das Wesen der Aufgabe in der Erzeugung eines gesamten Testplans, sinnvoll abstrahierbar zu einem mehr oder weniger komplexen Entscheidungsbaum. Auch hier wird also Verhaltensvorhersage basierend auf Wissen über Komponentenverhalten und Systemstruktur verlangt sowie Information über *Aktionen* und deren *Kosten*.

2.1.3 Architektur der Prototypen

Wenn man aus diesen Skizzen der technischen Lösungen die verschiedenen Elemente des zu repräsentierenden Wissens und der Algorithmen extrahiert, ergibt sich folgendes Bild:

- Grundlegend für alle drei Lösungen sind eine Bibliothek von **Verhaltensmodellen** der verwendeten **Komponententypen** und eine **Strukturbeschreibung** („Netzliste“) des betrachteten technischen Systems. Aus diesen beiden Elementen wird dessen Verhaltensmodell konstruiert (Systemmodell, „Device model“), was durch eine Softwarekomponente, die *Modellgenerierung* („Model composition“) automatisch geschieht (vgl. Abbildung 2-1).
- Das Verhaltensmodell des Systems stellt die Basis für die *Verhaltensvorhersage* dar, die ebenfalls zum gemeinsamen Kern der drei Werkzeuge gehört (Abbildung 2-1), wenn auch möglicherweise in verschiedenen technischen Realisierungen. Bei der FMEA und der Diagnose kommt es darauf an, aus verschiedenen angenommenen Betriebszuständen bzw. vorgenommenen Messungen und Beobachtungen möglichst viele Informationen über den zugehörigen Zustand des Systems aus dem Modell abzuleiten. Für die Testgenerierung hingegen ist wesentlich, dass Verhaltensvorhersage die Systemzustände im fehlerfreien Zustand bzw. in Fehlerfällen *für alle möglichen Eingaben und Einflüsse* in einem globalen Bild darstellt, um daraus geeignete Unterscheidungen zu ermitteln.
- Während bei der FMEA die Verhaltensvorhersage bereits den Kern der Aufgabe löst, nämlich Bestimmung der möglichen Auswirkungen von Komponentenfehlern auf bestimmte, funktional relevante Größen, sind für die anderen Aufgaben weitere Algorithmen notwendig. Ein *Diagnosegenerator* ermittelt aus den Ergebnissen der Verhaltensvorhersage (genauer: aus deren Widersprüchen zu den tatsächlichen Beobachtungen) Hypothesen über defekte Komponenten. Der *Testgenerator* hingegen macht umgekehrt zu unterscheidende Systemverhalten (Fehlerfälle) zum Ausgangspunkt und ermittelt daraus Kandidaten für diskriminierende Tests, d.h. vorzunehmende Zustandsänderungen des Systems, die für bestimmte Systemgrößen zu unterscheidbaren Beobachtungen führen. Auf diese Weise kann entweder ein gesamter Entscheidungsbaum (für die Prüfplangenerierung) oder ein Vorschlag für einen einzelnen Test (als nächstem Schritt im Rahmen der Werkstattdiagnose) erzeugt werden.
- Soweit im Werkstattdiagnose-System Messwerte aus einem Tester verarbeitet werden sollen, so sind die gewonnenen Rohsignale durch eine *Signalabstraktion* so zu transformieren, daß sie von dem Modell verarbeitet werden können.
- Die Generierung geeigneter Tests muss offenkundig auf eine Darstellung der praktischen Aspekte ihrer Durchführung gestützt sein, nämlich geeignete *Tätigkeiten* (Eingaben, Beobachtungen), ihrer Voraussetzungen (z.B. Geräte, Demontage) und des erforderlichen Aufwands, in Abbildung 2-1 in „Kosten“ zusammengefaßt.
- Schließlich sei noch erwähnt, dass nur für die FMEA und die Testgenerierung explizite Modelle der relevanten Komponentenfehler unabdingbar sind, während Diagnoseverfahren existieren, die Fehlerdetektion und -lokalisierung nur auf der Basis von Modellen korrekten Komponentenverhaltens vornehmen ([Dressler Struss 1996], [Hamscher Console de Kleer 1992]).

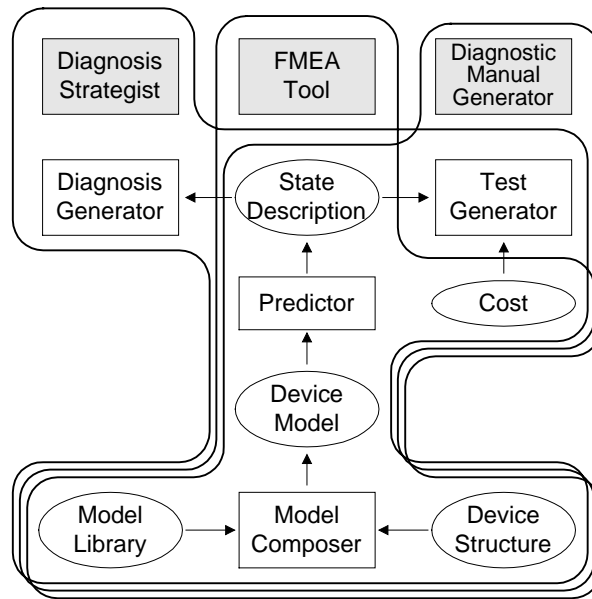


Abbildung 2-1: Zusammenhang zwischen Softwarekomponenten und Elementen der Wissensbasis

Abbildung 2-1 zeigt eine Übersicht über die erwähnten Softwarekomponenten und Elemente der Wissensbasis sowie ihren Zusammenhang untereinander und mit den verschiedenen aufgabenbezogenen Werkzeugen. Dabei wird deutlich, dass bei der vorgenommenen Strukturierung ein hoher Grad an Wiederverwendbarkeit sowohl des repräsentierten Wissens als auch der Softwarekomponenten quer zu den Einzelaufgaben erzielt werden kann. Dies betrifft zum einen natürlich die *Modellbibliothek* und die *Strukturbeschreibung*, was eine grundlegende Zielsetzung modellbasierter Systeme ist. Gemeinsam sind auch die *Modellkomposition* und *Verhaltensvorhersage*. Hier wurden die entsprechenden Komponenten wie auch der Diagnosealgorithmus aus dem Diagnose- und Modellierungssystem RAZ'R von OCC'M Software genutzt. Dies gestattet ferner eine Abtrennung der Erstellung des Systemmodells von den aufgabenspezifischen modellbasierten Problemlösungen: Erfassung und Erprobung der Komponententypmodelle und Strukturbeschreibung finden im Entwicklungssystem statt, aus dem letztlich nur das Verhaltensmodell eines Systems exportiert wird, das dann als Input für das RAZ'R-Diagnoselaufzeitsystem, den FMEA-Generator und die Testplan-Generierung dient.

In den folgenden Abschnitten werden die im Projekt realisierten Anwendungs-Prototypen näher beschrieben. Dabei sollen nur die Grundideen der verwendeten Algorithmen und Modellierung vermittelt werden. Für Details und wissenschaftliche Grundlagen wird jeweils auf die zugehörigen Fachbeiträge verwiesen.

2.2 Modellbasierte Verhaltensanalyse für die FMEA

2.2.1 Ziele und Anforderungen

Zu den meisten Kfz-Subsystemen verlangt der Automobilhersteller heute eine ausführliche Fehlermöglichkeits- und -Einfluß-Analyse (FMEA) als Nachweis der präventiven Qualitätssicherung.

In der FMEA werden alle denkbaren Fehlerarten und -ursachen während der Entwurfs-, Konstruktions- und Fertigungsphase in Betracht gezogen und es wird versucht, ihnen Fehlerauswirkungen auf Ebene der Komponenten, Subsysteme und des Fahrzeugs zuzuordnen.

Die dazu notwendigen Informationen und Schlussfolgerungen werden auf FMEA-Teamsitzungen nach methodischen Ansätzen (Tabellenformular) erarbeitet, gesammelt und dokumentiert. Abbildung 2-2 zeigt die Grundstruktur einer so entstehenden Tabelle.

Da sich ein FMEA-Team aus Fachleuten von der Entwicklung bis hin zur Fertigungsvorbereitung zusammensetzt, ist das Verfahren sehr personalintensiv, weshalb meist noch eine Phase der Wissenserhebung durch einen Moderator vorgeschaltet wird. Durch diese Arbeit wird insgesamt jedoch sehr viel wertvolle Ingenieur-Kapazität gebunden.

Der Anwender Bosch verfolgt daher das Ziel, sowohl zur Vorbereitung durch den FMEA-Moderator als auch zur Verwendung während der FMEA-Sitzungen rechnerbasierte Techniken einzusetzen.

Dazu wird ein Werkzeug benötigt, das die FMEA in der bisher gewohnten Weise (Tabellenformular) als strukturierender Editor unterstützt und das die zentrale Aufgabe der FMEA, eine Verhaltensanalyse durch Verfolgen von Fehlerarten und ihren Auswirkungen im System durchzuführen, automatisch vornimmt.

Component Process	Function Purpose	Failure Mode	Failure Effect	Failure Cause	...
...

Abbildung 2-2: Aufbau eines FMEA-Dokuments

2.2.2 Modellbasierte Vorhersage von Fehlerursache-Fehlerauswirkungs-Beziehungen

Der Rahmen für die Realisierung eines modellbasierten FMEA-Werkzeugs ist durch die in Abbildung 2-1 abgebildeten Komponenten gegeben. Die daraus für die FMEA benötigte Kernfunktionalität wird durch das Verhaltensvorhersagemodul („Predictor“) abgedeckt. Der Predictor liefert für eine Menge von Beobachtungen (sog. „Szenario“) und einen gegebenen Kontext (d.h. Normalverhalten oder Fehlverhalten von Komponenten) mit Hilfe des Modells Vorhersagen über die Werte von Modellvariablen, die unter diesem Kontext gelten.

Unter der Voraussetzung, dass Verhaltensmodelle für die relevanten Fehlerfälle von Komponenten vorhanden sind, ermöglicht dies die automatische Bestimmung von Zusammenhängen zwischen den auftretenden Fehlerursachen (d.h. Fehlermodus der entsprechenden Komponente) und deren Auswirkungen auf das System (d.h. abgeleitete Werte für weitere Systemvariablen).

Wie bei der manuell durchgeführten FMEA beschränkt sich der Prototyp dabei auf die Betrachtung von Einfachfehlern. Die auftretenden Kontexte für den Predictor können also in Form einer Schleife erzeugt werden, die alle Einfachfehlerannahmen durchläuft und diese mit den Korrektheitsannahmen für alle übrigen Komponenten kombiniert.

2.2.3 Architektur des FMEA-Prototyps

Die Basis für die automatisierte Vorhersage von Fehlerauswirkungen aus Fehlerursachen nach dem oben dargestellten Ansatz ist:

- eine Bibliothek von Verhaltensmodellen für Komponenten, die das korrekte Verhalten und das Verhalten unter relevanten Fehlern abdecken (Model Library),
- eine Beschreibung der Systemstruktur, d.h. der auftretenden Komponententypen und ihrer Verbindungen (System Model),
- eine Beschreibung relevanter Situationen (meist Worst-Case-Szenarien) in Form von Sequenzen von Zuweisungen an Variablen, die bestimmte Betriebszustände des Systems charakterisieren, z.B. die Druckaufbauphase in einem ABS (Situation Description).

Abbildung 2-3 zeigt die Architektur des FMEA-Prototyps. Es wird vorausgesetzt, daß alle aufgeführten Eingaben in Form von XML-Dateien vorliegen. Der Prototyp durchläuft anhand dieser Informationen systematisch die einzelnen Fehlermodi und Situationen und bereitet die Ergebnisse der Verhaltensvorhersage in Form einer FMEA-Tabelle auf.

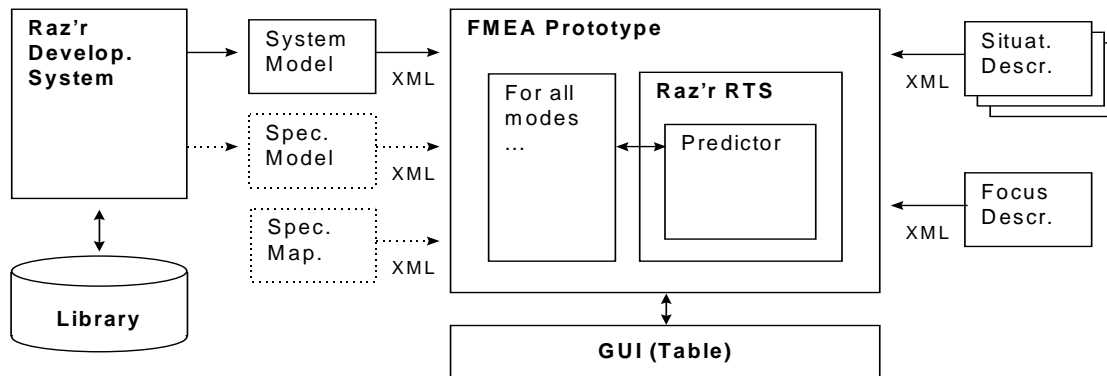


Abbildung 2-3: Architektur des FMEA-Prototyps

Bei der Generierung der FMEA-Tabelle kann nicht davon ausgegangen werden, dass jeder für eine Variable vorhergesagte Wert eine Fehlerauswirkung darstellt und als Eintrag in die FMEA-Tabelle übernommen werden kann. Dies liegt daran, dass in der FMEA nicht das Normalverhalten, sondern nur der Unterschied von Normal- zu Fehlverhalten betrachtet wird. Außerdem geschieht dies meist noch bezogen auf eine bestimmte Auswahl von relevanten Systemgrößen, d.h. in der FMEA-Tabelle wird meist nicht die gesamte Wirkungskette eines Fehlers wiedergegeben, sondern man beschränkt sich auf "interessante" Auswirkungen.

In der Grundstufe des FMEA-Prototyps wird deshalb zunächst für jede Situation das Verhalten des Systems vorhergesagt unter der Annahme, dass alle Komponenten korrekt sind. Ein unter einer bestimmten Fehlerannahme vorhergesagter Wert wird dann als Fehlerauswirkung betrachtet, wenn er von dem entsprechenden Ergebnis für korrekt funktionierende Komponenten abweicht.

Die gewünschte Beschränkung auf eine Teilmenge von „interessanten“ Variablen wird dabei durch die Angabe einer zusätzlichen Fokusbeschreibung (Focus Description) ermöglicht. Diese kann z.B. Output-Variablen enthalten (etwa das Giermoment des Fahrzeugs bei der FMEA eines Antiblockiersystems), wodurch der gewünschte Effekt erzielt wird, „uninteressante“ interne Variablen (z.B. hydraulischer Druck an verschiedenen Stellen im Hydroaggregat) bei der FMEA eines Antiblockiersystems auszublenden.

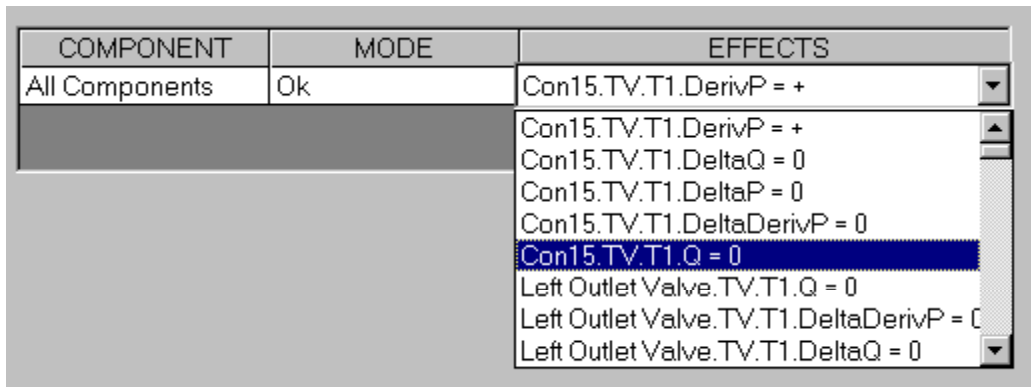
In diesem FMEA-Basissystem wird das Referenzverhalten des Systems also implizit durch sein Normalverhalten festgelegt. Eine Erweiterung dieser Basisfunktionalität ist in Abbildung 2-3 gestrichelt dargestellt. Bei dieser Erweiterung kann zusätzlich zum Systemmodell die explizite Repräsentation eines spezifizierten Systemverhaltens mit angegeben werden. Dieses „Specification Model“ ist im Grunde nichts weiter als ein zusätzliches Verhaltensmodell, welches nun aber dazu dient, das intendierte Verhalten des Systems zu beschreiben. Ein „Specification Mapping“ ist nötig, um die Abbildung der Variablen, die in der Spezifikation auftreten, auf die Variablen des betrachteten Systemmodells herzustellen. Dadurch wird es möglich, mehrere verschiedene Realisierungen oder Modifikationen eines Systems mit einer gemeinsamen, vorher festgelegten Spezifikation zu vergleichen.

Trotz der Beschränkung auf Einfachfehler kann die Menge der zu betrachtenden Kontexte (Fehlermoduszuweisungen) noch sehr groß werden. Die Laufzeit des Predictors würde mit jeder zusätzlichen Komponente im System multiplikativ anwachsen. Man stellt jedoch fest, dass nicht alle Vorhersagen für Modellvariablen in jedem Kontext unterschiedlich sind. Besonders wenn ein System sehr groß ist, wirkt sich ein Fehler oft nur in bestimmten Bereichen des Systems aus, während andere Bereiche des Systems von Fehlerauswirkungen unberührt bleiben. Dieser Effekt kann ausgenutzt werden, indem bereits abgeleitete Vorhersagen für weitere Kontexte wiederverwendet werden. In dem vorgestellten Prototyp wird dies durch die Verwendung eines sog. Assumptionbased Truth Maintenance Systems (ATMS) ([de Kleer 1986]) erreicht. Das ATMS ermöglicht es, dass für jeden neu zu betrachtenden Fehler nur noch die Differenzen zu bereits untersuchten Fehlern behandelt werden müssen. Bereits abgeleitete und in einem neuen Kontext noch gültige Ergebnisse werden dadurch nicht nochmals neu erzeugt. Anhängig von der Art der auftretenden Fehler und der Struktur des Systems kann dies einen erheblichen Effizienzgewinn bewirken.

2.2.4 Realisierung des FMEA-Prototyps

Im Projektzeitraum von INDIA wurde basierend auf den dargestellten Konzepten ein Demonstrator für die modellbasierte Unterstützung von FMEA spezifiziert und unter Verwendung vorhandener Softwarekomponenten des Systems RAZ'R implementiert.

Während des Projektzeitraums gab es noch keinen konkreten Anwendungspartner bei Bosch, der spezifische Anforderungen für eine Validierung bereitstellen und eine Evaluierung hätte vornehmen können. Insbesondere existierten damit für den FMEA-Prototyp noch keine Anforderungen an die Bedienoberfläche, so dass diese noch sehr rudimentär ist. Abbildung 2-4 zeigt ein Beispiel. Die zugrundeliegende Implementierung umfasst jedoch alle in Abbildung 2-3 dargestellten Schnittstellen, inklusive der im vorigen Abschnitt beschriebenen Erweiterung um die Spezifikation eines Referenzmodells.



COMPONENT	MODE	EFFECTS
All Components	Ok	Con15.TV.T1.DerivP = +
		Con15.TV.T1.DerivP = +
		Con15.TV.T1.DeltaQ = 0
		Con15.TV.T1.DeltaP = 0
		Con15.TV.T1.DeltaDerivP = 0
		Con15.TV.T1.Q = 0
		Left Outlet Valve.TV.T1.Q = 0
		Left Outlet Valve.TV.T1.DeltaDerivP = 0
		Left Outlet Valve.TV.T1.DeltaQ = 0

Abbildung 2-4: Beispiel für die Ausgabe des FMEA-Prototyps

2.2.5 Verwandte Ansätze

Die FMEA ist eine in der Industrie weit verbreitete Methode. Daher existieren verschiedenste Ansätze, den Prozess der FMEA zu unterstützen. Die meisten dieser Systeme bieten jedoch nur graphische Werkzeuge und Textverarbeitungs-Funktionen zur Verwaltung der Einträge in den FMEA-Tabellen an. Sie unterstützen dagegen nicht die hier zum Gegenstand gemachte zentrale Aufgabe der FMEA, die Verhaltensanalyse von Fehlern. Eine Ausnahme bildet das kommerzielle System AutoSteve ([Price 1998]), das z.B. bei den Automobilherstellern Jaguar und Ford eingesetzt wird. AutoSteve automatisiert einen Großteil des FMEA-Prozesses für eine spezielle Anwendungsdomäne, nämlich elektrische Schaltungen. Dazu wird das Verhalten des elektrischen Schaltkreises im Normalfall und im Fehlerfall simuliert. Aus dem Ergebnis werden mittels einer Auswahl vorgegebener Textbausteine weitgehend natürlichsprachliche FMEA-Einträge erzeugt.

Die wesentlichen Unterschiede zwischen dem hier beschriebenen modellbasierten FMEA-Prototyp und AutoSteve sind:

- AutoSteve testet nicht systematisch alle möglichen Einfachfehler, sondern nur eine vom Benutzer spezifizierte Auswahl. Der Grund liegt in einem Effizienzproblem des eingesetzten Simulationsverfahrens, das für jeden Fehlerfall neu angestoßen werden muss. Der hier vorgestellte Prototyp basiert dagegen auf einem ATMS, welches wie ein Cache für bereits abgeleitete Ergebnisse funktioniert. Für weitere Fehler müssen deshalb nur noch die Differenzen zu den bereits untersuchten Fehlern betrachtet werden, wodurch es möglich wird, systematisch alle Einfachfehler zu analysieren.
- AutoSteve verwendet einen speziellen Simulationsalgorithmus (QCAT), der auf Modelle elektrischer Netze spezialisiert ist. Der hier vorgestellte Prototyp basiert dagegen auf einem generischen Predictor-Modul, welches nicht auf eine spezifische Art von Modellen festgelegt ist, und kann daher beliebige Verhaltensmodelle verarbeiten.
- AutoSteve ist beschränkt auf die implizite Definition von Fehlverhalten als Differenz zum Verhalten des Systems, wenn alle seine Komponenten korrekt arbeiten. Der vorgestellte FMEA-Prototyp erlaubt dagegen den Vergleich zwischen einem Systemmodell und einer beliebigen, explizit vorgegebenen Spezifikation seines intendierten Verhaltens.

2.3 Modellbasierte Unterstützung der Online- und Werkstattdiagnose

2.3.1 Ziele und Anforderungen

Die durchgeführten Arbeiten zur FMEA und zur Generierung von Fehlersuchanleitungen lassen die Verwendung der Modelle und Diagnose-Algorithmen auch für Aufgaben in der Werkstattdiagnose bzw. der Onboard-Diagnose im Fahrzeug vielversprechend erscheinen. Einerseits hat diese Problemstellung einen Bezug zu den bisherigen Arbeiten, da auf Seite des physikalischen Systems die gleichen Komponenten wie in den bisher behandelten Anwendungen auftreten und damit die vorhandenen Modellbibliotheken benutzt werden können. Andererseits müssen für die Übertragung der modellbasierten Ansätze zusätzliche Probleme in Angriff genommen werden, welche in den Systemen zur Unterstützung der Verhaltensanalyse bei der FMEA bzw. Generierung von Fehlersuchanleitungen für die Werkstätten nicht auftreten.

Die entwickelten FMEA- und Testgenerierungssysteme arbeiten interaktiv. Insbesondere sind bei diesen Systemen die Beobachtungen des Systemverhaltens jeweils vom Benutzer zu spezifizieren. In der Werkstattdiagnose entspricht dies den Kundenbeanstandungen, die meist in vagen qualitativen Beschreibungen wie z.B. „Bremspedal zu weich“ vorliegen. Der in diesem Band wiedergegebene Beitrag [Struss Sachenbacher Dummert 1997] beschäftigt sich mit der Frage, wie solche Beschreibungen im Rahmen eines modellbasierten Diagnoseansatzes nutzbar gemacht werden können.

Daneben stützt sich die Werkstattdiagnose aber auch auf die mit Hilfe eines Werkstatt-Testers gewonnenen Messwerte. Onboard-Diagnose ist sogar nur auf die laufenden Signale des Steuergeräts angewiesen. Soweit die Diagnose solche elektronisch erfaßten Messwerte mit einbezieht, stellt sich deshalb die Aufgabe, wie zeitlich indizierte Sensorsignale in einem modellbasierten Diagnosesystem geeignet verarbeitet werden können. Die Realisierung eines entsprechenden Prototypen wird in diesem Abschnitt beschrieben. Da die Komponentenbibliothek und auch die Basis-Software-Komponenten hierfür vorhanden sind, beschränken sich die Arbeiten für einen solchen Prototypen im wesentlichen auf zwei Komponenten: Die Erfassung und Aufbereitung von Signalen des Fahrzeug-Subsystems zur Verarbeitung in einem modellbasierten Diagnose-System und die Transformation des Systemmodells gemäß den Anforderungen der Online-Diagnose. Die On-Board-Diagnose wirft darüber hinaus das Problem der Diagnose im Umfeld limitierter Ressourcen (Verarbeitung der Daten möglichst in Echtzeit, begrenzter Speicherplatz für das Diagnosesystem) auf.

Ein im Rahmen des Projekts betrachtetes Leitbeispiel für die Erprobung modellbasierter Online-Diagnose im Bereich mechatronischer Kfz-Subsysteme waren abgasrelevante Fehler in der Dieselregelung, die von der bisherigen On-Board-Diagnose nicht abgedeckt werden können. Für diese Szenarien standen beim Anwendungspartner Bosch entsprechende Messdaten zur Verfügung.

2.3.2 Verfahren zur modellbasierten Online-Diagnose

Die dargestellten Anforderungen führten auf Forschungsseite zur Entwicklung eines Diagnoseverfahrens, das sich durch besondere Effizienz im Bereich der Diagnose dynamischer Systeme auszeichnet. Im folgenden Abschnitt werden die Grundlagen dieses Ansatzes skizziert. Eine ausführliche Darstellung findet sich in [Struss 1997].

2.3.3 Zustandsbasierte Diagnose dynamischer Systeme

Die Aufgabenstellung der konsistenzbasierten Diagnose ist es, zu untersuchen, ob das vom Modell vorhergesagte Verhalten mit den Beobachtungen konsistent ist. Wenn $Model(Mode)$ das Modell eines Verhaltensmodus und Obs eine Menge von Beobachtungen beschreibt, bedeutet dies, die Konsistenz von

$$Model(Mode) \cup Obs$$

zu prüfen. Wenn dieser Ansatz für die Diagnose eines dynamischen Systems angewendet wird, ergibt sich die Frage, ob dies eine Vorhersage des Verhaltens über die Zeit, d.h. Simulation, erfordert. Häufig wird dies bei der Diagnose dynamischer Systeme sogar von vorneherein als gegeben angenommen. Es zeigt sich aber, dass Diagnoseergebnisse auch auf der Grundlage eines Konsistenzchecks von Modell und beobachteten Zuständen gewonnen werden können, d.h. ohne Simulation von Verhalten über die Zeit. Unter bestimmten Voraussetzungen hat dies sogar überhaupt keinen Verlust an Diagnoseinformation zur Folge ([Struss 1997]).

Die Idee dieser sogenannten zustandsbasierten Diagnose ist es, das Modell eines dynamischen Systems in zwei Klassen von Constraints zu zerlegen. Die eine Klasse von Constraints beschränkt die möglichen Zustände des

Systems zu einem bestimmten Zeitpunkt (State-Constraints). Die andere Klasse von Constraints beschränkt die möglichen Übergänge von einem Zustand zu einem anderen beschränken (Temp-Constraints):

$$\text{Model}(\text{Mode}) = \text{State-Constraints}(\text{Mode}) \cup \text{Temp-Constraints}(\text{Mode}).$$

Für die zustandsbasierte Diagnose wird dann lediglich geprüft, ob

$$\text{State-Constraints}(\text{Mode}) \cup \text{Obs}$$

konsistent ist, d.h. es wird nur geprüft, ob die Menge der beobachteten Zustände eine Teilmenge der konsistenten Zustände des Systems ist. Da folglich die Reihenfolge der beobachteten Zustände nicht mehr in die Betrachtung eingeht, scheint dieser Ansatz auf den ersten Blick weitaus schwächer als ein simulationsbasierter Ansatz zu sein, der das beobachtete Verhalten mit dem simulierten Verhalten des Modells vergleicht. Wenn jedoch temp-constraints nur aus Constraints besteht, welche generell gültige mathematische Gesetze über Stetigkeit, Differenzierbarkeit etc. beinhalten, dann kann temp-constraints keine zusätzlichen Widersprüche im Modell liefern. Solche Constraints können deshalb von der Konsistenzprüfung ausgenommen werden, ohne das Diagnoseergebnis zu beeinflussen.

Durch zustandsbasierte Diagnose wird die Notwendigkeit vermieden, das Modell über verschiedene Zeitpunkte hinweg zu simulieren. Stattdessen kann die Diagnose auf die einfache Konsistenzprüfung von Zuständen abgebildet werden. Dies kann einen großen Effizienzgewinn bedeuten, insbesondere, wenn dadurch Fehler nicht simuliert werden müssen. Zustandsbasierte Diagnose stellt damit einen wichtigen Beitrag dar, um die geforderten Reaktionszeiten des Online-Diagnoseprototyps zu gewährleisten.

2.3.4 Automatische Modelltransformation für kompositionale Modelle

Ein anderer Schlüssel zur Erreichung der nötigen Effizienz, um modellbasierte Diagnose im Online-Betrieb einsetzen zu können, liegt auf Seiten des Modells.

Im Hinblick auf die zeit- und speicherkritische Anwendung für Online-Diagnose ist es notwendig, dass im Modell nur solche Unterscheidungen berücksichtigt werden, die zur Erreichung eines bestimmten Ziels (z.B. Diskriminierung zweier Diagnosekandidaten mit unterschiedlichen Recovery Actions) auch unbedingt nötig sind. Da das Diagnosesystem automatisch aus einer Bibliothek von Modellfragmenten konfiguriert wird, die unabhängig vom Verwendungskontext einsetzbar sein sollen, können diese Unterscheidungen nicht in den einzelnen Modellfragmenten angelegt sein. Stattdessen stellen sie eine Eigenschaft des Gesamtmodells und seines Verwendungszwecks dar.

Deshalb ist eine zwischengeschaltete Komponente zur automatischen Modelltransformation notwendig, welche die relevanten Unterscheidungen (d.h. qualitative Werte) für einen bestimmten Anwendungskontext generiert. Ziel dieses Arbeitspakets war daher die Entwicklung von Techniken zur automatischen Transformation eines kompositionalen Modells auf eine spezifische, durch den Anwendungskontext vorgegebene Granularität. Dies umfasst die Berechnung von qualitativen Werten im Modell abhängig von dem zu erreichenden Diagnoseziel und der Beobachtbarkeit, z.B. der vorgegebenen Messgenauigkeit oder Abtastfrequenz der Sensorsignale.

Die theoretischen Grundlagen für die aufgabenbezogene Modellabstraktion und eine Reihe von Algorithmen hierfür sind in [Sachenbacher Struss 2000] (in diesem Band) und [Struss Sachenbacher 1999] zusammengefasst.

2.3.5 Automatische Signalaufbereitung für die Online-Diagnose

Die Anbindung des modellbasierten Diagnosesystems an reelle Daten, wie sie z.B. auf dem Fahrzeug vom Steuergerät oder in der Werkstätte von Messgeräten geliefert werden, erfordert aber auch die Transformation der Daten selbst auf eine für das zugrundeliegende Modell geeignete Darstellung und Granularität. Das Ziel eines weiteren Arbeitspakets war daher die Entwicklung einer Komponente für die automatische Signaltransformation, d.h. die Umwandlung numerischer, zeitlich indizierter Sensorsignale auf die Repräsentationsebene der qualitativen Modelle. Dies umfasst u.a. die Abbildung quantitativer Werte auf qualitative Wertebereiche des Modells, die Berechnung abgeleiteter Werte aus den numerischen Daten (z.B. Abweichungen) und die Aggregation einzelner quantitativer Zustände zu qualitativen Beobachtungsvektoren als Input für das Diagnose-Runtimesystem.

2.3.6 Architektur des Online-Diagnoseprototyps

Die Software für den Online-Diagnoseprototypen besteht damit aus den folgenden Komponenten:

- ein Modul zur Transformation der Wertebereiche von Modellvariablen auf die erforderliche (qualitative) Granularität,

- ein Modul zur Umwandlung der Rohsignale in qualitative Beobachtungen,
- ein modellbasiertes Runtime-Diagnosesystem, das basierend auf diesen Beobachtungen und dem Modell eine zustandsbasierte Diagnose durchführt.

Der erste Punkt besteht aus einer prototypischen Implementierung der Verfahren aus [Sachenbacher Struss 2000], welche auf dem Modellformat des kommerziellen Diagnoseframeworks RAZ'R aufsetzt.

Der zweite Punkt umfasst eine Komponente zur Umwandlung quantitativer Signale in qualitative Werte und Abweichungen. Von diesem Modul wird jedes Mal, wenn eine Veränderung der beobachteten Variablen oder ihrer Abweichungen auf der qualitativen Ebene eintritt, ein neuer Vektor von Beobachtungen erzeugt und an die Diagnosemaschine übergeben.

Für den dritten Punkt wurden Komponenten des kommerziellen Diagnoseframeworks RAZ'R verwendet, das ein Entwicklungssystem für Diagnosemodelle und ein Runtime-System einer konsistenzbasierten Diagnosemaschine enthält. Die Diagnosemaschine führt zustandsbasierte Diagnose mit dem Modell und den qualitativen Beobachtungsvektoren durch. Das Diagnoseergebnis erhält man schließlich aus der Kombination der Diagnoseergebnisse für die einzelnen Beobachtungsvektoren.

2.3.7 Evaluierung des Prototyps an Daten eines Versuchsfahrzeugs

Während der Laufzeit von INDIA bestand die Möglichkeit, Experimente im Zusammenhang mit der Online-Diagnose anhand der konkreten Daten eines Versuchsfahrzeug durchzuführen, das bei Bosch im Rahmen des thematisch verwandten EU-Projekts "VMBD" (Vehicle Model Based Diagnosis) zur Verfügung stand. Es handelte sich um einen Volvo 850 TDI mit einer elektronisch gesteuerten, verteilerbasierten Einspritzung (distributor type injection, abgekürzt DTI), für die im Rahmen des Projekts VMBD ein kompositionales Modell entwickelt worden war. In dem genannten Versuchsfahrzeug konnten Fehler in verschiedenen Betriebszuständen injiziert und Messungen der Sensorsignale vorgenommen werden. Die verschiedenen im Fahrzeug eingebauten Fehler wurden hierbei durch Schalter vom Fahrzeuginnenraum aus aktiviert. Ein pneumatisches Leck wurde beispielsweise durch zusätzliche Ventile simuliert, die elektrisch geöffnet oder geschlossen werden konnten.

Für die Gewinnung der Signale zur Online-Diagnose mussten verschiedene zusätzliche Interfaces und Geräte im Fahrzeug installiert werden. Voraussetzung war dabei, dass aus Sicherheitsgründen das Seriensteuergerät und seine Diagnosefunktionen ohne Unterbrechung laufen konnten, und der modellbasierte Diagnoseprototyp Zugriff auf die die gleichen Sensorsignale wie das Seriensteuergerät erhalten musste.

Momentan ist die Rechenleistung von Steuergeräten noch sehr beschränkt, so dass das modellbasierte Online-Diagnosesystem nicht direkt in die Steuergerätesoftware integriert werden konnte. Die Messdaten für den Prototypen wurden deshalb über einen Umweg mit Hilfe eines sogenannten Applikationssteuergeräts gewonnen. Applikationssteuergeräte werden normalerweise zur Parametrisierung und Kalibrierung der Steuergerätesoftware für einen bestimmten Fahrzeugtyp verwendet und enthalten spezielle Speicherbausteine, so dass die Variablen des Steuergeräts und damit auch die Sensorsignale in Echtzeit ausgelesen werden können, ohne die normale Funktion des Steuergeräts zu beeinträchtigen. Dadurch standen die Sensorsignale dem Online-Prototyp in der gleichen zeitlichen Auflösung wie dem Seriensteuergerät zur Verfügung. Die Ergebnisse des Online-Prototyps konnten dadurch auch mit den Diagnosefähigkeiten eines normalen Seriensteuergeräts verglichen werden.

2.3.8 Diagnoseszenarien

Die in dem beschriebenen Versuchsfahrzeug eingebauten Fehler waren im Wesentlichen solche, die mit traditioneller On-Board-Diagnose (d.h. basierend auf Schwellwerten und fest eingebauten Plausibilitätsprüfungen) nicht oder nur unzureichend diagnostiziert werden können. Dazu zählen vor allem emissions-relevante Symptome wie erhöhte Kohlenstoffemissionen aufgrund von übermäßiger Kraftstoff- oder unzureichender Verbrennungsluftzufuhr zum Motor, die durch elektrische Fehler oder undichte pneumatischen Leitungen verursacht werden.

Eines der im Versuchsfahrzeug realisierten Szenarien besteht aus einem Leck in der Verbindung zwischen dem Turbolader-Auslass und dem Motorkrümmen. Wenn das Leck geöffnet wird, strömt - abhängig vom Betriebszustand - eine signifikante Menge an Ladeluft aus, die zuvor bereits den Luftmassenmesser passiert hat. Die Einspritzmenge, die vom Steuergerät in Abhängigkeit von diesem Signal berechnet wird, ist deshalb zu hoch für die Menge an Verbrennungsluft, die in der Verbrennungskammer des Motors tatsächlich vorhanden ist. Dieser Fehler führt deshalb zu unvollständiger Verbrennung des Dieselkraftstoffs und damit zu erhöhten Emissionen und verminderter Leistung des Motors.

2.3.9 Messerfassung der Rohdaten

Der Online-Diagnoseprototyp benutzt die gleichen Signale, die auch dem Seriensteuergerät zur Verfügung stehen. Von den verfügbaren Steuergerätevariablen wurden für das beschriebene Szenario die folgenden Variablen an den Prototypen übergeben:

- Atmosphärendruck
- Ladedrucksignal
- Luftmassensignal
- Motordrehzahl
- Ansteuersignal des Ladedruckventils
- aktuelle Einspritzmenge

Die einzelnen quantitativen Messwerte dieser Variablen liegen nicht notwendigerweise zu äquidistanten Zeitpunkten vor. Der Grund ist, dass die Frequenz, mit der das Steuergerät die Sensorwerte abliest, von der Drehzahl des Motors abhängt.

2.3.10 Beispiel für ein Diagnoseergebnis des Prototyps

In diesem Abschnitt werden die Ergebnisse eines Probelaufs des Diagnoseprototypen für das beschriebene Szenario dargestellt. Die Messung für das exemplarische Beispiel läuft über 9,75 Sekunden und umfasst 1064 quantitative Beobachtungsvektoren. Das oben beschriebene Modul zur Signaltransformation reduziert diese Daten zu lediglich 12 qualitativen Beobachtungsvektoren. Basierend auf diesen Daten und dem Modell generiert das Diagnosemodul drei Konfliktmengen:

- {Junction1.ok, IntakeTurbine.ok, Junction3.ok, Engine.ok, AirflowSensor.ok, Junction2.ok, PressureSensor.ok} zu den qualitativen Beobachtungsvektoren 3, 5 und 6,
- {Junction1.ok, IntakeTurbine.ok, Junction3.ok, Engine.ok, AirflowSensor.ok, Junction2.ok, SpeedSensor.ok} zu den qualitativen Beobachtungsvektoren 5, 6 und 10,
- {Junction3.ok, Engine.ok, SpeedSensor.ok, PressureSensor.ok} zum qualitativen Beobachtungsvektor 10.

Die Komponentennahmen stehen hier für die Annahme, daß die entsprechende Komponente korrekt funktioniert. Die drei Konflikte kombinieren sich zu zwei Einfachfehlerhypothesen und einer Reihe von Mehrfachfehlerhypothesen:

- {Junction3.ok}
- {Engine.ok}
- {PressureSensor, Junction1.ok}
- {PressureSensor, IntakeTurbine.ok}
- {PressureSensor, AirflowSensor.ok}
- {PressureSensor, Junction2.ok}
- {SpeedSensor.ok, Junction1.ok}
- {SpeedSensor.ok, IntakeTurbine.ok}
- {SpeedSensor.ok, AirflowSensor.ok}
- {SpeedSensor.ok, Junction2.ok}
- {SpeedSensor.ok, PressureSensor.ok}

Die zwei Einfachfehlerhypothesen enthalten das Verbindungsstück, wo der Fehler tatsächlich injiziert wurde (Junction3). Die Laufzeit für das Beispiel beträgt 2,87 Sekunden auf einem Pentium PC unter Windows NT. Das bedeutet, dass für dieses Beispiel die Performance des Online-Diagnoseprototyps im Bereich der Echtzeit liegt (die Messung für das Szenario lief über 9,75 Sekunden). Für das Beispiel wurde lediglich ein Modell des Normalverhaltens verwendet. Zusätzliches Wissen über die auftretenden Fehlerfälle kann verwendet werden, um die Menge der Diagnosekandidaten weiter einzuschränken, z.B. um den Motor aus den Diagnosekandidaten zu eliminieren.

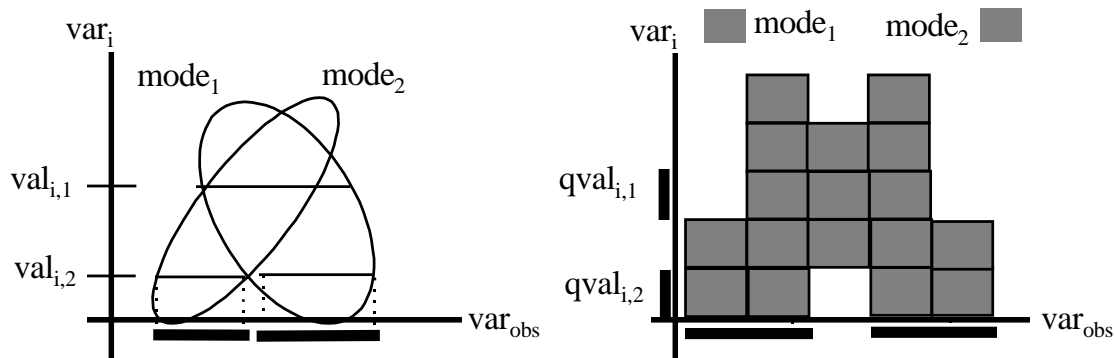


Abbildung 2-5: Zwei durch Relationen repräsentierte Verhaltensmodus a über kontinuierlichen Wertebereichen b in abstrahierter Form

Mögliche Eingabewerte für die Inputvariable var_i auf der vertikalen Achse, die zugehörigen möglichen Werte der beobachtbaren Variable var_{obs} als Balken an der horizontalen Achse

Der entstandene Diagnoseprototyp hat gezeigt, dass die modellbasierte Online-Diagnose einer Motorsteuerung in Echtzeit möglich ist. Der Prototyp besitzt jedoch ebenso wie der FMEA-Prototyp noch keine Bedienoberfläche, die auf einen Anwender in der Werkstätte zugeschnitten wäre.

2.3.11 Ausblick: Jenseits komponenten-orientierter Modellierung und Diagnose

Die geschilderten Ansätze zur Werkstatt- und Online-Diagnose sind gemäß dem allgemeinen Stand der Kunst geeignet, Komponentenfehler in stabilen Strukturen zu lokalisieren. Dies ist aber bei einigen Problemen im Fahrzeug (und in stärkerem Maße in der Prozeßindustrie sowie bei der Diagnose und Therapie natürlicher Systeme) zu beschränkt. So läßt sich etwa die mögliche Ursache für erhöhte CO-Emissionen nicht nur an defekten Komponenten festmachen. z.B. kann Treibstoff mit einem erhöhten Anteil schwerer Kohlenwasserstoffe einen veränderten Verbrennungsprozeß und damit ebenfalls erhöhte Emissionen hervorrufen.

Im Projekt wurden daher theoretische Grundlagen entwickelt und implementiert, die den modellbasierten Diagnoseansatz auf Systeme mit einer dynamischen Struktur aus Prozessen ausweiten (siehe [Struss Heller 1999], [Struss Heller 2000], in diesem Band). Die Prozeßmodellierung kann sich wiederum auf Modelle stützen, deren ursprüngliche Elemente nicht Constraints (d.h. Relationen), sondern kombinierbare Einflüsse sind (s. [Heller Struss 1996], Nachdruck in diesem Band). Diese Erweiterungen öffnen der Anwendung modellbasierter Diagnoseverfahren eine große neue Klasse von Systemen und Fehlern.

2.4 Generierung von Fehlersuchanleitungen

2.4.1 Automatische Testgenerierung

Testen eines Systems bedeutet, es durch eine Änderung seiner Struktur und/oder einen bestimmten Stimulus (Eingabe) so zu beeinflussen, daß dadurch Beobachtungen möglich werden, die Information über den „Verhaltensmodus“ des Systems (insbesondere vorhandene Defekte) bereitstellen. Für ein System, das sich aus Komponenten in einer festen Struktur zusammensetzt, ist dies gleichbedeutend mit dem Ziel, die Verhaltensmodi der Systemkomponenten, d.h. Komponentenfehler, zu erkennen. Beispielsweise geht es beim Testen nach der Fertigung um die Feststellung, ob alle Komponenten korrekt funktionieren oder ob eine von ihnen fehlerhaft arbeitet. Dies wird normalerweise realisiert durch den Versuch, alle Einfachfehler auszuschließen. Im Diagnosekontext dient Testen dazu, einen vorhandenen Fehler oder eine Klasse von Fehlern zu identifizieren. Da es prinzipiell unmöglich ist, einen bestimmten Fehler oder eine Fehlerklasse direkt durch Beobachtungen zu bestätigen, ist die Aufgabe nur durch die Durchführung von Tests zu lösen, die das Vorhandensein aller anderen möglichen Fehler ausschließen können. Im Gegensatz zur Diagnose, die sich auch allein auf die Kenntnis des intendierten, korrekten Verhaltens stützen kann (vgl. Abschnitt 0), benötigen Testgenerierung und Testen immer das Wissen über die Menge der möglichen Fehler(klassen).

Da modellbasierte Methoden Wissen über korrektes und fehlerhaftes Verhalten von Komponenten und dessen Auswirkungen repräsentieren bzw. ableiten können, bieten sie sich zur Unterstützung und Automatisierung an. In früheren Arbeiten ([Struss 1994a], [Struss 1994b]) haben wir die theoretischen Grundlagen für modellbasierte

(komponentenorientierte) Testgenerierung geschaffen und deren Implementierung an einem realistischen Anwendungsproblem erprobt ([Inderst et al.1995]). Wir verzichten an dieser Stelle auf eine formale Präsentation und beschränken uns darauf, die grundlegenden Ideen zu vermitteln und zu illustrieren.

In der entwickelten Theorie und den darauf gegründeten Algorithmen werden die Verhaltensmodelle des zu testenden Systems (etwa alle Verhaltensweisen, die sich durch Einfachfehler ergeben) als Relationen über der Menge der System- und Zustandsvariablen sowie der Parameter repräsentiert. Abbildung 2-5a zeigt zwei abstrakte Verhaltensrelationen als Venn-Diagramme, wobei die zeitlichen Aspekte des Verhaltens ignoriert werden. Das Testen zur Diskriminierung zwischen Verhaltensmodi ist dann die Aufgabe, die Unterschiede zwischen diesen Mengen zu entdecken, also Tupel, die nicht in beiden Relationen enthalten sind. Genauer gesagt geht es darum, eine geeignete Eingabe zu bestimmen, d.h. den Wert von beeinflussbaren Variablen zu fixieren, in Abbildung 2-5a als $var_i=val_{i,1}$ auf der vertikalen Achse. Geeignete Eingaben sind solche, die unter den verschiedenen Modellen (Relationen) möglichst unterschiedliche Beobachtungen hervorrufen. Die möglichen Beobachtungen, in Abbildung 2-5a als Variable v_{obs} auf der horizontalen Achse dargestellt, sind gegeben durch die Projektionen des Durchschnitts der Eingabe $var_i=val_{i,1}$ mit den jeweiligen Relationen auf die beobachtbaren Variablen, im Beispiel v_{obs} . Offenkundig ist hier $var_i=val_{i,1}$ eine schlechte Wahl, da die resultierenden Projektionen einander sehr stark überlappen, während $var_i=val_{i,2}$ mit Sicherheit $mode_1$ von $mode_2$ unterscheidet. Wenn Fehlerwahrscheinlichkeiten und/oder Wahrscheinlichkeitsverteilungen über den Relationen gegeben sind, kann die Minimierung der Entropie als Kriterium für die Auswahl des Tests mit dem größten Informationsgewinn genutzt werden (siehe [Struss 1994b]).

Offenkundig ist das Berechnen der jeweiligen Projektionen für alle möglichen Inputs sehr aufwendig, wenn nicht unmöglich. Es wird aber ermöglicht, indem wir qualitative anstelle von numerischen Modellen verwenden, wie in Abbildung 2-5b symbolisiert durch die Darstellung der Verhaltensrelationen als endliche Menge von Rechtecken (qualitativen Werten). Unter der Voraussetzung, daß die qualitativen Verhaltensrelationen die feinkörnigeren vollständig überdecken, ist es möglich, die oben genannten Berechnungen nunmehr auf endlichen Mengen von Werte-Tupeln durchzuführen. Außerdem wird durch diesen Schritt die Menge der Fehler endlich, da diese qualitative Abstraktion Mengen kontinuierlich variierender Fehler zu einer Beschreibung zusammenfaßt (z.B. unterschiedlich hohe Abweichung eines Widerstands).

Übertragen auf elektrische Schaltungen bedeutet dies beispielsweise, daß qualitative Modelle nicht exakte Strom- und Spannungswerte in Relation setzen, sondern den Wertebereich für Stromvariablen auf "negativ"(neg), "null"(zero) und "positiv"(pos) reduzieren. Entsprechend sind für den Wertebereich von Spannungsgrößen "Masse"(gnd) und "Batterie"(bat) zwei wichtige Größen, ein dritter Bereich umfaßt alle Werte dazwischen (btw).

Mit diesen Wertebereichen ergibt sich das in Tabelle 2-12-1 links dargestellte Modell für das korrekte Verhalten eines Temperaturfühlers. Das fehlerhafte Verhalten eines offenen Temperaturfühlers ist durch den Strom „null“ für alle Inputs gegeben (Tabelle 1, rechts). Die Inputs, die (potentiell) verschiedene Beobachtungen über den Strom produzieren, sind direkt abzulesen. Durch die qualitative Abstraktion gelten diese Modelle sogar für alle elektrischen Widerstände.

(gnd, gnd)		(gnd, gnd)	
(gnd, btw)		(gnd, btw)	
(gnd, bat)		(gnd, bat)	
(btw, gnd)		(btw, gnd)	
(btw, btw)		(btw, btw)	
(btw, bat)		(btw, bat)	
(bat, gnd)		(bat, gnd)	
(bat, btw)		(bat, btw)	
(bat, bat)		(bat, bat)	
	neg 0 pos		neg 0 pos

Tabelle 2-1 Qualitatives Modell des korrekten (links) und offenen Temperaturfühlers (rechts). Vertikal sind die Paare der Potentiale links und rechts aufgetragen, horizontal der Strom

Auf dieser Grundlage wurden in vorangegangenen Arbeiten Testgenerierungsalgorithmen implementiert und auf Relais-Schaltungen angewendet, ein Beispiel aus einer Anwendung zur Programmierung von Testautomaten für

Weichenschaltungen ([Inderst et al. 1995]). Das System produzierte beweisbar korrekte Mengen von Tests und identifizierte Fehlerpaare, die unter den gegebenen Voraussetzungen nicht unterscheidbar waren, für Schaltungen mit ca. 50 Komponenten. Dies liegt bereits jenseits dessen, was menschliche Programmierer garantiert erschöpfend behandeln können. Dennoch war der implementierte Prototyp noch relativ weit von einem tatsächlich einsetzbaren Werkzeug entfernt. Dies liegt an zwei grundlegenden Beschränkungen.

Zum einen betrachtet die skizzierte Lösung Testgenerierung als sehr abstrakte Aufgabe und ignoriert wesentliche praktische Aspekte des Testens, z.B. benötigte Ausrüstung und Kosten, Anordnung der Tests, um die Kosten zu reduzieren, die Tatsache, daß einige Tests riskant oder unmöglich sind, solange nicht bestimmte Fehler ausgeschlossen sind.

Zweitens kann jedes modellbasierte System zur Testgenerierung, selbst wenn es die genannten Aspekte beinhaltet, nur ein wirksames Werkzeug werden, wenn es den tatsächlichen Arbeitsprozeß der Erzeugung und Anwendung von Testplänen berücksichtigt. In unserem Anwendungsbereich geschieht dies in den Kundendienstabteilungen. Hier werden mit relativ großem Personalaufwand sogenannte Fehlersuchanleitungen geschrieben, geprüft, angepaßt, aktualisiert und übersetzt, die dann, in unserem Fall auf CD-ROM, an die Kundendienstwerkstätten zur Anleitung der Diagnose und Reparatur ausgeliefert werden. Den Kern dieser Dokumente bilden natürlich-sprachlich beschriebene Pläne für die Durchführung von Werkstatt-Tests mit dem Ziel der Fehlerlokalisierung.

Damit ist ein natürlicher Weg der Einführung modellbasierter Testgenerierung, die Autoren in den zentralen Kundendienstabteilungen bei der Erstellung von Fehlersuchanleitungen mit geeigneten Werkzeugen zu unterstützen. Wir haben dies in INDIA begonnen, indem wir einen neuen Typ von Autorensystem, genannt Genesis (Generation of Electronic Service Information Systems) spezifiziert und implementiert haben. Wir beschreiben im folgenden kurz dieses System und kommen dann auf die Nutzung modellbasierter Testgenerierung in seinem Kontext zurück.

2.4.2 Genesis - ein Autorensystem für Fehlersuchanleitungen

Genesis zielt auf die Integration automatischer, modellbasierter Testgenerierung in ein Autorensystem. Es kombiniert die Wiederverwendung von Modellen mit der verbesserten Wiederverwendung von Texten und bietet eine Ebene der Wissensrepräsentation für Daten und Informationen über die Fahrzeugsysteme.

Im Projekt wurde zunächst der gegenwärtige Arbeitsprozeß der Autoren von Fehlersuchanleitungen systematisch untersucht und formal repräsentiert, um eine Basis für die Kopplung mit der Testgenerierung und Reparaturplanung zu schaffen. Traditionelle Diagnose-Autorensysteme sind im wesentlichen Textverarbeitungssysteme, eventuell mit einer Datenbasis von Textbausteinen, die zudem Teile des weiteren Arbeitsprozesses unterstützen, wie etwa Freigabe und Übersetzung von Dokumenten. Gewöhnlich wird die vorhandene relativ starke Strukturierung des Inhalts der Diagnoseanleitungen nicht besonders ausgenutzt, und aufgrund der resultierenden reinen Textsicht gibt es keine Unterstützung für die Überprüfung und Aufrechterhaltung der Konsistenz der Anleitungen. Ferner ist es auf dieser Basis nicht möglich, semantisch annotierte Dokumentformate, z.B. sog. mark-up languages wie SGML oder XML zu erzeugen. Dies gerät zunehmend in Konflikt mit aktuellen Anforderungen wie dem in SGML formulierten Standard J2008 für Diagnosedokumente und der Präsentation der Manuale im Internet (vgl. www.w3c.org, www.sae.org).

In INDIA verfolgten wir das Ziel, die konventionellen, textorientierten Autorensysteme in drei Stufen zu verbessern und zu erweitern:

1. durch die explizite und detaillierte Strukturierung der Fehlersuchanleitungen und die semantische Annotation von Textelementen (Satzfragmenten, Phrasen, Dokument-abschnitten),
2. durch die Bereitstellung einer Ebene der Wissensrepräsentation, die es dem System ermöglicht, die Beziehung zwischen Textelementen der Fehlersuchanleitung und Struktur und Komponenten des jeweiligen Fahrzeugsystems herzustellen,
3. durch die Verbindung dieser Strukturdarstellung mit einer Bibliothek von Verhaltensmodellen als Basis für Anbindung und Ausnutzung modellbasierter Techniken wie etwa der Testgenerierung.

Die erste Stufe bringt bereits den gravierenden Vorteil der wesentlich weiter reichenden Wiederverwendung von Textfragmenten in verschiedenen Fehlersuchanleitungen. Dies bewirkt auch eine drastische Reduktion des Übersetzungsaufwands, der einen wesentlichen Kostenfaktor in der Bereitstellung der Fehlersuchanleitungen darstellt. Es gibt hunderte von Fahrzeugvarianten, während die Fehlersuche normalerweise aus Sequenzen von einigen wenigen stereotypen Tests und Aktionen besteht. Daher tauchen in den Anleitungen vielfache

Wiederholungen bestimmter Phrasen, mit leichten Variationen, auf. Dieser Tatsache wird in Genesis Rechnung getragen, indem der Text der Fehlersuchanleitungen bis auf die Ebene kleiner kohärenter Phrasen gegliedert wird (die auch individuell übersetzt werden können). Eine derart strukturierte Fehlersuchanleitung kann damit automatisch übersetzt werden, sofern sie keine neuen Textfragmente einführt. Ein Seiteneffekt ist, daß die Manuale einheitlicher werden, selbst wenn sie von verschiedenen Autoren geschrieben werden, da sie sich auf dasselbe Repertoire von Formulierungen stützen. Die Textfragmente können gemäß ihrer unterschiedlichen Rolle in der Anleitung organisiert werden, was ihr Retrieval unterstützt. z.B. kann ein Autor leicht zu den verfügbaren Varianten für die Formulierung von Spannungsmessungen geführt werden. Darüber hinaus ermöglicht die semantische Annotation der Textelemente die automatische Transformation in Dokumentformate wie SGML oder XML.

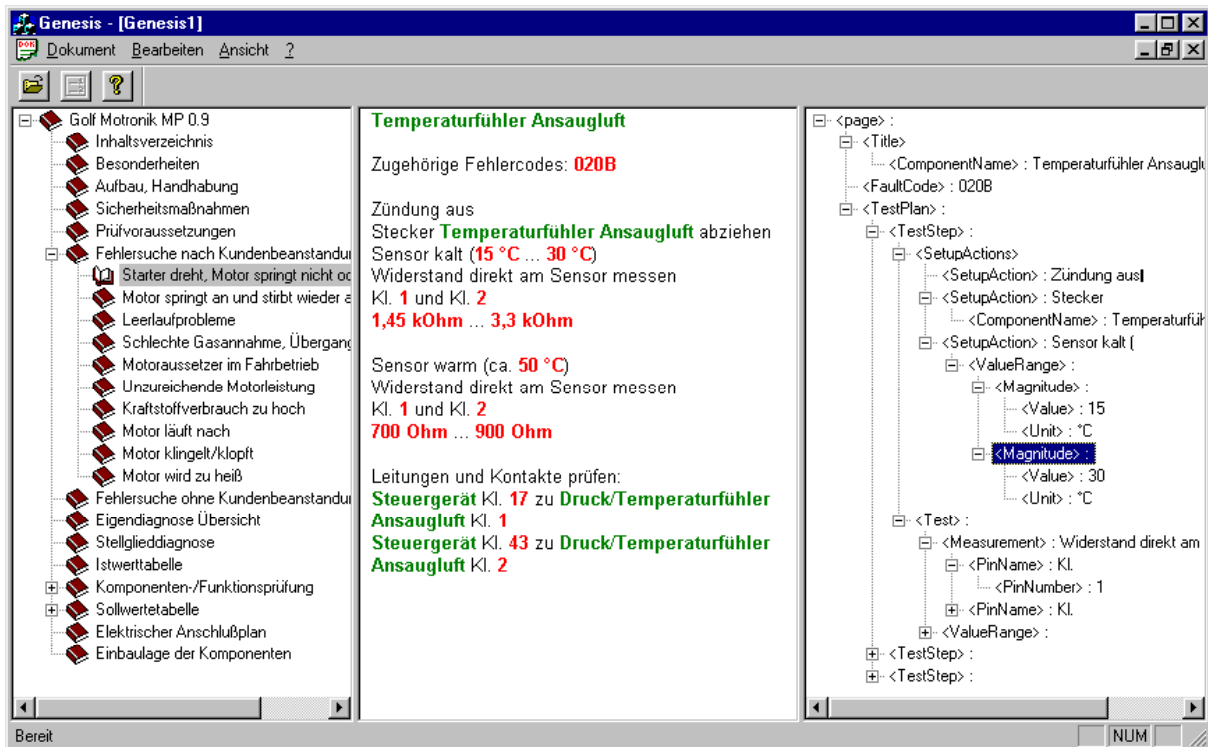


Abbildung 2-6: Bildschirmabzug des Autorensystems Genesis, in den Fenstern: links: Kapitelstruktur der Fehlersuchanleitung, Mitte: bearbeiteter Abschnitt (hier: Prüfplan), rechts: Struktur dieses Abschnitts, nur partiell entfaltet

Abbildung 2-6 zeigt einen Bildschirmabzug dieser ersten Stufe von Genesis. Im linken Fenster wird die Struktur der Fehlersuchanleitung (hier für eine Dieseleinspritzung) dargestellt, mit deren Hilfe der Autor durch die Kapitel und Seiten des Dokuments navigieren kann. Nach Auswahl einer Seite wird deren Inhalt im mittleren Fenster gezeigt. In diesem Fenster kann der Text editiert werden. Dabei sind Satzteile, die sich auf Eigenschaften von Fahrzeugkomponenten beziehen (z.B. "Temperaturfühler Kühlmittel"), farbig hervorgehoben, die Komponenten sind grün, Parameter rot. Die dritte Sicht (im Fenster rechts) zeigt die hierarchische Strukturierung der Seite in Textfragmente. Sie illustriert, wie solche Fragmente selbst aus einfacheren aufgebaut sein können. z.B. setzt sich ein *TestStep* aus einer Liste von *SetupActions* und einem *Test* zusammen. Der Baum in Abbildung zeigt die Seite nicht völlig entfaltet.

In der zweiten Phase von Genesis wurde diese semantische Unterlegung der Textfragmente mit einer modellbasierten Repräsentation des Fahrzeugs und seiner Subsysteme verbunden. Dieses Fahrzeugmodell besteht aus einer hierarchischen Darstellung der Komponenten und ihrer Beziehungen („Teil von“), einer Beschreibung der Gerätetopologie (Strukturdarstellung unter Benutzung der jeweiligen Komponenten) und Datenblättern, die die entsprechenden Parameter enthalten. Die Elemente dieser Darstellung, also z.B. Komponenten, Klemmen, Parameter, werden mit den entsprechenden Textfragmenten verbunden. Der Hauptvorteil liegt darin, daß der Autor in dieser Darstellung die Beschreibung des Subsystems ändern kann, ohne im einzelnen die textuelle Fehlersuchanleitung editieren zu müssen. z.B. erscheinen im Datenblatt geänderte Parameterwerte oder in der Grafik geänderte Klemmenbezeichnungen unter allen anderen Sichten, also auch in der Textsicht und der Struktursicht auf die Seiten. Grundsätzlich (aber in Genesis nicht realisiert) besteht die Möglichkeit, elektronisch

vorliegende Fahrzeugdaten aus anderen (etwa CAD-)Systemen zu importieren. Umgekehrt führen Eingaben, die etwa im Text vorgenommen wurden, auch zu den entsprechenden Änderungen in der Systemdarstellung.

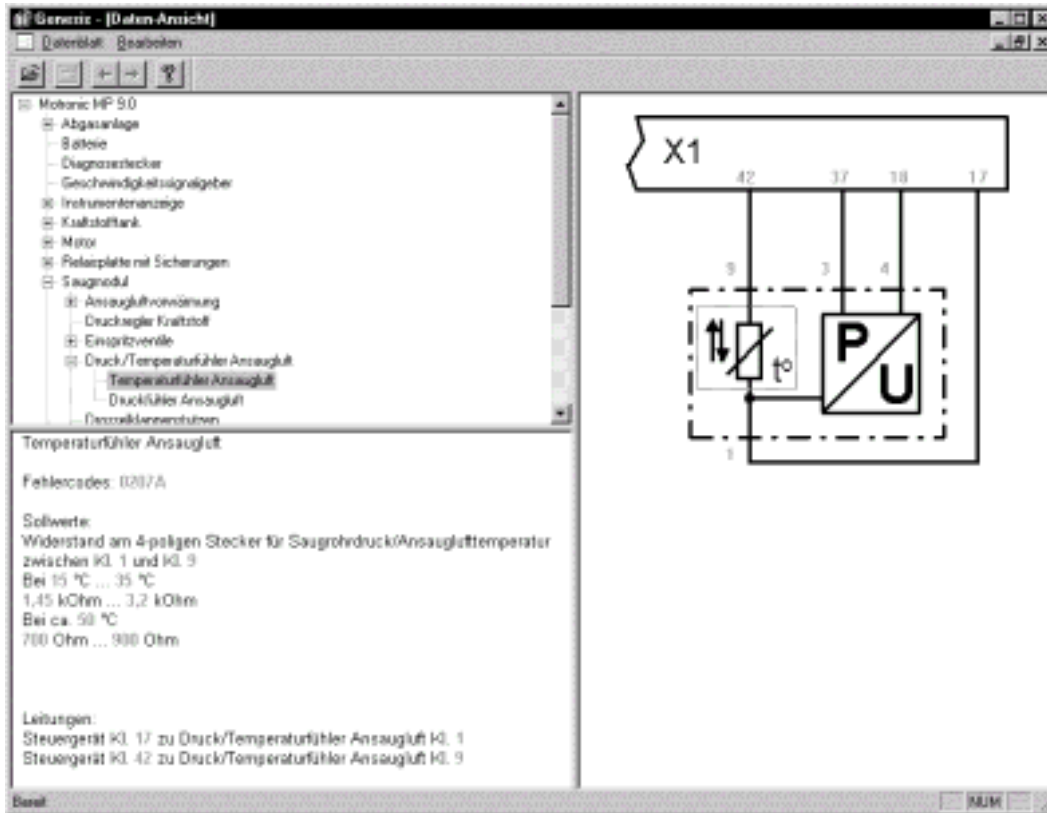


Abbildung 2-7: Bildschirmabzug: zusätzliche Views der zweiten Stufe. Links oben: Systemhierarchie, rechts: Struktur mit Eingabemöglichkeiten, links unten: Datenblatt einer Komponente

Abbildung 2-7 zeigt die zusätzlichen Sichten der zweiten Stufe. Das obere linke Fenster enthält die hierarchische Systemdarstellung. Darunter wird das Datenblatt einer ausgewählten Komponente bzw. Funktionsgruppe gezeigt, hier des Temperatursensors für die Ansaugluft. Im rechten Fenster wird der Schaltplan dieser Funktionsgruppe dargestellt. In jeder dieser Sichten kann der Autor Komponenten, Klemmen oder Parameter auswählen und die zugehörige Information editieren.

Die explizite und eindeutige Repräsentation der Information, die variieren kann, stellt automatisch die Konsistenz des Dokuments sicher und entbindet den Autor von der Notwendigkeit, gleichartige Anpassungen bei allen betroffenen Textteilen selbst vornehmen zu müssen. Etwaige Auswirkungen auf die Struktur des Prüfplans selbst können auf dieser Stufe natürlich nicht erkannt oder gar automatisch vorgenommen werden.

Dies wird in der dritten und letzten Stufe von Genesis auf der Grundlage modellbasierter Techniken möglich. In der zweiten Stufe werden durch die explizite Repräsentation der Struktur und physikalischen Eigenschaften des Systems die Grundlage dafür gelegt. Stufe 3 nutzt sie im Zusammenhang mit Bibliotheken von Verhaltensmodellen für die beteiligten Komponententypen, wie sie auch für die modellbasierte Unterstützung von FMEA und Diagnose verwendet werden. Damit werden die oben geschilderten Techniken der Testgenerierung im Kontext des Autorensystems nutzbar. Und da in ihm auch die Prüfpläne der Fehlersuchanleitung formal repräsentiert sind, wird es auch möglich, die Resultate der automatischen Testgenerierung zu importieren und als Text zu präsentieren.

Damit ist ein Weg realisiert, modellbasierte Testgenerierung ohne gravierende Änderungen im Arbeitsprozeß und in der Sicht der Autoren nutzbar zu machen, und somit eines der oben genannten Hindernisse für den Transfer dieser Technologie ausgeräumt. Hinzutreten muß aber die Erweiterung der geschilderten abstrakten Testgenerierung um die Behandlung der praktischen Dimension. Dies soll im folgenden skizziert werden.

2.4.3 Modellbasierte Erzeugung von Prüfplänen für Genesis

Abbildung 2-8 zeigt ein typisches Beispiel einer Funktionsgruppe, wie sie in der Prüfanleitung eine Motorsteuergeräts behandelt wird.

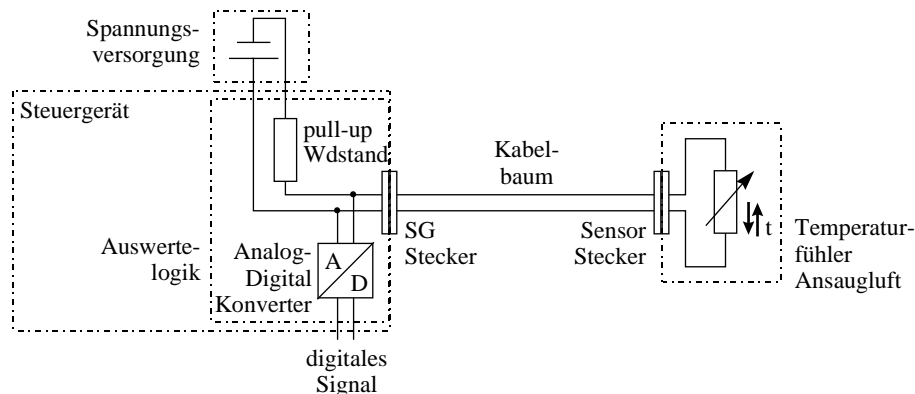


Abbildung 2-8: Funktionsgruppe „Temperaturfühler Ansaugluft“ bestehend aus Steuergerät mit Spannungsversorgung, Steckern und Sensor

Dargestellt ist der Temperaturfühler für die Ansaugluft mit seinen Steckern und Verbindungen sowie die Auswertung des analogen Sensorsignals im Motorsteuergerät. Aus der Sicht der Werkstattdiagnose ist es angemessen, zwischen den Fehlern von kleinsten austauschbaren Einheiten zu unterscheiden. Dies wäre in dieser Baugruppe ein Steuergerätefehler, ein Fehler auf der zwei-adrigen Verbindungsleitung oder ein Sensordefekt (die Steckerkontakte werden immer gemeinsam mit den Komponenten, an denen sie hängen, ausgetauscht). Typische Komponentenfehler dieser Schaltung sind eine zu niedrige Versorgungsspannung, wie auch Kurzschlüsse und Unterbrechungen des Temperaturfühlers, der Stecker und Leitungen (Kabelbaum).

Durch die Anforderung, daß die Prüfanleitung mit in der Werkstatt verfügbaren Mitteln abgearbeitet werden soll, ergeben sich Möglichkeiten und Einschränkungen der verwendbaren Beeinflussungen und Messungen: So gilt beispielsweise für den Temperaturfühler die im Steuergerät ermittelte Spannung bzw. der damit verbundene Temperaturwert als beobachtbar, da er mit Hilfe eines speziellen Testgerätes ausgelesen werden kann. Ebenso kann man ein Vielfachmeßgerät anschließen und damit Spannungen oder Widerstandswerte bestimmen. Änderungen, die an der Baugruppe vorgenommen werden können, sind im wesentlichen das Öffnen und Schließen von Steckverbindungen. An offenen Steckern kann alternativ das Vielfachmeßgerät angeschlossen werden, welches man zwischen den Betriebsmodi "Spannung messen" und "Widerstand messen" umschalten kann.

Diese Zusammenhänge und Informationen sind natürlich für die eigentlichen Handlungsanweisungen zur praktischen Durchführung der Fehlersuche und Reparatur das Wesentliche und legen auch Voraussetzungen, Kosten, Aufwand und damit die Reihenfolge der einzelnen Schritte fest. Z.B. ist das Ablesen des Ist-Wertes des Sensors am Steuergerät billig, vorausgesetzt, der Tester ist bereits angeschlossen. In dem in Abschnitt 0 skizzierten Verfahren zur automatischen Testgenerierung wurde von solchen praktischen Bedingungen völlig abstrahiert. Sie liefert auf Basis des Verhaltensmodells lediglich eine abstrakte Charakterisierung von Tests. Dies ist unter Gesichtspunkten der Wiederverwendung von Software und Ergebnissen durchaus erwünscht: Wenn sich nur die praktischen Randbedingungen (wie praktischer Aufwand von Messungen, verfügbare Geräte etc.), aber nicht die Struktur des Testgegenstandes ändern, bleiben die prinzipiell hilfreichen Tests dieselben, und lediglich ihre Auswahl oder Reihenfolge ist zu modifizieren. Im betrachteten Anwendungsbereich sind z.B. der Aufbau der Funktionsgruppen und ihre Funktionalität relativ stabil, während ihre Einbaulage und damit Zugänglichkeit und Meßmöglichkeiten abhängig vom Hersteller- oder Fahrzeugtyp sind.

Daher wurde die sehr generische Testgenerierungskomponente (im folgenden als Generator für abstrakte Tests, abstract test generator, bezeichnet) bewahrt, statt sie um die Darstellung und Berücksichtigung der praktischen, handlungsbezogenen und Kostengesichtspunkte zu erweitern, die ihrer Natur nach sehr anwendungsspezifisch sein können. Die Verarbeitung dieses Wissens zur Planung konkreter Prüfabläufe wurde in einer gesonderten Komponente, dem Aktionsplaner (action planner) realisiert. Der Einsatz dieser beiden Komponenten wird einer weiteren Komponente, dem Strategen („strategist“) kontrolliert und koordiniert (vgl. Abbildung 2-9). Die Funktion der einzelnen Komponenten und ihre Interaktion werden im folgenden anhand der Schritte in einem

Zyklus der Prüfplangenerierung diskutiert. Dies ist beispielhaft zu verstehen, denn die Architektur soll es gerade ermöglichen, verschiedene Strategien unter Verwendung der Basiskomponenten zu realisieren.

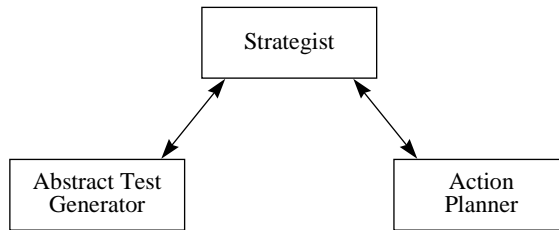


Abbildung 2-9: Die drei Komponenten der Prüfplangenerierung

Der Strategie

Diese Komponente ist Ausgangspunkt und Koordinator des gesamten Prozesses der Testgenerierung und Prüfplanung. Die Beschreibung eines Prüfplanungsproblems als Input umfaßt, wie in Abbildung 2-10 dargestellt, zumindest

- ein Verhaltensmodell des Geräts („device model“)
- eine Beschreibung der grundsätzlich verfügbaren Aktionen und Beobachtungsmöglichkeiten („actions and observations“), sowie
- zugehörige „pragmatische“ Information über deren Kosten, Vorbedingungen etc. („costs and pragmatics“), also z.B., daß der Anschluß des Meßgeräts das Öffnen der Steckverbindungen voraussetzt und einen gewissen Zeitbedarf aufweist.
- eine Beschreibung des jeweiligen aktuellen Zustandes („current situation“).

Die Charakterisierung des Zustands wiederum umfaßt drei Aspekte:

- den Systemzustand in engeren Sinne, wie er im Verhaltensmodell ausgedrückt ist, eine (möglicherweise partielle) Beschreibung von Zustandsvariablen: Schalterstellungen, etwa „Zündung ein“ repräsentierend, „Motortemperatur betriebswarm“ etc.,
- den Zustand des physikalischen Systems als Resultat vorangegangener Aktionen, insbesondere im Hinblick auf Demontage, Anschluß von Meßgeräten etc., und
- den Zustand des Diagnoseprozesses, ausgedrückt als Menge von verbleibenden Fehlerhypothesen, die es noch zu diskriminieren gilt, ggf. mit Information über deren aktuelle Wahrscheinlichkeiten.

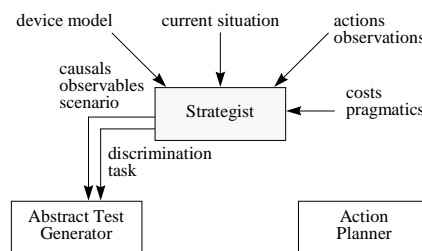


Abbildung 2-10: Input des Strategen und Aufruf des abstrakten Testgenerators

Der Strategie erstellt aus der aktuellen Situation eine (abstrakte) Diskriminierungsaufgabe, also z.B. daß der nächste Test zur Diskriminierung aller gegenwärtigen Fehlerhypothesen (oder einer Auswahl daraus) erzeugt werden soll, falls ein solcher aufgrund der gegebenen Möglichkeiten der Beeinflussung und Beobachtung überhaupt existiert. Der Strategie übergibt diese Aufgabe an den Generator für abstrakte Tests. Daneben legt er fest, welche Variablen dieser als Inputvariable ("causals") oder als beobachtbar ("observables") behandeln soll. Ferner kann der Strategie ein Szenario als Fokus festlegen, d.h. eine Menge von Systemzuständen (im engeren Sinne). Damit kann zum Beispiel kontrolliert werden, daß vorgeschlagene Tests nur beschränkte oder gar keine Veränderung des gegenwärtigen Zustands vornehmen sollen, bestimmte kritische Zustände vermieden werden u.s.w. Dies erscheint einfach als Einschränkung des Verhaltensmodells durch eine weitere Relation.

Der Generator für abstrakte Tests

Dieser im Abschnitt 0 dargestellte Modul verarbeitet die Diskriminierungsaufgabe (Abbildung 2-11) und bedient sich dabei des Gerätemodells und zusätzlich der Information über Wahrscheinlichkeiten (über Fehlerhypothesen und Variablenwerte) im aktuellen Zustand. Das Ergebnis der Testgenerierung ist eine Menge von abstrakten Tests. Jeder abstrakte Test löst zumindest eine Teilaufgabe des Diskriminierungsproblems. Wie betont, beinhaltet diese Darstellung nur die Information, **was grundsätzlich** zu tun ist, d.h. welche Variablenwerte eingestellt und welche Variablen beobachtet werden sollen, jedoch nicht, **wie dies in der Realität** geschehen kann. Der Testgenerator kann bereits ableiten, welche Diskriminierung wie gut durch die einzelnen Tests vorgenommen wird („discriminative power“, repräsentiert als diskriminierte Mengen von Fehlerhypothesen und Entropieänderung), jedoch noch nicht, wie teuer die Durchführung des Tests wird.

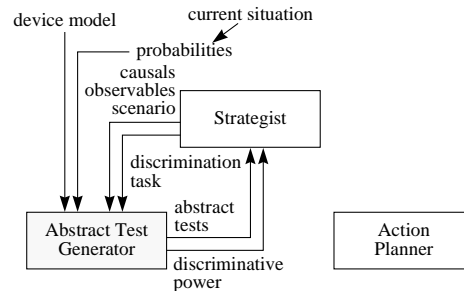


Abbildung 2-11: Input und Output des Generators für abstrakte Tests

Der **Strategie** erhält die generierten Vorschläge für Tests und deren diskriminierende Wirkung und wählt aus diesen nach statistischen oder heuristischen Kriterien eine Teilmenge aus. Sollten keine Tests möglich sein, so muß er, solange dies möglich ist, unter Variation der Diskriminierungsaufgabe, der Menge der beobachtbaren und Input-Variablen und/oder des Focus den Testgenerator erneut aufrufen. Die ausgewählten abstrakten Tests leitet er an die Planungskomponente weiter (Abbildung 2-12).

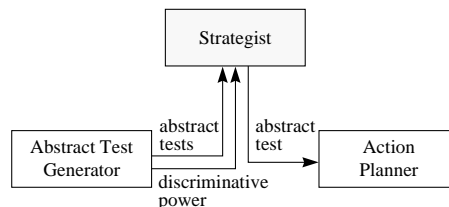


Abbildung 2-12: Der Strategie als Brücke zwischen den möglichen abstrakten Tests und deren praktischer Realisierung

Die Planungskomponente für Aktionsfolgen

Die Komponente hat die Aufgabe, einen abstrakten Test in einen konkreten Test, d.h. eine Sequenz von Handlungen umzusetzen, falls dies möglich ist (Abbildung 2-13). Die erstellte Handlungsfolge muß also zuerst die durch den abstrakten Test bestimmten Variablenwerte ausgehend vom aktuellen Zustand des Systems (einschließlich etwa der vorgenommenen Zerlegung) einstellen und dann entsprechende Beobachtungsaktionen durchführen. Die dafür zur Verfügung stehenden Handlungen und Beobachtungen und deren Kosten sind ein weiterer Input. Die Handlungsfolge sollte natürlich möglichst geringe Kosten verursachen.

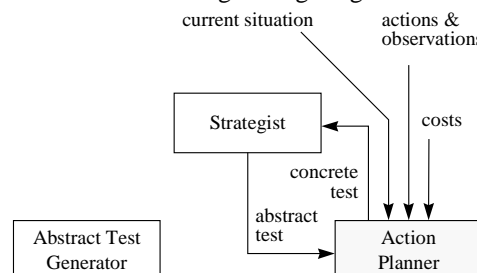


Abbildung 2-13: Input und Output des Aktionsplaners

Aktionen werden jeweils durch ihre Vorbedingungen und Nachbedingungen beschrieben. Beobachtungen sind charakterisiert durch die Variablen, deren Werte ermittelt werden und können ebenfalls Vorbedingungen enthalten. In obigen Beispiel sind alle Aktionen recht einfach gestaltet. Lediglich das Anschließen des Vielfachmeßgerätes unterliegt der Bedingung, daß Widerstände nur gemessen werden dürfen, wenn die entsprechenden Meßpunkte garantiert potentialfrei sind (d.h. keine Verbindung zur Spannungsversorgung besitzen). Andere Aktionen, wie z.B. das Zerlegen des Motors, haben komplizierte Vor- und Nachbedingungen. Im Rahmen der Arbeiten an Genesis wurde kein allgemeiner, ausgefeilter Planer entwickelt, was aber auch durch die betrachteten Prüfpläne nicht erfordert wurde. Durch den modularen Aufbau sind hier mächtigere Planungskomponenten integrierbar.

Der **Strategie** hat nun wieder die Möglichkeit, aus den von der Planungskomponente erstellten Handlungsfolgen eine auszuwählen. Bei dieser Wahl berücksichtigt er vor allem das Verhältnis von Kosten der Handlungsfolge und zu erwartendem Gewinn der Testdurchführung. Wurde kein konkreter Test als möglich zurück geliefert oder erfüllt keiner der Tests gewisse (zumeist kostenorientierte) Kriterien, so kann der Strategie weitere abstrakte Tests an den Planer übergeben oder auch alternativ Reparaturschritte einleiten (komplexere Reparaturanweisungen können dabei wieder von der Planungskomponente erstellt werden, was aber in der gegenwärtigen Implementierung nicht realisiert und hier nicht bildlich dargestellt ist).

Die vom Strategen ausgewählten Handlungen erweitern den Prüfplan, der beim ersten Durchlaufen des Zyklus nur aus der Wurzel besteht, die die anfängliche Diskriminierungsaufgabe repräsentiert. Durch Anwendung der modellbasierten Verhaltensvorhersage kann der Strategie bestimmen lassen, wie sich die Durchführung der Handlung auf den Zustand des Gerätes auswirkt (Abbildung 2-14). Je nach Beobachtungsergebnis der Tests werden dabei unterschiedliche Zustände erreicht. Jeder neue Zustand bestimmt für sich genommen wieder den Ausgangspunkt für einen neuen Durchgang der Testgenerierung. Dabei muß für die sich neu ergebenden Szenarien nicht wieder der ganze Zyklus durchlaufen werden. Die abstrakten Tests behalten im wesentlichen ihre Gültigkeit, lediglich die Wahrscheinlichkeiten, und damit ihre Diskriminierungskraft, verändert sich. Der Strategie bricht ab, wenn alle erreichbaren gewünschten Unterscheidungen durchgeführt und die entsprechenden Reparaturschritte dem Prüfplan hinzugefügt wurden.

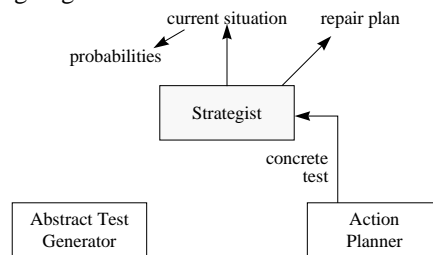


Abbildung 2-14: Der Strategie erweitert den Plan und bestimmt die aus diesem Schritt resultierenden Situationen

Wie erwähnt, sind andere Vorgehensweisen durch Austausch der Strategiekomponente realisierbar: Z.B. könnte diese jeweils vom Planer den mit den geringsten Kosten zu erreichenden nächsten Zerlegungszustand erzeugen lassen, um anschließend alle abstrakten Tests, die in diesem Zustand möglich sind, zu generieren und zu planen. Diese Flexibilität bei der Berücksichtigung anwendungsspezifischer Kostengesichtspunkte und Prüfstrategien war ein wesentliches Kriterium für die gewählte Architektur und die Entscheidung gegen die Verwendung von herkömmlichen Algorithmen zur Erzeugung von Entscheidungs- oder Klassifikationsbäumen.

Für die Anwendung des oben skizzierten Algorithmus auf das Beispiel des Temperaturfühlers an wurde zur Vereinfachung vorgegeben, daß der Stecker des Temperaturfühlers unzugänglich ist und sich somit nur der Stecker des Steuergerätes öffnen läßt. Unter diesen Voraussetzungen wird folgende Prüfplan generiert, der aus Gründen der Verständlichkeit z.T. verbalisiert wurde.

```
<TestPlan>
  <TestStep ID=1>
    <SetupAction> SchlieÙe Steuergeräte-Tester an Motorsteuergerät an
    <Test> Lese digitales Steuergerätesignal für Ansauglufttemperatur aus
      <TestResult NextStep=9> Signal liegt zwischen -20°C und +40°C
      <TestResult NextStep=2> Signal liegt unter -20°C
```

```

        <TestResult NextStep=5> Signal liegt über +40°C
<TestStep ID=2>
    <SetupAction> Öffne Steuergeräte Stecker
    <SetupAction> Schalte Vielfachmessgerät auf Spannungsmessung
    <SetupAction> Schließe Vielfachmessgerät an Steuergerät an
    <Test> Lese digitales Steuergerätesignal für Ansauglufttemperatur aus
        <TestResult NextStep=3> Signal liegt zwischen -20°C und +40°C
        <TestResult NextStep=4> Signal liegt unter -20°C
<TestStep ID=3>
    <RepairAction> Tausche Temperaturfühler mit Zuleitungen aus
<TestStep ID=4>
    <RepairAction> Tausche Steuergerät aus
<TestStep ID=5>
    <SetupAction> Öffne Steuergeräte Stecker
    <SetupAction> Schalte Vielfachmessgerät auf Spannungsmessung
    <SetupAction> Schließe Vielfachmessgerät an Steuergerät an
    <Test> Lese digitales Steuergerätesignal für Ansauglufttemperatur aus
        <TestResult NextStep=6> Signal liegt über +40°C
        <TestResult NextStep=8> Signal liegt zwischen -20°C und +40°C
<TestStep ID=6>
    <RepairAction> Tausche Spannungsversorgung aus
    <Test> Lese digitales Steuergerätesignal für Ansauglufttemperatur aus
        <TestResult NextStep=9> Signal liegt zwischen -20°C und +40°C
        <TestResult NextStep=7> Signal liegt über +40°C
<TestStep ID=7>
    <RepairAction> Tausche Steuergerät aus
<TestStep ID=8>
    <RepairAction> Tausche Temperaturfühler mit Zuleitungen aus
<TestStep ID=9>
    <Comment> alles ok, kein Fehler gefunden

```

Dieser automatisch generierte Prüfplan beginnt, wie auch der von Menschen erstellte, mit dem Auslesen von Istwerten aus dem Steuergerät. Dieser Prüfschritt wird deshalb bevorzugt, weil er günstig auszuführen ist (wenig Vorbereitungsaktionen) und bereits im ersten Prüfschritt die am häufigsten anzutreffende Situation, nämlich die, in der kein Fehler vorliegt, von den bei weitem unwahrscheinlicheren Fehlersituationen abgrenzt. In der Regel kann man also ein Prüfergebnis zwischen -20°C und +40°C erwarten, welches auf korrektes Verhalten der Temperaturerfassung hinweist. Nach diesem Schritt bricht der Prüfplan ab.

Eine erste Überraschung tritt im Fehlerfall auf. Zwar wird erwartungsgemäß vorgeschlagen, den Steuergerätestecker zu öffnen und das Vielfachmeßgerät als Spannungsmesser anzuschließen. Abgelesen wird dann allerdings nicht das Meßgerät selbst, sondern wiederum die Anzeige des Steuergeräte-Testers! Diese Wahl ist zufällig, denn das Ablesen beider Meßgeräte bringt im vorliegenden Fall den gleichen Nutzen für die Diskriminierung und ist mit den gleichen assoziierten Kosten modelliert (wäre das Ablesen des Vielfachmeßgerätes gegenüber dem Ablesen des Steuergeräte-Testers als billiger modelliert, so würde ersteres bevorzugt). Man kann den vorgeschlagenen, unkonventionellen Test so interpretieren, daß das Vielfachmeßgerät als Prüfwiderstand fungiert, der in Ersatz für den potentiellen Fehlerkandidat Temperaturfühler eingesetzt wird. Liegt der ermittelte Digitalwert im gültigen Bereich, so ist offensichtlich der Sensor defekt und wird ausgetauscht. Ist andererseits der ermittelte Digitalwert auch bei der zweiten Messung im unerlaubten Bereich, so sind das Steuergerät und die Spannungsversorgung potentielle Fehlerkandidaten, letztere aber nur bei zu niedriger Spannung oder entsprechend zu hohem ermitteltem Digitalwert, da die Spannungsversorgung im Fehlerfall gemäß Modell nur zu niedriger Spannung, nicht jedoch zu hoher Spannung liefern kann. Bei einem Digitalwert über +40°C wird zuerst die Spannungsversorgung ausgetauscht, da diese zum einen eine höhere Fehlerwahrscheinlichkeit aufweist, zum anderen billiger zu ersetzen ist. Erst wenn der Fehler auch nach dieser Reparatur noch vorliegt, wird das Steuergerät ausgetauscht.

Wie der dargestellte Plan illustriert, erzeugt die automatische Prüfplangenerierung eine Struktur mit Markierungen, wie sie in der Repräsentation der textuellen Prüfpläne in der ersten Stufe von Genesis verwendet werden. Dies ist die Basis dafür, den generierten Handlungsplan automatisch ebenfalls in einen Text umzusetzen. Voraussetzung hierfür ist die Existenz von Textbausteinen für die verschiedenen Typen von Aktionen,

Messungen und Entscheidungen, deren sich die Planungskomponente bedient. Dies ist ein überschaubarer (und in der Basis für den Planer aufgezählter) Vorrat. Die variierenden Textbestandteile (Soll- und Istwerte, Namen von Komponenten und Meßgeräten etc.) sind bereits in der ersten Stufe von Genesis als Parameter in die Textvorlagen integrierbar.

Damit ist in diesem Anwendungsbereich erfolgreich an einem Prototypen demonstrierbar, wie modellbasierte Lösungen mit geringfügigen Änderungen im gegenwärtigen Arbeitsprozeß nutzbar gemacht werden können.

2.5 Erkenntnisse

Unsere Arbeit im INDIA-Projekt verfolgte die Zielsetzung, den Transfer von Forschungsergebnissen im Bereich modellbasierter Systeme in mehrere Anwendungen in der Fahrzeugindustrie einen wesentlichen Schritt vorwärts zu bringen. Auf dem Weg von der Demonstration prinzipieller Machbarkeit zum echten Einsatz sollten prototypische Lösungen für die Unterstützung vorhandener Arbeitsabläufe unter Berücksichtigung ihrer realistischen Randbedingungen erstellt werden.

Die geschilderten Prototypen haben dies (mit unterschiedlicher Reife) geleistet. An ihnen lässt sich das Potential der modellbasierten Technologie demonstrieren:

- Es ist mit den vorhandenen Mitteln möglich, einzelne **Arbeitsschritte wirksam zu unterstützen** und sogar zu automatisieren. Letzteres gilt insbesondere für den Anteil der Arbeit, der zwar Fachgebietenwissen erfordert, aber dieses in klar strukturierter Weise anwendet, wie etwa die Bestimmung der Fehlerauswirkung für eine Menge von Fehlerursachen.
- Die Kombination von kompositionaler Modellierung auf der Basis von Modellbibliotheken und generischen Problemlösungskomponenten stellt eine wirksame **Antwort auf das Problem der Variantenvielfalt** dar.
- Darüber hinaus stellt der Ansatz die entsprechenden Arbeitsprozesse auf eine **klare methodische Grundlage**, vor allem durch die Prinzipien und Organisation der Modelle und die Trennung zwischen Gegenstandswissen und Problemlösungswissen.
- Dabei lassen sich computergestützte Lösungen durch **Widerverwendung** sowohl von Modellen als auch von Softwarebausteinen z.T. sehr effizient produzieren.

Die demonstrierbaren Möglichkeiten der Technologie beziehen sich aber nicht nur auf die Unterstützung einzelner Arbeitsprozesse:

- Modellbibliotheken, aber auch die klaren Schnittstellen zwischen Softwarekomponenten stellen die Basis für eine **Integration von Arbeitsprozessen** und den elektronischen Austausch von Information und Ergebnissen von Arbeitsschritten dar. Die entwickelten Modellbibliotheken wurden, ebenso wie die Softwarekernkomponenten in allen drei Prototypen eingesetzt.
- Noch allgemeiner gesehen, sind die Erstellung und Pflege von Modellbibliotheken als ein Beitrag zum **Wissensmanagement** in technologie-orientierten Unternehmen zu sehen.

Natürlich und durchaus beabsichtigt hat die Arbeit auch Grenzen der gegenwärtigen Technologie und Aufgaben künftiger Forschungs- und Entwicklungsarbeiten aufgedeckt oder erhellt. Zu den wichtigsten dieser Aufgabenkomplexe gehören:

- Besser **Unterstützung und Automatisierung der Modellierung**. Dies beinhaltet insbesondere die Nutzung der im Ingenieurbereich entwickelten Modelle und die Erzeugung von Modellen, die in Effizienz und Kompetenz für die jeweiligen Systeme und Aufgaben maßgeschneidert sind.
- Systematische **Charakterisierung der Voraussetzungen und Ergebnisse** der vorhandenen Problemlösungen, vor allem im Hinblick auf die Behandlung dynamischer Abläufe und die Verwendung unvollständiger Lösungsalgorithmen (s. die Ausführungen zur zustandsbasierten Diagnose).
- Verbesserte theoretische Grundlagen und technische Lösungen, die die komplexe Natur von Arbeitsprozessen als **Handlungsabläufe** widerspiegeln und deren praktische Randbedingungen und Kosten systematisch berücksichtigen.

Das Projekt hat gezeigt, dass trotz dieser notwendigen Forschungsarbeiten vorhandene Lösungen in die industrielle Praxis überführt werden können. Dies erfordert geeignete Strategien, um die Einführung in die Arbeitsprozesse und deren Umgestaltung behutsam aber wirksam vorzunehmen. Einige Ansätze zur Lösung dieses Problems werden durch die geschilderten Arbeiten aufgewiesen.