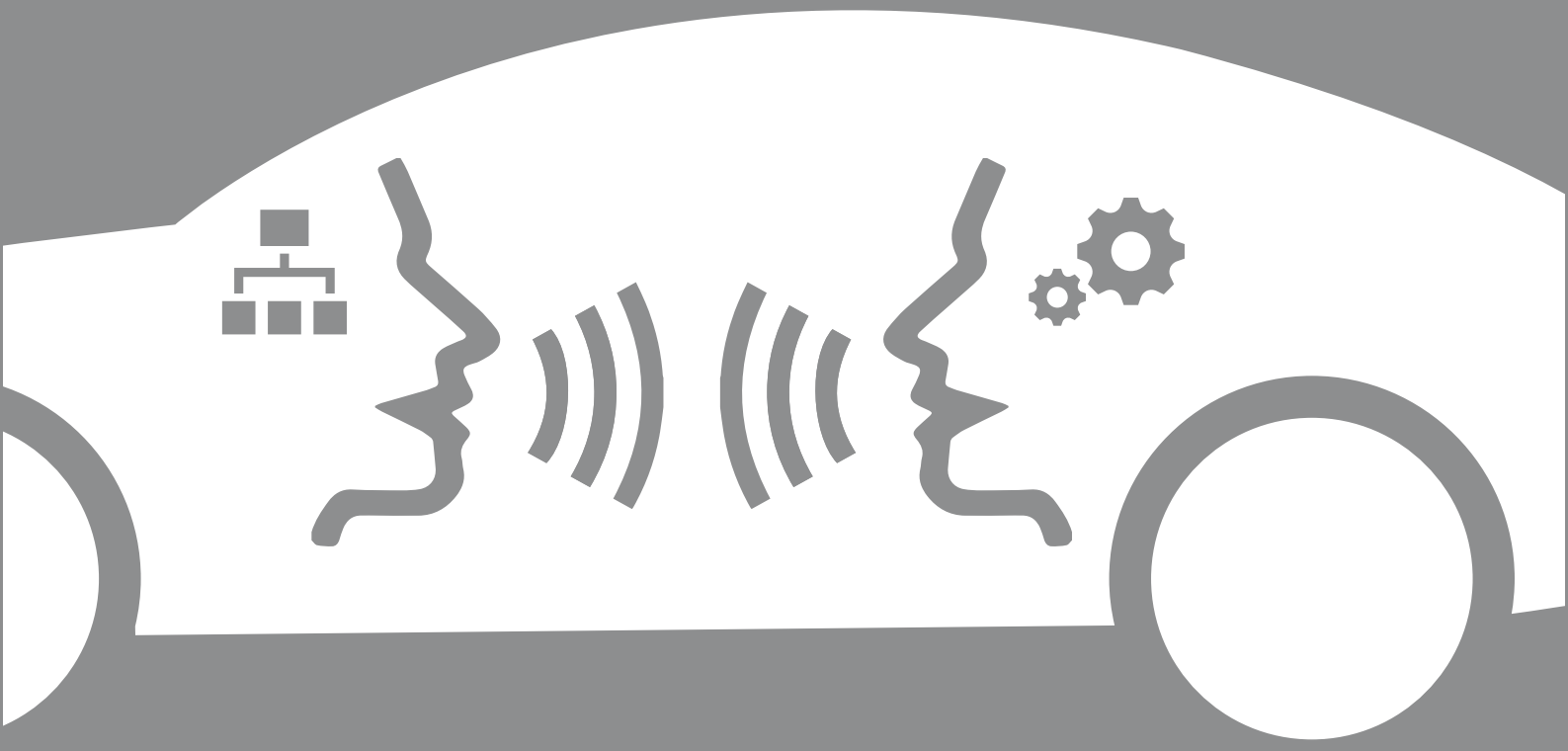
18

SCHRIFTENREIHE ZU ARBEITSWISSENSCHAFT UND TECHNOLOGIEMANAGEMENT

Lars Lütze

Modellbasierter Testprozess der akustischen Mensch-Maschine-Schnittstelle eines Infotainmentsystems im Kraftfahrzeug



Fraunhofer IAO

Universität Stuttgart
Institut für Arbeitswissenschaft und Technologiemanagement IAT

SCHRIFTENREIHE ZU ARBEITSWISSENSCHAFT UND TECHNOLOGIEMANAGEMENT

Herausgeber

Univ.-Prof. Dr.-Ing. Dr.-Ing. E. h. Dr. h. c. Dieter Spath
Univ.-Prof. Dr.-Ing. habil. Prof. e. h. mult. Dr. h. c. mult. Hans-Jörg Bullinger

Institut für Arbeitswissenschaft und Technologiemanagement IAT
der Universität Stuttgart, Stuttgart
Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO, Stuttgart

Band 18

Lars Lütze

Modellbasierter Testprozess der akustischen Mensch-Maschine-Schnittstelle eines Infotainmentsystems im Kraftfahrzeug

FRAUNHOFER VERLAG

Impressum

Kontaktadresse:
Institut für Arbeitswissenschaft
und Technologiemanagement IAT
der Universität Stuttgart und
Fraunhofer-Institut für Arbeitswirtschaft
und Organisation IAO
Nobelstraße 12, 70569 Stuttgart
Telefon +49 711 970-01, Fax -2299
www.iat.uni-stuttgart.de
www.iao.fraunhofer.de

Schriftenreihe zu Arbeitswissenschaft
und Technologiemanagement

Herausgeber:
Univ. Prof. Dr.-Ing Dr.-Ing. E.h. Dr. h. c. Dieter Spath
Univ. Prof. Dr.-Ing. habil. Prof. e.h. mult.
Dr. h.c. mult. Hans-Jörg Bullinger

Institut für Arbeitswissenschaft
und Technologiemanagement IAT
der Universität Stuttgart und
Fraunhofer-Institut für Arbeitswirtschaft
und Organisation IAO

Bibliografische Information der
Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über www.dnb.de abrufbar.

ISSN 2195-3414
ISBN 978-3-8396-0837-1

D 93
Zugl.: Stuttgart, Univ., Diss., 2014

Druck und Weiterverarbeitung:
IRB Mediendienstleistungen
Fraunhofer-Informationszentrum
Raum und Bau IRB, Stuttgart

Für den Druck des Buches wurde chlor- und säurefreies Papier verwendet.

Fraunhofer-Informationszentrum
Raum und Bau IRB
Postfach 800469, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon +49 711 970-2500, Fax -2508
E-Mail verlag@fraunhofer.de
http://verlag.fraunhofer.de

Geleitwort

Grundlage der Arbeiten am Institut für Arbeitswissenschaft und Technologiemanagement IAT der Universität Stuttgart und am kooperierenden Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO ist die Überzeugung, dass unternehmerischer Erfolg in Zeiten globalen Wettbewerbs vor allem bedeutet, neue technologische Potenziale nutzbringend einzusetzen. Deren erfolgreicher Einsatz wird vor allem durch die Fähigkeit bestimmt, kunden- und mitarbeiterorientiert Technologien schneller als die Mitbewerber zu entwickeln und anzuwenden. Dabei müssen gleichzeitig innovative und anthropozentrische Konzepte der Arbeitsorganisation zum Einsatz kommen. Die systematische Gestaltung wird also erst durch die Bündelung von Management- und Technologiekompetenz ermöglicht. Dabei wird durch eine ganzheitliche Betrachtung der Forschungs- und Entwicklungsthemen gewährleistet, dass wirtschaftlicher Erfolg, Mitarbeiterinteressen und gesellschaftliche Auswirkungen immer gleichwertig berücksichtigt werden.

Die im Rahmen der Forschungsarbeiten an den Instituten entstandenen Dissertationen werden in der »Schriftenreihe zu Arbeitswissenschaft und Technologiemanagement« veröffentlicht. Die Schriftenreihe ersetzt die Reihe »IPA-IAO Forschung und Praxis«, herausgegeben von H. J. Warnecke, H.-J. Bullinger, E. Westkämper und D. Spath. In dieser Reihe sind in den vergangenen Jahren über 500 Dissertationen erschienen. Die Herausgeber wünschen den Autoren, dass ihre Dissertationen aus den Bereichen Arbeitswissenschaft und Technologiemanagement in der breiten Fachwelt als wichtige und maßgebliche Beiträge wahrgenommen werden und so den Wissensstand auf ein neues Niveau heben.

Dieter Spath

Hans-Jörg Bullinger

Vorwort

Mit Fertigstellung der vorliegenden Arbeit gebührt es sich, den daran beteiligten Personen zu danken.

An erster Stelle möchte ich Prof. Dr.-Ing. Dr.-Ing. E.h. Dr. h.c. Dieter Spath als Doktorvater und ehemaligem Leiter des Fraunhofer-Instituts für Arbeitswirtschaft und Organisation IAO und des Instituts für Arbeitswissenschaft und Technologiemanagement IAT der Universität Stuttgart danken. Herrn Prof. Dr.-Ing. Thomas Maier danke ich herzlich für die Übernahme des Mitberichts.

Besonderer Dank gilt Herrn Dr. Metsch, Herrn Dr. Blossey und Herrn Peter Häußermann der Daimler AG. Sie haben diese Arbeit ermöglicht.

Vielen Dank an Herrn Dr. Manfred Dangelmaier (IAO) und Herrn Dr. Steffen Werner (Daimler AG) für die konstruktive Unterstützung und wissenschaftliche Betreuung der Arbeit, welche nicht besser hätte sein können.

Dank gilt auch vielen Kollegen/-innen der Daimler AG, insbesondere Herrn Thomas Albrecht und dessen Team. Viele haben diese Arbeit durch die Weitergabe ihrer Erfahrungen und ihres Wissens direkt oder indirekt beeinflusst. Besonderes danken möchte ich an dieser Stelle Herrn Andreas Dorawa. Durch ihn und seine Kollegen wurde die schnelle technische Umsetzung entwickelter Konzepte möglich.

Die Erstellung einer Dissertation erfordert viel Zeit. Zu danken ist aus diesem Grund auch meiner Familie und meiner Freundin Ulrike, die mir nicht nur die notwendige Zeit eingeräumt haben, sondern auch als unermüdliche Motivatoren und Lektoren auffällig wurden.

Es besteht die Erkenntnis, dass die hier bedankte Unterstützung als Notwendigkeit für den Erfolg der Arbeit bewertet werden darf.

Esslingen am Neckar, Dezember 2014
Lars Lütze

Modellbasierter Testprozess der akustischen Mensch-Maschine-Schnittstelle eines Infotainmentsystems im Kraftfahrzeug

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik
der Universität Stuttgart
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Dipl.-Ing. Lars Lütze
aus Stuttgart-Bad Cannstatt

Hauptberichter: Prof. Dr.-Ing. Dr.-Ing. E.h. Dr. h.c. Dieter Spath
Mitberichter: Prof. Dr.-Ing. Thomas Maier

Tag der mündlichen Prüfung: 31.10.2014

Institut für Arbeitswissenschaft und Technologiemanagement (IAT)
der Universität Stuttgart

2014

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

ASR	Spracherkenner (engl. Automated Speech Recognizer)
AVS	Adaptive Variable Stellteile
CAN-Bus	Controller Area Network Bussystem
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CTP	Critical Testing Process
DM	Dialogmanager
DTW	Dynamic-Time-Warping
HU	Head Unit - Haupteinheit des Infotainmentsystems
ISTQB ®	International Software Testing Qualifications Board
Kfz	Kraftfahrzeug
KI	Kombiinstrument
MBT	Modellbasierter Test/ Modellbasiertes Testen
MFL	Multifunktionslenkrad(-Tasten)
MMS	Mensch-Maschine-Schnittstelle
MOST-Bus	Media Oriented Systems Transport Bussystem
OCR	Optical Character Recognition
PIN	Persönliche Identifikationsnummer
PTA	Push-To-Activate
PTE	Push-To-End
PTT	Push-To-Talk
SDS	Sprachdialogsysteme
SoP	Start of Production
SPICE	Software Process Improvement and Capability Determination

STEP	Systematic Test and Evaluation Process
SUT	System Under Test (Das zu testende System)
SW	Software
Tmap	Test Management approach
TMM	Testing Maturity Model
TMM-AM	TMM-Assessment Modell
TPI	Test Process Improvement
TQM	Total Quality Management
TS	Testschritt (engl. Test step)
TTS	Text-to-Speech
UML	Unified Modeling Languange
VP	Verifikationspunkt (engl. Verification point)
XMI	XML Metadata Interchange
XML	Extensible Markup Language
ZBE	Zentrale Bedien Einheit (Dreh-, Drück-, Schiebesteller)

1 Einleitung

Die fortschreitende Entwicklung der Informationstechnologien veränderte das Leben und die Gesellschaft in den vergangenen Jahrzehnten ebenso nachhaltig wie die Erfindung des Automobils. Mobile Kommunikation, weltweite Vernetzung, satellitengestützte Positionsbestimmung und multimediale Möglichkeiten sind heute feste Bestandteile unseres Lebens. 125 Jahre nach der Erfindung des Automobils wächst die Fahrzeugtechnik mit den sich entwickelnden Informationstechnologien zusammen und konfrontiert die Automobilhersteller mit neuen Herausforderungen. Von besonderer Bedeutung ist hierbei die Gestaltung der Fahrzeugergonomie im Kontext der neuen, softwarebasierten Funktionen. Die Ergonomie des Fahrerplatzes stellt ohnehin eine besondere Herausforderung dar, schließlich hat der Fahrzeug-führer seine Aufmerksamkeit primär den Steuer- und Regelaufgaben zu widmen [Dan00]. Es ergibt sich die Notwendigkeit einer besonders sorg-fältigen, ergonomischen Gestaltung aller Bedienelemente und Funktionen.

Heute finden sich in fast allen Fahrzeugmodellen Bordcomputer und Infotainmentsysteme die dem Fahrer eine Vielzahl an Funktionalitäten zur Verfügung stellen. Dazu gehören:

- Kommunikation durch Freisprechtelefonie mit SMS-Funktion und Adressbuch.
- Navigation mit individualisierbarer Routenberechnung unter Berücksichtigung aktueller Verkehrsinformationen, sowie umfangreichen Informationen zur Verkehrsinfrastruktur und touristischen oder kulturellen Zielen.

- Unterhaltung durch Radio und Multimedia (z. B. Audio CD, Audio/Video DVD, personalisierte digitale Speichermedien und TV).
- Bedienung von Komfortfunktionen und Systemeinstellungen des Fahrzeugs wie der Klimaanlage, ambienter Lichtverteilung und Sitzeinstellungen.
- Zugriff auf E-Mail und Internetinhalte.

Abbildung 1 veranschaulicht die Entwicklung der dem Fahrer zur Verfügung gestellten Funktionalitäten mit dem Einzug moderner Infotainmentsysteme.

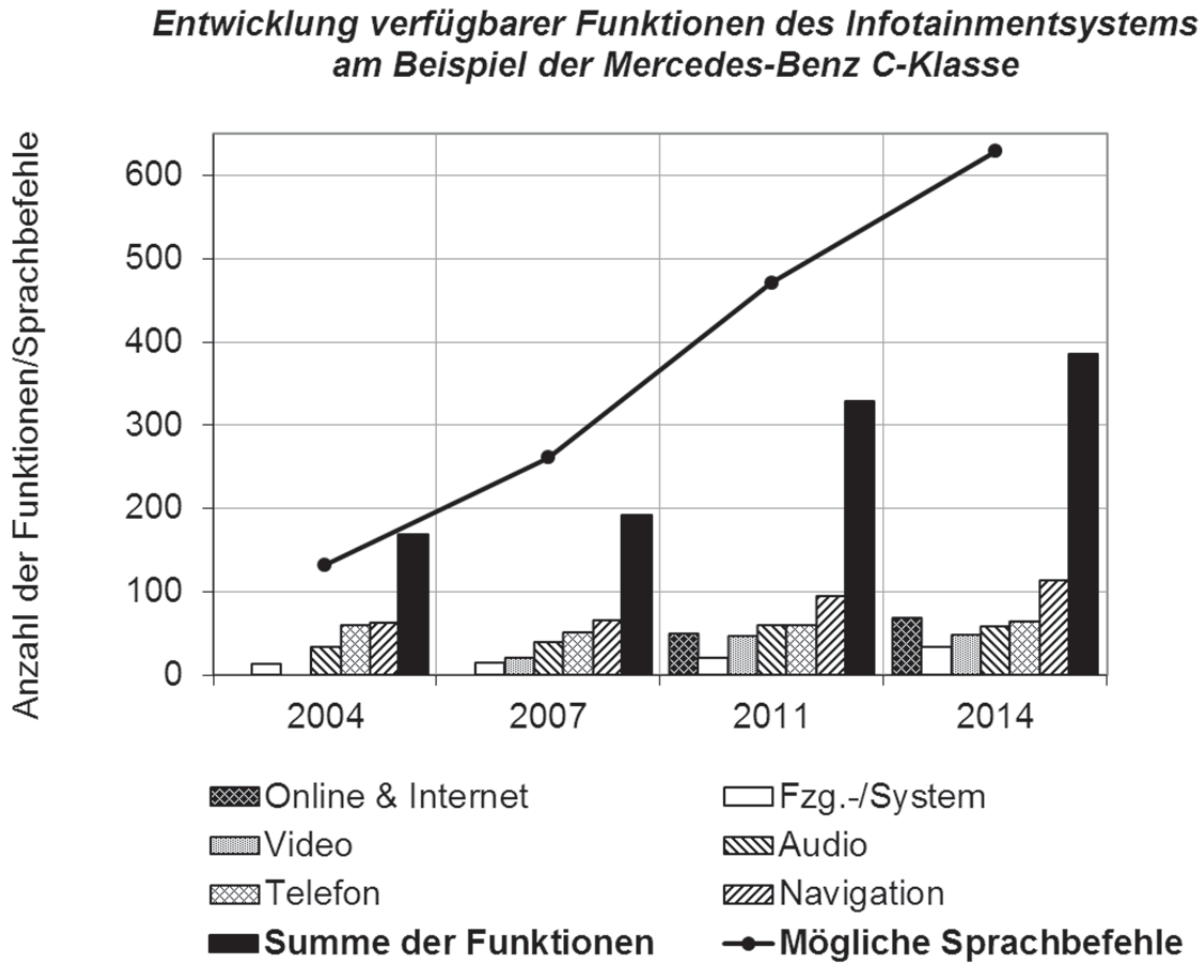


Abbildung 1: Entwicklung der Funktionalitäten in einem Infotainmentsystem

Um die Ablenkung des Fahrers durch die im Infotainmentsystem zusammengefassten Telematik- und Multimediafunktionen zu regulieren, gibt es strenge ergonomische Vorgaben [DIN15008] [DIN9241].

Der Umgang des Menschen mit dem Automobil basiert seit seiner Erfindung auf haptischer und visueller Interaktion. Die Entwicklungen in der Informatik ermöglichen es uns heute, eine weitere, natürlichere Form der Interaktion hinzuzufügen: die Interaktion durch Sprache.

Automatisierte Spracherkennung, die Ausführung komplexer Dialogabläufe und die Generierung von Sprachausgaben ermöglichen eine sinnfällige und natürliche Interaktion zwischen Mensch und Maschine. Die Ablenkung des Fahrers lässt sich auf ein Minimum reduzieren, bei gleichzeitiger Erhöhung des Komfort und der Sicherheit. Dabei sind die an die Sprachbedienung gestellten Anforderungen ebenso hoch wie jene an die gesamte Fahrzeugergonomie.

Durch die ansteigende Dichte der zu bedienenden Funktionen rückt in der Entwicklung neben den konzeptionellen Herausforderungen die Problematik der funktionalen Qualitätsabsicherung in den Vordergrund. Kurze Entwicklungszyklen, hohe Variantenvielfalt und globale Märkte mit unterschiedlichen Sprachen verschärfen die Herausforderungen in der Entwicklung einer fehlerfreien Mensch-Maschine-Schnittstelle.

Um die erforderliche funktionale Absicherung in diesem Kontext sicherzustellen, sind erweiterte Kompetenzen aus dem Bereich der Softwareentwicklung und des Softwaretests notwendig. Kompetenzen, welche in der Fahrzeugtechnik und im klassischen Maschinenbau bislang kaum von

Bedeutung waren. Die Übernahme und Entwicklung neuer Prozesse und Methoden ist notwendig, um die funktionale Qualität der Mensch-Maschine-Schnittstelle (MMS) im Spannungsfeld begrenzter Ressourcen und wachsender Anforderungen für die Zukunft sicherzustellen.

Im Rahmen dieser Arbeit soll gezeigt werden, dass der modellbasierte Testprozess das Potenzial besitzt, die beschriebenen Herausforderungen zu lösen. Durch dessen Implementierung kann ein virtueller Tester geschaffen werden, welcher in der Lage ist den Menschen im Test der akustischen MMS zu ersetzen und die funktionale Absicherung der akustischen MMS durchzuführen.

Nachdem in Kapitel 2 die Grundlagen zu relevanten Testprozessen und Testinhalten eingeführt sind, wird in Kapitel 3 der Ansatz zur Lösung beschriebener Herausforderungen ausgearbeitet. Kapitel 4 bis 8 zeigen durch Nachweis der eingeführten Thesen die Möglichkeit auf, eine akustische MMS im Kfz modellbasiert und automatisiert zu testen, um die beschriebenen Herausforderungen zu lösen.

2 Stand der Technik

2.1 Die Mensch-Maschine-Schnittstelle im Kraftfahrzeug

2.1.1 Menschliche Informationsverarbeitung

Im Jahr 1985 stellte der japanische Automobilhersteller Subaru sein erstes Sportcoupé vor. Das Fahrzeug war in Summe eher durchschnittlich, dank eines Details jedoch einzigartig. Der Subaru XT verfügt über einen Allradantrieb, zuschaltbar durch die Bedienung der Bremse und paralleler Betätigung des Scheibenwischers. [Sub10]

Derartige Bedienkonzepte für Fahrzeugfunktionen sind in der Entwicklungsgeschichte des Automobils eher die Ausnahme. Seit dessen Erfindung wird der Interaktion zwischen Mensch und Automobil eine zunehmende Aufmerksamkeit zuteil. Um eine möglichst effektive, sichere und wirkungsvolle Interaktion zu ermöglichen, wird diese in verschiedenen Ansätzen beschrieben. Die Kenntnisse physiologischer und psychologischer Fähigkeiten des Menschen sind dabei von zentraler Bedeutung.

Die Lehre der Anthropometrie führte schon im frühen zwanzigsten Jahrhundert zur Entwicklung zunehmend ergonomischer Mensch-Maschine-Schnittstellen [Rüh90]. Die Maße und Physiologie des menschlichen Körpers rückten dabei in den Mittelpunkt der Entwicklung. Die Gestaltung des Automobils wurde hinsichtlich seiner Abmessungen, Anzeigen und Bedienelemente den neuen wissenschaftlichen Erkenntnissen angepasst.

Die Entwicklung von Computersystemen und deren fortschreitende Durchdringung aller Lebensbereiche hatte auch Folgen für die vom Mensch zu bedienenden Maschinen [Joh93]. Die zunehmende Komplexität der Bedienung machte es notwendig, insbesondere die psychologischen Aspekte des Schnittstellendesigns zu erforschen und zu berücksichtigen. Das Erkennen und die Erkenntnis des Menschen im Umgang mit technischen Systemen beschäftigen die kognitive Psychologie. Die Forschungsarbeiten dieses Wissenschaftsfelds zeigen, dass der Mensch kein passiver Empfänger von Reizen und Informationen ist, sondern ein aktives, dynamisches Element, das die auf ihn einströmenden Informationen mit Flexibilität aufsucht, verarbeitet und umsetzt [Lac79] [Pal86].

Nach [Ras86] ist beim Mensch die bewusste, kontrollierte Informationsverarbeitung von der unbewussten Informationsverarbeitung zu unterscheiden. Das unbewusste System kann als verteiltes Verarbeitungssystem mit hoher Verarbeitungskapazität verstanden werden. Es leitet unsere Wahrnehmung und ermöglicht dem Menschen, auf wahrgenommene Veränderungen im Umfeld durch entsprechende Aktivitäten zu reagieren. Die bewusste Informationsverarbeitung kann als parallel arbeitendes System verstanden werden. Dieses System unterliegt jedoch Grenzen. Die Kapazität und Schnelligkeit der überwiegend sequenziell durchgeführten Verarbeitung ist eingeschränkt. Es wird zur Improvisation oder im logischen Denken verwendet. Die bewusste Informationsverarbeitung beruht auf der Erfahrung des semantischen Gehalts aufgenommener Information. Zur Verarbeitung anstehende Informationen bleiben im Bewusstsein für maximal 10 Sekunden erhalten [Mai12].

Von besonderem Augenmerk sollen an dieser Stelle die Grenzen des menschlichen Informationsverarbeitungssystems sein. Eine bewusste Informationsverarbeitung kann nur dann stattfinden, wenn ausreichend kognitive Kapazität zur Verfügung steht. Erreicht das Reizniveau aus der Umgebung des Menschen eine individuell kritische Schwelle, können die Reize aus der Umgebung zwar noch unbewusst wahrgenommen und bearbeitet werden, vom Bewusstsein aber nur sequenziell verarbeitet werden. Dies birgt besonders in zeitkritischen Situationen die Gefahr, dass Umgebungsreize im Rahmen der verfügbaren Zeit nicht bewusst verarbeitet werden. Der Mensch handelt folglich unbewusst und reflexartig. Ein Ergebnis, das insbesondere im Straßenverkehr von hohem Risiko und daher zu vermeiden ist.

Die Erkenntnisse der Wissenschaft zeigen die Notwendigkeit einer MMS, welche die kognitiven Fähigkeiten des Menschen berücksichtigt. Im Straßenverkehr muss der Fahrzeuglenker, trotz aller Umgebungseinflüsse und Reize, immer über ausreichend kognitive Kapazität verfügen, um notwendige Steuer- und Regelungsaufgaben bewusst durchführen zu können. Möglichst alle Aufgaben und Bedienaktivitäten im Kraftfahrzeug sind dementsprechend so zu konzipieren, dass die bewusste Informationsverarbeitung in der kürzest möglichen Zeit abgeschlossen ist oder jederzeit abgebrochen werden kann und die kognitive Kapazität für folgende oder priorisierte Aktivitäten zur Verfügung steht.

Moderne Infotainmentsysteme bergen durch die Vielzahl der zur Verfügung stehenden Informationen und Funktionen die Gefahr, die kognitive Kapazität des Fahrers zu binden. Um dies zu vermeiden, werden in der Entwicklung

derartiger Systeme große Anstrengungen zur Maximierung der Qualität und Gebrauchstauglichkeit der MMS unternommen.

In diesem Zusammenhang gewinnt die aufgabenorientierte System-entwicklung an Bedeutung. Ziel dieser ist eine MMS, welche den Menschen bei der Erledigung seiner Aufgaben bestmöglich unterstützt. Die Bedienung der Maschine soll sinnfällig gestaltet und die Gebrauchstauglichkeit (engl. Usability) maximiert werden. Die internationale Norm EN ISO 9241-11 definiert Gebrauchstauglichkeit als „das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskonzept genutzt werden kann, um bestimmte Ziele effektiv, effizient und mit Zufriedenheit zu erreichen“ [DIN9241]. Die Norm besteht aus 26 Teilen und beschreibt Grundsätze zur Gestaltung einer ergonomischen Mensch-System-Interaktion (siehe auch Kapitel 2.1.2).

2.1.2 Die MMS von Infotainmentsystemen

Das Infotainmentsystem im Kraftfahrzeug besteht aus der „Head Unit“ (HU), dem zentralen Rechner des Infotainmentsystems, sowie den Anzeige- und Bedien-elementen. Aktuell lassen sich zwei Bedienkonzepte unterscheiden. Konzepte mit berührungsempfindlichen Anzeigeelementen und Konzepte mit „Adaptiven Variablen Stellteilen“ (AVS) wie haptischen Dreh-, Drück- und Schiebestellern. Abbildung 2 zeigt beispielhaft die Elemente eines Infotain-mentsystems mit AVS.

Seit den neunziger Jahren des 20. Jahrhunderts werden Infotainmentsysteme im Kfz verbaut. Infotainment, zusammengesetzt aus „Information“ und

„Entertainment“, beschreibt ein System von Komponenten, welches dem Fahrer unterschiedliche Funktionalitäten zur Verfügung stellt.

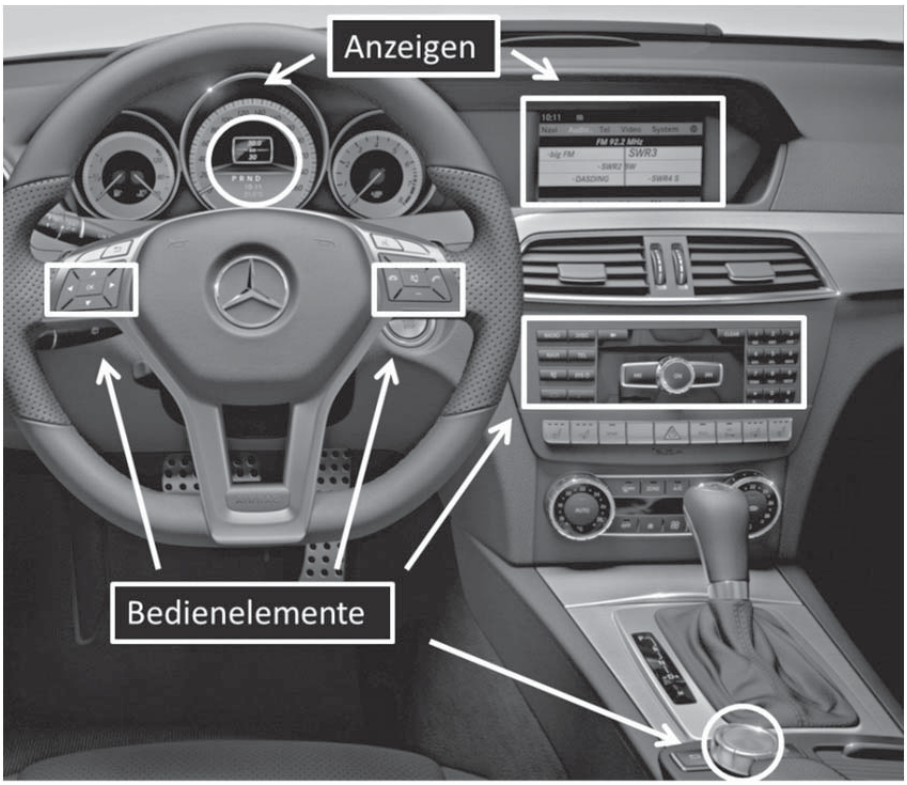


Abbildung 2: Elemente eines Infotainmentsystems im Kfz, MB C-Klasse

Abbildung 1 (S. 23) veranschaulicht die Entwicklung der dem Fahrer zur Verfügung gestellten Funktionen in einem Infotainmentsystem. Hierfür wurden die seit 2004 erhältlichen Systeme der C-Klasse Baureihe von Mercedes-Benz analysiert. Zu den bereits anfänglich vorhandenen Umfängen der Navigation, Telefonie, Audio und Systemfunktionen kamen später Video- und Onlinefunktionalitäten hinzu. Gleichzeitig vergrößerte sich der Funktionsumfang bereits vorhandener Applikationen über die Systemgenerationen hinweg. Es lässt sich feststellen, dass sich die Summe aller bereitgestellten Funktionen in den letzten sieben Jahren mehr als verdoppelt hat.

Die Entwicklung der MMS sieht sich der enormen Herausforderung gegenübergestellt, die Gebrauchstauglichkeit des Systems trotz der immer weiter ansteigenden Funktionsumfänge zu gewährleisten. Dabei ist davon auszugehen, dass sich der Trend zu einer noch höheren Funktionsdichte fortsetzt. Durch den Ausbau von Hochgeschwindigkeitsnetzen wie LTE (ca. 100 Mbit/s) steigt die verfügbare mobile Datenrate um mehr als das Fünfzigfache gegenüber UMTS (ca. 2000 kbit/s). Auch große Datenmengen werden mobil verfügbar und die flächendeckende Nutzung der Inhalte des World Wide Web wird möglich. Insbesondere Online-Funktionalitäten werden in den nächsten Generationen der Infotainmentsysteme an Dichte und Umfang gewinnen.

Möchte der Fahrer all die ihm zur Verfügung gestellten Funktionen nutzen, besteht die reale Gefahr, dass umfangreiche kognitive Kapazitäten durch die Möglichkeiten des Infotainment gebunden werden und die eigentlichen Fahraufgaben nicht primär behandelt werden können. Da die Verkehrssicherheit oberste Priorität hat, gibt es international verschiedene Richtlinien, Normen und Gesetze, welche die Bedienung angebotener Funktionalitäten während des Fahrens teilweise stark einschränken. In der DIN EN ISO 15008 (2003) werden ergonomische Aspekte von Fahrerinformations- und Assistenzsystemen für Straßenfahrzeuge festgelegt. Die Norm definiert Richtlinien der Displaygestaltung, für Schriftgröße, Zeichenformat, Wort- und Zeilenabstände. Hinzu kommen Richtlinien für den Farbabstand und den Kontrast von Vorder- und Hintergrundfarben [DIN15008]. Die Norm EN ISO 9241-110 bezieht sich auf die Gestaltung interaktiver Systeme und ist folglich auch für Infotainmentsysteme relevant. Sie formuliert Grundsätze für die Gestaltung

und Bewertung von Dialogsystemen. In [DIN9241] sind folgende Grundsätze definiert:

Aufgabenangemessenheit: „Ein Dialog ist aufgabenangemessen, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen.“ [DIN9241]. Die Effizienz schließt die Berücksichtigung der notwendigen Zeit zur Erledigung ein.

Selbstbeschreibungsfähigkeit: „Ein Dialog ist selbstbeschreibungsfähig, wenn jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer auf Anfrage erklärt wird.“ [DIN9241]. Notwendige Elemente und Inhalte zur Generierung einer intuitiven Bedienung stehen im Fokus.

Steuerbarkeit: „Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie eine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist.“ [DIN9241]. (z. B. die Möglichkeit zum Abbruch des Dialogs).

Erwartungskonformität: „Ein Dialog ist erwartungsgemäß, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, zum Beispiel den Kenntnissen aus dem Arbeitsgebiet, der Ausbildung und Erfahrung des Benutzers sowie den allgemein anerkannten Konventionen.“ [DIN9241]. Auf diese Weise sollen z. B. Erfahrungen, die ein Nutzer bei ähnlichen Systemen erlangt hat, bestätigt und genutzt werden.

Fehlertoleranz: „Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingabe entweder mit keinem oder mit

minimalem Korrekturaufwand durch den Benutzer erreicht werden kann.“ [DIN9241].

Individualisierbarkeit: „Ein Dialog ist individualisierbar, wenn das Dialogsystem Anpassungen an die Erfordernisse der Arbeitsaufgabe, individuelle Vorlieben des Benutzers und Benutzerfähigkeiten zulässt“ [DIN9241]. Zum Beispiel die individuell einstellbare Ausrichtung von Navigationskarten (Nord-/ Fahrtrichtung).

Lernförderlichkeit: „Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen des Dialogsystems unterstützt und anleitet.“ [DIN9241].

An den Auszügen wird bereits deutlich, dass die Norm inhaltlich vage bleibt. Freiheitsgrade für die Entwicklung und Umsetzung neuer Konzepte und Innovationen bleiben somit bestehen.

Im letzten Jahrzehnt hat sich neben den klassischen Bedien- und Anzeigeelementen eine weitere Schnittstelle zwischen Mensch und Maschine etabliert. Durch Fortschritte in der Informatik, Signalverarbeitung und Softwaretechnik sind Sprachdialogsysteme entwickelt worden, die eine auditive Interaktion ermöglichen. Insbesondere im Automobil ist diese, ergänzende Bedienungsmodalität von großem Interesse.

Eine multimodale MMS nutzt mehrere Wahrnehmungskanäle des Menschen (Abbildung 3). In Versuchen konnte nachgewiesen werden, dass die Fehlerrate in der Bedienung multimodaler Systeme bei sich ergänzenden oder redundanten Reizen über mehrere Sinneskanäle deutlich geringer ausfällt als in Systemen mit nur einer Bedienungsmodalität [Mai12].

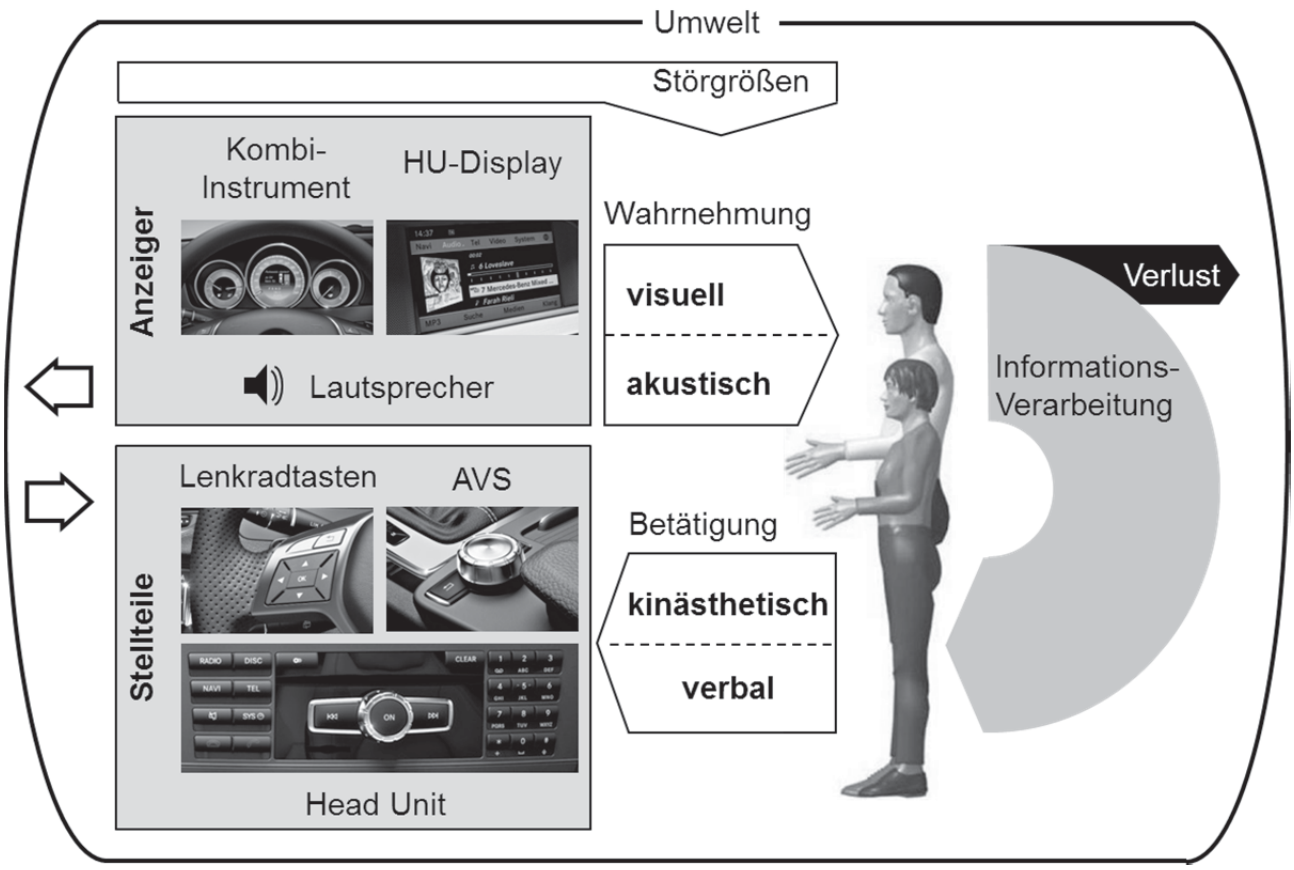


Abbildung 3: Schema multimodaler Informationsverarbeitung, nach [Mai12]

[Boe01] vergleicht in einer Studie, welchen Einfluss Aufgaben, mit haptisch/ visueller Beanspruchung, im Vergleich zu Aufgaben mit sprachlich auditiver Beanspruchung auf das Lenkverhalten von Probanden haben. Dabei konnte gezeigt werden, dass Bedienaktivitäten, welche haptisch/visuelle Ressourcen beanspruchen, in 90 % zu einem Anstieg unvorhersehbarer und ineffizienter Lenkaktivitäten führten. Die Reaktionszeiten stiegen gleichzeitig um 74 %. Bei der Nutzung sprachlicher Ressourcen wurde ein Anstieg ineffizienter Lenkaktivitäten von nur 10 % festgestellt. Die Reaktionszeit verlängert sich um 26 %.

Es lässt sich schlussfolgern, dass durch die Integration eines Sprachdialogsystems als ergänzende Bedienungsmodalität im Automobil die Bediensicherheit weiter erhöht werden kann. Kognitive Kapazitäten bleiben für die primären Fahraufgaben verfügbar.

Im folgenden Kapitel werden die Grundlagen der akustischen MMS im Automobil ausführlicher dargestellt.

2.1.3 Die Sprachbedienung

Nach Hamerich sind Sprachdialogsysteme sprachverstehende Systeme, welche „dem Erreichen von Zielen im gegenseitigen kooperativen Dialog“ dienen [Ham09]. Hierfür muss das System „Benutzeräußerungen adäquat verarbeiten und darauf sinnvoll reagieren können.“ Sprachdialogsysteme (SDS) differenzieren sich von anderen sprachverstehenden Systemen, indem sie durch an den Benutzer gerichtete Fragen mit selbigem interagieren und nicht nur gesprochene Kommandos entgegennehmen. Wird ein Sprachdialogsystem zur Steuerung von Geräten per Sprache genutzt, wie beispielsweise im Automobil, werden diese auch als Sprachbediensystem bezeichnet [Ham09]. Die Begriffe werden im Folgenden synonym verwendet.

In Abbildung 4 ist der schematische Aufbau eines SDS dargestellt. Die Schnittstelle zum Menschen ist einerseits durch ein Mikrofon realisiert, welches die akustische Spracheingabe des Nutzers erfasst, andererseits durch Lautsprecher zur akustischen Weitergabe der Systemausgaben.

Im Automobil ist mindestens eines, in der Regel jedoch mehrere Mikrofone verbaut, z. B. im Dachhimmel, der Dachbedieneinheit, der A-Säule oder im

Rückspiegel. Nachdem ein Sprachsignal das Sprachdialogsystem erreicht hat, wird zunächst eine Vorverarbeitung des Signals durchgeführt, um dessen Qualität zu maximieren. Ziel des Spracherkenners (engl. Automated Speech Recognitizer, ASR) ist, die vom Benutzer gesprochenen Worte über die Anwendung verschiedener Methoden zu erfassen und zu interpretieren, welche Zeichen, Wörter oder Sätze eingesprochen wurden.

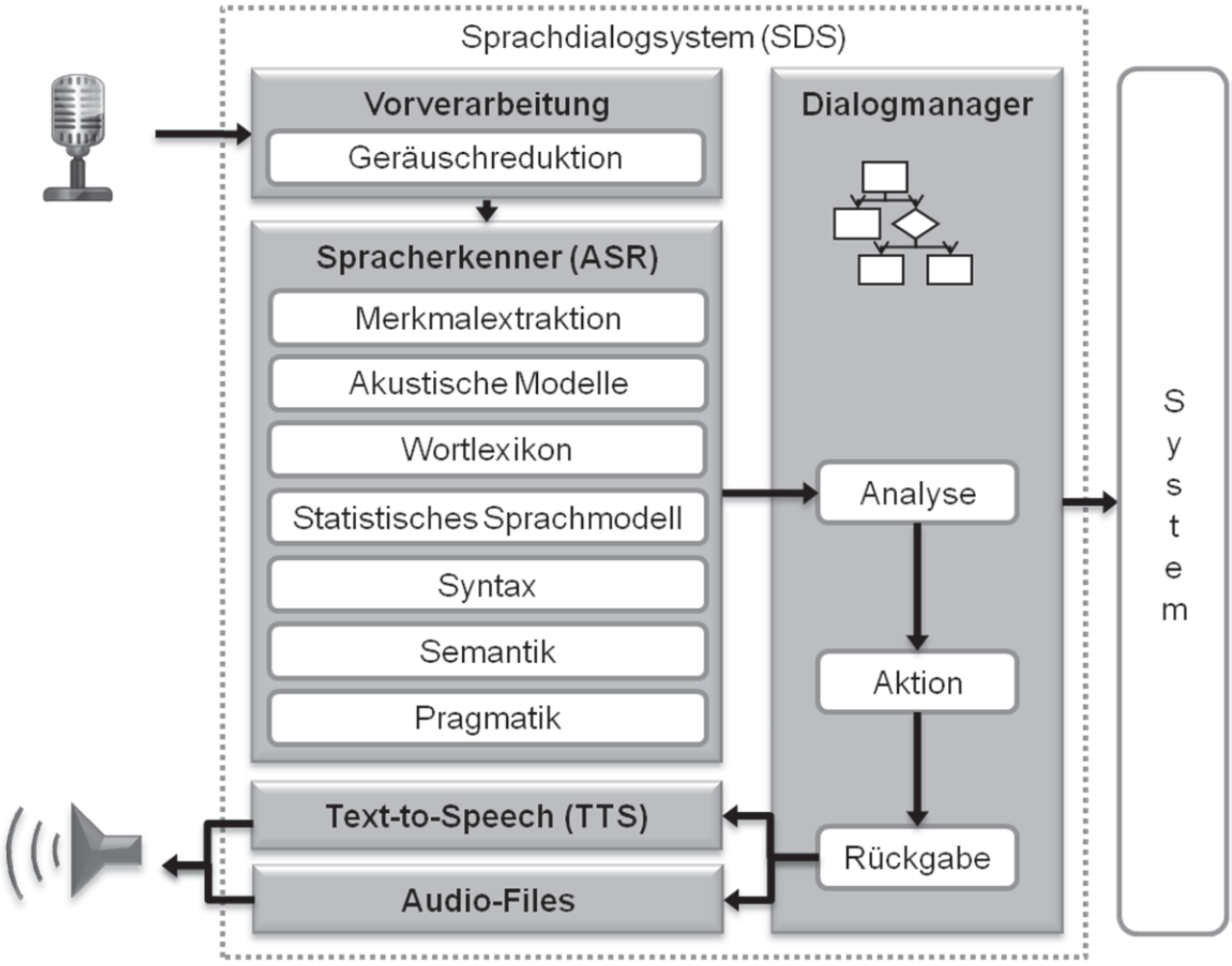


Abbildung 4: Aufbau eines SDS, nach [Eul06], [Ham09]

Der Wortschatz eines Spracherkenners lässt sich durch das Wortlexikon definieren. Alle hierin vorhandenen Wörter können vom ASR potenziell

erkannt werden. Um die Bedeutung der erkannten Zeichenketten zu erfassen, werden die Wortfolgen durch Analyse syntaktischer Strukturen und semantischer Beziehungen innerhalb von Wortfolgen herausgearbeitet. Für weiterführende Informationen zur Sprachsignalerkennung sei auf Fachliteratur wie [Var98], [Jel98] und [Eul06] verwiesen.

Der Dialogmanager (DM) hat die Kontrolle über das SDS. Im DM ist das Dialogmodell hinterlegt, welches vom DM interpretiert wird. Die Ergebnisse der Erkennung werden an den DM weitergeleitet, welcher kontextabhängig den weiteren Dialogfluss steuert. Entsprechend werden Rückgaben an den Benutzer angestoßen. Diese können über eine akustische Reaktion des SDS, durch Sprachausgaben (Voice-Prompts) erfolgen oder durch Ansteuerung weiterer Komponenten außerhalb des SDS (z. B. optische Anzeigen). Die Sprachausgaben des SDS können zwei verschiedenen Methoden zugeordnet werden. Zum einen können Sprachaufnahmen definierter Systemausgaben angelegt und im Dialogmodell als akustische Datei hinterlegt werden. Wird ein entsprechender Dialogschritt erreicht, kann die gespeicherte Audioausgabe abgespielt werden. Diese Methodik eignet sich insbesondere bei Dialogmodellen mit statischen, sich wiederholenden Systemausgaben einer begrenzten Anzahl. Zum anderen werden Methoden zur Sprachsynthese angewandt. Umgesetzt wird dies durch Text-to-Speech (TTS) Engines. Dabei können beliebige Texte, die z. B. dynamisch zur Laufzeit generiert werden, ausgegeben werden. Die Qualität und Natürlichkeit der generierten Sprachausgaben kann in Abhängigkeit der verwendeten TTS Engine erheblich variieren und ist im Allgemeinen, trotz signifikanter Verbesserungen in den vergangenen Jahren, vom Menschen als synthetisch generiert erkennbar.

Insbesondere bei der Ausgabe variabler Dialoginhalte (z. B. Ansage <aktueller Straßenname>) ist der Einsatz von TTS Systemen sinnvoll oder notwendig.

Ausgegeben werden die Systemrückmeldungen des SDS über das Lautsprechersystem im Automobil. Dabei werden sowohl die aufgezeichneten Audiobausteine menschlicher Sprecher, als auch synthetisch erzeugte Ausgaben verwendet und kombiniert. Etwaige Überlagerungen durch parallele Soundausgaben werden durch ein Audiomanagement gesteuert.

Aus den in Kapitel 2.1.2 beschriebenen Sicherheitsgründen hat sich die Sprachbedienung im Automobil bereits als weitere Bedienmodalität, neben der haptischen Bedienung, etabliert. Hinzu kommt ein weiterer Vorteil. Einige Funktionen lassen sich in einem Infotainmentsystem erst nach mehreren haptischen Eingaben, in der dritten oder vierten Ebene, einer Menüstruktur bedienen oder einstellen. Über die Sprachbedienung können solche Funktionen gegebenenfalls direkt, durch nur eine Nutzeraktion, aufgerufen werden. Des Weiteren können komplexe Eingaben wie die eines Navigationszieles in einem Bedienschritt durchgeführt werden, wohingegen in der haptischen Modalität jeder Buchstabe einzeln eingegeben oder aus einer langen Liste ausgewählt werden muss. Aufgrund dessen ist die Sprachbedienung im Automobil nicht ausschließlich ein Element zur Erhöhung der Verkehrssicherheit, sondern trägt auch erheblich zur Steigerung des Bedienkomforts bei.

Abbildung 1 (S.23) veranschaulicht auch die Entwicklung der Anzahl, aller dem Fahrer zur Verfügung gestellten, möglichen Sprachbefehle.

Welche Bedeutung der Sprachbedienung bereits heute zukommt, wird in deren Entwicklung über die vergangenen Systemgeneration deutlich. Dabei ist zu beachten, dass neben den abgebildeten Sprachbefehlen zusätzlich eine große Anzahl alternativer Sprachbefehle definiert ist, welche synonym verwendet werden, um die Eingaben im gewohnten Vokabular durchführen zu können.

Im Automobil ist das SDS üblicherweise nicht permanent aktiviert. Möchte der Fahrzeuglenker das System aktivieren, wird dieses im Allgemeinen über ein haptisches Stellteil realisiert. Aktuell wird dieses in nahezu allen Fahr-zeugen als Lenkradtaste realisiert (Push-To-Talk, PTT oder Push-To-Activate, PTA), siehe Abbildung 5.

Abbildung 5: Push-To-Talk-(PTT) im Bedienfeld des Lenkrads

Die Aktivierung des SDS wird dem Bediener durch eine akustische und/oder optische Rückmeldung bestätigt. In den meisten Fällen wird ein kurzer Signalton ausgegeben, um hinsichtlich des Bedienkonzepts die Aktivierung der sprachlich-akustischen Modalität zu signalisieren. Das Ende eines Sprachdialogs kann entweder durch eine vom SDS angestoßene Systemreaktion oder einen Abbruch des Dialogs erreicht werden. Wird ein Dialog abgebrochen, wird dies analog zur Aktivierung meist durch ein optisches oder akustisches Signal bestätigt.

In verschiedenen Fahrzeugen wird auch ein haptisches Element zum Abbruch des Sprachdialogs angeboten (Push-To-End PTE).

Die Anforderungen an das SDS im Kfz unterscheiden sich von jenen anderer Sprachsysteme [Ham09]. Die in einem Fahrzeug verbaute Hardware ist aufgrund der Umgebungsbedingungen nicht mit handelsüblichen IT-Bausteinen vergleichbar. Verbaute Elemente müssen Resistenzen gegen große Temperaturunterschiede, Erschütterungen und elektromagnetische Felder besitzen. Zudem ist die maximale Spannungsversorgung ebenso wie die Möglichkeit zur Kühlung eingesetzter Bauteile beschränkt. Aufgrund des hohen Kostendrucks und hoher Kostenrelevanz sind meist Hardwareelemente mit minimal notwendiger Leistung verbaut. Vor dem Hintergrund, dass Sprachdialogsysteme komplexe Softwareprodukte mit entsprechend hohen Systemanforderungen sind, ist die Umsetzung einer Sprachbedienung im Automobil angesichts der beschränkten Ressourcen eine besondere Herausforderung.

Auch die Akustik und Qualität der Spracheingabe im Automobil ist für ein SDS herausfordernd. Jedes Fahrzeugmodell verursacht aufgrund der individuellen räumlichen Gegebenheiten und Umgebungsbedingungen unterschiedliche Umgebungsgeräusche. Hinzu kommen variable Störgeräusche der Fahrbahn, Wetter, Klimaanlage, Blinker, Scheibenwischer usw. Die Qualität des Eingangssignals ist aber für die Spracherkennrate von großer Bedeutung. Aus diesem Grund ist eine akustische Vorverarbeitung der Signale im Automobil zwingend notwendig. Möglichkeiten zur Umsetzung sind in [Hän04] und [Nor01] beschrieben.

Mit dem Ziel, ein optimales Bedienkonzept für die akustische MMS zu entwickeln, fließen erhebliche Aufwände in die Gestaltung des Dialogmodells. Die akustische MMS wird ergänzend zur haptischen Bedienung in das Gesamtsystem integriert. Um eine multimodale Bedienung zu ermöglichen, sind Schnittstellen zur haptischen Bedienung zu definieren, welche die Durchgängigkeit im gesamten Bedienkonzept gewährleisten.

Die Kernkompetenzen zur Softwareentwicklung und Implementierung von sprachbedienbaren Systemen finden sich traditionell nicht bei den Automobilherstellern, sondern spezialisierten Softwareunternehmen. Aus diesem Grund wird die Konzeption des SDS vom jeweiligen Automobilhersteller vollständig und detailliert in einer Spezifikation beschrieben, um die Implementierung der individuell festgelegten Dialoge durch einen Systemlieferanten durchführen zu lassen. Dieses Arbeits- bzw. Geschäftsmodell führt für die auftraggebenden Automobilhersteller zu Notwendigkeiten in der funktionalen Qualitätsabsicherung.

2.2 Der Entwicklungs- und Testprozess

Das Automobil als Produkt hoher Komplexität erfordert in der Entwicklung leistungsfähige Prozesse, welche den Anforderungen interdisziplinärer Fachbereiche unternehmensübergreifend Rechnung tragen. Um das Gesamtfahrzeug zu einem definierten Termin („Start of Produktion“, SOP) in die Serienproduktion zu übergeben, wird in der Automobilindustrie ein strukturierter Fahrzeugentwicklungsprozess gelebt. Während sich die Entwicklung in den verschiedenen Fachbereichen parallel vollzieht, wird innerhalb der

einzelnen Fachbereiche, wie beispielsweise der „Elektrik/ Elektronik“ (E/E), sequenziell entwickelt.

2.2.1 Der Entwicklungsprozess

Abbildung 6 veranschaulicht den Fahrzeugentwicklungsprozess schematisch.

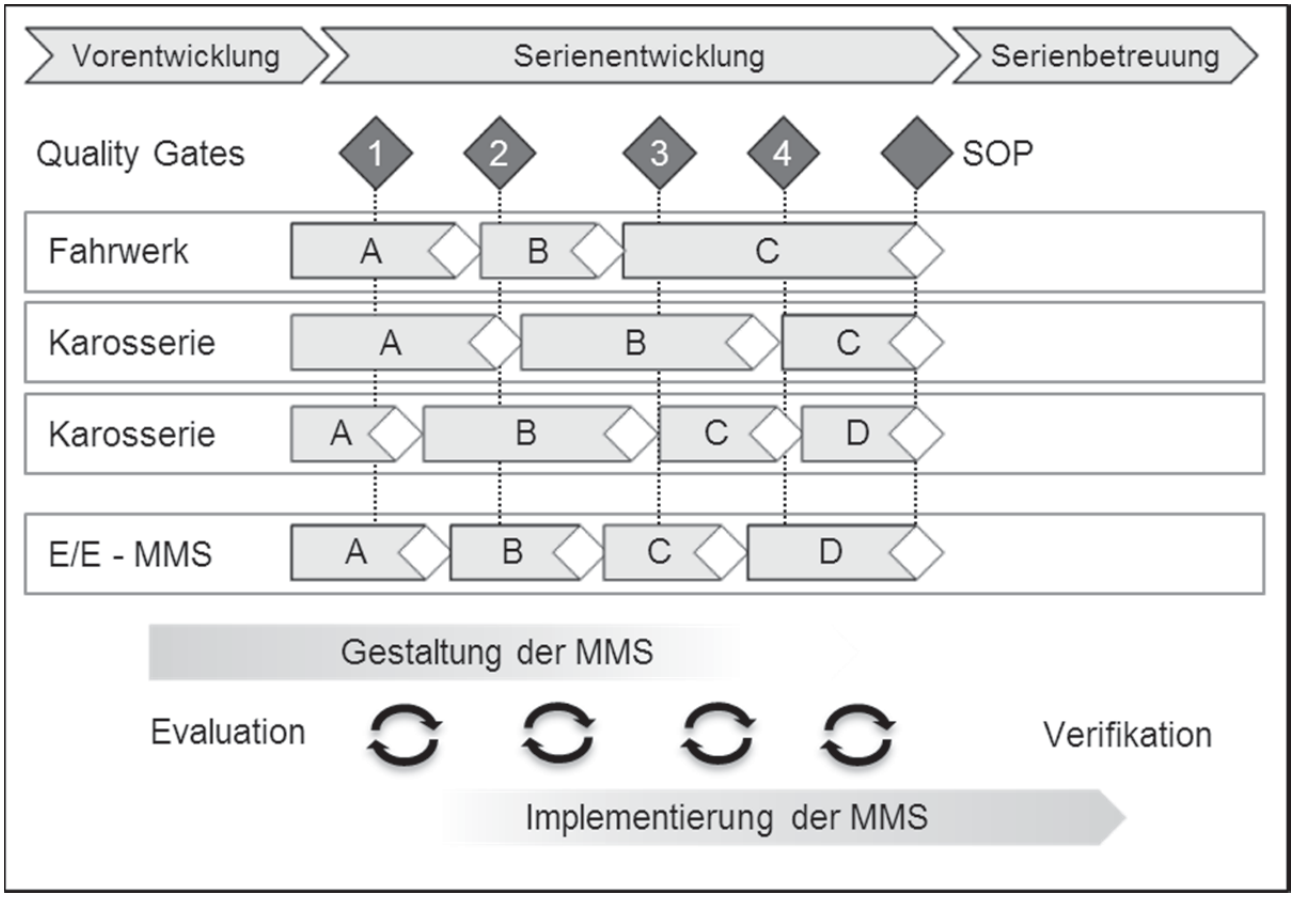


Abbildung 6: Schematischer Entwicklungsprozess in der Automobilindustrie

Die Serienentwicklung folgt dabei einer Vorentwicklung, deren Ziel die Erforschung von Innovationen zu Produkt und Prozessen, sowie die konzeptionelle Weiterentwicklung und Absicherung neuer Lösungen ist. Ergebnisse fließen in die Serienentwicklung ein.

In der Serienentwicklung werden alle Komponenten des Gesamtfahrzeugs in Zusammenarbeit mit den jeweils relevanten Zulieferern zur Serienreife gebracht. In diesem Produktentstehungsprozess erstellen die Fachbereiche des Automobilherstellers Anforderungsdokumente, welche die Anforderungen und Erwartungen an das Zielprodukt beschreiben. Auf Basis dieser Dokumente werden für die zu vergebenden Fahrzeugteile Lieferanten-verhandlungen und Aufwandsanalysen durchgeführt, die schließlich zur Auftragsvergabe führen. Im Folgenden erstellt der Auftraggeber die notwendigen Detailspezifikationen, in welchen detailliert beschrieben und spezifiziert wird, auf welche Weise die angeforderten Funktionen und Eigenschaften des Zielprodukts umgesetzt werden sollen [Rup09].

Die Serienentwicklung erstreckt sich über einen Zeitraum von mehreren Jahren, weswegen es notwendig ist, Meilensteine und/oder Quality Gates zu definieren, welche in bestimmten zeitlichen Abständen, endend mit dem SOP, Zwischenziele beschreiben. Für jeden Fachbereich sind individuelle Ziele definiert, welche zu einem bestimmten Meilenstein erreicht sein müssen. Auf diese Weise ist es möglich, die zahlreichen parallel verlaufenden Entwicklungsprojekte zu synchronisieren und eine Qualitätssicherung der erreichten Entwicklungsstände durchzuführen. Zur weiteren Unterstützung einer synchronen Entwicklung wird in den verschiedenen Fachbereichen nach festgelegten Musterphasen entwickelt. In der Softwareentwicklung wird beispielsweise im Rahmen der A-Musterphase das Systemverhalten definiert. Mit Abschluss der B- bzw. C-Musterphasen sollte das Feindesign der Software abgeschlossen sein, bevor am Ende der D-Musterphase das serienreife, qualitativ abgesicherte Produkt entwickelt sein soll.

Die Entwicklung des Infotainmentsystems und dessen MMS findet im Gesamtrahmen des beschriebenen Fahrzeugentstehungsprozesses statt. In der Serienentwicklung werden die Detailspezifikationen für die verschiedenen Komponenten des Infotainmentsystems festgeschrieben, auf Basis derer ein Systemlieferant die Hardware und/oder Software implementiert. Die Segmentierung der Softwareimplementierung ist im Vergleich zur Hardwareentwicklung deutlich granularer organisiert. So werden nicht ausschließlich die beschriebenen A-, B-, C-. oder D-Muster hinsichtlich der Softwarequalität überprüft, sondern auch definierte, zwischenliegende Entwicklungsstände. Aufgrund der langen Produktlebenszyklen werden die entwickelten Produkte und Ergebnisse nach dem SOP einer Serienbetreuung übergeben, welche im weiteren Verlauf notwendige Weiterentwicklungen oder Änderungen am Produkt verantwortet.

2.2.2 Das V-Modell

Im Folgenden werden sich alle verwendeten Begrifflichkeiten im Kontext der Testprozesse und Testaktivitäten an den „ISTQB®/GTB Standard Glossar der Testbegriffe“ [Int09] halten. Das „International Software Testing Qualifications Board“ (ISTQB®) ist eine 2002 gegründete, internationale Organisation und Plattform für Softwaretester. Fach- und Expertenkenntnisse zum Softwaretest werden zusammengeführt und in Form von international anerkannten Qualifikationsmaßnahmen an Interessierte weitergegeben. In vielen Unternehmen hat sich der durch das ISTQB® zusammengeführte und entwickelte Glossar als „Quasi-Standard“ etabliert. Nationale Organisationen wie das „German Testing Board“ (GTB) arbeiten eng mit dem ISTQB®

zusammen, entwickeln Inhalte weiter und setzen die internationalen Dokumente für die lokale, nationale Verwendung um [GTB12].

Neben dem skizzierten Entwicklungsprozess finden in der Serienentwicklung umfangreiche Testaktivitäten zur Qualitätssicherung statt, für welche separate Testprozesse notwendig sind und entsprechend definiert sein müssen. Der Zusammenhang zwischen Entwicklungs- und Testprozess wird im V-Modell deutlich. Abbildung 7 zeigt ein allgemeingültiges V-Modell der Softwareentwicklung [Spi08]. Auf der linken Seite des Modells befindet sich die (kognitive) Entwicklung und schriftliche Definition des Zielsystems.

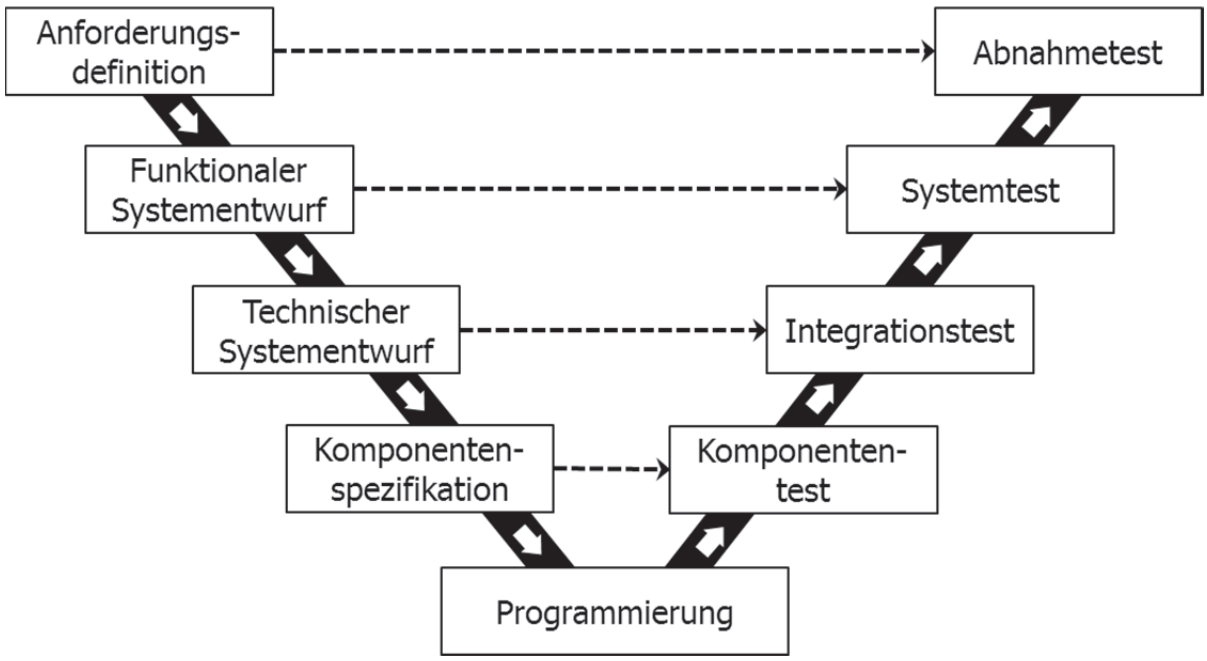


Abbildung 7: Allgemeines V-Modell in Anlehnung an [Spi08]

Ausgehend von einer Vision oder Idee werden die Anforderungen an das System immer detaillierter dargelegt und in der Spezifikation beschrieben. Auf der rechten Seite des V-Modells wird der entsprechende Testvorgang beschrieben.

Vorteil der Darstellung im V-Modell ist die übersichtliche Illustration der verschiedenen Teststufen und deren Zusammenhang mit dem jeweiligen Entwicklungsschritt oder Ausgangsdokument. Im Komponententest werden die einzelnen, elementaren Module oder Komponenten getestet, im Integrationstests das Zusammenspiel von zwei oder mehreren Komponenten, im Systemtests die Funktionsfähigkeit des gesamten Systems entsprechend der Anforderungen, und im Abnahmetest die Qualität und Gebrauchstauglichkeit des Systems unter Echtzeitbedingungen. Nachteilig an einer Darstellung im V-Modell ist festzuhalten, dass der reale zeitliche Ablauf ebenso wenig abgebildet ist wie die notwendigen Testaktivitäten zur Durchführung einer Teststufe [Pol02]. Testaktivitäten finden nicht gebündelt am Ende des Entwicklungsprozesses statt, sondern sind vielmehr iterativ während des gesamten Produktentstehungsprozesses notwendig und müssen entsprechend geplant werden. Ergänzend zum aufgezeigten V-Modell ist ein detailliertes Vorgehensmodell für die Testaktivitäten notwendig.

2.2.3 Der fundamentale Testprozess

Spillner et al. beschreiben den fundamentalen Testprozess (Abbildung 8), welcher die grundlegenden Aktivitäten des Testens gliedert und beschreibt [Spi08]. Der Prozess ist im Grundsatz für jede Teststufe des V-Modells (Abbildung 7) anwendbar.

Hierzu gehören Testplanung und –steuerung, Testanalyse und -design, Testrealisierung und -durchführung, Testauswertung und -bericht sowie der Abschluss der Testaktivitäten. Im Folgenden wird der Inhalt dieser Elemente skizziert. Für weiterführende Informationen sei auf [Spi08] verwiesen.

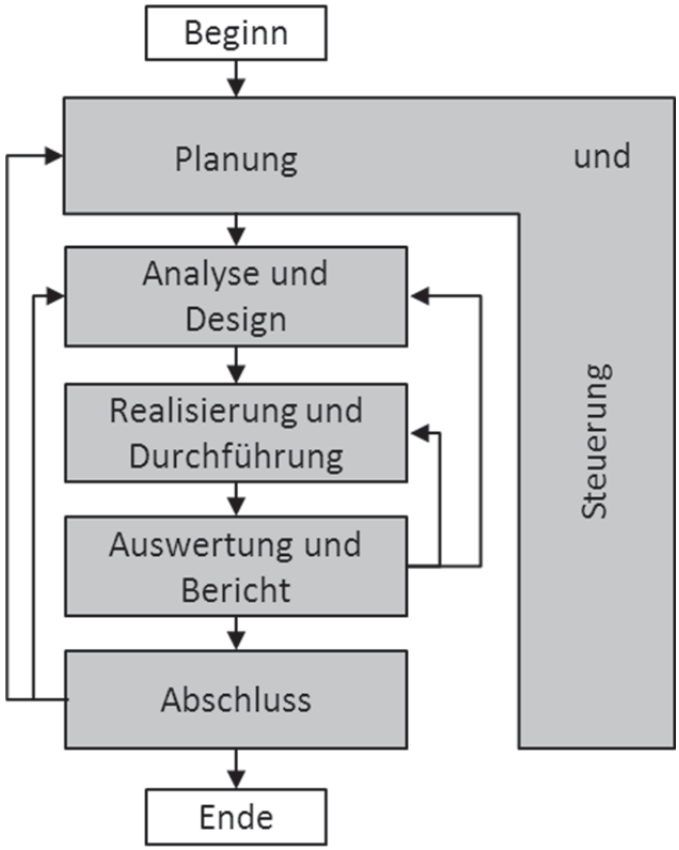


Abbildung 8: Der Fundamentale Testprozess [Spi08]

Die Aktivität **„Testplanung und -steuerung“** beginnt mit dem Start eines Softwareentwicklungsprojekts und bleibt während des gesamten Testprozesses steuerndes Element. Zur Testplanung gehört nicht nur die Definition eines Zieles welches durch den Test erreicht werden soll, sondern auch die Planung von Ressourcen zur Testdurchführung (Anzahl der Mitarbeiter, notwendige Zeit, notwendige Hardware zum Test etc.). Des Weiteren ist eine nachvollziehbare Teststrategie ebenso festzulegen (inkl. Definition der Testverfahren und anzuwendende Testtechniken), wie eine Methodik zur Priorisierung der Tests und Testendekriterien. Es liegt in der Verantwortung der Teststeuerung, dass der Ablauf aller geplanten und tatsächlichen Testaktivitäten übereinstimmt. Auf Basis der zur Teststeuerung notwendigen Bericht-

erstattung durch die Tester sind vom Testmanager gegebenenfalls Anpassungen oder steuernde Eingriffe erforderlich [Spi08].

Die Phase **„Testanalyse und -design“** ist inhaltlich einerseits noch Teil der beschriebenen Testplanung, andererseits Vorbereitung der operativen Durchführung. Die Testanalyse bezieht sich dabei auf die Definition von Testbedingungen (Funktionen, welche durch Testfälle verifiziert werden können) und die Planung der operativen Testdurchführung. Das Testdesign bezieht sich auf den Entwurf von Testfällen. Nach [Spi08] gehört zu jedem Testfall die Festlegung der Ausgangssituation (Vorbedingung), relevante Testdaten, dass Sollergebnis und Nachbedingungen des Tests. In jedem Fall sollen Testfälle reproduzierbar, nachprüfbar und rückverfolgbar sein. In diesem Prozessschritt können die Anforderungsdokumente, welche als Basis für die Erstellung von Testfällen herangezogen werden (Testorakel), einem Review unterzogen werden. Durch eine statische Analyse ist es möglich die Qualität des Testorakels während der Testfallerstellung sicherzustellen und zu verbessern (siehe auch 2.2.4).

Es folgt die **„Testrealisierung und -durchführung“**. Zur Testrealisierung gehört die Organisation der Testfälle, die optional notwendige Erstellung von Testskripten (Testablaufspezifikation für automatisierte Tests), die Festlegung der Reihenfolge zur Durchführung der Testfälle und die Überprüfung der Testumgebung. Sind alle vorbereitenden Aktivitäten abgeschlossen, können die Testaktivitäten durchgeführt werden. Bei der Testdurchführung lässt sich die manuelle Testdurchführung von der automatisierten unterscheiden. Für den manuellen Test sind explizite Testfälle nicht zwangsläufig notwendig (z. B. im Erfahrungsbasierten Test, siehe Kapitel 2.2.4). In den meisten Fällen

ist es jedoch sinnvoll, auch für die manuelle Testdurchführung Testfälle zu verwenden. In diesem Fall ist darauf zu achten, dass Testfälle klar aufgebaut und gut lesbar sind, um menschliche Fehler bei der Testdurchführung zu vermeiden. Manuelles Testen ist flexibel und einfach durchzuführen. Bei einer überschaubaren Anzahl an Testfällen ist die Vorgehensweise zudem günstig und effizient. Von Nachteil ist ein ausschließlich manuelles Testen dann, wenn eine große Anzahl an Testfällen zu prüfen ist, oder Testfälle häufig wiederholt werden müssen. Die Testdurchführung wird hier schnell zeitaufwändig, teuer und somit ineffizient. Besser ist in diesem Fall meist eine automatisierte Testdurchführung. Hierfür sind Testfälle, bzw. maschinenlesbare Testskripte notwendig. Der Aufbau und die Einrichtung einer Testumgebung zum automatischen Test ist kostenintensiver als für manuelle Testaktivitäten, weswegen die automatische Durchführung primär bei sich wiederholenden Testfällen mit hoher Anzahl effizient ist. In den vergangenen Jahren hat sich die Testautomatisierung immer weiter durchgesetzt und nach [Roß10] erheblich dazu beigetragen, die Akzeptanz des Testens generell zu erhöhen.

Durch die Testdurchführung gewonnene Ergebnisse sind im Anschluss gewissenhaft zu analysieren, um Fehlverhalten der Software festzustellen und einzuordnen. Werden Fehler identifiziert, müssen diese nicht zwangsläufig im zu testenden System (engl.: “System Under Test“, SUT) begründet sein. Ebenso können Fehler bei der Testdurchführung, in der Testdokumentation, den verwendeten Testdaten oder bereits im Testorakel ihre Ursache haben. Es ist sicherzustellen, dass Fehler und Fehlerursache richtig protokolliert werden [Spi08]. Hierzu gehört auch die Möglichkeit, Testfälle eindeutig identifizieren zu können, um nach der Fehlerkorrektur entsprechende

Regressionstests (siehe Kapitel 2.2.4) durchführen zu können. Es ist zu beachten, dass die Behebung der Fehler nicht Aufgabe der Qualitätssicherung, sondern der Entwicklung ist.

Die **„Testauswertung“**, zusammengefasst im **„Testbericht“**, ist prinzipiell für jede Teststufe anzufertigen. Ergänzend kann ein abschließender Testbericht, welcher die Testergebnisse und Erkenntnisse in ihrer Gesamtheit wiedergibt, sinnvoll sein. In den Berichten ist die Verwendung zuvor definierter Metriken von großer Bedeutung, um eine aussagekräftige Qualitätsbewertung des SUT zu ermöglichen. Es ist zielführend, die Metriken in Abhängigkeit eines Projekts individuell festzulegen und anzuwenden. Die GQM-Methodik (engl. Goal-Question-Metric) kann bei der Findung geeigneter Maße helfen. Hierbei sind die folgenden drei Fragen zu beantworten [Bas88]:

1. Wie lautet das Ziel (engl. Goal), welches durch die Messung erreicht werden soll?
2. Welche Fragen (engl. Question) müssen beantwortet werden, um das in (1) beschriebene Ziel zu erreichen? Welche Eigenschaften des SUT sind zu erfassen?
3. Wie, oder durch die Anwendung welcher Metriken (engl. Metric), können die relevanten Eigenschaften aus (2) gemessen werden?

Einige häufig verwendete und projektübergreifend sinnvolle Metriken und Maße werden in der Fachliteratur beschrieben. In Abhängigkeit ihrer Ziele und Ausrichtung werden diese in verschiedene Klassen eingeteilt. Dabei wird in Prozess-/Projektmetriken und Produktmetriken unterschieden. Tabelle 1 zeigt eine Übersicht ausgewählter Produktmetriken, in Tabelle 2 ist eine Aus-

wahl verschiedener Prozess-/Projektmetriken beschrieben. Die Einordung der Metriken in eine „Maßklasse“ ist aufgrund fließender Übergänge nicht immer eindeutig möglich.

Tabelle 1: Auswahl verschiedener Produktmetriken

(Produkt-) Metriken	Beschreibung
Anzahl Testfälle ohne Fehlerwirkung	Anzahl durchgeführter Testfälle ohne Fehlerwirkung
Anzahl Testfälle mit Fehlerwirkung	Anzahl durchgeführter Testfälle mit Fehlerwirkung
Anzahl blockierter Testfälle	Anzahl nicht vollständig ausgeführten Testfälle
Prozentual abgedeckte Anforderungen	Abgedeckte Anforderungen/ Alle Anforderungen
Prozentsatz bestandener Testfälle	Prozentsatz bestandener Testfälle/ Alle Testfälle
Prozentsatz Testfälle mit Fehlerwirkung	Testfälle mit Fehlerwirkung/ Alle Testfälle
Prozentsatz blockierter Testfälle	Blockierte Testfälle/ Alle Testfälle
Fehlerdichte	Fehlerhafte Anforderung/ Alle Anforderungen
Testabdeckung	Getestete Pfade (Zweige/Knoten)/ Alle Pfade (Zweige/Knoten)

Die Verwendung von Metriken in der Projekt- oder Prozessbewertung ist zielführend, um eine fundierte Qualitätsaussage ableiten zu können. Die Ergebnisse können in zukünftig angewandte Prozesse und durchzuführende Projekte einfließen, um eine kontinuierliche Verbesserung selbiger zu erreichen [Spi08], [Lig09].

Tabelle 2: Auswahl verschiedener Prozess-/Projektmetriken

(Prozess/Projekt-) Metriken	Beschreibung
Anzahl spezifizierter Testfälle	Anzahl der insgesamt formulierten Testfälle
Anzahl automatischer Testfälle	Testfälle zur automatischen Testdurchführung
Anzahl geänderter Testfälle	Anzahl geänderter Testfälle (z.B. weil fehlerhaft…)
Anzahl gelöschter Testfälle	Anzahl gelöschter Testfälle (z.B. weil ungültig)
Anzahl durchgeführter Testfälle	Anzahl aller durchgeführten Testfälle
Testzeit	Anzahl der Teststunden (h)
Prozentual durchgeführte Testfälle	Durchgeführte Testfälle/ Alle Testfälle
Fehlerfindungsrate	Anzahl der innerhalb eines def. Zeitraums gefundenen Fehler/ Alle gefundenen
Anzahl Personentage für manuellen Test	Anzahl der Personentage für manuelles Testen
Anzahl Personentage für automatischen Test	Anzahl der Personentage für automatisches Testen
Testproduktivität	Anzahl gefundener Fehler/ Test-Personentage
Testkosten	Summe der Testkosten
Testeffizienz	Testkosten/ Anzahl der identifizierten Fehler
Effizienz der Testfallerstellung	Anzahl erstellter Testfälle/ Personenstunde
Effizienz der Testdurchführung	Anzahl durchgeführter Testfälle/ Personenstunde in der Testdurchführung

Es ist festzuhalten, dass keine allgemeingültigen Grenzen oder Wertebereiche für eine bestimmte Metrik festgelegt werden können. Der Testmanager muss erfahrungsbasiert ermitteln, ob Ergebnisse innerhalb der Toleranzgrenzen liegen [Lig09].

Zum **„Abschluss der Testaktivitäten“** ist sicherzustellen, dass die Durchführung der eingeplanten Testaktivitäten vollständig abgeschlossen ist. Identifizierte Fehler sind behoben und alle erstellten Berichte und Dokumente vollständig. Rückblickend werden an dieser Stelle Verbesserungen und Potenziale dokumentiert, um die Testaktivitäten im folgenden Prozess zu verbessern. Zudem ist eine übergreifende Analyse der identifizierten Fehler zielführend, um etwaige Auffälligkeiten und Tendenzen in Fehlerursache und Fehlerzeitpunkt festzustellen. Auf Basis der hierdurch erlangten Erkenntnisse lassen sich Verbesserungspotenziale im Entwicklungsprozess heben. Zuletzt sind Testumgebung und verwendete Testkomponenten, einschließlich der Testfälle, Testinfrastruktur etc. (engl. Testware), zu archivieren. Diese Aktivität gewährleistet, dass im Bedarfsfall spezifische Testaktivitäten auch nach längerem Zeitraum, unter gleichen Testbedingungen wiederholt werden können. Zudem kann die archivierte Testware an die Serienentwicklung übergeben werden, um so eine fortlaufende Wartung des SUT im Serienbetrieb zu ermöglichen.

Die beschriebenen Testaktivitäten geben dem Testmanager die Rahmenbedingungen zum Testvorgehen und erforderliche Inhalte vor, lassen ihm aber auch die notwendige Freiheit, die konkrete Umsetzung an das individuelle Entwicklungsprojekt anzupassen. Nachdem der prozessuale Rahmen für das „Testen“ beschrieben ist, wird im Folgenden die Aktivität

Testen hinsichtlich der inhaltlichen Ausprägungen, Methoden und Vorgehensweisen vertieft.

2.2.4 Testmethoden, -techniken und -verfahren

Im technischen Sinne ist Testen „der Prozess, der aus allen Aktivitäten des Lebenszyklus besteht (sowohl statisch als auch dynamisch), die sich mit der Planung, Vorbereitung und Bewertung eines Softwareprodukts und dazugehöriger Arbeitsergebnisse befasst. Ziel des Prozesses ist sicherzustellen, dass diese allen festgelegten Anforderungen genügen, dass sie ihren Zweck erfüllen, und etwaige Fehlerzustände zu finden.“ [Int09]. In der Definition des Testbegriffs durch das [Int09] wird in **statisches und dynamisches Testen** unterschieden. Im statischen Testen wird das SUT (System Under Test) nicht ausgeführt. Der statische Test bezieht sich somit auf die Analyse, Reviews, syntaktische Prüfung, etc. von Anforderungsdokumenten, Spezifikationen oder Code. Das dynamische Testen erfordert eine ausführbare Software, Systeme oder Programmteile, um in der Ausführung des SUT Fehler zu finden. Hierbei werden die realen mit den erwarteten Systemausgaben, in Reaktion auf definierte Eingaben, auf Übereinstimmung verglichen. Im Folgenden wird der allgemeine Begriff „Testen“ auf den dynamischen Test bezogen.

Dynamische Tests können sowohl **funktionale** als auch **nicht funktionale** Testziele verfolgen. Während durch nicht funktionale Tests konzeptionelle Bedingungen wie Benutzbarkeit, Stabilität oder Performance verifiziert werden, wird im funktionalen Test geprüft, ob die implementierte Funktionalität alle Vorgaben erfüllt. [Roß10] beschreiben die Zusammenhänge des

dynamischen Tests wie in Abbildung 9 dargestellt. Hier wird deutlich, dass neben den Systemeingaben und folgenden Systemausgaben auch Vorbedingungen bestehen können und zu berücksichtigen sind, um einen Test auszuführen. In Abhängigkeit verschiedener Vorbedingungen können Eingaben ihre Gültigkeit verlieren oder zu unterschiedlichen Systemreaktionen führen. Die Reaktion des Systems kann analog um Nachbedingungen ergänzt werden, welche nach Bearbeitung der Systemeingaben bestehen (z. B. Änderungen interner Systemzustände oder Datenbankeinträge).

Die Zusammenhänge zwischen Systemeingaben und -ausgaben, sowie Vor- und Nachbedingungen leiten sich aus den Anforderungs- bzw. Spezifikationsdokumenten ab. Nach [IEE90] umfasst ein Testfall genau diese zur Systemausführung notwendigen Angaben („[...] notwendige Vorbedingungen, [...] Eingabewerte, [...] vorausgesagte Ergebnisse, sowie die erwarteten Nachbedingungen“), siehe Abbildung 9.

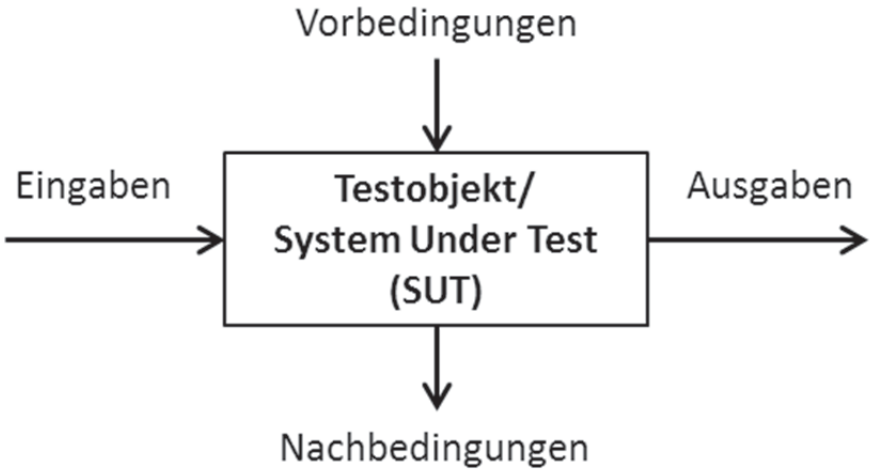


Abbildung 9: Parameter des dynamischen Tests aus [Roß10], S.17

Im Weiteren sind **abstrakte und konkrete Testfälle** zu unterscheiden. Im abstrakten Testfall werden ausschließlich die Wertebereiche für Ein- und Ausgaben des Systems beschrieben (z. B. „gültige Hausnummer 1-999“). Im konkreten Testfall werden tatsächliche, konkrete Werte verwendet (z. B. Hausnummer 16).

Es ist zu beachten, dass bei der Durchführung von Tests gegen die Spezifikation des SUT (mit oder ohne Testfälle) die grundsätzliche Problematik besteht, dass ausschließlich die Anwesenheit von Fehlern bewiesen werden kann, deren Abwesenheit jedoch nicht. Dieser Beweis kann nur indirekt erbracht werden, wenn ein System in seiner Ganzheit vollständig geprüft werden würde und dabei keine Fehler aufträten. In der Praxis ist dies schon bei kleinen Programmen aufgrund der Vielzahl möglicher Programmzustände kaum zu realisieren. Aus diesem Grund wurden in den vergangenen Jahrzehnten systematische Testtechniken entwickelt, welche durch systematische Methoden versuchen, die notwendige Anzahl an Testfällen gering zu halten und dabei trotzdem möglichst viele potentielle Fehler zu identifizieren. Einige dieser Testtechniken werden im Folgenden vorgestellt.

Vorausschickend sind zwei weitere, grundsätzlich zu unterscheidende Testverfahren von Bedeutung: Das **Black-Box- und White-Box-Testverfahren**. Im Black-Box Testverfahren wird das SUT als abgeschlossenes, nicht einsehbares System verstanden. Die Spezifikation wird als Testbasis verwendet, weitere Informationen aus der konkreten Implementierung, wie Datenstruktur, Code und Algorithmen, werden in diesem Testverfahren nicht verwendet. Die Testüberdeckung wird auf Basis der spezifizierten Ein- und Ausgaben gemessen. Funktionale oder funktionsorientierte Testtechniken

sind Black-Box-Techniken. Zudem gibt es weitere Black-Box-Testtechniken, welche nicht zwangsläufig funktionsorientiert sind [Lig09]. Black-Box-Techniken finden hauptsächlich im System- und Abnahmetest Anwendung. Im Gegensatz hierzu wird im White-Box Testverfahren die Programmstruktur oder der Programmcode selbst als Testbasis verwendet, um beispielsweise nicht erreichbare Elemente im Programm oder das interne Schnittstellenverhalten verschiedener Komponenten zu prüfen. Die Testüberdeckung wird auf Basis des abgedeckten Codes gemessen [Wir04], [Fra03]. White-Box Testverfahren werden im Komponententest und im Integrationstest angewendet. Das Testverfahren kann nur eingesetzt werden, wenn dem Tester entsprechende Informationen zur Software vorliegen (Programmstruktur etc.). Wie sich im weiteren Verlauf der Arbeit zeigen wird, liegen diese Informationen dem Automobilhersteller im konkreten Fall einer MMS Entwicklung für ein Kfz-Infotainmentsystem nicht vor. Aus diesem Grund wird im Folgenden eine Auswahl dynamischer Testtechniken vorgestellt, welche im Blackbox-Testverfahren Anwendung finden. Abbildung 10 zeigt eine Übersicht der beschriebenen Testtechniken.

(1) Der **funktionsorientierte Test** (= funktionale Test) bewertet, die Testvollständigkeit und die Korrektheit der Testergebnisse gegen die Spezifikation. Es lassen sich vier Testtechniken unterscheiden.

(1-1) **Die Äquivalenzklassenbildung** ist eine Technik, welche bei gedächtnislosen Systemen Anwendung finden kann. Hierbei werden Werte oder Eingaben, die ein identisches funktionales Verhalten auslösen, zu einer (Eingaben-) Äquivalenzklasse zusammengefasst. Dabei können sowohl positive Äquivalenzklassen (mit gültigen Werten) als auch negative Äqui-

valenzklassen gebildet werden. Ein analoges Vorgehen ist für die Systemausgaben möglich. Werden diese in (Ausgabe-) Äquivalenzklassen eingeteilt, sind Eingabewerte zu bestimmen, welche die entsprechende Ausgabeäquivalenzklasse provozieren. Die Technik basiert auf der These, dass viele Fehler für alle Werte einer bestimmten Äquivalenzklasse auftreten. Aus diesem Grund genügt es, einen Repräsentanten der Äquivalenzklasse zu testen.

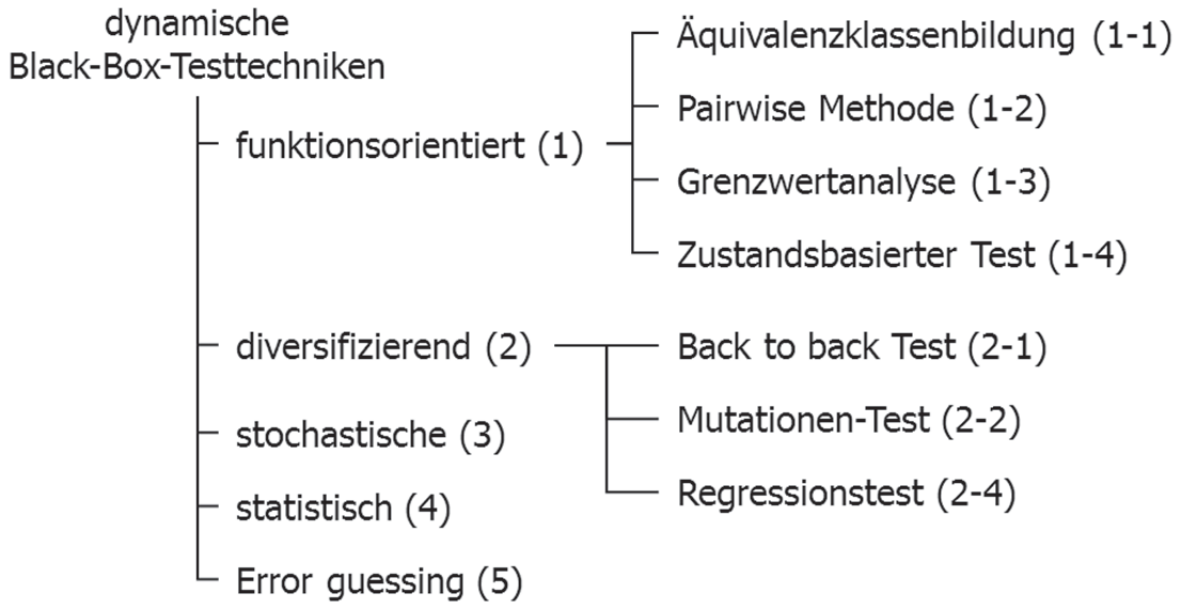


Abbildung 10: Dynamische Testtechniken, nach [Lig09]

(1-2) **Die Pairwise Methode** ist eine wirkungsvolle Technik, um die Anzahl von Testfällen mit jeweils mehreren „parallelen“ Eingabewerten zu reduzieren. Anstatt alle möglichen Kombinationen zu testen (z. B. drei alternative Fahrzeugklassen, drei alternative Motorisierungsvarianten, drei alternative Fahrzeugfarben = 27 Testfälle) wird jeder Eingabewert einer Kategorie lediglich paarweise mit jedem Eingabewert der anderen Kategorien getestet (aus den 27 Testfällen im Beispiel werden neun Testfälle).

Hintergrund dieser Technik ist die These, dass Fehler bei der Kombination zweier bestimmter Eingaben auftreten und somit lediglich sichergestellt sein muss, dass jede Kombination an Paaren mindestens einmal geprüft wird. In allen weiteren theoretisch möglichen Kombinationen, in welchen diese Paarung Auftritt, wiederholt sich der entsprechende Fehler.

(1-3) Die **Grenzwertanalyse** ist ein Spezialfall der Technik zur Äquivalenzklassenbildung. Die Testtechnik der Grenzwertanalyse basiert auf der These, dass Fehler primär an den Systemgrenzen auftreten. Aus diesem Grund wird mit den folgenden Strategien das Systemverhalten insbesondere an vorhandenen Grenzwerten geprüft.

- Ein Grenzwert (z. B. min. oder max.) wird pro Variable ausgewählt und getestet.
- Es werden für jede Variable zwei Grenzwerte (min. und max.) getestet.
- Ist die Variable ähnlich einer Funktion, kann diese mehrere Definitionsbereiche besitzen. Zum Test sind dann drei Testfälle zweckmäßig. Zwei Testfälle mit gültigen Werten innerhalb des Definitionsbereichs (von beiden Seiten an die Grenze annähernd), sowie ein Testfall mit einem Wert außerhalb der Definition.
- Jeder vorhandene Grenzwert einer jeden Variablen wird getestet. Insbesondere dann, wenn viele Variablen im System vorhanden sind, ist dieses Vorgehen aufgrund der hohen Anzahl an Grenzwerten meist nicht realisierbar.

(1-4) **Zustandsbasierte Testtechniken** können dann angewendet werden, wenn die Software als Zustandsautomat/Zustandsdiagramm spezifiziert ist oder so dargestellt werden kann. Systemzustände (engl. states) sind stabil

und werden ohne äußere Stimuli nicht verlassen. Aktivitäten werden hier als Zustandsübergang (engl. transitions) beschrieben, welche zum nächsten Systemzustand führen. Zustandsübergänge können an ein und demselben Zustand beginnen und enden (engl. selftransition). Des Weiteren sind Schleifen möglich, wenn Pfade über mehrere Zustände zu einem vorher durchlaufenen Zustand zurückkehren. Ist ein Zustandsdiagramm vorhanden, kann dies unter anderem mit folgenden Testtechniken systematisch geprüft werden. Zur Veranschaulichung der unterschiedlichen Techniken wird in den jeweils aufgeführten Beispielen Bezug auf den Graphen in Abbildung 11 genommen.

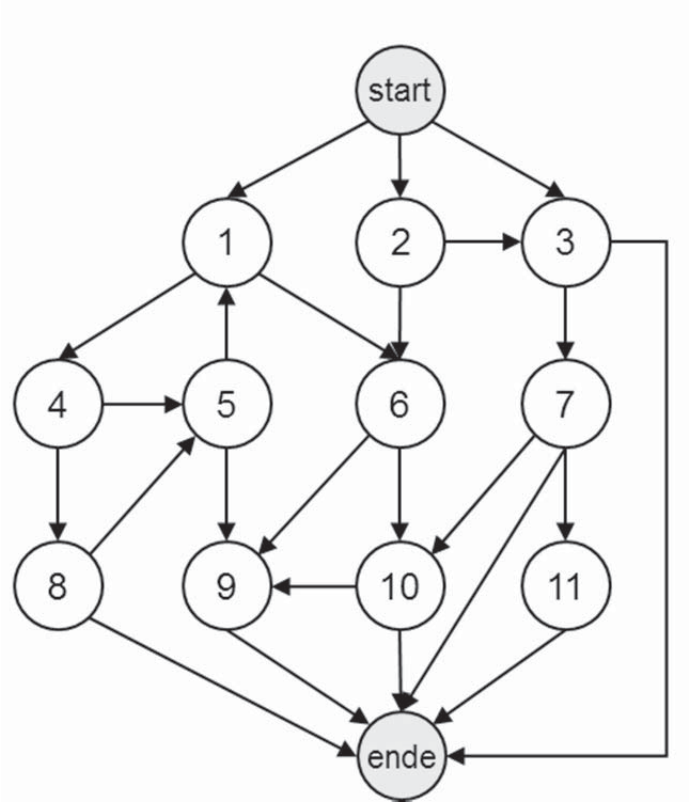


Abbildung 11: Bsp. eines Kontrollflussgraphen

- Die vollständige Zustandsüberdeckung ist erreicht, wenn alle Systemzustände mindestens einmal durchlaufen bzw. aktiviert sind. Im Beispiel

können alle Zustände (1 bis 11) mit 3 Testfällen abgedeckt werden (1-4-8-5-9, 2-6-10, 3-7-11). Dabei werden 10 von 24 Zustandsübergängen nicht durchlaufen.

- Eine vollständige Transitionsüberdeckung ist erreicht, wenn alle Zustandsübergänge mindestens einmal durchlaufen sind. Diese Technik subsumiert die vollständige Zustandsüberdeckung, da hierbei auch alle Systemzustände mindestens einmal durchlaufen werden. Eine vollständige Transitionsüberdeckung ist mächtiger als die vollständige Zustandsüberdeckung, führt aber auch zu einer höheren Anzahl notwendiger Testfälle. Im Beispiel sind zur Abdeckung aller 24 Transitionen acht Testfälle notwendig (1-4-8-5-9, 2-6-10, 3-7-11, 1-4-5-1-6-9, 2-3-7, 3-7-10-9, 3, 1-4-8)
- Die vollständige Pfadüberdeckung ist erreicht, wenn alle möglichen Pfade im Zustandsdiagramm durch einen jeweiligen Testfall abgedeckt sind. Die Technik subsumiert die vollständige Transitionsüberdeckung und ist die mächtigste der möglichen Testtechniken. In der Praxis ist diese Vorgehensweise allerdings bereits bei einfachen Zustandsdiagrammen aufgrund der enormen Anzahl an notwendigen Testfällen kaum anzuwenden. Eine besondere Herausforderung bilden insbesondere Schleifen, welche zu einer unendlich hohen Anzahl an Testfällen führen. Aus diesem Grund kann die vollständige Pfadüberdeckung nur praktikabel eingesetzt werden, wenn die Anzahl der Schleifendurchgänge begrenzt wird. Im Beispiel sind bei dieser Technik und einer Einschränkung auf einen Schleifendurchgang mindestens 26 Testfälle notwendig.

Ergänzend sei an dieser Stelle erwähnt, dass neben Zustandsdiagrammen auch andere beschreibende Diagramme wie Ablaufdiagramme, Fluss-

diagramme oder Sequenzdiagramme mit diesen oder analogen Techniken geprüft werden können. In der entsprechenden Fachliteratur sind diese teilweise als Transaktionsflussbasierte Testtechniken zusammengefasst [Utt07] [Lig09].

(2) Des Weiteren werden in [Lig09] drei **diversifizierende Testtechniken** beschrieben. Hierbei werden die Testergebnisse mehrerer Versionen der zu testenden Software miteinander verglichen. Ein Abgleich der Testergebnisse mit der Spezifikation ist bei diesen Testtechniken nicht vorgesehen.

(2-1) Bei einem **Back to Back Test** werden auf Basis einer Spezifikation mehrere Versionen der Software unabhängig voneinander implementiert. Anschließend werden alle Versionen einem identischen Test unterzogen und die Ergebnisse verglichen. Die Testtechnik basiert auf der Theorie, dass bei einer unabhängigen Softwareimplementierung an unterschiedlichen Stellen Fehler entstehen. Diese würden im Abgleich der Testergebnisse auffällig. Nachteil: Ist ein in allen Versionen identischer Fehler vorhanden, wird dieser folglich nicht entdeckt werden können [Lig09].

(2-2) Um einen **Mutations-Test** durchführen zu können, werden aus einer zu prüfenden Software weitere Versionen abgeleitet, in welchen definierten Fehlern implementiert werden. Aus den so erstellten Softwaremutationen werden Testfälle abgeleitet, die auf die ursprüngliche, zu prüfende Software angewendet werden. Durch die Anwendung dieser Technik kann sowohl die Leistungsfähigkeit der Testfälle überprüft werden (da bekannt ist, welche zu Fehlern führen müssen), als auch der Beweis erbracht werden, dass die zuvor definierten oder bekannten Fehler im SUT nicht vorhanden sind. [Utt07]

(2-3) Der **Regressionstest** ist die Durchführung bereits durchgeführter Testfälle. Regressionstests sind unverzichtbar, um nach Änderungen in der Software sicherzustellen, dass zuvor identifizierte Fehler korrigiert wurden und/oder in bereits geprüften Softwareteilen durch die Änderungen keine weiteren Fehler entstanden sind. Nach [Lig09] sind Regressionstests bevorzugt automatisiert durchzuführen.

Es folgen drei weitere Testtechniken, welche ebenfalls dem dynamischen Black-Box-Test zugeordnet werden können.

(3) Der **stochastische Test** ist eine zufallsbasierte Testtechnik. Ein zufällig ausgewählter Testbereich des SUT wird mit zufällig ausgewählten Testfällen und Testdaten geprüft. Ein stochastisches Vorgehen scheint zwar wenig systematisch, ist jedoch meist effektiv und empfehlenswert, um weitere Testtechniken zu ergänzen.

(4) Der **statistische Test** ist eine Testtechnik, bei welcher die Erstellung von Testfällen und Testdaten auf Basis einer statistischen Auswahl stattfindet. So können die statistisch am häufigsten genutzten oder fehlerhaften Softwareelemente gezielt überprüft werden. Von besonderem Nutzen ist diese Technik, wenn (z. B.) in einem Zustandsdiagramm Wahrscheinlichkeitswerte an den vorhandenen Transitionen ergänzt werden. Diese können bei der Testfallerstellung für einen statistischen Test Verwendung finden.

(5) Das **Error Guessing** ist eine auf Erfahrungswerten basierte Testtechnik. Diese wird meist manuell und weder stochastisch noch deterministisch ausgeführt. Insbesondere bei der Anwendung durch erfahrene Tester ist die Methodik wirkungsvoll. Da weder die Vollständigkeit noch die erreichte Test-

abdeckung nachgewiesen werden kann, ist diese Technik nur in Ergänzung zu anderen Testtechniken anzuwenden.

Eine allgemeingültige Definition von Minimalanforderungen hinsichtlich der durchzuführenden Testtechniken ist nicht gegeben. Die Wahl der Testtechniken basiert letztendlich auf den Erfahrungswerten das Testmanagers. Für weitere Informationen und zu Hintergründen der verschiedenen Testtechniken sei auf entsprechende Fachliteratur wie [Lig09] verwiesen.

Bis auf das „Error Guessing“ haben alle beschriebenen Testtechniken das Ziel, Testinhalte und Testfälle systematisch herzuleiten. Bei einigen Testtechniken, wie beispielsweise den zustandsbasierten, ist es notwendig, eine mehr oder weniger formale Beschreibung des zu testenden Systems als Basis vorliegen zu haben. In diesem Zusammenhang wird im folgenden Kapitel ausführlicher auf das modellbasierte Testen eingegangen. Hierbei handelt es sich um keine Testtechnik [Lig09], sondern um eine Testmethodik oder Vorgehensweise, welche die formale Beschreibung des SUT in den Mittelpunkt der Test-aktivitäten rückt.

2.2.5 Modellbasierter Test

Der Modellbasierte Test (MBT) beschreibt ein Vorgehen, in welchem das zu testende System, dessen Umgebung oder die Testfälle selbst als Modell dargestellt werden. [Roß10] definieren modellbasiertes Testen als „die Nutzung von Modellen für die Automatisierung von Testaktivitäten und/oder die Modellierung von Artefakten im Testprozess“. Utting et. al. definieren MBT als „die automatisierte Konstruktion eines Black-Box-Tests“ [Utt06]. Die Bedeutung des Modellbegriffs, welche außerhalb der Softwaretechnik

Verwendung findet, ist hierbei beachtenswert. Modelle sind im Allgemeinen Abbildungen oder Repräsentationen eines Originals, welche nur beschränkt die Attribute des Originals abbilden. In der Praxis lässt sich hier bereits eine zentrale Herausforderung der Modellierung ableiten. Einerseits soll das Modell auf das Wesentliche reduziert sein, andererseits aber auch alle notwendigen Details zur vollständigen Beschreibung umfassen.

In Abhängigkeit der Umgebungsbedingungen eines Entwicklungsprojekts sowie der definierten Zielsetzung ist MBT von unterschiedlicher Ausprägung. [Utt06] stellen in einer Übersicht vor, welche Ausprägung die verschiedenen Elemente des MBT annehmen können (Abbildung 12).

Nach der Modellierung folgen die Testfallgenerierung und schließlich die Durchführung der generierten Testfälle. Die grundsätzlich möglichen Ausprägungen, welche ein Modell annehmen kann, beschreiben [Utt06] in vier Kategorien:

Gegenstand der Modellierung (Abbildung 12)
Es kann sowohl das System selbst als auch die Systemumgebung modelliert werden. Ebenfalls sind Modelle möglich, welche beide Blickwinkel in unterschiedlicher Ausprägung beschreiben.

Ab-/Unabhängigkeit von System- und Testmodell (Abbildung 12)
[Roß10] beschreibt mit Verweis auf [Sch07] sechs verschiedene Varianten des MBT, welche sich auf diese Kategorie von [Utt06] beziehen und sich durch verschiedene Ausprägungen in der Anwendung von System- und Testmodellen unterscheiden. Systemmodelle beschreiben die Funktionalität sowie das Systemverhalten und sind im Entwicklungsprozess oft als formale

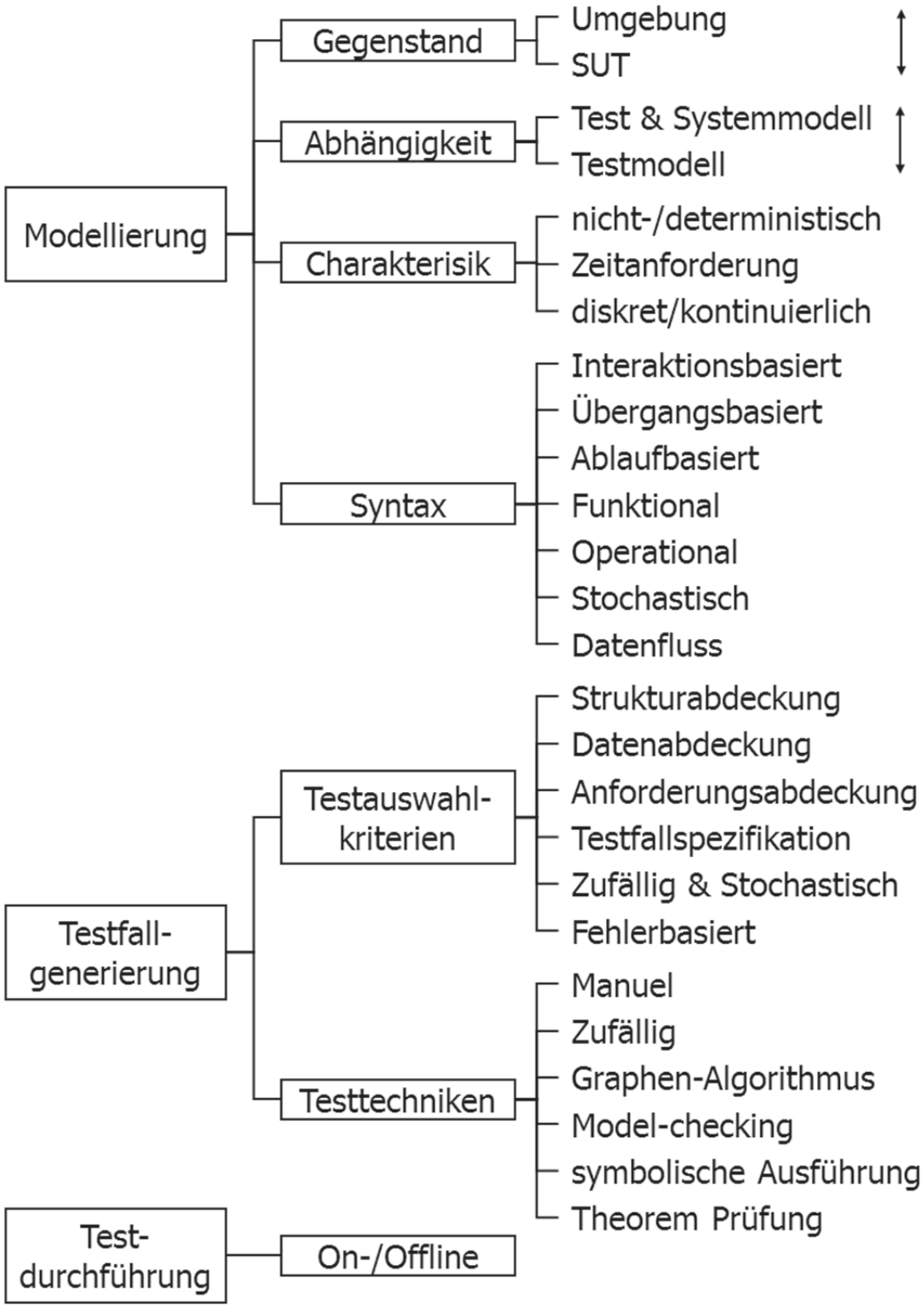


Abbildung 12: Taxonomy of Model-Based Testing, nach [Utt06]

Anforderungsbeschreibung oder Spezifikation vorhanden. In Testmodellen werden die für den Test relevanten Teile des Systems oder der Testumgebung modelliert, um daraus Testfälle zu generieren.

In Abbildung 13 sind die **Systemmodellgetriebenen** Varianten des MBT dargestellt.

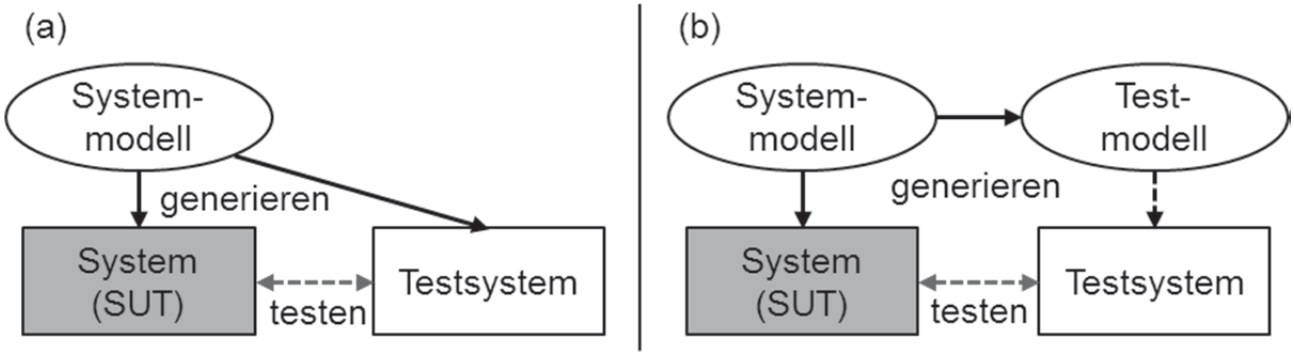


Abbildung 13: Systemmodellgetriebene Varianten des MBT [Roß10]

Darstellung (a) beschreibt eine Vorgehensweise, in welcher aus dem Systemmodell sowohl das Zielsystem (Testobjekt/SUT) als auch das Testsystem (Testfälle etc.) generiert wird. Diese augenscheinlich einfachste Variante (da nur ein Modell notwendig ist und dieses in Form einer Spezifikation möglicherweise schon vorliegt) hat einen entscheidenden Nachteil. Im Systemmodell vorhandene Fehler werden sowohl im SUT als auch im Testsystem implementiert. In der Testdurchführung ist es folglich unmöglich, Fehler zu identifizieren. Es ist von zentraler Bedeutung, die Unabhängigkeit von SUT und Testsystem zu wahren. Dies wird in Variante (b) besser umgesetzt. Hierbei wird aus dem Systemmodell das SUT sowie ein Testmodell generiert. Wird das Testsystem auf Basis des Testmodells erstellt und

nicht generiert, können vorhandene Fehler im Testsystem korrigiert werden. Im anschließenden Test sind diese folglich auch im SUT identifizierbar.

Abbildung 14 zeigt verschiedene Varianten des **Testmodellgetriebenen** Vorgehens.

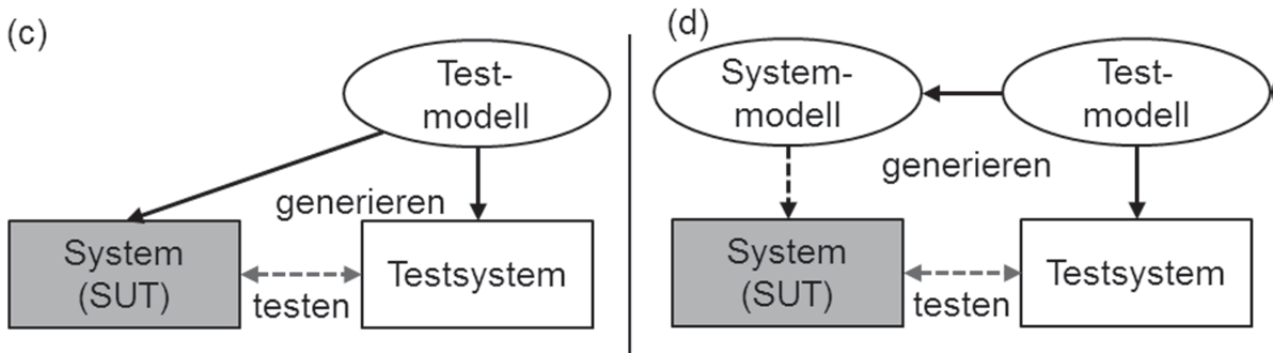


Abbildung 14: Testmodellgetriebene Varianten des MBT [Roß10]

Die Variante in Darstellung (c) kommt gänzlich ohne Systemmodell aus. Hierbei wird aus dem Testmodell nicht nur das Testsystem abgeleitet, sondern auch Teile des Systems selbst generiert. Diese Vorgehensweise legt den Fokus im Vergleich zu Varianten (a) aus Abbildung 13 auf das Testen. Die prinzipiellen Nachteile durch die Verwendung von nur einem Modell sind in (a) und (c) kongruent. Daher ist im Falle einer Testmodellorientierten Ausrichtung Variante (d) vorzuziehen. In Analogie zur Darstellung (b) aus Abbildung 13 findet hier sowohl ein System- als auch Testmodell Verwendung. Im Gegensatz zu (b) ist die Vorgehensweise jedoch Testmodellgetrieben. Es ist darauf zu achten, dass die Unabhängigkeit der Modelle gewahrt wird. Auf diese Weise können in (d) Testfälle aus dem Testmodell generiert werden, eine Codegenerierung auf Basis des Systemmodells sollte jedoch unterbunden werden.

Die nach [Roß10] zu bevorzugende Variante des MBT ist in Abbildung 15 dargestellt. Ausgehend von den definierten Anforderungen an das System wird unabhängig voneinander sowohl ein System- als auch Testmodell erstellt.

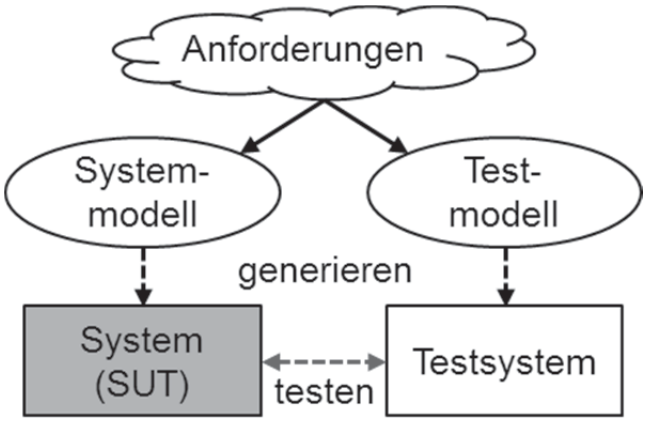


Abbildung 15: Optimale Variante des MBT [Roß10]

Dieses Vorgehen gewährleistet eine vollständige Unabhängigkeit der Modelle. Das Systemmodell kann zur Codegenerierung, das Testmodell zur Generierung von Testfällen verwendet werden. Nach [Roß10] wird der Nachteil, welcher durch den Aufwand einer parallelen Modellerstellung entsteht, durch den folgenden Effizienzgewinn in Implementierung und Testdurchführung ausgeglichen.

Die Charakteristik des Modells (Abbildung 12, S66)

- Das Modell ist entweder von deterministischem Charakter, wenn eine eindeutige Beziehung zwischen Eingabedaten und Ausgabedaten besteht, oder nichtdeterministisch, wenn auf ein und dieselbe Eingabe mehrere Möglichkeiten für den Übergang in den nachfolgenden Zustand gegeben sind.

- Es können optional Echtzeitanforderungen im Modell hinterlegt sein.
- Es sind diskrete oder kontinuierliche Signale/Datenpakete modelliert.

Die Modellierungssyntax im Modell (Abbildung 12, S66)
Das Modell kann in einer von mehreren möglichen Modellierungssprachen umgesetzt sein. In Abbildung 12 sind einige Möglichkeiten dargestellt. Zu den übergangsbasierten Modellen gehören unter anderem auch Zustandsdiagramme. Liggesmeyer beschreibt Zustandsautomaten als Beschreibungsmittel für gedächtnisbehaftetes Systemverhalten, welche meist deterministisch ausgeführt werden, und somit endliche Zustandsautomaten (engl. „Finite-State-Machines“, FSM) sind [Lig09].

Ist ein Modell vorhanden, können aus diesem Testfälle generiert werden. [Utt06] beschreiben die Charakteristik der Testfallgenerierung über die angewandten **Testtechniken** (engl.: „Technology“) bei der Generierung von Testfällen und die damit verknüpften Strategien und Techniken zur Steuerung, Auswahl bzw. Einschränkung selbiger (engl.: „Test Selection Criteria“), siehe auch Kapitel 2.2.4.

Zuletzt wird die **Testfalldurchführung** (engl.: „Test Execution“) in „online-“ und „offline-“ Tests differenziert. Im Online-Test reagiert die Testfallgenerierung auf die Ausgaben des SUT. Die Testfallgenerierung findet parallel zur Durchführung statt. Im Offline-Test werden zuerst Testfälle generiert, dann der Test durchgeführt.

Neben diesen Möglichkeiten zur Klassifizierung des MBT nach [Utt07] haben [Roß10] das Modellbasierte Testen in Abhängigkeit seiner Reife und Ausprägung in **modellorientiertes-, modellgetriebenes- und modell-**

zentrisches Testen differenziert. Im modellorientierten Testen werden die verwendeten Modelle hauptsächlich zur Visualisierung der Systemeigenschaften eingesetzt. Sie können als Orientierung und Diskussionsgrundlage in der Entwicklung des Systemverhaltens ebenso eingesetzt werden wie im späteren Verlauf des Entwicklungsprozesses als Grundlage für die zu erzielende Testabdeckung. Das modellgetriebene Testen hat im Vergleich zur Modellorientierung eine höhere Reife und Ausprägung hinsichtlich der Fokussierung auf das Modell, welches nunmehr im Mittelpunkt des Testprozesses steht. So werden durch den Einsatz entsprechender Modellierungswerkzeuge Modelle erzeugt, aus welchen im Folgenden durch den Einsatz von Testfallgeneratoren, abstrakte oder konkrete Testfälle generiert werden können. Im modellzentrischen Testen wird der MBT zum Allgemeingut und Standardvorgehen. In Ergänzung zur Modellorientierung sind hier neben den „vorwärts gerichteten“ Generierungsschritten auch „rückwärts gerichtete“ Transformationen und Rückführungen möglich.

Zum produktiven Einsatz des MBT sind verschiedene **Werkzeuge** erforderlich. In Abhängigkeit der primären Zielsetzung ist jedoch nicht zwangsläufig für jede Testaktivität ein entsprechendes Werkzeug notwendig. Im Folgenden werden einige Werkzeuge vorgestellt.

Zur Planung und Steuerung der Testaktivitäten gibt es Werkzeuge zur Bearbeitung folgender Themen:

- Organisation der Anforderungen
- Organisation der Fehler
- Auswertung und Darstellung von Fehlern

- Generierung der Testdokumentation
- Verwaltung von Testdaten
- Verwaltung von Versions- und Konfigurationsdaten

In den Prozessphasen „Analyse & Design“ sowie „Realisierung & Durchführung“ (siehe Abbildung 8, S. 47) sind die zentralen Werkzeuge des MBT einzusetzen. Hierzu gehören:

- Testdatengeneratoren (Datenbankbasiert, Codebasiert, Schnittstellenbasiert oder Spezifikationsbasiert). Auf Basis eines Modells werden Testdaten für die Erstellung von abstrakten und/oder konkreten Testfällen erzeugt.
- Testfalleditoren, welche auf Basis eines abstrakten Testfallmodells konkrete Repräsentationen des Testfalls zur manuellen oder automatischen Testdurchführung erzeugen.
- Testfallgeneratoren, die basierend auf einem Modell des Systemverhaltens, mehrere Testfälle bzw. Testskripte, automatisch nach konfigurierbaren Abdeckungskriterien erzeugen.
- Testwerkzeuge zur automatisierten Testdurchführung funktionaler Systeminhalte wie Testroboter oder Capture/Replay-Tools
- Last- und Performancetestwerkzeuge zur automatisierten Durchführung nicht funktionaler Tests.

Die großen Potenziale des MBT liegen in der Testfallgenerierung und Testdurchführung. Dies ist insofern von besonderer Bedeutung, da durchschnittlich 85 % der Testaufwände in die Analyse des SUT und die Erstellung von Testfällen, sowie deren Durchführung investiert werden [Roß10]. Aus

einem System- oder Testmodell können durch Anwendung definierter Testtechniken Testfälle automatisch generiert werden. Die Generierung kann schnell durchgeführt werden, ist transparent und ermöglicht eine eindeutige und nachvollziehbare Systemabdeckung.

Besteht die Notwendigkeit oder das Vorhaben, vorhandene Testaktivitäten und Prozesse weiterzuentwickeln, um das modellbasierte Testen in der Praxis umzusetzen, ist die viel zitierte Aussage „Automating chaos leads to faster chaos“ von [Gra96] zu beachten. Es ist von Bedeutung, dass die verwendeten Testprozesse und deren Inhalte eine gewisse Reife besitzen, um die Weiterentwicklung, hin zum MBT, realisieren zu können. Andernfalls besteht die Gefahr, dass ohnehin instabile und unreife Testprozesse durch die Ergänzung komplexer, modellbasierter Inhalte in der Praxis nicht umgesetzt werden können und die weiteren Testaktivitäten chaotisch und ineffizient verlaufen. Im folgenden Kapitel wird daher auf Modelle und Methoden eingegangen, welche die Reife eines Testprozesses bewerten und zu verbessern vermögen.

2.3 Modelle & Methoden zur Verbesserung von Prozessen

Um einen konkreten Testprozess zu verbessern ist es notwendig, den aktuellen Prozess und alle verwendete Methoden und Tools in Ihrer Gesamtheit zu erfassen. Die Festlegung der konkreten Ist-Situation erweist sich in der Praxis oft schon als Herausforderung, insbesondere wenn Testprozesse und Methoden nur oberflächlich dokumentiert sind [Pol02]. In der Literatur findet sich eine Vielzahl verschiedener Modelle und Methoden, welche zur Verbesserung von Prozessen eingesetzt werden können. Durch

deren Einsatz lassen sich Prozessverbesserung auf einer argumentierbaren, wissenschaftlichen Grundlage aufbauen.

Ein Modell zur Testprozessoptimierung hat nach [Pol02] im Gegensatz zu einer Methodik keinen bestimmten Testprozess als Grundlage, sondern gilt prinzipiell für jeden Testprozess. Das Modell ist abstrakt-theoretischer Idealzustand und damit Benchmark für den realen Ist-Prozess. Eine Methodik schreibt hingegen einen bestimmten Entwurf für den Testprozess vor. Sowohl die Verwendung eines Modells als auch einer Methodik führt zu Ansätzen der Prozessverbesserung. Im Folgenden sollen einige Methoden und Modelle vorgestellt werden, die im Kontext der Verbesserung eines Prozesses zum Test der softwarebasierten MMS von Interesse sind. In Kapitel 2.4 werde diese evaluiert, um das optimale Modell/die optimale Methodik zur Bewertung und Verbesserung der Reife im Testprozess einer MMS für ein Infotainmentsystem zu identifizieren.

2.3.1 Total Quality Management

Das TQM ist eine Methode zur kontinuierlichen Verbesserung der Produktqualität mit präventivem Charakter. Zentrales Element ist die Kundenorientierung. Die Verbesserung des Entwicklungsprozesses wird zu einem kontinuierlichen und ganzheitlichen Prozess, der kein Ende hat. Die zentralen Ansätze des TQM sind nach [Lig09] und [Spi08]:

- „Null-Fehler-Programm“: Nur fehlerfreie Produkte sind akzeptabel.
- „Continuous Improvement Process“ (CIP): Das Prinzip der ständigen Veränderung zum Besseren (auch Kaizen genannt).

- „Total Quality Control“ (TQC): Ein System zur Entwicklung, Aufrechterhaltung und Verbesserung der Qualität im Marketing, in der Entwicklung, in der Produktion und im Kundendienst.
- „Company-Wide Quality Control“ ist ein Konzept, das TQC um die Komponente der Mitarbeiterorientierung erweitert.

Betrachtet man die MMS als ein Produkt, dessen Qualität verbessert werden soll, sind durch die Anwendung der Methodik durchaus Ansätze zu erwarten, welche zu Verbesserungen im Entwicklungsprozess und folglich auch im Produkt führen können. Weitere Informationen zur Methodik finden sich in entsprechender Fachliteratur wie [Lig09] oder [Spi08].

2.3.2 Six Sigma

Ähnlich wie das TQM ist auch „Six Sigma“ eine Methode zur allgemeinen, kontinuierlichen Qualitätsverbesserung. Im Gegensatz zum TQM steht jedoch kein „Null-Fehler-Programm“ im Mittelpunkt, sondern die Kontrolle von Prozess- und Produkt-Beziehungen. Schwerpunkt ist hier der Einsatz definierter Fehlermaße. Der Fokus liegt dabei aber ebenfalls auf der Kundenzufriedenheit [Spi08]. Umfassende Beschreibungen zu Six Sigma finden sich in zahlreicher Literatur, so z. B. in [Tou09] oder [Mag01].

2.3.3 Capability Maturity Model Integration

Entwickelt an der Carnegie Mellon University, ist das „CMMI“ ein Modell zur Messung und Verbesserung des Reifegrades einer Softwareentwicklung. Es ist die im Jahre 2002 veröffentlichte Weiterentwicklung des „CMM“ (Capability Maturity Model). Ziel des CMMI ist es, durch eine Verbesserung der Prozesse

zur Softwareerstellung auch die Qualität des Produkts zu verbessern. Der Entwicklungsprozess ist hierfür in 22 Prozessgebiete aufgeteilt. Zu jedem der Prozessgebiete kann jeweils auf Basis definierter, spezifischer Ziele ein separater Fähigkeitsgrad (0 = „Incomplete“ bis 5 = „Optimizing“) ermittelt werden [Utt06] [Spi08]. Je nach Fähigkeitsgrad der einzelnen Prozessgebiete lässt sich der Reifegrad für den gesamten Softwareentwicklungsprozess bestimmen. Rückschlüsse aus dem Reifegrad zu einzelnen Fähigkeiten sind nicht zwangsläufig möglich. Die 5 möglichen Reifegrade werden wie folgt unterschieden:

Reifegrad 1: „Initial“ (Keine Anforderungen, unzureichende Prozessdefinition).

Reifegrad 2: „Managed“ (Projekte sind organisiert, ein gleichartiges Projekt kann erfolgreich wiederholt werden).

Reifegrad 3: „Defined“ (Reifegrad zwei erfüllt und Projekte folgen einem angepassten Standardprozess, es gibt eine kontinuierliche Prozessverbesserung).

Reifegrad 4: „Quantitatively Managed“ (Reifegrad drei erfüllt, ergänzt durch statistische Prozesskontrolle).

Reifegrad 5: „Optimizing“ (Reifegrad vier erfüllt und Prozesse werden auf Basis einer statistischen/quantitativen Prozesskontrolle systematisch und zielorientiert verbessert).

Das CMMI stellt dem Anwender ein Modell zur Verfügung, durch welches der Ist-Zustand des eigenen Prozesses feststellbar ist und im Weiteren mögliche Schritte zur Optimierung des Prozesses genannt werden.

2.3.4 SPICE: ISO/IEC 15504

Das Modell „Software Process Improvement and Capability Determination“, oder auch „SPICE“, ist ein im Rahmen der [ISO04] festgelegter, internationaler Standard, um Unternehmensprozesse mit Schwerpunkt in der Softwareentwicklung zu bewerten. SPICE ist ein Verbesserungsansatz, mit welchem einerseits die Prozessfähigkeit von Lieferanten bestimmt werden kann und andererseits Verbesserungen in Prozessen der eigenen Organisation ermittelt werden können [Spi08]. SPICE unterscheidet 49 Teilprozesse, welche neun verschiedenen Prozessgruppen und drei Prozesskategorien zugeordnet werden können (siehe Abbildung 16).

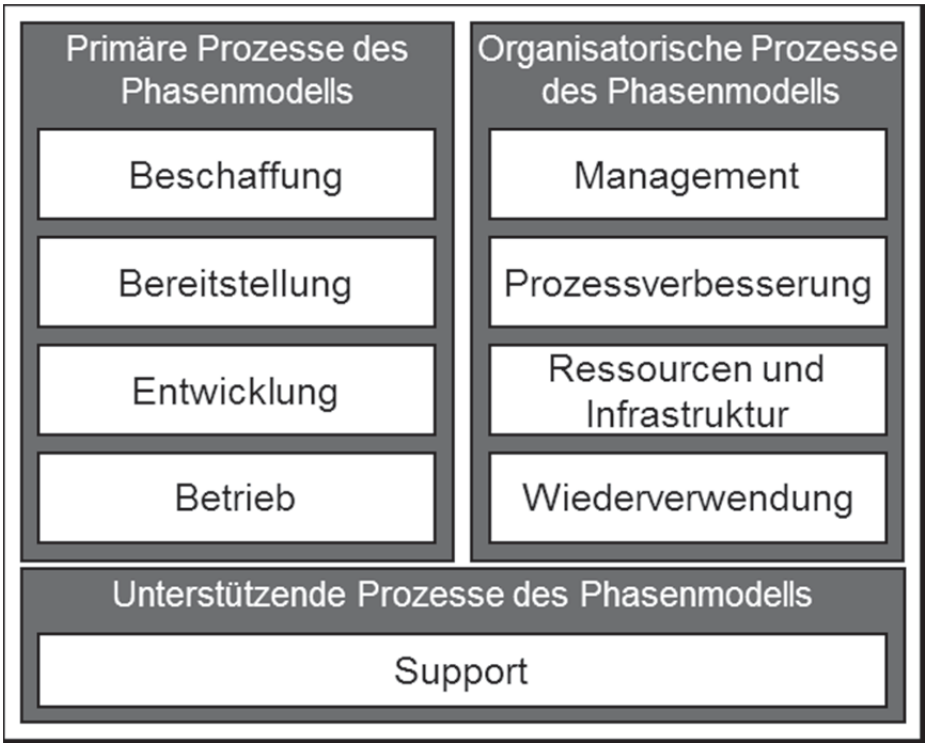


Abbildung 16: „SPICE“ Prozessgruppen und Kategorien

Anhand der in der Norm vorgegebenen Prozessattribute lässt sich im Assessment eine vorher definierte Teilmenge der 49 Teilprozesse auf ihre

Fähigkeit/Reife hin überprüfen und einem der bestehenden 5 Fähigkeitsgrade zuordnen [Lig09]. Diese sind in Anlehnung an die Fähigkeitsgrade des CMMI (0 = „Incomplete“ bis 5 = „Optimizing“) formuliert.

Durch die Auswechslung des exemplarischen Prozesses (ISO/IEC 15504-5:2006) wurden branchenspezifische Modelle ermöglicht. So konnte „Automotive SPICE“ speziell für die Bedürfnisse der Automobilindustrie entwickelt werden.

Dieses Modell eignet sich insbesondere zur Bewertung der Leistungsfähigkeit von Entwicklungsprozessen verschiedener Steuergeräte-lieferanten. Auch das CMMI-Modell kann als Referenz und exemplarischer Prozess in SPICE eingesetzt werden. Weiterführende Literatur zum Thema findet sich unter anderem in [Höh09], [Hör06] und [Spi08].

2.3.5 Critical Testing Process

Der CTP von [Bla03] beschreibt den Testprozess, indem nur die kritischen Aspekte des Testens betrachtet werden. „Kritisch“ bedeutet hier, dass der zu betrachtende Aspekt entscheidend für den Erfolg des Testens als Ganzes ist. Insgesamt werden zwölf kritische Aspekte definiert. Der erste Aspekt ist der Testprozess an sich, welcher in vier Phasen unterteilt werden kann:

1 Planung (plan):

- Umfeld berücksichtigen
- Risiken managen
- Testaufwand abschätzen
- Tests planen

2 Vorbereiten (prepare):

- Testtteam aufstellen
- Testsystem entwickeln

3 Durchführen (perform):

- Testreleases managen
- Tests durchführen

4 Vervollständigen (perfect):

- Abweichungen managen
- Ergebnisse berichten
- Änderungen managen

Zu jedem Aspekt gibt es Checklisten mit Fragestellungen, die mit „erfüllt“ oder „nicht erfüllt“ beantwortet werden können. Die Checklisten beschreiben auf diese Weise einen Modellprozess, an dem sich der konkrete Prozess messen muss.

Ergänzend hierzu werden quantitative Metriken berücksichtigt: „Fehlerfindungsrate“, „Return of Investment“ des Testens, die erreichte Überdeckung der Anforderungen und Risiken, indirekte Kosten eines Testrelease sowie die Anzahl der zurückgewiesenen Fehlerberichte [Bla03], [Spi08]. Eine ausführliche Beschreibung des CTP findet sich in [Bla03].

2.3.6 Systematic Test and Evaluation Process

Die STEP-Methodik wurde 1985 vorgestellt und beschreibt einen Testprozess, der als paralleler Prozess zum Softwareentwicklungsprozess zu verstehen ist. Basis der Methodik ist das Prinzip der frühzeitigen Fehlererkennung durch Validation der Anforderungen und des Entwurfs, bevor die tatsächliche Implementierung stattfindet. Der Testprozess wird von der Methodik in sechs Phasen eingeteilt:

1. Planen: Risiken werden ermittelt und die Teststrategie bestimmt.
2. Analysieren: Testziele, Testbedingungen und Testanforderungen werden festgelegt.

3. Entwurf: Die Spezifikation der Testfälle wird erstellt.
4. Implementierung: Testfälle und Testumgebung werden generiert und bereitgestellt.
5. Ausführung: Die Testfälle werden inkl. notwendiger Re-Tests durchlaufen.
6. Wartung: Tests werden bei Änderungen gesichert und aktualisiert.

Es wird mindestens ein Komponenten- und Akzeptanztest vorgesehen. Bei umfangreichen Neuentwicklungen sind weitere Teststufen notwendig. Für jede Stufe ist die Planung der Strategie, Bereitstellung der Testware und Messung des Verhaltens durchzuführen. Hier sind einzelne Aktivitäten definiert, um das Testziel erreichen zu können. Weitere Informationen zur Methodik können [Cra02] entnommen werden.

2.3.7 Testing Maturity Model

Im Jahr 1996 veröffentlichte das „Illinois Institute of Technology“ das TMM, erstellt auf Grundlage des CMM. Die Entwickler erkannten die Notwendigkeit eines Modells, das sich im Gegensatz zu CMM gänzlich auf den Testprozess konzentriert. Ähnlich dem CMM wurden für das TMM fünf Reifegradstufen definiert, um die Fähigkeiten eines Testprozesses objektiv und anschaulich einordnen zu können. Jede der Reifegradstufen beinhaltet mehrere Reifeziele:

1. Initial

   Die Phase enthält keine Reifeziele. Testen findet unstrukturiert und nicht definiert statt. Die Software ist von minderer Qualität. Notwendige

Ressourcen zum Testen sind nicht oder in nicht ausreichendem Umfang vorhanden.

2. Phasendefinition

   Reifeziele: Testpolitik und Testziele, Testplanung, Testtechniken, Testmethoden und Testumgebung. Testen ist ein eigenständiger Prozess. Es findet eine Testplanung und Strategiebestimmung statt. Es werden Verfahren verwendet, um Testfälle aus Anforderungen abzuleiten. Die Tests finden erst spät im Entwicklungsprozess statt.

3. Integration

   Reifeziele: Testorganisation, Testschulungsprogramm, Testprozessintegration im Entwicklungsprozess, Kontrolle und Überwachung. Die Testplanung findet frühzeitig statt und der Testprozess ist im SW-Entwicklungsprozess integriert. Eine Risikobetrachtung fließt mit in die Anforderungen an die Teststrategie ein. Teilweise werden Reviews durchgeführt.

4. Management und Metriken

   Reifeziele: Peer Reviews, Testmetriken, Reifeziele und SW-Qualitätsmanagement. Der Testprozess ist definiert und quantifizierbar. Reviews werden systematisch durchgeführt. Testfälle werden zentral in einer Datenbank verwaltet und Testmetriken werden verwendet.

5. Optimierung, Fehlerprävention, Qualitätsregelung

   Reifeziele : Fehlerprävention, Testprozessoptimierung, Qualitätskontrolle und Qualitätsregelung. Die vorangegangenen Stufen sind erreicht, Kosten und Effektivität der Testmethoden sind messbar. Testwerkzeuge werden ausgewählt und evaluiert. Durch ständige Verbesserungen des Testprozesses werden Fehler immer früher im Entwicklungsprozess erkannt und vermieden.

Zu jedem Reifeziel sind Teilziele formuliert, um Aktivitäten und Durchführung noch konkreter zu beschreiben [Bur96]. Die Bewertung des eigenen Testprozesses kann mit Hilfe des TMM-Assessment Modells (TMM-AM) stattfinden. Zur Durchführung gibt das TMM-AM dem Nutzer einen Fragebogen, den Ablauf des Assessments sowie Teamschulung und Auswahlkriterien vor. Mit Hilfe des Fragebogens kann der untersuchte Prozess in das Testmodell eingeordnet werden. Eine detaillierte Beschreibung des TMM findet sich in [Bur96].

2.3.8 Test Process Improvement

TPI ist ein Bewertungsmodell zur Bestimmung der Reife eines Testprozesses in einer Organisation. Das Modell wurde auf Basis praktischer Erfahrung von der Firma „Sogeti Nederland B.V.“ entwickelt. Der Testprozess wird vom TPI-Modell in 20 Kerngebiete aufgeteilt. Jedes der 20 Gebiete wird unter Verwendung einer umfassenden Checkliste separat bewertet. Zu jedem Kerngebiet sind dazu zu erreichende Kontrollpunkte formuliert. Neben den Kontrollpunkten sind für jede Ebene eines jeden Kerngebiets konkrete Optimierungsvorschläge formuliert, um die nächste Ebene zu erreichen. Mit Hilfe einer Matrix (siehe Kapitel 4.2), welcher im Modell des TPI eine zentrale Bedeutung zukommt, kann die Ist-Situation eines Testprozesses schließlich anschaulich ermittelt werden. Ebenso kann eine angestrebte Situation bestimmt werden und die notwendigen Schritte zum Erreichen dieser sind bereitgestellt [Sog09], [Pol02], [Spi08]. Eine detaillierte Beschreibung des Modells und der 20 Kerngebiete findet sich in Kapitel 4.2 sowie in [Sog09].

2.3.9 Test Management approach

TMap wurde 1995 von Pol, Teunissen und Van Veenendaal veröffentlicht [Pol02]. Die Methodik war Grundlage zur Entwicklung des TPI-Modells (Kapitel 2.3.8). TMap ist im Vergleich zum TPI-Modell stärker auf den Test selbst fokussiert und strukturiert selbigen, wohingegen das TPI den gesamten Testprozess optimiert.

TMap stellt dem Anwender ein strukturiertes Testkonzept zur Verfügung. Dieses Konzept betrachtet vier Aspekte:

1. *Phasenmodell*:

   Testaktivitäten werden in einem generischen Phasenmodell dargestellt. Es definiert die Vorgehensweise und kann in verschiedenen Teststufen eingesetzt werden. Es besteht aus den Phasen „Planung und Verwaltung“, „Vorbereitung“, „Spezifikation“, „Testdurchführung“ und „Abschluss“.

2. *Technik*:

   TMap stellt dem Tester Techniken zur Verfügung, um die verschiedenen Testaktivitäten im Rahmen des Testprozesses durchführen zu können. Hierzu gehören eine Strategiebestimmung, eine Testpunktanalyse, verschiedene Checklisten und Test-Spezifikationstechniken.

3. *Infrastruktur und Tools*:

   Die notwendige Testumgebung, Test-Tools und Büroeinrichtung für anforderungsgerechtes Testen werden erläutert.

4. *Organisation*:

   Wie sich Testaktivitäten in die Organisation Unternehmens einfügen lassen und welche Aspekte hierbei Beachtung finden sollten, wird thematisiert.

Betrieblicher Testprozess, strukturelle Testorganisation, Testmanagement, Personal und Ausbildung sind nach TMap dabei die zentralen Aspekte der Organisation.

Zu jeder Phase des beschriebenen Phasenmodells beschreibt TMap detailliert durchzuführende Aktivitäten. Zu jeder Aktivität werden jeweils Ziel, Arbeitsweise, Produkt und anzuwendende Techniken beschrieben. Weiterführende Informationen und Hintergründe zu TMap finden sich in [Pol02].

Es zeigt sich, dass verschiedene Ansätze zur Verbesserung eines (Test-) Prozesses vorhanden sind. Im folgenden Kapitel soll einer dieser Ansätze ausgewählt werden, um den konkreten Testprozess einer softwarebasierten MMS für ein Infotainmentsystem hinsichtlich seiner Reife zu bewerten und zu verbessern. Aus Gründen der Übersichtlichkeit werden hier die Begriffe „Modell“ und „Methodik“ Synonym verwendet.

2.4 Prozessmodell für den Test einer softwarebasierten MMS

2.4.1 Kriterien zur Auswahl eines Prozessmodells

Wie in Kapitel 2.2.5 beschrieben, ist für die Weiterentwicklung eines Testprozesses und verwendete Methoden ein ausreichend hoher Prozessreifegrad von zentraler Bedeutung. Ist dieser nicht erreicht, steigt die Gefahr, dass die Einführung neuer Prozesselemente und Methoden kontraproduktive oder chaotische Auswirkungen hat. Um ein geeignetes Modell zur Verbesserung des Testprozesses der MMS im Kfz zu bestimmen, werden im Folgenden verschiedenen Kriterien zur Bewertung der Prozessmodelle definiert. Anhand

dieser Kriterien werden im Anschluss die verschiedenen, in Kapitel 2.3 vorgestellten Ansätze zur Prozessverbesserung, verglichen und bewertet. Das Prozessmodell sollte folgenden Kriterien genügen:

- **Durchführbar:** Die hinreichende Beschreibung einer Methodik zur Prozessverbesserung ist ein notwendiges Kriterium, um in die Auswahl aufgenommen zu werden. Ohne eine ausreichende Beschreibung können die Eigenschaften eines Ansatzes nicht hinlänglich beurteilt werden.
- **Praxisorientiert:** Das Prozessmodell soll der Realität möglichst nahe kommen und dem Prozess zum Test einer MMS im Kfz möglichst entsprechen. Der praktische Nutzen steht im Vordergrund.
- **Objektiv und detailliert:** Das Modell soll möglichst objektiv den Zustand eines Testprozesses beschreiben. Wird der Prozess von zwei verschiedenen Gutachtern bewertet, sollte ein nahezu identisches Ergebnis resultieren. Ein zu allgemeines Modell mit großem Interpretationsspielraum ist an dieser Stelle nicht zielführend.
- **Einordnung des Ist-Zustands:** Das Modell soll eine einfache Feststellung des Ist-Zustandes und der zwischenzeitlichen Fortschritte verschiedener Verbesserungsaktivitäten ermöglichen. Das Aufzeigen von Reifegraden ist erwünscht.
- **Konkrete Optimierung:** Das Modell soll konkrete Lösungen und Möglichkeiten zur Verbesserung aufzeigen, um notwendige Verbesse-rungen gezielt durchführen zu können.
- **Aufzeigen der Prioritäten:** Das Modell soll die Priorität verschiedener Verbesserungen aufzeigen. Auf diese Weise kann die Vorgehensweise zur Optimierung des Testprozesses methodisch bestimmt werden.

Auf Basis dieser Kriterien soll die optimale Methodik zur Reifegradbestimmung und Prozessverbesserung gewählt werden. Es wird unterschieden, ob die Methodik ein Kriterium nicht, teilweise oder ganz erfüllt.

- ○ Kriterium nicht erfüllt.
- ◐ Kriterium teilweise erfüllt.
- ● Kriterium erfüllt.

Im Folgenden werden die in Kapitel 2.3 vorgestellten Ansätze zur Verbesserung eines Testprozesses hinsichtlich der hier definierten Kriterien bewertet.

2.4.2 Bewertung und Auswahl eines Prozessmodells

In Kapitel 2.3 wurden insgesamt neun verschiedene Methoden und Modelle vorgestellt, durch deren Verwendung ein Entwicklungs-/Testprozess verbessert werden kann. Die Bewertung der verschiedenen Ansätze nach den in Kapitel 2.4.1 definierten Kriterien führte zu dem im Tabelle 3 dargestellten Ergebnis.

Total Quality Management (TQM):

Die Prinzipien des TQM lassen sich auch auf den Testprozess übertragen, beziehen sich aber grundsätzlich auf den gesamten Entwicklungsprozess. Eine Reifegradbestimmung des Status quo ist mit TQM nicht möglich. Auch konkrete Optimierungsmaßnahmen und deren Priorisierung sind durch das Anwenden der Methode nur bedingt möglich.

Six Sigma:

Die Grundsätze aus Six Sigma können auch auf den Testprozess übertragen

werden, beziehen sich aber generell auf den gesamten Entwicklungsprozess. Eine Reifegradbestimmung des aktuellen Status ist mit Six Sigma nicht möglich. Konkrete Optimierungsmaßnahmen sind nach Anwendung der Methode nur bedingt zu erwarten. Auch deren Priorisierung wird durch die Methode nicht unterstützt.

Tabelle 3: Vergleich verschiedener Methoden zur Prozessverbesserung

Methodik/Modelle / Kriterien	TQM	Six Sigma	CMMI	SPICE	CTP	STEP	TMM	TPI	Tmap
Praxisorientiert	◐	◐	◐	◐	●	●	●	●	●
Objektiv und detailliert	○	○	◐	◐	◐	◐	◐	●	●
Einordung des Ist-Zustands	○	○	●	●	◐	◐	●	●	◐
Konkrete Optimierung	◐	◐	◐	◐	◐	◐	●	●	●
Prioritäten	○	○	◐	◐	◐	○	◐	●	◐

○ Kriterium nicht erfüllt; ◐ Kriterium teilweise erfüllt.
● Kriterium erfüllt.

Capability Maturity Model Integration (CMMI):

Der modellbasierte Verbesserungsansatz CMMI kann zur Bewertung von Prozessen, als auch zur Prozessverbesserung genutzt werden. Allerdings findet sich die Aktivität „Test“ nur in zwei der 22 Prozessgebiete wieder. Beide Prozessgebiete, „Verifikation“ und „Validation“, müssen den Fähigkeitsgrad drei erreichen, um mit der Organisationseinheit den dritten Reifegrad zu erlangen. CMMI gibt dem Anwender ein Modell an die Hand, welches es ermöglicht, den eigenen Entwicklungsprozess in Reifegradstufen einzuteilen. Der Ist-Zustand des eigenen Prozesses ist so feststellbar. Dabei wird der Testprozess aber nicht explizit betrachtet und ist wenig detailliert beschrie-

ben. Anhand der beschriebenen Ziele für die einzelnen Prozessgebiete ist eine Richtung zur weiteren Optimierung vorgegeben, konkrete Maßnahmen zur Erreichung der Ziele sind aber nicht vorhanden. Nachdem die Aktivität „Testen“ in nur zwei Prozessgebieten konkret behandelt wird, lassen sich die gegebenen Optimierungsvorschläge auch nur bedingt priorisieren.

SPICE: ISO/IEC 15504:

Wer sich mit dem Testen im Entwicklungsprozess beschäftigt, findet Hinweise in den Teilprozessen „Software construction, -integration, -testing“, „System-integration“ und „-testing“. Aus der Prozessgruppe „Support“ ist zudem der Teilprozess „Verification“ und „Validation“ relevant. SPICE gibt dem Anwender die Möglichkeit, den eigenen Entwicklungsprozess hinsichtlich seiner Fähigkeiten zu beurteilen. Der Ist-Zustand des ausgewählten Prozesses kann festgestellt werden. Dabei ist der Testprozess aber nicht detailliert beschrieben. Anhand der beschriebenen Ziele für die einzelnen Prozessgebiete ist eine Richtung zur weiteren Optimierung vorgegeben, konkrete Maßnahmen zur Erreichung der Ziele sind nicht vorhanden. Auch eine Priorisierung der notwendigen Aktivität zur Verbesserung des Testprozesses ist nur bedingt möglich.

Critical Testing Process (CTP):

Ergebnis nach Anwendung der Methode ist eine Übersicht der aktuellen Schwachstellen im Prozess. Empfehlungen zur Verbesserung sind anschließend umzusetzen. Hier liefert CTP jedoch nur allgemeine Verbesserungsvorschläge. Die konkrete Formulierung und Priorisierung von Verbesserungen wird vom Anwender gefordert, vorhandene Checklisten können hier als allgemeine Hilfestellung betrachtet werden. Der Testprozess wird durch die

Methode zwar analysiert und einem idealen Modell gegenübergestellt, eine konkrete Einordnung des Ist-Zustandes in Reifegradstufen findet jedoch nicht statt.

Systematic Test and Evaluation Process (STEP):

STEP ist, ähnlich dem CTP-Ansatz, eine Definition der Vorgehensweise im Testprozess. Der Fokus liegt auf dem Testprozess und nicht dem gesamten Entwicklungsprozess. Es lassen sich gut Schwachstellen im Testprozess identifizieren, eine konkrete Einstufung in Reifegrade ist jedoch nicht vorgesehen. Auch konkrete Vorschläge zu Verbesserungen sucht der Anwender vergeblich. Eine Priorisierung der nötigen Verbesserungen ist folglich ebenfalls nicht möglich.

Testing Maturity Model (TMM):

Das TMM bezieht sich speziell auf den Testprozess und gibt neben einer Einordnung des Status Quo auch Empfehlungen zu Verbesserungen. Nötige Verbesserungen können anhand eines Fragebogens nur in eingeschränktem Umfang priorisiert werden.

Test Process Improvement (TPI):

Das TPI-Modell ähnelt dem TMM, ist aber deutlich detaillierter. Das Modell gibt neben einer detaillierten Beschreibung des Status Quo auch konkrete Empfehlungen zur Verbesserung des Testprozesses. Durch den Einsatz der TPI-Matrix ist eine Priorisierung identifizierter Optimierungsaktivitäten möglich. TPI hat sich in seiner Anwendung bereits etabliert und wird unter anderem von [Göt09] empfohlen, um konkrete Verbesserungsschritte zu definieren.

Test Management approach (TMap):

TMap verfolgt einen praxisorientierten Ansatz mit Fokussierung auf den Test. Die beschriebenen Techniken wie die Testpunktanalyse (TPA®) und verschiedene Checklisten geben dem Anwender konkrete Werkzeuge an die Hand, um die Testaktivität zu verbessern. TMap ist eine Test-Methodik und kein Test-Modell wie etwa TPI. Aus diesem Grund ist die Bestimmung des Ist-Zustands und eine Priorisierung der notwendigen Optimierungsschritte mit TMap kaum möglich. Lücken und Verbesserungsmöglichkeiten lassen sich durch die Anwendung der Methodik zwar gut erkennen, es findet aber keine Einordnung des „absoluten“ Reifegrades statt. [Pol02]

Ergebnis:

Das TPI-Modell erfüllt in dieser Beurteilung, die in Kapitel 2.4.1 definierten Kriterien am besten. Möchte man den Prozess zum Test einer softwarebasierten MMS im Kfz verbessern, empfiehlt sich die Verwendung des Modells. Das TPI überzeugt insbesondere durch die Möglichkeit der Priorisierung notwendiger Verbesserungsschritte. Die Aufteilung des Testprozesses in verschiedene Kernbereiche ermöglicht dabei zunächst eine detaillierte und differenzierte Betrachtung des Testprozesses. Die Zusammenführung der verschiedenen Kernbereiche in der TPI Matrix fügt die aus dem ersten Schritt gewonnenen Erkenntnisse, unter Berücksichtigung der vorhandenen Abhängigkeiten, wieder zu einem Gesamtbild zusammen. Diese Systematik überzeugt und ermöglicht die Einleitung notwendiger Schritte zur Verbesserung der existierenden Prozesse. Kritisch zu bemerken ist der hohe zeitliche Aufwand, welcher durch die detaillierte und differenzierte Betrachtung des Testprozesses in der Anwendung des TPI entsteht.

3 Problemstellung, Ansatz und Vorgehensweise

Über die softwarebasierte MMS im Kfz werden immer komplexere Infotainmentsysteme bedient. Neben den funktionalen Erweiterungen in der Software wurde die MMS multimodal und ermöglicht neben der haptischen auch eine akustische Bedienung der Systeminhalte. Sich verändernde Rahmenbedingungen und eine fortschreitende inhaltliche Erweiterungen der MMS führen in der Entwicklung zu erheblichen Herausforderungen. Diese sind zu lösen, um im zukünftigen Marktumfeld weiterhin qualitativ hochwertige Produkte anbieten zu können. Im Folgenden werden zentrale Herausforderungen ausgearbeitet und ein Ansatz vorgestellt, diese zu bewältigen.

3.1 Herausforderungen in der Infotainment-Entwicklung

Moderne Infotainmentsysteme haben mittlerweile einen wertbestimmenden und markenspezifischen Rang im Automobil erobert. Die Gestaltung der MMS kann Alleinstellungsmerkmale schaffen und die Innovationskraft sowie den Qualitätsanspruch des Herstellers untermauern [Spa11]. Die Funktionsdichte der Systeme erhöht sich dabei mit jedem Entwicklungszyklus. Das Aufgabenspektrum reicht von der Wiedergabe verschiedener, multimedialer Inhalte über die Bereitstellung von Fahrzeuginformationen bis hin zur Kommunikation mit anderen Menschen oder Anbietern von Diensten [Lüt12]. Durch die Vergrößerung der Funktionsumfänge muss die MMS hinsichtlich des Bedienkomforts und der Bediensicherheit stetig verbessert und erweitert werden. Gleichzeitig ist von Bedeutung, dass die funktionale Qualität der implementierten MMS sichergestellt werden kann. Revolutionäre Bedienkonzepte sind

wertlos, wenn die Umsetzung scheitert und eine mangelhafte funktionale Qualität den Bedienkomfort und die Bediensicherheit einschränkt. Westkämper und Warnecke stellen fest, dass „der Kunde ein komplexes Produkt gewöhnlich nur von seiner Oberfläche her wahrnimmt“ und sich zunächst hauptsächlich mit dessen Bedienung befasst [Wes02]. Qualitative Mängel der MMS sind für den Kunden folglich mit hoher Wahrscheinlichkeit wahrnehmbar und haben einen unmittelbaren Einfluss auf die Wertigkeit und den Qualitätseindruck des gesamten Produkts.

Die Entwicklung von Bedienkonzepten und deren Validierung zur Sicherstellung der Bediensicherheit und Gewährleistung eines hohen Bedienkomforts ist für die Automobilhersteller aufgrund des vorrangig softwarebasierten Umfeldes eine Herausforderung. Trotzdem können die Hersteller hinsichtlich der Entwicklung von Bedienkonzepten auf umfangreiche Erfahrungen und Fachkompetenzen aus mehreren Jahrzehnten Automobilentwicklung zurückgreifen und aufbauen. Anders ist dies, wenn es um die qualitative Verifikation von Softwaresystemen geht. Software-entwicklung und deren Qualitätsabsicherung gehört nicht zu den (ursprünglichen) Kompetenzbereichen der Automobilindustrie. Eine zentrale Problematik in der Qualitätsabsicherung lässt sich am so genannten „Magischen Dreieck“ zeigen. In Abbildung 17 ist dargestellt, wie sich ein Entwicklungsprojekt im Kontext der Zielkonflikte Qualität, Kosten, Zeit verhält.

Zielkonflikte entstehen, weil die Qualität eines Produkts direkt von den investierten Ressourcen abhängig ist. Eine Verbesserung der Qualität führt zu höheren Zeitaufwänden und damit auch zu höheren Kosten in der Entwicklung.

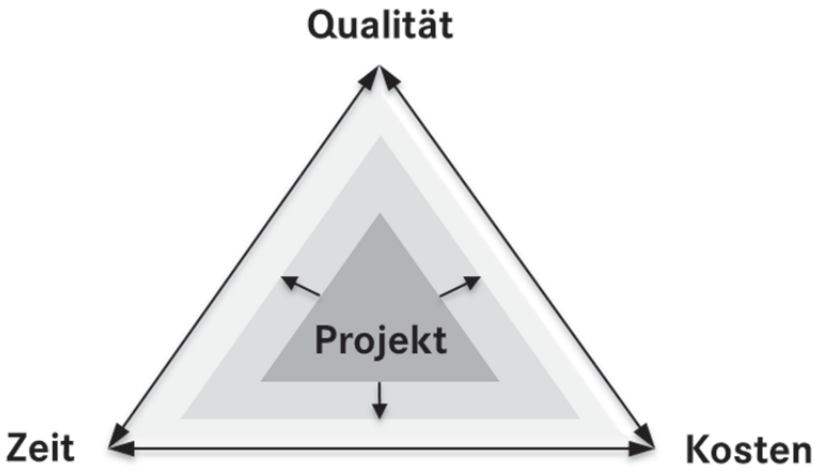


Abbildung 17: Das „Magische Dreieck“, Qualität-Zeit-Kosten

Wird ein Entwicklungsprojekt inhaltlich umfangreicher, folgt daraus zwangsläufig die Notwendigkeit höherer Ressourceneinsätze, um das Qualitätsniveau zu halten. In der Praxis führen diese Zusammenhänge oft zu einer erforderlichen Managemententscheidung, ob ein Projekt innerhalb der vorgegebenen Zeiten und/oder den festgelegten Kostengrenzen abgeschlossen werden muss oder nach Erreichen der Spezifikations- und Qualitätsziele abgeschlossen werden kann. Hierbei zeigt sich, dass die Erreichung terminlich festgelegter Ziele und die Einhaltung des Budgets meist priorisiert wird. Oft scheint es eher vertretbar, ein qualitativ unausgereiftes Produkt auf den Markt zu bringen, als den Verkaufsstart zu verschieben oder das Budget zu überziehen. Dabei ist zu bedenken, dass Softwarefehler erhebliche Folgekosten nach sich ziehen können.

Durch die beschriebene, stetig ansteigende Funktionsdichte der MMS werden die Entwicklungsprojekte immer umfangreicher. Hinzu kommen weitere Faktoren, welche die Entwicklungsprojekte inhaltlich vergrößern. Das

Produktportfolio der Automobilhersteller hat sich in den vergangenen Jahren enorm erweitert. Verschiedene Baureihen werden mit Infotainmentsysteme ausgerüstet. Zwar werden die Systeme nach Möglichkeit baureihen-übergreifend verbaut, trotzdem bleiben insbesondere im Themenfeld der MMS oft Varianten erhalten. Neben diesem Ausbau an Varianten führen marktspezifische Systemvarianten zu einer Erweiterung der Entwicklungs-umfänge. Die Anforderungen an ein Infotainmentsystem und die MMS unterscheiden sich international erheblich. Gesetzliche Vorgaben, technische, sprachliche und kulturelle Charakteristika machen marktspezifische Varianten in Hard- und Software notwendig.

Um die Softwarequalität des Systems und der MMS mit der beschriebenen Komplexität, Vernetzung und Variantenvielfalt weiterhin sicherzustellen, ist in der Folge ein deutlich höherer Ressourceneinsatz notwendig. In der Praxis stehen diese Ressourcen jedoch nicht zur Verfügung. Dass Entwicklungs-kosten von zentraler Bedeutung sind, ist nicht neu. Die Märkte der westlichen Welt sind weitestgehend gesättigt und die automobile Konkurrenz ist groß. Im internationalen Wettbewerb um Gewinn und Rendite zur Befriedigung von Aktionären und Investoren werden Sparprogramme und Effizienzziele zur Unternehmensleitkultur. Zudem haben sich die Entwicklungszeiten in den vergangenen Jahren tendenziell verkürzt. Auf den ersten Blick scheint dies für die Softwareentwicklung von geringerer Bedeutung, schließlich werden Entwicklungszeiten von fünf Jahren [Eur10] in der Softwareindustrie meist deutlich unterboten.

Branchenspezifische Aspekte führen in der Entwicklung von Infotainment-systemen und deren MMS zu oft zeitkritischen Projektabläufen. Wie bereits

beschrieben, sind komplexe Softwaresysteme erst seit kurzem fester Bestandteil in der Automobilentwicklung. Die entsprechenden Fachkompetenzen zur eigenständigen Entwicklung der Software sind nicht oder nur gering vorhanden. Bereits einfache Radiogeräte werden seit Beginn der Automobilgeschichte von Systemlieferanten zugekauft.

Moderne Infotainmentsysteme sind hoch entwickelte und komplexe Softwareprodukte mit hohen Anforderungen und individuellen Schnittstellen zur Fahrzeugarchitektur und dem Menschen. Das Produkt soll sich funktional, optisch und hinsichtlich der Bedienung in das Gesamtkonzept des Fahrzeugs integrieren. Vor diesem Hintergrund hat sich ein Prozess etabliert, in dem der Automobilhersteller die Anforderungen an das System in einer funktionalen Spezifikation detailliert beschreibt und dem Systemlieferanten als Grundlage zur Implementierung der Software überlässt. Dieser führt die weiteren Entwicklungsschritte zur Implementierung durch und übergibt dem Auftraggeber zu definierten Zeiten Hardwaremuster mit festgelegten Entwicklungsständen der Software. Diese Produkte werden vom Automobilhersteller hinsichtlich der Erfüllung gestellter Anforderungen überprüft. Fehlverhalten der SW und konzeptionelle Änderungswünsche werden schließlich dem Systemlieferanten zurückgemeldet, welcher das System entsprechend anpasst. Diese Iterationen werden bis zum Entwicklungsende wiederholt.

Die Automobilhersteller spezifizieren das erwartete Infotainmentsystem hauptsächlich textuell. Für die Spezifikation der MMS werden im konkreten Fall verschiedene Spezifikationsmodelle parallel verwendet. Textuelle Beschreibungen in Kombination mit Grafikelementen, Dialogstrukturen in teilweise formalen, meist proprietären Zustands- oder Ablaufdiagrammen. Die

Beschreibungen sind detailliert, formal jedoch inkonsistent. Dies führt zu Schwierigkeiten in der Abstimmung zwischen Auftraggeber und Lieferant. Abbildung 18 verdeutlicht die horizontal verlaufende Schnittstelle zwischen Auftraggeber und Softwarelieferanten im V-Modell.

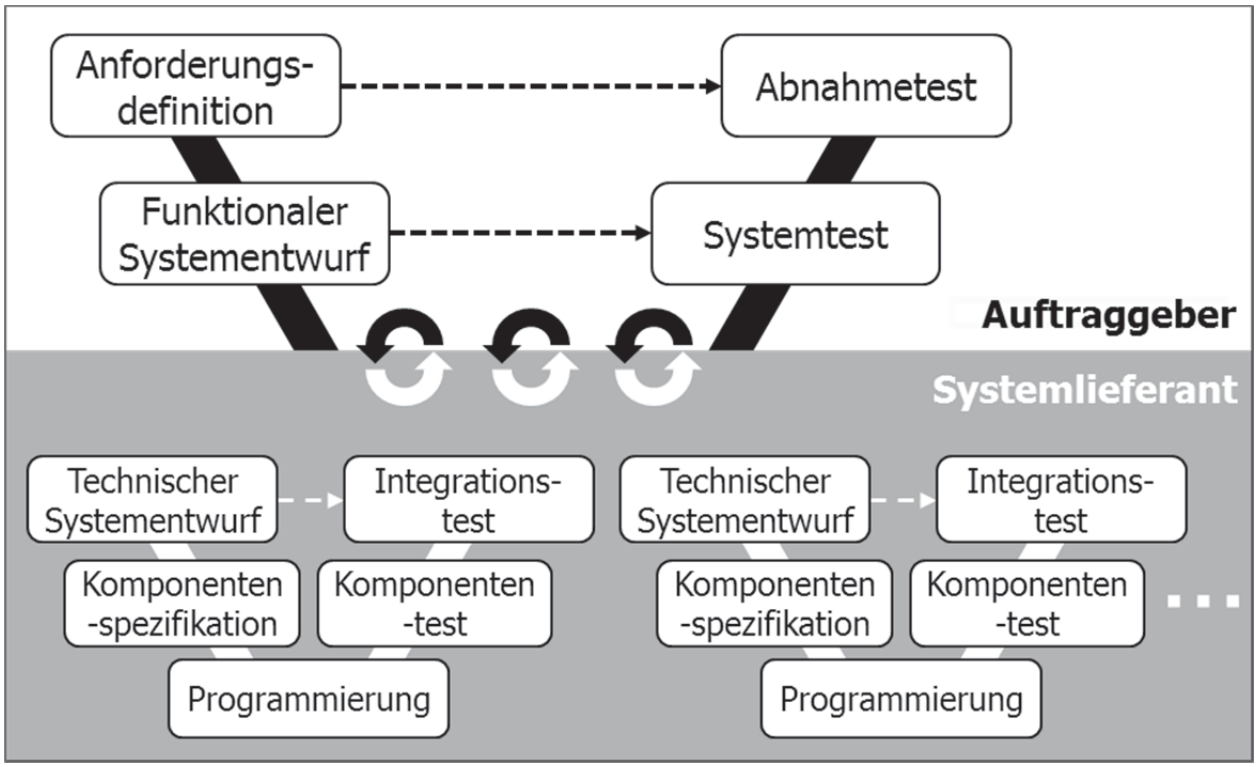


Abbildung 18: Horizontale Lieferantenschnittstelle im V-Modell

Nicht zu vernachlässigen ist im Folgenden der Kostendruck, welcher auf dem Systemlieferanten lastet. In den meisten Fällen stehen mehrere Softwarelieferanten in Konkurrenz um eine Ausschreibung. Die vorgelegten Angebote sind knapp kalkuliert, für die Implementierung und Durchführung von Testaktivitäten sind nur beschränkte Ressourcen verfügbar, um die Wirtschaftlichkeit aus Sicht des Lieferanten zu gewährleisten.

Nach Implementierung der Software ist diese für den Auftraggeber eine „Black-Box“ und kann lediglich gegen die Spezifikation getestet werden. Da eine Abstimmung der Testinhalte zwischen Auftraggeber und Systemlieferanten oft nicht oder nur eingeschränkt stattfindet und dem Systemlieferanten meist unzureichende Vorgaben zur erwarteten Softwarequalität vorgeschrieben sind, werden Testaktivitäten durch den Automobilhersteller notwendig. Hier verlangen die funktionalen Tests der fachlich komplexen Systeme nach Fachkräften für die erforderlichen Testaktivitäten. Die Erstellung von Testfällen zur manuellen Testdurchführung und der Aufbau von automatisierten Tests ist zeitaufwändig und meist wenig effizient. Diese Rahmenbedingungen verschärfen eine bekannte Problematik: Fehler werden in frühen Projektphasen begangen, aber erst in späten Phasen entdeckt.

Dies führt schließlich dazu, dass die Entwicklung der softwarebasierten MMS in einem immer stärker ausgeprägt Spannungsfeld zwischen Qualität, Zeit und Kosten stattfindet. Setzt sich dieser Trend fort, besteht das Risiko, dass die Qualität in einem nicht ausreichenden Maße abgesichert werden kann und das Vertrauen der Kunden in die Produktqualität schwindet.

Um diese Herausforderungen in der Entwicklung einer Softwarebasierten MMS zu lösen, sind im Folgenden Ansätze aufgezeigt, welche das beschriebene Spannungsfeld und die resultierenden Risiken entschärfen können. Im Fokus des weiteren Vorgehens steht die verbale, akustische Mensch-Maschine-Schnittstelle als Teil der softwarebasierten MMS. Bei immer größeren Funktionsumfängen in modernen Infotainmentsystemen kommt der Sprachbedienung zunehmend Bedeutung zu. Bereits heute sind viele Funktionen aufgrund gesetzlicher Vorgaben während des Fahrbetriebs

ausschließlich über ein Sprachbediensystem ansteuerbar. Bislang gibt es jedoch keine praktikable Umsetzung, um die Qualität dieser zukunftsweisenden MMS im beschriebenen, herausfordernden Umfeld auch zukünftig sicherzustellen.

3.2 Ansatz zur Lösung der Herausforderung

Die in Kapitel 3.1 beschriebenen Herausforderungen sind zu bewältigen, um die Qualität der softwarebasierten MMS sicherzustellen und die Potenziale der Sprachbedienung für den Kunden durch abgesicherte Funktionen auch in Zukunft nutzbar zu machen. Um die Qualität weiterentwickelter Dialogsysteme zu gewährleisten, ist heute die Umsetzung neuer Ansätze notwendig, welche den Anforderungen der gegebenen Rahmenbedingungen gerecht werden.

In der nichtfunktionalen Entwicklung und Bewertung von Mensch-Maschine-Schnittstellen werden seit Jahrzehnten Modelle des Menschen verwendet, um die festgelegten Anforderungen zu verifizieren. Mit dem Beweis der im Folgenden vorgestellten Ansätze, könnte gezeigt werden, dass auch in der funktionalen Absicherung ein „Mensch-Modell“ oder „virtueller Tester“ entwickelt werden kann, welcher den Menschen im Test der akustischen MMS zu ersetzen vermag. Zudem soll geprüft werden, ob sich die hierbei entwickelten Prozesse, Methoden und Techniken auch auf den Test der haptischen MMS übertragen lassen.

Der in Kapitel 2.2.5 aufgezeigte, modellbasierte Testprozess hat Potenzial, zum „Gehirn“ des virtuellen Testers zu werden und das Spannungsfeld

zwischen Qualität, Zeit und Kosten in der Entwicklung einer softwarebasierten MMS zu entlasten. In der Softwaretechnik konnte bereits gezeigt werden, dass MBT die Effizienz der Testaktivitäten erhöhen und die Qualitätsabsicherung von Softwaresystemen verbessern kann [Roß10]. Würde sich der Entwicklungs-, bzw. Testprozess der akustischen MMS eines Infotainmentsystems modellbasiert umsetzen lassen, könnten die beschriebenen Herausforderungen gelöst werden. Dieser Ansatz führt zur These der Arbeit, welche wie folgt zusammengefasst werden kann:

Es ist möglich, die akustische MMS eines Infotainmentsystems im Kfz modellbasiert und automatisiert zu testen. Dabei ist es möglich die MMS als Testschnittstelle zu verwenden. Hierdurch entsteht ein „virtueller“ Tester. Die Effizienz und Qualität des Entwicklungs- und Testprozesses kann durch die Implementierung des modellbasierten Testprozesses sowie neue Methoden und Werkzeuge gesteigert werden.

Um die Gültigkeit der These nachzuweisen, steht zunächst der Testprozess im Fokus. Wie in Kapitel 2.2.5 beschrieben, ist für die Umsetzung des MBT-Prozesses zunächst der Reifegrad des aktuellen Testprozesses zu bestimmen. Anschließend können Aktivitäten definiert werden, um den MBT umzusetzen. Zum Beweis der These ist es zielführend, diese in granulare Thesen zu zerlegen. Kann deren Gültigkeit beweisen werden, ist auch der zentrale Ansatz bewiesen.

These A:
Es ist möglich, die akustische MMS eines Infotainmentsystems in einem Standardformat formal zu modellieren.

Die These A bezieht sich auf das erste Elemente im MBT-Prozess: Spezifikation und Modell. Soll der MBT-Prozess umgesetzt werden, sind bereits vorhandene Anforderungsbeschreibungen und Spezifikationen als Systemmodell zu verwenden oder diese in ein nutzbares, formales Testmodell zu überführen. In Bezug auf die in Abbildung 13 (S. 67), Abbildung 14 (S. 68) und Abbildung 15 (S. 69) dargestellten Möglichkeiten zur Verwendung von System- und Testmodell ist eine entsprechende Lösung für die Modellierung der MMS zu entwickeln und umzusetzen.

These B:
Es ist möglich, aus dem durch Beweis von These A definierten Modell der akustischen MMS, Testfälle anhand festgelegter Testtechniken zu generieren. Dabei kann eine definierte Anzahl an Testfällen mit bekanntem Systemabdeckungsgrad generiert werden.

Die These B bezieht sich auf die Möglichkeiten der Testfallgenerierung. Es ist zu beweisen, dass aus dem zum Beweis der These A definierten Modell automatisch Testfälle generiert werden können.

These C:
Es ist möglich, die akustische MMS automatisiert zu testen, ohne auf eine zusätzlich implementierte Testschnittstelle zurückgreifen zu müssen. Die Testumgebung nutzt die MMS selbst als Testschnittstelle und wird durch Nutzung der im Beweis von These B generierten Testfälle zu einem „virtuellen Benutzer“ des Systems.

Die These C bezieht sich schließlich auf die Testdurchführung. Generierte Testfälle sollen nicht nur manuell durchgeführt werden, sondern auch

automatisch. Zum Beweis der These ist eine Testumgebung zu entwickeln, welche zu einem automatischen Test der akustischen MMS in der „Black-Box“ des SUT verwendet werden kann.

Mit dem Beweis der Thesen A bis C wird ein „virtueller“ Tester realisiert, durch dessen Einsatz die Testtiefe und Transparenz der Testaktivitäten verbessert werden kann. Können die Thesen bewiesen werden, ist es möglich, die Qualität der akustischen MMS in komplexen Entwicklungsprojekten abzusichern, ohne die bisher zwangsläufige Folge eines höheren Ressourceneinsatzes. Es ist zu zeigen, dass die Effizienz der Testaktivitäten, trotz der möglicherweise zunächst höheren Ressourceneinsätze (durch die Umsetzung der neuen Testmethodik) gewährleistet ist.

3.3 Vorgehensweise

Zunächst steht der Testprozess als Ganzes im Vordergrund der Betrachtung. In Kapitel 4 werden die Anforderungen an den Testprozess ebenso ausgearbeitet wie die notwendigen Testaktivitäten, um ein modellbasiertes Vorgehen zu ermöglichen. Unter Anwendung der TPI Methode wird der Reifegrad der aktuellen Testprozesse bestimmt. Auf Basis der Ergebnisse können schließlich notwendige Maßnahmen zur Verbesserung festlegt werden. Aufbauend auf die hier geschaffenen Grundlagen können weitere Entwicklungsschritte zur Umsetzung des MBT stattfinden.

In Kapitel 5 werden die Möglichkeiten zur Modellierung der akustischen MMS beleuchtet und damit die Beweisführung zu These A verfolgt. Anforderungen an das Modell werden formuliert und Möglichkeiten zur Bewältigung

praktischer Herausforderungen aufgezeigt. An einem Beispiel wird eine Modellierungssyntax zur Beschreibung der akustischen MMS entwickelt. Abschließend wird geprüft, ob die entwickelte Methodik auf die Beschreibung der haptischen MMS übertragen werden kann.

Nach Abhandlung der Modellierung wird in Kapitel 6 die Beweisführung zu These B aufgenommen. Verschiedene, kommerzielle Werkzeuge zur Testfallgenerierung sind zu analysieren, um schließlich ein Werkzeug zu identifizieren welches den zuvor ausgearbeiteten Anforderungen gerecht wird und die automatische Erstellung von Testfällen für die akustische MMS ermöglicht. Die in Kapitel 5 erstellten Modelle sollen Verwendung finden, um die Prinziptauglichkeit der Vorgehensweise nachzuweisen.

Kapitel 7 beschreibt die Beweisführung von These C und Entwicklung einer Testumgebung zur automatischen Testdurchführung. Ausgehend von bisherigen Testaktivitäten werden neue funktionale, automatische Tests entwickelt, um schließlich den Dialog der akustischen MMS automatisiert zu testen. Hierfür ist die Testumgebung weiterzuentwickeln, um die MMS als Testschnittstelle verwenden zu können.

Abschließend werden in Kapitel 8 die Möglichkeiten zum Test der akustischen MMS hinsichtlich der wirtschaftlichen Aspekte analysiert. Es ist zu bestimmen, ob die erwartete Effizienzsteigerung durch die modellbasierte Vorgehensweise aufgezeigt werden kann.

4 Optimierung des Prozesses zum Test einer SW MMS

Die in Kapitel 3 aufgezeigten Thesen und Ansätze zur Lösung der beschriebenen Herausforderungen sollen im Folgenden bewiesen werden. Die Basis der angestrebten Weiterentwicklung ist die Umsetzung des MBT-Prozesses in der Entwicklung einer softwarebasierten MMS. In diesem Kapitel wird die konkrete Vorgehensweise zur Umsetzung eines solchen Testprozesses aufgezeigt, die erwarteten Vorteile und die möglicherweise auftretenden Schwierigkeiten beschrieben. Nach Beschreibung und Gegenüberstellung der konkreten Testartefakte im Prozess wird die Methodik des „Test Process Improvement“ (TPI) eingesetzt, um die modellbasierte Vorgehensweise zu realisieren.

4.1 Vom manuellen zum modellbasierten Testprozess

4.1.1 Der modellbasierte Testprozess im Entwicklungsprozess

In Kapitel 2.2 wurden die Zusammenhänge zwischen Entwicklungs- und Testprozess eingeführt. Abbildung 7 (S. 45) veranschaulicht das allgemeine V-Modell, wobei bereits darauf hingewiesen wurde, dass Testaktivitäten nicht ausschließlich am Ende des Entwicklungsprozesses stattfinden, sondern iterativ während des Produktentstehungsprozesses. [Spi03] beschreibt in diesem Kontext das W-Modell (Abbildung 19), welches die Parallelität der Testaktivitäten zur Systementwicklung besser veranschaulicht.

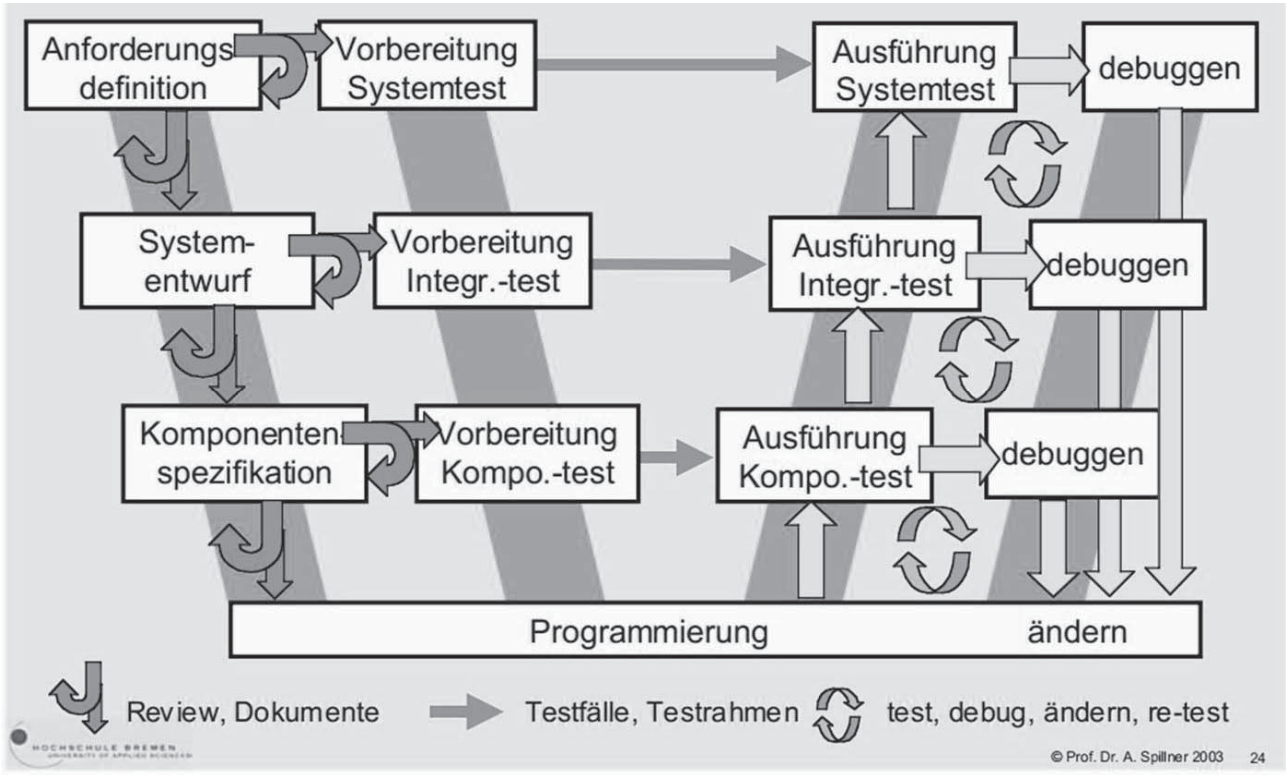


Abbildung 19: Das W-Modell [Spi03]

Die Testaktivitäten beginnen bereits mit dem Projektstart, wodurch ein Großteil der Testaktivitäten bis zum Beginn der Systemimplementierung bereits abgeschlossen sein kann.

In Anlehnung an das W-Modell wurde eine erweiterte Darstellung entwickelt, welche den Zusammenhang zwischen modellbasiertem Testprozess und der Systementwicklung verdeutlicht. Das in Abbildung 20 dargestellte Modell berücksichtigt dabei zusätzlich die relevante Schnittstelle zwischen Auftraggeber und Systemlieferant.

Der funktionale Systementwurf und die Anforderungsdefinition bilden den Ausgangspunkt im Entwicklungsprojekt und die zentrale Schnittstelle zwischen Auftraggeber und Auftragnehmer (Abbildung 20, links oben).

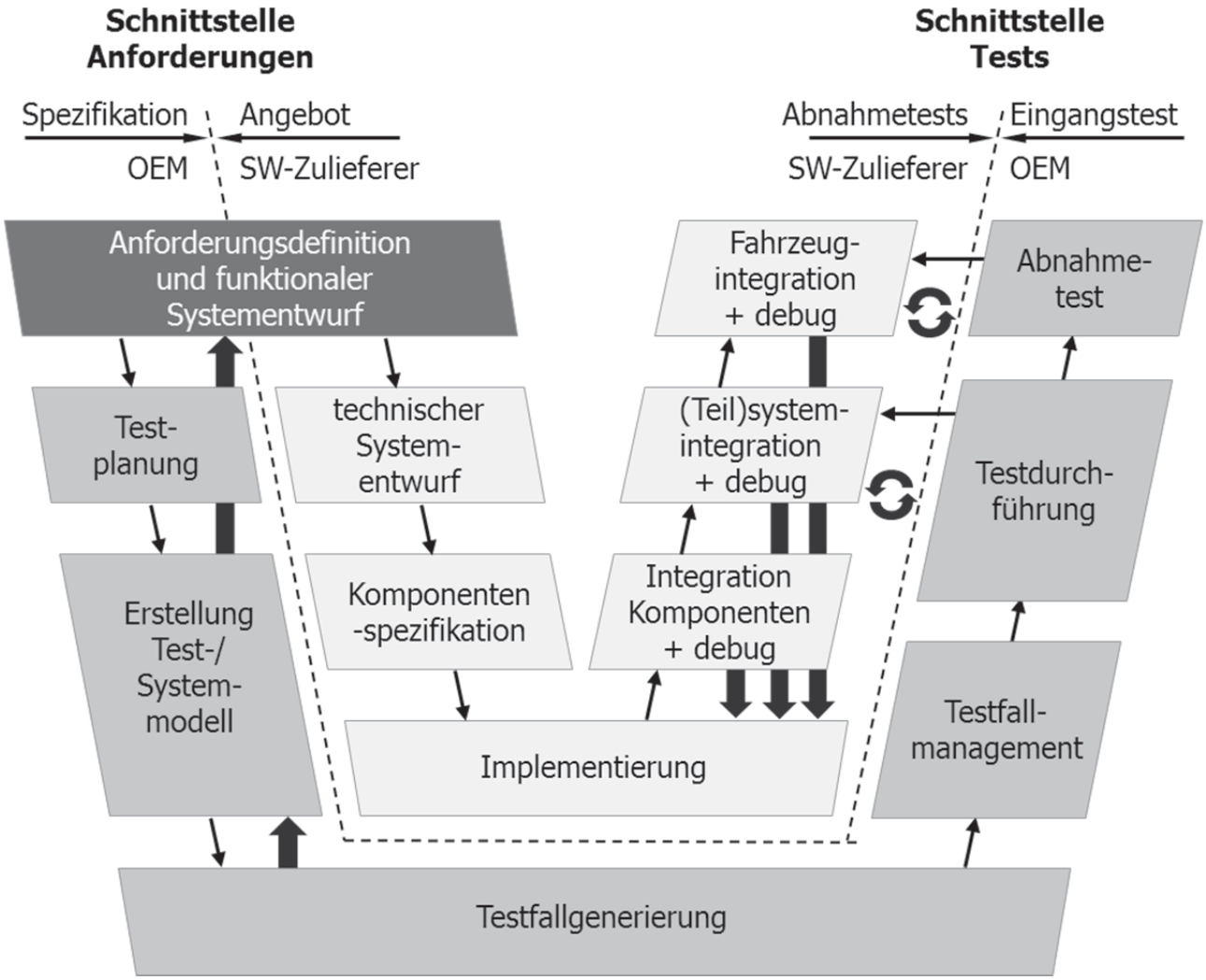


Abbildung 20: Erweitertes W-Modell des Modellbasierten Testprozesses

Auf Seiten des Auftragnehmers kann auf Basis dieser Dokumente das „klassische“ V-Modell gelebt werden und nach technischem Systementwurf und Komponentenspezifikation die Implementierung stattfinden. Neu ist, dass der Auftraggeber parallel zum Entwicklungsprozess des Systemlieferanten Testaktivitäten durchführt. Auf Basis der Spezifikationsdokumente können bereits zum Projektstart die Testaktivitäten geplant werden und das für den modellbasierten Testprozess zentrale Test- oder Systemmodell erstellt werden (siehe auch Kapitel 2.2.5). Durch die Erstellung des Modells können

konzeptionelle, inhaltliche oder formale Fehler in den Spezifikationsdokumenten bereits frühzeitig identifiziert und korrigiert werden. Gleiches gilt nach Generierung der Testfälle. Ein Review der generierten Testfälle ermöglicht nicht nur die Identifikation von Modellierungsfehlern, sondern offenbart auch konzeptionelle Schwächen des Systems. Ein Großteil der Testaktivitäten kann bereits zu einem Zeitpunkt stattfinden, an welchem die Implementierung des Systems noch nicht begonnen wurde oder noch am Anfang steht. Eine fehlerhafte Implementierung aufgrund fehlerhafter Anforderungsdokumente kann deutlich besser verhindert werden als im bisherigen „klassischen“ V-Modell. Hier ist ein Systemtest für den Auftraggeber erst mit Lieferung einer ersten Softwareimplementierung möglich. Während der Systemlieferant die Software implementiert und die Komponentenintegration durchführt, bleibt dem Auftraggeber Zeit, um Testfälle zu organisieren und priorisieren. Mit Lieferung der ersten Software sind optimale Umgebungsbedingungen und Voraussetzungen für einen effizienten und systematischen Systemtest geschaffen. Die in Kapitel 3.1 beschriebene Herausforderung der „horizontalen Lieferantenschnittstelle im V-Modell“ (Abbildung 18) kann durch Einführung des erweiterten W-Modells deutlich besser bewältigt werden. Durch das vorhandene Test- oder Systemmodell kann der Auftraggeber aktiv an der Produktentwicklung mitwirken. Welche Testaktivitäten konkret notwendig sind, um den MBT für die softwarebasierte MMS im Kfz umzusetzen, wird im Folgenden hergeleitet.

4.1.2 Der manuelle Testprozess

Nachdem in Kapitel 2.2 die allgemeingültigen Vor- und Nachteile der manuellen Testdurchführung behandelt wurden, sollen in diesem Kapitel die

konkreten Testaktivitäten erläutert werden, um den manuell durchgeführten, funktionalen Test einer softwarebasierten MMS durchzuführen. Hierbei besteht keine Notwendigkeit zur Modellierung des Systems. Vielmehr handelt es sich um die einfachste und ursprünglichste Möglichkeit der Testdurchführung. In der Praxis nimmt diese Vorgehensweise nach wie vor eine zentrale Rolle ein. Allerdings ist die Qualität des manuellen Testens oft stark von den individuellen Erfahrungen und Testkenntnissen des Testpersonals abhängig. Eine systematische Vorgehensweise im manuellen Test führt hier zu einer allgemeinen Verbesserung der Testqualität und kann durch die Etablierung entsprechender Strukturen und Prozesse erreicht werden.

In Tabelle 4 sind die Testaktivitäten zum manuellen Test der funktionalen Anforderungen einer MMS im Kontext des fundamentalen Testprozesses dargestellt. Alle hier enthaltenen Aktivitäten sind bei der Weiterentwicklung des Testprozesses hin zum MBT von Bedeutung. Aus diesem Grund werden die einzelnen Aktivitäten in der Folge genauer beschrieben.

Ausgangspunkt des Testprozesses zum manuellen Test einer softwarebasierten MMS ist die Testplanung und -steuerung. Hierzu gehören folgende Testaktivitäten:

- Bestimmung der Testbasis: Die Methodik zur Beschreibung der MMS muss festgelegt werden. Dabei ist darauf zu achten, dass die gewählte Spezifikations- oder Anforderungsbeschreibung als Testbasis Verwendung finden kann.

Tabelle 4: Testaktivitäten zum manuellen Test einer softwarebasierten MMS

Planung & Steuerung	Testplanung	Testbasis bestimmen
		Teststrategie festlegen
		Testtechniken & Metriken definieren
		Testendkriterien bestimmen
		Zeitliche Planung
		Kostenvoranschlag erstellen
		Infrastruktur und Werkzeugunterstützung definieren
		Verantwortlichkeiten festlegen
		Dokumentation der Planung
	Teststeuerung	Organisation der Testmittel
		Berichterstattung
Analyse & Design	Infrastruktur	Aufbau der Testinfrastruktur
	Prüfung der Testbasis	Statische Tests durchführen
	Testfälle	Erstellung abstrakter oder konkreter Testfälle
		Vor- und Nachbedingungen erfassen
		Rückverfolgbarkeit sicherstellen
		Sollwerte/Testorakel aufbauen
		Dokumentation der Testfallerstellung
Realisierung & Durchführung	Realisierung der Testfälle	Konkrete Testfälle realisieren
	Organisation der Testfälle	Testsequenzen/Testsuite zusammenstellen
		Priorisierung der Tests
		Change-/Versionsmanagement
	Testausführung	Prüfung der Testumgebung
		Testfälle durchführen
		Soll-Ist Vergleich
		Prüfung im Fehlerfall
		Regressionstest
	Berichterstattung	Testprotokollierung
		Abweichungsmanagement

Steuerung	Auswertung & Projektabschluss	Verwendung von Metriken	Produktmetriken einsetzen
			Prozessmetriken einsetzen
			Testabschlussbericht weiterleiten
		Abschluss der Aktivitäten	Lessons Learned
			Archivierung der Testmittel
			Wiederverwendung der Testmittel
			Abschlussbericht

- Festlegung der Teststrategie: Die Vorgehensweise für den Test ist (pro Teststufe) festzulegen. Hierzu gehören jeweils auch die Aktivitäten zur „Definition der Testverfahren“, „Bestimmung der Testendkriterien“ und die „zeitliche Planung“, welche im Folgenden beschrieben werden.
- Definition der Testtechniken und Metriken: Die Vorgehensweise, nach welcher Testfälle erstellt und ausgewählt werden, ist festzulegen. Zudem sind Metriken festzulegen, um die Qualität der MMS zu beschreiben. Werden für den manuellen Test keine Testfälle erstellt, ist die Auswahl einer Testtechnik obsolet.
- Bestimmung der Testendekriterien: Es sind Bedingungen festzulegen, welche den Abschluss des Testprozesses definieren, z. B. die Testabdeckung bestimmter Anforderungen.
- Zeitliche Planung: Die zeitliche Abfolge von Testaktivitäten, Meilensteine des Projekts, Abhängigkeiten sowie Start- und Enddaten werden festgelegt.
- Erstellung des Kostenvoranschlags: Es wird ein Kostenvoranschlag für die Testaktivitäten erstellt. Die wirtschaftlichen Rahmenbedingungen werden berücksichtigt und die Effizienz der Testaktivitäten messbar.

- Definition der Infrastruktur und Werkzeuge: Die notwendige Testumgebung, Büro- oder Laboreinrichtung sowie Werkzeuge zum Test sind zu bestimmen.
- Festlegung der Verantwortlichkeiten: Aufgaben, Kompetenzen und Verantwortlichkeiten im Testbetrieb werden zugewiesen und die Hierarchie der Berichterstattung festgelegt.
- Dokumentation der Planung: Die Ergebnisse der Testplanung sind im Testkonzept schriftlich zu dokumentieren.
- Organisation der Testmittel: Ein Testmanagement zur Überwachung und Anpassung der Testmittel sowie Testaktivitäten ist einzurichten. Ebenso ein Change- und Versionsmanagement für Testbasis und Testfälle.
- Organisation der Berichterstattung: Es findet eine zyklische Berichterstattung mit Verwendung der festgelegten Metriken statt.

Es folgen die Testaktivitäten zur Vorbereitung der Testdurchführung (Analyse & Design). Hierzu gehören folgende Testaktivitäten:

- Aufbau der Testinfrastruktur: Realisierung der Infrastruktur zur Durchführung der geplanten Testaktivitäten. Hierzu gehört die Beschaffung und Installation der Labor- und Büroräume, der Testumgebung und Testmittel, Testwerkzeuge, Organisationsstrukturen etc.
- Durchführung statischer Tests: Durch statisches Testen wird die Testbasis anhand Checklisten, Reviews etc. überprüft, um mögliche Fehler zu identifizieren.
- Erstellung abstrakter oder konkrete Testfälle: Sollen bei der manuellen Testdurchführung Testfälle verwendet werden, sind diese zu formu-

lieren. Dabei können zunächst abstrakte Testfälle erstellt werden, welche zu einem späteren Zeitpunkt mit konkreten Daten angereichert werden. Alternativ ist es möglich, direkt und ausschließlich konkrete Testfälle zu formulieren. Die Testfälle können aus der Testbasis abgeleitet werden oder auf Basis explorativer und erfahrungsbasierter Kenntnisse entstehen.

- Erfassung von Vor- und Nachbedingungen: Der für die Testausführung eines bestimmten Testfalls notwendige Zustand des SUT und/oder dessen Umgebung wird aus der Testbasis ermittelt. Diese Vorbedingungen werden dem Testfall zugeordnet. Nachbedingungen werden analog behandelt.
- Sicherstellung der Rückverfolgbarkeit: Es ist sicherzustellen, dass die Rückverfolgbarkeit der erstellten Testfälle auf die Spezifikations- und/oder. Anforderungsdokumente gewährleistet ist. Auf diese Weise kann die Testabdeckung der gestellten Anforderungen gesteuert werden.
- Aufbau des Testorakels/ der Sollwerte: Das festgelegte (Soll) oder vorausgesagte (Orakel) Verhalten eines Systems ist im Testfall, der Testbasis oder einer Datenbank zu hinterlegen, um etwaige Abweichungen des SUT identifizieren zu können.
- Dokumentation der Testfallerstellung: Die Erstellung der Testfälle ist zu dokumentieren, um einen Überblick der vorhandenen Testfälle, deren Abdeckung und Zielsetzungen festzuhalten.

Schließlich folgt die manuelle Testdurchführung zum Test der softwarebasierten MMS. Folgende Testaktivitäten sind dabei auszuführen:

- Realisierung der Testfälle: Wurden in der Testvorbereitung abstrakte Testfälle erstellt, werden aus diesen nun konkrete, ausführbare Testfälle erstellt.
- Zusammenstellung von Testsequenzen/Testsuiten: Um die Testdurchführung zu optimieren, werden mehrere Testfälle zu Testsequenzen oder Testtypen zusammengefasst. Die Zusammenstellung kann zum Beispiel in Abhängigkeit der benötigten Zeit, notwendiger Testmittel, Vorbedingungen etc. erfolgen.
- Priorisierung der Tests: Die Testfälle/Testsuiten werden nach gegebenen Anforderungen oder Rahmenbedingungen priorisiert und eine definierte Reihenfolge zur Durchführung festgelegt.
- Versionsmanagement der Testfälle: Durch die Verwendung eines Change- und Versionsmanagements werden anforderungskonforme Testfälle sichergestellt. Die Beziehungen zwischen Testbasis, Testfall und SUT lassen sich über mehrere Versionen hinweg nachvollziehen.
- Prüfung der Testumgebung: Die Korrektheit der Testumgebung wird vor Durchführung der Tests überprüft, um sicherzustellen, dass die Ursache später identifizierter Fehler im SUT und nicht in der Testumgebung begründet ist.
- Ausführung der Testfälle: Erstellte Testfälle werden manuell durchgeführt. Zudem ist ein vollkommen freies Testvorgehen ohne Vorgaben zum „kreativen“ Test einer MMS sinnvoll.
- Vergleich der Soll- und Ist-Werte: Der Abgleich von Testergebnissen (Ist-Werten) mit den Sollwerten findet im manuellen Testen meist parallel zur Testdurchführung stattfinden. Alternativ ist es jedoch auch

möglich, die Durchführung aufzuzeichnen und den Abgleich zwischen Soll- und Ist-Werten im Anschluss durchzuführen.

- Überprüfung im Fehlerfall: Im Fehlerfall wird sichergestellt, dass der betroffene Testfall korrekt ist und als Fehlerquelle ausgeschlossen werden kann.
- Durchführung von Regressionstests: Wurde das SUT modifiziert oder Fehler korrigiert, ist es notwendig, Regressionstests durchzuführen.
- Protokollierung der Testfälle: Die Durchführung von Testfällen ist inklusive aller Rahmenbedingungen und Einzelheiten der Testausführung zu protokollieren. Auf diese Weise wird sichergestellt, dass ein Testdurchlauf stattgefunden hat und bei Bedarf exakt nachgestellt werden kann.
- Durchführung des Abweichungsmanagements: Es werden Werkzeuge oder Datenbanken verwendet, um aufgedeckte Fehler zu dokumentieren, zu analysieren und zu bearbeiten. Sind verschiedene Entwicklungsparteien (wie z. B. OEM und Systemlieferant oder Entwickler und Tester) vorhanden, sind die Inhalte des Abweichungsmanagements den beteiligten Fraktionen zugänglich und werden synchronisiert.

Zum Ende des Testprozesses finden die Testaktivitäten zur Auswertung der Ergebnisse und bei Abschluss des gesamten Entwicklungsprojekts schließlich jene zur Beendigung des Testprojekts statt. Folgende Aktivitäten sind zu berücksichtigen:

- Einsatz von Produktmetriken: Definierte Produktmetriken werden in der Berichterstattung verwendet, um die Qualität der MMS einheitlich zu beschreiben (siehe auch Kapitel 2.2, Tabelle 1).

- Einsatz von Prozessmetriken: Definierte Prozess-/Projektmetriken werden in der Berichterstattung verwendet, um die Qualität des Entwicklungsprojekts zu beschreiben (siehe Kapitel 2.2, Tabelle 2).
- Weiterleitung des Testabschlussberichts: Im Testabschlussbericht sind die Testaktivitäten und -ergebnisse zusammengefasst. Er enthält eine Bewertung der durchgeführten Tests mit Bezug auf die definierten Testendekriterien.
- Zusammenfassung der „Lessons Learned“: Es findet ein Review/Walkthrough des Testprozesses/der Testaktivitäten statt, um Erfahrungen aus dem Projekt und Verbesserungspotentiale zu identifizieren. Diese sollen in folgende Projekte einfließen.
- Archivierung der Testmittel: Die Testmittel, Testfälle, Testumgebung etc. werden archiviert, um Testbedingungen reproduzierbar und die Wiederverwendung möglich zu machen.
- Wiederverwendung der Testmittel: Die Testmittel, Testfälle, Testumgebung etc. werden, so weit möglich, auf zukünftige Entwicklungsprojekte übertragen.
- Verfassen des Abschlussberichts: Im Abschlussbericht werden die Testaktivitäten des gesamten Entwicklungsprojekts abschließend zusammengefasst.

Wie erwähnt werden in der Praxis viele Aktivitäten zum Test der MMS eines Infotainmentsystems manuell durchgeführt. Nur selten werden jedoch alle hier aufgeführten Testaktivitäten tatsächlich ausgeführt. In den folgenden Kapiteln wird sich aber zeigen, dass diese Aktivitäten für die Weiterentwicklung des Testprozesses von Bedeutung sind. Werden diese Test-

aktivitäten nicht berücksichtigt, sind komplexere Testprozesse, wie der für den modellbasierten Test notwendige, zum Scheitern verurteilt.

Möchte man die Testaktivitäten verbessern, ist es sinnvoll, die funktionalen Inhalte einer softwarebasierten MMS differenziert zu betrachten. Ein manueller oder auch akustischer Dialog zwischen Mensch und Maschine besteht aus verschiedenen Elementen, welche in Abhängigkeit des Testziels im Fokus der Aktivitäten stehen können. Es erweist sich daher als sinnvoll, die MMS wie in Abbildung 21 dargestellt zu partitionieren und diese Einteilung bei der Durchführung von Testaktivitäten zu berücksichtigen.

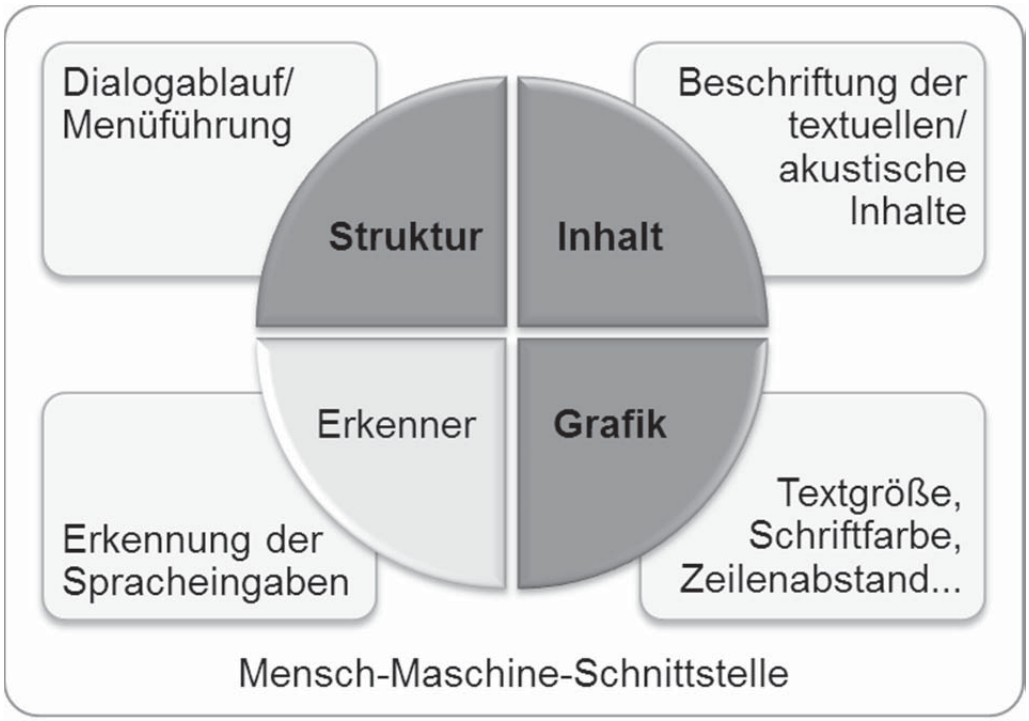


Abbildung 21: Elemente einer softwarebasierten MMS im Kfz

Grundlage eines jeden Dialogs zwischen Mensch und Maschine ist die Dialog- oder Menüstruktur. Hierin ist beispielsweise definiert, ob nach einer be-

stimmten Systemeingabe die entsprechende Funktion direkt aufgerufen wird oder zunächst eine akustische Rückfrage des Systems erfolgt.

Die Dialogstruktur der MMS wird durch die Ergänzung von Dialoginhalten für den Menschen nutzbar. Menüfelder und akustische Ausgaben besitzen einen textuellen Inhalt, Symbole oder hinterlegte Daten. Ein definiertes Menüfeld erhält beispielsweise den Inhalt „Ziel eingeben“, „Enter Destinition“, „Entre destination“, etc. Die Dialoginhalte sind, wie im Beispiel bereits aufgezeigt, sprachenabhängig, wohingegen die Dialogstruktur auch länderübergreifend weitgehend identisch sein kann.

Weitere Testaktivitäten können sich auf die grafische und akustische Umsetzung der MMS fokussieren. Zeichengröße, Zeilenabstand, Farbe, Hintergrund und weitere Eigenschaften lassen sich dem grafischen Aspekt der MMS zuordnen. Sprechgeschwindigkeit und Tonlage der Sprachausgabe lassen sich analog der nicht funktionalen, akustischen Umsetzung einer MMS zuordnen. Während die Entwicklung der Dialogstruktur und -inhalte meist im Kompetenzbereich einer entsprechenden Entwicklungsabteilung akkumuliert ist, liegt die Entwicklung und Absicherung der grafischen Umsetzung einer MMS meist im Kompetenzbereich der Ergonomen und Designer. Im Falle der akustischen MMS kann zudem die Spracherkennung als Element der MMS beschrieben werden (siehe auch Kapitel 2.1.3).

Im Vergleich zu anderen Systemelementen eines Infotainmentsystems ist der manuelle Test einer MMS leicht durchzuführen, da das Testobjekt als Schnittstelle zwischen Mensch und Maschine direkt angesprochen werden kann. Dies geschieht beispielsweise im Rahmen freier Testaktivitäten.

Der Tester nutzt das Infotainmentsystem im Labor oder Fahrzeug ohne konkrete Vorgaben zur Testdurchführung. Dabei können die Tester erfahrungsbasiert vorgehen und/oder fokussiert jene Systemteile prüfen, welche in der Vergangenheit besonders fehleranfällig waren. Voraussetzung eines solchen Tests sind Kenntnisse zum spezifizierten Systemverhalten und Erfahrung. Oft hängt die Qualität des Tests dabei stark von den individuellen Kompetenzen des Testers ab. Fachwissen zum Testen ist oft nur implizit vorhanden und nicht schriftlich dokumentiert. Um die Testqualität insgesamt zu erhöhen ist es wichtig, das implizit vorhandene Wissen explizit verfügbar zu machen. Zudem kann manuelles Testen auch durch die Nutzung von Testfällen unterstützt und verbessert werden. Die verwendeten Testfälle decken meist kundentypisches Nutzungsverhalten (engl.: Use cases) ab und werden manuell aus den Anforderungsdokumenten abgeleitet.

Durch manuelles Testen können alle Elemente der MMS geprüft werden. Nutzt der Tester das SDS, bekommt er einen Eindruck der Erkennungsqualität, kann den Inhalt der Dialoge genauso verifizieren wie die Dialogstruktur und kann dabei die grafische Umsetzung der MMS bewerten. Um fremdsprachliche Inhalte der MMS ebenfalls manuell zu verifizieren, werden Tester unterschiedlicher Nationalitäten bemüht. Grundsätzlich ist jedoch zu bedenken, dass ein Tester kaum alle Systemanforderungen kennt. Es ist leicht möglich, dass bei einem Testdurchlauf inhaltliche Fehler der MMS identifiziert werden, gleichzeitig aber Fehler in der Dialogstruktur oder grafischen Umsetzung übersehen werden. Die Fokussierung der Verifikation auf eines der beschriebenen Elemente (Struktur/Inhalt/Grafik) wirkt dem entgegen.

Nachdem in Kapitel 2.2.5 die Grundlagen des MBT behandelt wurden, soll im Folgenden betrachtet werden, welche konkreten Testartefakte und -aktivitäten im fundamentalen Testprozess gegenüber der manuellen Vorgehensweise verändert oder ergänzt werden müssen, um ein automatisiertes und modellbasiertes Testen zu realisieren.

4.1.3 Der automatisierte und modellbasierte Testprozess

Die Grafik in Abbildung 22 zeigt die abstrakten Unterschiede zwischen den Testprozessen zum manuellen, automatisierten und modellbasierten Testvorgehen.

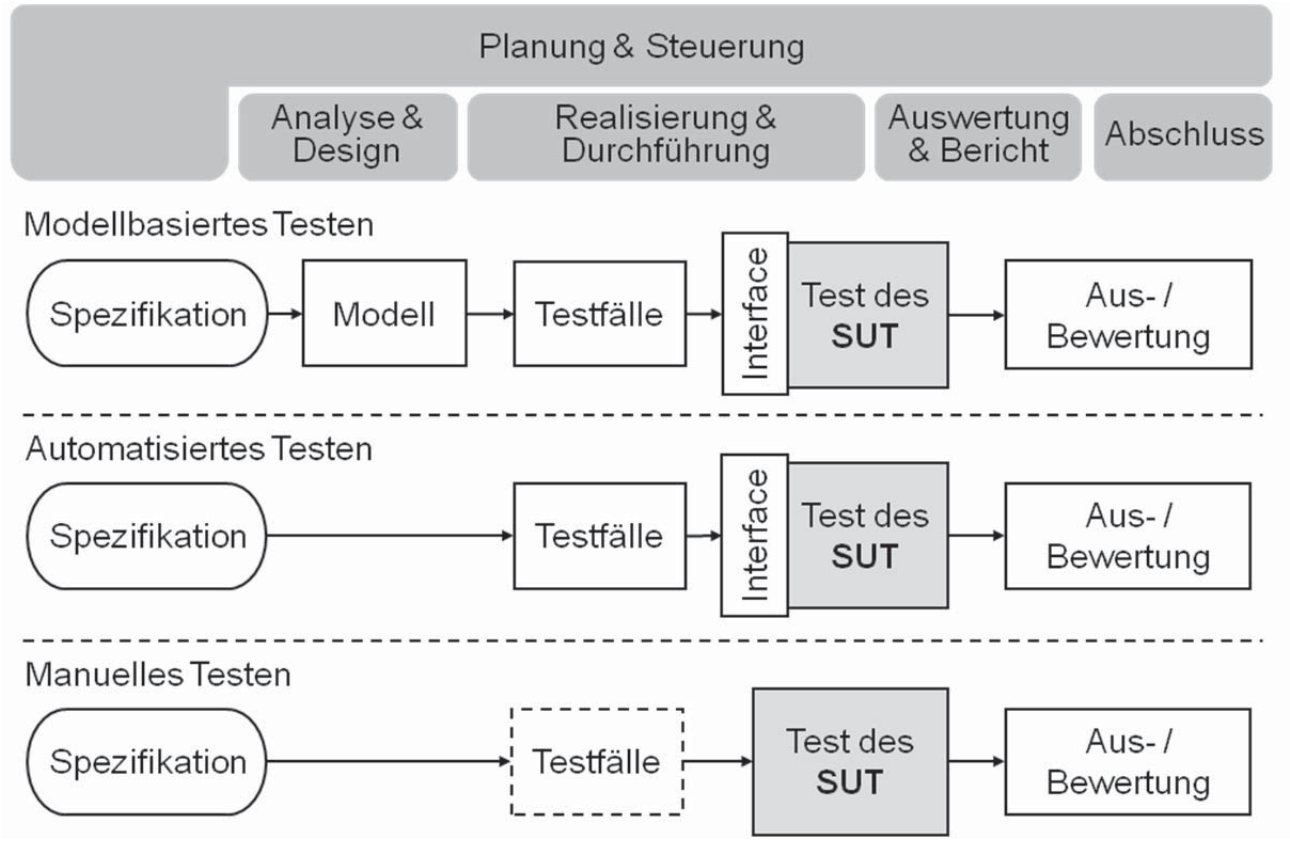


Abbildung 22: Der Prozess zum automatisierten und modellbasierten Test

Das zentrale Element im modellbasierten Testprozess ist, wie die Prozessbezeichnung bereits impliziert, das Modell. Durch die Erstellung eines System- oder Testmodells wird die Möglichkeit geschaffen, Testfallgeneratoren einzusetzen und Testfälle oder Testskripte aus dem Modell zu generieren. Diese können manuell oder automatisch auf dem SUT ausgeführt werden.

Zur Durchführung manueller Tests sind Testfälle nicht zwangsläufig notwendig. Dies ändert sich, soll die Testausführung automatisch stattfinden. Hier ist es unumgänglich, Testfälle zu beschreiben. Dabei entstehen neue Anforderungen an das Format und den Inhalt eines Testfalls. Um bei der Abarbeitung manueller Testfälle Fehler durch den Menschen zu vermeiden, ist deren Format leserlich, eindeutig und möglichst einfach zu gestalten. Eine automatische Testdurchführung erfordert maschinell ausführbare Testfälle in einer formalen, durchgängigen Beschreibung. Ausführbare Testfälle werden im Folgenden als Testskripte bezeichnet. Testskripte können sowohl aus bereits bestehenden oder erstellten Testfällen abgeleitet werden, als auch direkt aus den Anforderungsdokumenten oder der Spezifikation. Testskripte können analog der Testfälle abstrakt formuliert oder mit konkreten Inhalten angereichert sein. Dabei besteht auch die Möglichkeit, konkrete Werte in einer Datenbank zu hinterlegen, welche bei Ausführung des abstrakten Testskripts angezogen werden. Um erstellte Testskripte auszuführen, ist eine entsprechende Testumgebung sowie Schnittstelle zum SUT erforderlich.

In Tabelle 5 sind die notwendigen Aktivitäten aufgeführt, um eine softwarebasierte MMS im Kfz automatisch und modellbasiert zu testen. Dabei ist zu beachten, dass alle im manuellen Test beschriebenen Aktivitäten relevant bleiben, diese jedoch meist geändert oder erweitert werden müssen. Auch

zusätzliche Testaktivitäten sind notwendig. Tabelle 5 veranschaulicht, welche der Testaktivitäten zum automatischen bzw. modellbasierten Test Veränderung erfahren. Im Folgenden werden diese Veränderungen beschrieben, um die notwendigen Erweiterungen im Testprozess darzustellen.

Tabelle 5: Testaktivitäten zum automatisierten und MBT einer SW MMS

			Veränderung am (manuellen) Prozess, zum automatischen Testen	Neue Testaktivität im automatisierten Testprozess	Veränderung am (manuellen) Prozess, zum modellbasierten Testen erforderlich	Neue Testaktivität im modellbasierten Testprozess
Planung & Steuerung	Testplanung	Testbasis bestimmen			X	
		Teststrategie festlegen	X		X	
		Testtechniken & Metriken definieren			X	
		Testendkriterien bestimmen			X	
		Zeitliche Planung	X		X	
		Kostenvoranschlag erstellen	X		X	
		Infrastruktur und Werkzeug-unterstützung definieren	X		X	
		Verantwortlichkeiten festlegen	X		X	
		Dokumentation der Planung				
	Teststeuerung	Organisation der Testmittel			X	
		Berichterstattung				
	Infrastruktur	Aufbau der Testinfrastruktur	X		X	
Analyse & Design	Modellierung	Testmodell/Testbasis erstellen				X
	Prüfung der Testbasis	Statische Tests durchführen			X	
	Testfälle	Generierung abstrakter oder konkreter Testfälle				X
		Erstellung abstrakter oder konkreter Testfälle	X		X	
		Vor- und Nachbedingungen erfassen			X	
		Reproduzierbarkeit der Testfälle sicherstellen				X
		Rückverfolgbarkeit sicherstellen	X		X	
		Sollwerte/Testorakel aufbauen	X		X	
		Dokumentation der Testfallgenerierung			X	

			Neue Testaktivität im modellbasierten Testprozess	Veränderung am (manuellen) Prozess, zum modellbasierten Testen erforderlich	Neue Testaktivität im automatisierten Testprozess	Veränderung am (manuellen) Prozess, zum automatischen Testen
Planung & Steuerung	Realisierung & Durchführung	Realisierung der Testfälle	Konkrete Testfälle realisieren	X		X	
		Organisation der Testfälle	Testsequenzen/Testsuite zusammenstellen				
			Priorisierung der Tests			X	
			Change-/Versionsmanagement	X		X	
		Testausführung	Prüfung der Testumgebung				
			Testfälle ausführen	X			
			Fortsetzung im Fehlerfall		X		
			Exploratives/Erfahrungsbasiertes Testen		X		
			Soll-Ist Vergleich	X			
			Prüfung im Fehlerfall	X		X	
			Regressionstest				
		Berichterstattung	Testprotokollierung				
			Abweichungsmanagement				
	Auswertung & Projektabschluss	Verwendung von Metriken	Produktmetriken einsetzen			X	
			Prozessmetriken einsetzen	X		X	
			Testabschlussbericht weiterleiten				
		Abschluss der Aktivitäten	Lessons Learned	X		X	
			Archivierung der Testmittel	X		X	
			Wiederverwendung der Testmittel	X		X	
			Abschlussbericht				

In der Prozessphase zur Planung automatischer und modellbasierter Tests einer MMS sind folgende, erweiterte Aktivitäten notwendig:

- Bestimmung der Testbasis: Neben der Möglichkeit, die Spezifikation oder Anforderungsbeschreibung als Testbasis zu verwenden, wird im MBT das System- oder Testmodell als Testbasis verwendet. Es ist die Methodik zur Modellierung des SUT zu bestimmen (siehe Kapitel 5).

- Festlegung der Teststrategie: Ergänzend zu den in Kapitel 4.1.2 beschriebenen Inhalten der Planung, sind die automatisch und/oder modellbasiert durchzuführenden Testaktivitäten (pro Teststufe) festzulegen.
- Definition der Testtechniken und Metriken: Im MBT ist die Vorgehensweise, nach welcher Testfälle automatisch generiert und ausgewählt werden, festzulegen (siehe Kapitel 2.2.4). Hierbei entstehen Anforderungen an die Generierungswerkzeuge. Auch Metriken sind zu definieren, welche das erstellte System- oder Testmodell mit einbeziehen (z. B. die Modellüberdeckung).
- Bestimmung der Testendekriterien: Es sind Bedingungen festzulegen, die den Abschluss des Testprozesses definieren. Hierbei können im MBT Modellüberdeckungskriterien Verwendung finden.
- Zeitliche Planung: Die zeitliche Abfolge der Testaktivitäten ist unter Berücksichtigung der automatisierten und/oder modellbasierten Vorgehensweise festzulegen. Die (zusätzliche) Erstellung von Testskripten und Test- oder Systemmodellen ist dabei ebenso zu berücksichtigen wie die meist kürzeren Testdurchlaufzeiten durch die Automatisierung.
- Erstellung des Kostenvoranschlags: Der Kostenvoranschlag für die Testaktivitäten wird unter Berücksichtigung der automatischen Testdurchführung und/oder modellbasierten Vorgehensweise erstellt. Aufwendungen für Testwerkzeuge zur Modellierung, Testfallgenerierung und automatischen Durchführung werden einbezogen, sowie ggf. geringere Personalkosten in der Testfallerstellung und Testdurchführung.

- Definition der Infrastruktur und Werkzeuge: Die notwendige Testumgebung, Büro- oder Laboreinrichtung sowie Werkzeuge zum automatischen- und/oder MBT werden bestimmt (inkl. Werkzeuge zur Modellierung und Testfallgenerierung).
- Festlegung der Verantwortlichkeiten: Ergänzende, durch den automatischen und/oder MBT entstehende Aufgaben, Kompetenzen und Verantwortlichkeiten werden zugewiesen (z. B. die Erstellung von Modellen und Überwachung der Testfallgenerierung oder die Erstellung von Testskripten und die Überwachung der automatischen Testdurchführung).
- Organisation der Testmittel: Ein Testmanagement zur Überwachung und Anpassung der Testmittel und Testaktivitäten wird für den MBT eingerichtet. Ebenso ein Change- und Versionsmanagement für Testbasis, System- oder Testmodell und generierte Testfälle.

Die Testaktivitäten zur Vorbereitung der modellbasierten und automatischen Testdurchführung (Analyse & Design) erfordern folgende ergänzende Aktivitäten. Es ist insbesondere zu beachten, dass die „Erstellung von Testmodellen“ und die „Generierung von Testfällen“ sowie die „Organisation von Modellen und Testfällen“ als zusätzliche Testaktivitäten des MBT-Prozesses zu implementieren sind.

- Aufbau der Testinfrastruktur: Realisierung der Infrastruktur zur Durchführung der geplanten, modellbasierten und/oder automatischen Testaktivitäten. Hierzu gehört die Beschaffung der entsprechenden Modellierungs- und Testfallgenerierungswerkzeuge, der Testumgebung und Testmittel.

- Modellierung des System- oder Testmodells/Erstellung der Testbasis: Das als Testbasis vorgesehene System- oder Testmodell zur Generierung der Testfälle ist zu erstellen. (Nur im MBT-Prozess)
- Durchführung statischer Tests: Durch statisches Testen (siehe Kapitel 2.2.4) wird das System- oder Testmodell anhand Checklisten, Reviews, etc. überprüft, um Fehler zu identifizieren. (Nur im MBT-Prozess)
- Generierung abstrakter oder konkrete Testfälle: Aus dem erstellten System- oder Testmodell werden, durch den Einsatz entsprechender Werkzeuge, Testfälle mit oder ohne konkrete Ein- und Ausgabewerte generiert. (Nur im MBT-Prozess)
- Erstellung abstrakter oder konkrete Testfälle: Analog der in Kapitel 4.1.2 beschriebenen Erstellung von Testfällen können in dieser Prozessphase bereits Testskripte für die automatische Testdurchführung erstellt werden.
- Erfassung von Vor- und Nachbedingungen: Die für die Testausführung eines bestimmten Testfalls notwendigen Vor- und Nachbedingungen können im MBT-Prozess gegebenenfalls auch aus dem System- oder Testmodell abgeleitet werden.
- Sicherstellung der Reproduzierbarkeit von generierten Testfällen: Um im MBT-Prozess eine systematische Testfallgenerierung auch über mehrere Modellversionen hinweg sicherzustellen ist es notwendig, einen Algorithmus zur Generierung reproduzierbarer Testfälle einzu-setzen.
- Sicherstellung der Rückverfolgbarkeit: Es ist sicherzustellen, dass die Rückverfolgbarkeit der generierten Testfälle/erstellten Testskripte auf das System- oder Testmodell/die Spezifikations- oder Anforderungsdokumente gewährleistet ist.

- Aufbau des Testorakels/ der Sollwerte: Die Soll-Werte oder das vorausgesagte Verhalten eines Systems kann direkt im Testskript oder im System- bzw. Testmodell sowie alternativ in einer verknüpften Datenbank hinterlegt werden, um Abweichungen des SUT bei der automatischen Testdurchführung identifizieren zu können.
- Dokumentation der Testfallerstellung: Zusätzlich sind alle relevanten Parameter der Testfallgenerierung zu dokumentieren, um eine wiederholte Generierung zu ermöglichen (die Reproduzierbarkeit der Testfallgenerierung vorausgesetzt).

Es folgt die automatische Testdurchführung der erstellten oder generierten Testfälle. Folgende Testaktivitäten sind ergänzend zu den in Kapitel 4.1.2 beschriebenen Aktivitäten auszuführen. Dabei sind insbesondere die „Fortsetzung im Fehlerfall“ und der „explorative, erfahrungsbasierte Test“ als neue Testartefakte zu beachten.

- Realisierung der Testfälle: Wurden in der Testvorbereitung abstrakte Testfälle erstellt, werden aus diesen nun konkrete, automatisch ausführbare Testskripte erstellt.
- Priorisierung der Tests: Im MBT-Prozess besteht die Möglichkeit, dass Prioritäten bereits im System- oder Testmodell hinterlegt sind (vergl. Kapitel 6.2.2). Ist dies der Fall, kann die Priorisierung der Testfälle gegebenenfalls automatisch erfolgen.
- Versionsmanagement der Testfälle/Testskripte: Die Beziehungen zwischen Testbasis, Testfall und/oder Testskript, sowie SUT lassen sich über mehrere Versionen hinweg nachvollziehen. Anforderungskonforme Modelle und Testfälle werden sichergestellt.

- Ausführung der Testfälle: Testskripte werden automatisch durchgeführt.
- Fortsetzung Fehlerfall: Bei der automatischen Testdurchführung ist sicherzustellen, dass im Fehlerfall (ein Testskript wird nicht vollständig ausgeführt) die Fortsetzung der Testdurchführung gewährleistet ist (z. B. durch ein automatisches Zurücksetzen des Systems und anschließendem Start des nachfolgenden Testskripts).
- Exploratives-/Erfahrungsbasiertes Testen: Es ist wichtig, neben der automatischen Testdurchführung auch explorative und erfahrungsbasierte Testaktivitäten durchzuführen. Die „kreative Testdurchführung“ durch den Mensch erweitert die systematische Vorgehensweise um ein wichtiges Element. Aufgrund des Umstandes, dass der Mensch Teil des zu testenden Systems ist, sind dessen teilweise unvorhersehbaren Aktivitäten im Umgang mit dem System dringend zu berücksichtigen. Das automatische Testen der MMS ist als wertvolle, oft notwendige Ergänzung zum manuellen Test zu betrachten, jedoch nicht als hinreichend.
- Vergleich der Soll- und Ist-Werte: Der Abgleich von Testergebnissen (Ist-Werten) mit den Sollwerten kann im automatisierten Testen einer MMS ebenfalls automatisiert stattfinden. Die reale Systemreaktion kann dabei mit der erwarteten, in einer Datenbank oder im Testskript hinterlegten Systemreaktion verglichen werden. Stimmen die Werte überein, ist das anforderungskonforme Systemverhalten bestätigt.
- Überprüfung im Fehlerfall: Im Fehlerfall wird sichergestellt, dass der betroffene Testfall oder Testskript und das zu Grunde liegende System-

oder Testmodell korrekt ist und als Fehlerquelle ausgeschlossen werden kann.

Die Testaktivitäten zur Auswertung der Ergebnisse und zum Abschluss des gesamten Entwicklungsprojekts sind bei einem modellbasierten/automatisierten Testvorgehen ebenfalls zu erweitern.

- Einsatz von Produktmetriken: Produktmetriken können um modellbasierte Elemente erweitert werden, z. B. die Beschreibung der Testabdeckung in Bezug auf das verwendete System- oder Testmodell, siehe auch Tabelle 1 (S.51).
- Einsatz von Prozessmetriken: Prozess-/Projektmetriken werden in der Berichterstattung erweitert, um die automatisierte Vorgehensweise zu bewerten. Auch modellbasierte Elemente können Verwendung finden, so z. B. die Anzahl generierter Testfälle und daraus identifizierte Fehler im Vergleich zu manuell erstellten Testfällen.
- Zusammenfassung der „Lessons Learned“: Im Review/Walkthrough des Testprozesses/der Testaktivitäten wird explizit auch die Modellierung und Testfallgenerierung, sowie die automatisierte Testdurchführung betrachtet und „Lessons Learned“ formuliert.
- Archivierung der Testmittel: Ergänzend zu den in Kapitel 4.1.2 beschriebenen Aktivitäten werden auch Testmittel und Werkzeuge zur Modellierung, Testfallgenerierung und automatischen Testdurchführung archiviert.
- Wiederverwendung der Testmittel: Auch Testmittel und Werkzeuge der Modellierung, Testfallgenerierung und der automatischen Testdurchführung werden auf folgende Entwicklungsprojekte übertragen.

In der Praxis sind verschiedene Funktionalitäten der softwarebasierten MMS prädestiniert für ein automatisiertes Vorgehen. Hierzu gehören prinzipiell eher einfache, abstrakte Testfälle mit einer großen Vielfalt möglicher, konkreter Ein- und/oder Ausgabewerte. In Kapitel 7.2 wird das praktische Vorgehen zum automatischen Test ausführlicher behandelt.

Es zeigt sich, dass die Einführung des modellbasierten Testvorgehens Auswirkungen auf fast alle Testaktivitäten des fundamentalen Testprozesses hat. Modellbasiertes Testen ist nicht nur als weitere Ergänzung für vorhandene Testprozesse zu begreifen, sondern kann vielmehr als eine ganzheitliche Methodik verstanden werden, welche das grundsätzliche Verständnis zum Test verändert und im konkreten Testvorgehen nahezu alle fundamentalen Testaktivitäten verändert. Eine tabellarische Gegenüberstellung der in diesem Kapitel ausgearbeiteten, fundamentalen Aktivitäten zum manuellen, automatisierten und modellbasierten Test einer softwarebasierten MMS ist in Anhang 12.1 dargestellt.

Diese Gegenüberstellung unterstreicht das Potenzial des MBT-Prozesses. Gleichzeitig führt sie zu praktischen Herausforderungen. Weitreichende Veränderungen vorhandener Prozesse sind in der Praxis schwieriger umzusetzen als „punktuelle“ Ergänzungen im Testprozess. Im folgenden Kapitel wird daher die konkrete Anwendung des aus Kapitel 2.3.8 und 2.4.2 bekannten Prozessmodells „TPI“ aufgezeigt.

Durch dessen Anwendung kann ermittelt werden, welche Verbesserungen in der Praxis am konkreten Testprozess notwendig sind, um das MBT der softwarebasierten MMS durchgängig umzusetzen.

4.2 Das Test Process Improvement in der Praxis

Die Eigenschaften des TPI wurden in Kapitel 2.3.8 aufgezeigt. Ziel der Anwendung ist die Feststellung des Status quo bestehender Test-prozesse und die Ableitung konkreter Maßnahmen, um die Prozessreife zu verbessern.

4.2.1 Anwendung des Test Process Improvement

Auf Basis des allgemeinen TPI wurden weitere, branchenspezifische TPI-Modelle entwickelt. So z. B. „TPI Automotive“ der „Sogeti Deutschland GmbH“. Dieses Prozessmodell wurde gegenüber dem allgemeinen TPI geringfügig erweitert und ist Grundlage der folgenden Bewertung. Der Testprozess ist in folgende 21 Kerngebiete aufgeteilt [Sog09]:

- Die Teststrategie
- Einsatz des Phasenmodells
- Zeitpunkt der Beteiligung
- Kostenvoranschlag und Planung
- Test-Spezifikationstechniken
- Statische Testtechniken
- Metriken
- Testautomatisierung
- Testumgebung
- Testarbeitsplatz
- Engagement und Motivation
- Testfunktionen und Ausbildungen
- Reichweite der Methodik
- Kommunikation
- Berichterstattung
- Dokumentation der Abweichungen
- Testwaremanagement
- Testprozessmanagement
- Prüfen
- Low-Level-Tests
- Integrationstests

Eine detaillierte Beschreibung der Kerngebiete findet sich in [Sog09].

Zu jedem der 21 Kerngebiete des Testprozesses werden umfangreiche „Fragenkataloge“ bereitgestellt. Alle Fragen lassen sich mit „Ja“ oder „Nein“

beantworten und werden als Kontrollpunkte verstanden. Im Kerngebiet „Metriken“ gibt es beispielsweise folgenden Kontrollpunkt:

> *„Im (Test-)Projekt werden Input Metriken geführt: Eingesetzte Ressourcen (Stunden), ausgeführt Aktivitäten (Stunden und Durchlaufzeit), Umfang und Komplexität des zu testenden Systems (Anzahl der Funktionen bzw., Programmieraufwand, Anzahl der Systemanforderungen, Anzahl der Programmzeilen etc.“* [Sog09].

Sind die beschriebenen Inhalte und Aktivitäten bereits Teil des etablierten Testprozesses, ist der entsprechende Kontrollpunkt erreicht. In Abhängigkeit davon, welche und wie viele Kontrollpunkte eines bestimmten Kerngebiets erfüllt sind, wird dieses einer bestimmten Reifegradebene (A bis D) zugeordnet. Die unterste Ebene (Ebene A) stellt die geringsten Anforderungen an ein Kerngebiet. Zum Erreichen höherer Ebenen (B, C, D) müssen jeweils weitere Kontrollpunkte erreicht werden. Das Erreichen der nächsthöheren Ebene bedeutet folglich eine höhere Prozessreife (in zeitlicher, finanzieller und/oder qualitativer Hinsicht). Ergänzend sind in einer jeden Ebene konkrete Optimierungsvorschläge bereitgestellt, um das Erreichen der nächsthöheren Reifegradebene zu unterstützen.

Zentrales Element des TPI-Modells ist die TPI-Matrix. Diese stellt die Abhängigkeiten zwischen den definierten Kernbereichen her (Tabelle 6). Auf Basis der in den verschiedenen Kerngebieten erreichten Reifegradebenen kann der gesamte Testprozess nun einer „Entwicklungsstufe“ (1-13) zugewiesen werden. Je höher die Entwicklungsstufe, umso höher die allgemeine Prozessreife.

Tabelle 6: Die TPI Matrix

		beherrschbar					effizient					optimierend		
Stufe / Kernbereich	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Kernbereich 1		A				B						C		
Kernbereich 2				A			B			C				
Kernbereich 3		A					B				C		D	
Kernbereich 4		A			B		C					D		
Kernbereich 5		A				B		C						
Kernbereich 6			A			B				C				D
Kernbereich 7		A		B								C		
Kernbereich 8			A				B			C				
Kernbereich 9				A				B						D
Kernbereich 10		A			B									
Kernbereich 11			A				B				C		D	
Kernbereich 12				A							B			
Kernbereich 13		A		B						C				
Kernbereich 14					A		B							
Kernbereich 15						A			B			C		D
Kernbereich 16				A				B			C			
Kernbereich 17				A										
Kernbereich 18					A		B				C			D
Kernbereich 19			A		B							D		
Kernbereich 20					A		B		C					
Kernbereich 21				A			B				C			

Erreicht ein Testprozess die Entwicklungsstufen 1-5, gilt er als „beherrschbar“. Stufe 6-10 beschreibt den Testprozess als „effizient“, Entwicklungsstufe 11-13 wird bei einer hohen, „optimalen“ Prozessreife erreicht.

Um modellbasiertes Testen ohne Bedenken einführen zu können, sollte nach [Utt07] die Entwicklungsstufe 7 erreicht werden. Unter Berücksichtigung der Matrix müssen hierfür z.B. die Kernbereiche 1, 2 und 3 die Reifegradebene

„B“ erreichen, wohingegen im Kernbereich 4 das Erreichen der Reifegradebene „C“ erforderlich ist (vergl. Tabelle 6).

In der praktischen Anwendung der Methodik ergeben sich Herausforderungen. Die der Methodik zu Grunde liegenden Kontrollpunkte sind meist abstrakt formuliert. Die Erarbeitung der Inhalte durch Experteninterviews und Aufarbeitung vorhandener Dokumente gestaltet sich in der Folge immer wieder schwierig, da der Bezug zwischen abstrakten Inhalten und konkreten Vorgehensweisen teilweise nur schwer hergestellt werden kann.

Das Kerngebiet „Low-Level-Test“ behandelt Unit- und Komponententests in einem frühen Stadium der Systementwicklung. Diese Testaktivitäten werden meist von den Entwicklern selbst durchgeführt, z. B durch Verwendung von „Code Checkern“. Nachdem im konkreten Fall die Systementwicklung vom Systemlieferanten durchgeführt wird, ist das Kerngebiet ausgenommen.

Die softwarebasierte MMS verschiedener Hardwarekomponenten und Bedienungsmodalitäten des Infotainmentsystems werden in der Praxis von drei verschiedenen, eng verzahnten Gewerken entwickelt. Dabei werden sowohl unterschiedliche Methoden zur Anforderungsformulierung als auch verschiedene Testprozesse und -aktivitäten angewendet. Dies hat zur Folge, dass mehrere parallel vorhandene Testprozesse zu analysieren sind. Auf Basis der Vorgaben von [Utt07] und [Roß10] ist die Entwicklungsstufe 7 als Zielgröße für die Einführung des modellbasierten Testens einer softwarebasierten MMS im Kraftfahrzeug festgelegt. Kann die Entwicklungsstufe erreicht werden, steht der Umsetzung des modellbasierten Vorgehens nichts im Wege.

4.2.2 Ergebnisse des Test Process Improvement (TPI)

In Tabelle 7 sind die Ergebnisse des TPI dargestellt. Hierbei ist nach vollständig erfüllten Kontrollpunkten und eingeschränkt erfüllten Kontrollpunkten unterschieden. Bei zweiteren sind die zentralen Inhalte zwar erfüllt, jedoch bleibt Potenzial für Verbesserungen. Aus Gründen des Datenschutzes sind die Kernbereiche nicht explizit aufgeführt.

Wie in Kapitel 4.2 beschrieben, sind drei parallel vorhandene Prozesse zum Test der softwarebasierten MMS bewertet. In 65 % der analysierten Kernbereiche wurde für die drei bestehenden Prozesse eine ähnliche oder gleiche inhaltliche Reife festgestellt (Kernbereiche 1 bis 13). In den übrigen 35 % sind die Ergebnisse differenzierter zu betrachten, da die Reifegrade der bewerteten Prozesse abweichen (Kernbereiche 14 bis 20). Während im Kernbereich 1 (mit geringen Einschränkungen) die Entwicklungsstufe 11 erreicht ist, konnten in anderen Kernbereichen (16, 17, 19, 20) bei mindestens einem der analysierten Prozesse die Reifegradziele der ersten Entwicklungsstufe nicht erfüllt werden.

Insgesamt zeigt sich, dass über alle analysierten Testprozesse hinweg 70 % der Kerngebiete mindestens den Reifegrad der Entwicklungsstufe 7 einnehmen und noch 47 % die Bedingungen der Entwicklungsstufe 10 erfüllen.[1]

[1] Bei der Auswertung ist zu beachten, dass zum Erreichen einer Entwicklungsstufe X alle Reifegradebenen bis einschließlich der Entwicklungsstufe X erfüllt sein müssen, jedoch keine Reifegradebene der Stufe X+1. Um z. B. die Entwicklungsstufe 7 zu erreichen, genügt im Kernbereich „10“ das Erreichen der Reifegradebene „A“.

Tabelle 7: Ergebnisse der TPI Analyse

		beherrschbar					effizient					optimierend		
Stufe / Kernbereich	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Kernbereich 1		A		B								C		
Kernbereich 2			A			B				C				D
Kernbereich 3				A				B						D
Kernbereich 4		A				B		C						
Kernbereich 5				A			B				C			
Kernbereich 6				A			B			C				
Kernbereich 7						A			B			C		D
Kernbereich 8		A			B		C					D		
Kernbereich 9		A			B									
Kernbereich 10				A							B			
Kernbereich 11				A										
Kernbereich 12			A				B				C		D	
Kernbereich 13		A					B				C		D	
Kernbereich 14	Proz.1				A		B				C			
	Proz.2				A		B				C			
	Proz.3				A		B				C			D
Kernbereich 15	Proz.1	A		B						C				
	Proz.2	A		B						C				
	Proz.3	A		B						C				
Kernbereich 16	Proz.1			A				B			C			
	Proz.2			A				B			C			
	Proz.3			A				B			C			
Kernbereich 17	Proz.1	A				B						C		
	Proz.2	A				B						C		
	Proz.3	A				B						C		
Kernbereich 18	Proz.1		A		B							D		
	Proz.2		A		B							D		
	Proz.3		A		B							D		
Kernbereich 19	Proz.1		A				B			C				
	Proz.2		A				B			C				
	Proz.3		A				B			C				
Kernbereich 20	Proz.1				A		B							
	Proz.2				A		B							
	Proz.3				A		B							

Legende: erfüllt erfüllt mit Einschränkungen

Abbildung 23 gibt einen Überblick, zu welchem Anteil die verschiedenen Entwicklungsstufen durch die Reifegrade der verschiedenen Kernbereiche erreicht sind.

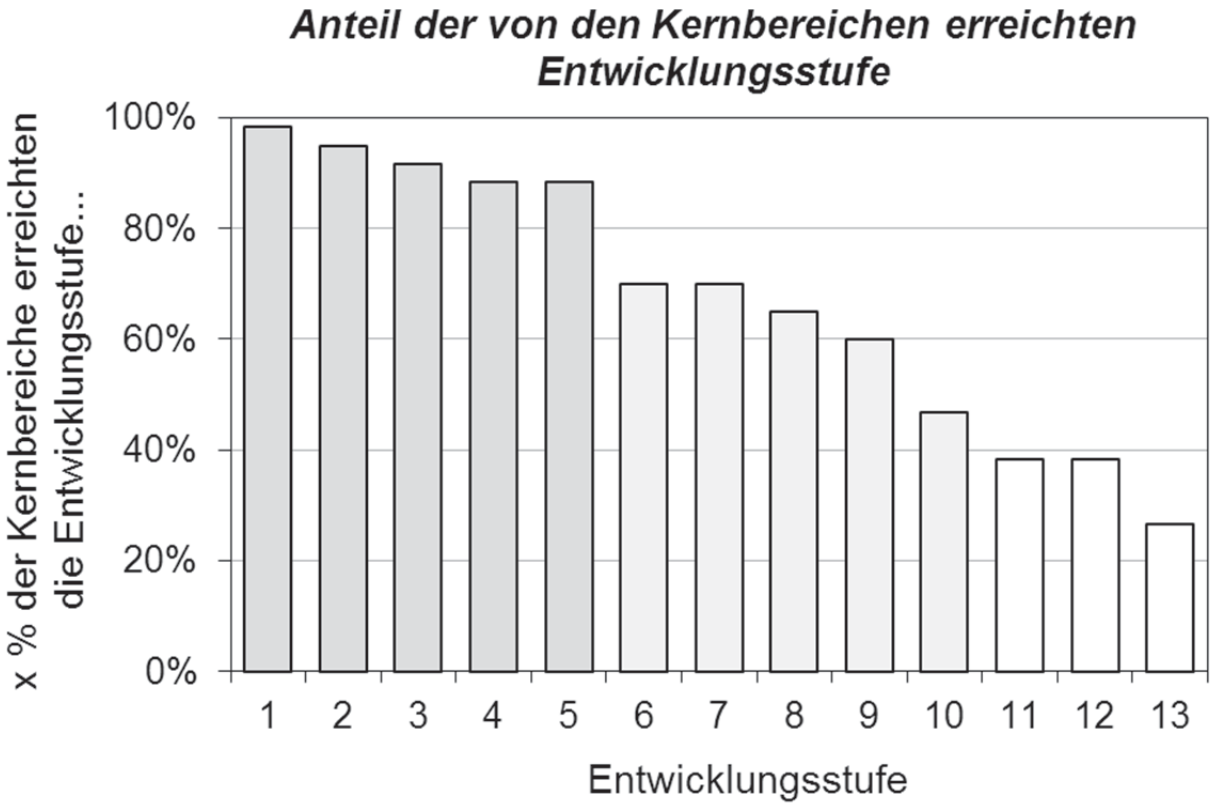


Abbildung 23: Übersicht der erreichten TPI-Reifegrade

Es bleibt zu beachten, dass einige Reifegradebenen nicht uneingeschränkt erfüllt sind. Soll wie bereits erläutert die Entwicklungsstufe 7 erreicht werden, sind verschiedene Verbesserungen im Prozess zum Test der MMS notwendig. Während in einigen Kernbereichen des Testprozesses umfangreiche Verbesserungsmaßnahmen notwendig sind (z. B. in den Kernbereichen 12, 13, 18 und 19), sind in anderen nur geringfügige Erweiterungen erforderlich, um die mit Einschränkungen erfüllten Kontrollpunkte vollständig zu erreichen.

Zur Implementierung des modellbasierten Vorgehens werden aus der TPI-Methodik schließlich konkrete Maßnahmen und Optimierungsvorschläge abgeleitet, um den erforderlichen Reifegrad zu erreichen. Ausgerichtet an den Phasen des fundamentalen Testprozesses sind hierbei folgende Themen zu berücksichtigen:

Planung und Steuerung: Im Rahmen der Testplanung ist ein schriftliches Testkonzept mit relevanten Testaktivitäten und einer Teststrategie zu erstellen. Dies geschieht zeitlich parallel zur Formulierung der Systemanforderungen. Der Testplan ist in den folgenden Aktivitäten des Testprozesses einzuhalten und die beschriebenen Inhalte umzusetzen. Abweichungen vom Testplan sind zu dokumentieren. In jedem Entwicklerteam ist mindestens eine Person zu bestimmen, die ausschließlich für das Testen zuständig ist. Zudem sind regelmäßig Testbesprechungen einzuführen, welche protokolliert werden. Außerdem sollen regelmäßige Testergebnisbesprechung zwischen Entwicklern, Systemlieferanten und Testmanagern stattfinden. Hierbei sind Absprachen der Testmanager mit dem Systemlieferanten zum Testvorhaben zu treffen.

Analyse und Design: Die erstellten Spezifikationen sind auf Konsistenz und Fehler hin zu überprüfen. Dabei sind das „4-Augen Prinzip“, Checklisten oder gleichwertige Methoden anzuwenden. Die Prüfungsergebnisse werden berichtet. Auch die Testbasis ist hinsichtlich ihrer Testbarkeit zu überprüfen. Dabei werden statische Tests oder andere formale Arbeitsweisen eingesetzt.

Realisierung: Im Rahmen der Testdurchführung sind automatische Hilfsmittel/Tools einzusetzen. Die technischen Aspekte und Anforderungen

der Tools sind zu dokumentieren. Verschiedene Testsituationen sollen gespeichert und wiederhergestellt werden können. Zudem sollen die Tools wiederverwendbar sein. Nichtfunktionale Qualitätsmerkmale sind ggf. mit Checklisten statisch zu testen.

Auswertung: Dem Management sind Einblicke in die Testtiefe zu ermöglichen. Hierfür ist zu definierten Zeitpunkten ein Testbericht zu erstatten.

Abschließend lässt sich feststellen, dass die Prozessreife der analysierten Prozesse zum Test der softwarebasierten MMS im Kfz insgesamt überzeugt. Zwar wurden in Abhängigkeit des betrachteten Prozesses und Kernbereichs unterschiedliche und teilweise geringe Reifegrade ermittelt, im Durchschnitt kann aber eine für die Einführung des MBT-Prozesses ausreichende Prozessreife festgestellt werden. Die aus der TPI-Methodik abgeleiteten Optimierungsvorschläge sind umzusetzen und werden für eine Angleichung und Verbesserung der Reife in allen Testprozessen sorgen. Mit Durchführung der Maßnahmen ist das Erreichen der siebten Prozessentwicklungsstufe realisierbar. Auf Basis dieser Beurteilung und Optimierung ist die Voraussetzung geschaffen, um das modellbasierte Testvorgehen zu implementieren und die in Kapitel 3.2 definierten Thesen zu beweisen.

4.3 Zusammenfassung

Um den MBT-Prozess für die Verifikation der softwarebasierten MMS im Kfz umzusetzen, stehen zunächst dessen Eigenschaften und Anforderungen im Vordergrund der Betrachtung. Durch die Erweiterung des W-Modells wird gezeigt, welchen (positiven) Einfluss das modellbasierte Vorgehen auf den

gesamten Entwicklungsprozess hat. Die früh stattfindenden Testaktivitäten ermöglichen eine Fehleridentifikation in frühen Projektphasen.

Um die konkreten Inhalte des MBT einer softwarebasierten MMS zu ermitteln und die notwendigen Änderungen im gesamten Testprozess darzustellen, werden die erforderlichen Testaktivitäten des manuellen Testens, automatisierten Testens und modellbasierten Testens ausführlich beschrieben. In der praktischen Betrachtung des Testvorgehens zeigt sich, dass die funktionalen Inhalte einer softwarebasierten MMS sinnvollerweise differenziert betrachtet werden. So kann die MMS als Verbund aus Dialogstruktur, Dialoginhalten, grafischer Umsetzung und Spracherkennung dargestellt werden. Im systematischen Test kann es sinnvoll sein, die Fehlersuche gezielt auf eines oder wenige dieser Elemente einzuschränken.

Werden die Aktivitäten der Prozesse zum manuellen, automatischen und MBT gegenübergestellt, zeigt sich, dass eine Weiterentwicklung zum MBT auf nahezu alle Testaktivitäten Einfluss hat. Dies zeigt, wie grundsätzlich und ganzheitlich die Methodik des MBT das Verständnis zum Test verändern kann. Dieses Potenzial führt jedoch gleichzeitig auch zu einer praktischen Herausforderung. Die praktische Umsetzung solch weitreichender Veränderungen ist schwieriger zu realisieren als abgegrenzte, überschaubare methodische Ergänzungen im Testprozess. Aus diesem Grund wird das Prozessmodell des „Test Process Improvement“ (TPI) herangezogen. Die vorhandenen Testprozesse werden auf ihre Reife hin beurteilt und notwendige Verbesserungsmaßnahmen können definiert werden. Durch die Entwicklung dieser Basis wird das Risiko, bei der Einführung neuer Prozesse und Testaktivitäten zu scheitern, deutlich reduziert und die Implementierung

des MBT realisierbar. Die Ergebnisse des TPI zeigen, dass sich die reale Prozessreife der aktuell vorhandenen Testprozesse auf einem zufriedenstellenden Niveau bewegt. Durch die Umsetzung der abgeleiteten, konkreten Verbesserungsvorschläge können vorhandene Defizite im Prozess abgestellt werden. Die Voraussetzungen für eine Umsetzung des modellbasierten Tests der softwarebasierten, akustischen MMS im Kfz sind somit erfüllt.

5 Modellierung der softwarebasierten MMS eines Kfz

Nachdem die prozessualen Rahmenbedingungen für die Einführung einer modellbasierten Vorgehensweise zum Test der softwarebasierten MMS im Kfz analysiert und ausgearbeitet sind, steht nun der Beweis der in Kapitel 3.2 beschriebenen These A im Vordergrund. Um diesen anzutreten werden zunächst Möglichkeiten zur Modellierung aufgezeigt, um schließlich eine formale Syntax zu entwickeln, welche als Testbasis für die anschließende Testfallgenerierung und -durchführung Verwendung finden kann.

5.1 Modelltheorie und Grundlagen

5.1.1 Die Spezifikation der Anforderungen

Visuelle Informationen werden vom menschlichen Gehirn effizienter aufgenommen als eine rein textuelle Beschreibung selben Inhalts. Abbildung 24 veranschaulicht diese Tatsache. Auf der linken Seite ist ein einfacher Sprachdialog rein textuell dargestellt, auf der rechten Seite ist selbiger anhand visueller Elemente beschrieben.

An der Entwicklung der softwarebasierten MMS im Kfz sind mehrere Parteien beteiligt (Auftraggeber und mindestens ein Systemlieferant). Zur Umsetzung der komplexen Entwicklungsziele ist eine vollständige Systembeschreibung ohne inhaltlichen Interpretationsspielraum dringend erforderlich. Nur so kann gewährleistet werden, dass alle an der Entwicklung beteiligten Parteien die

gleiche Zielvorstellung des zu entwickelnden Systems besitzen [Rup09]. Aus diesem Grund werden in der Praxis neben der textuellen Anforderungsbeschreibung auch Spezifikationen erstellt, in welchen visuelle Elemente zur Beschreibung der MMS verwendet werden.

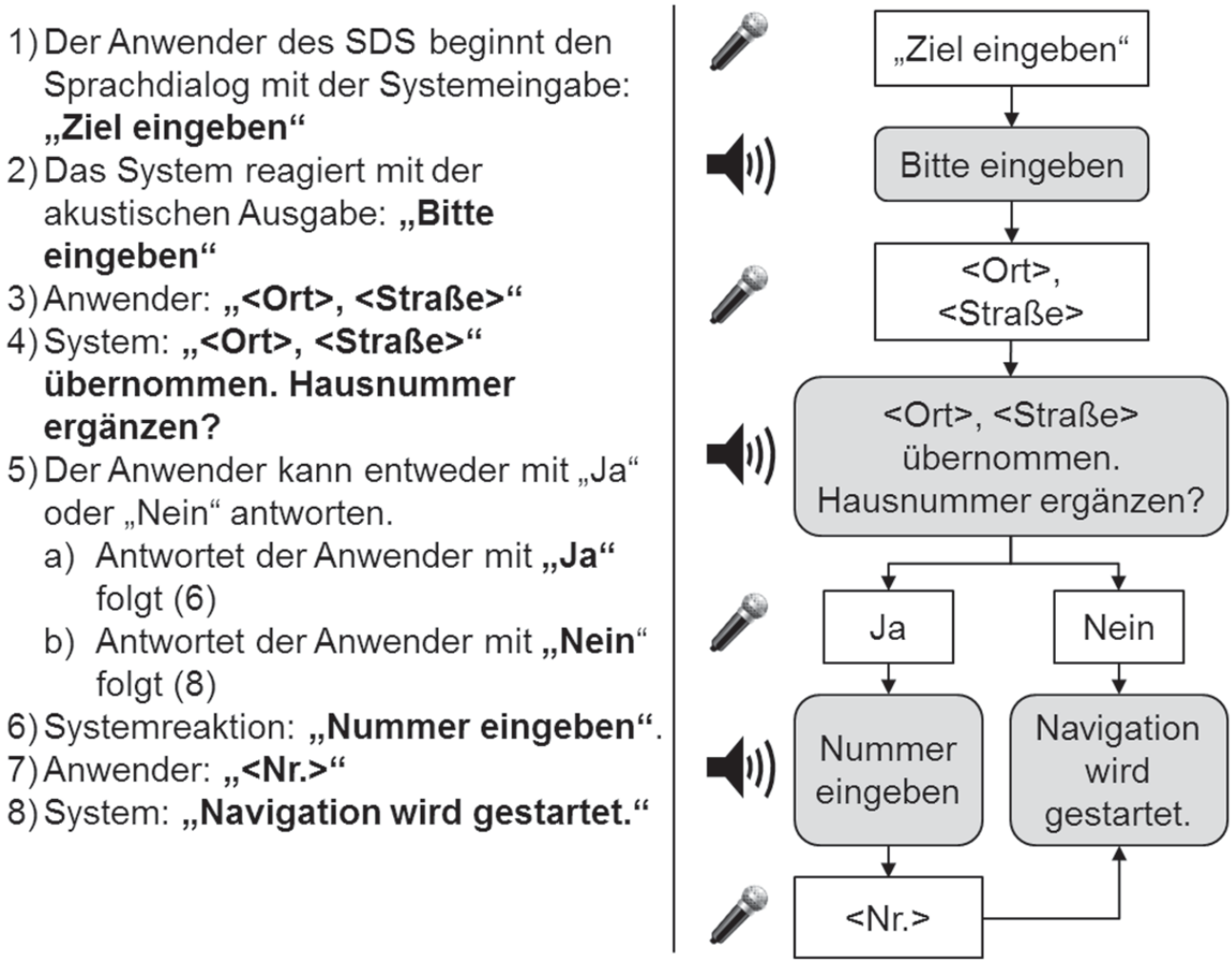


Abbildung 24: Gegenüberstellung textueller und grafischer Beschreibung

In diesen Spezifikationen werden sowohl die Dialogstruktur als auch Dialoginhalte dargestellt. Auch die grafische Umsetzung wird gegebenenfalls durch entsprechende Abbildungen aufgezeigt. Die vorhandenen Spezifikationen können dabei grundsätzlich als Systemmodell interpretiert werden, da

Aufbau, Funktionalität und Verhalten des Systems vollständig beschrieben sind. Die verwendeten Formalismen zur Systembeschreibung sollen im Folgenden kurz beschrieben werden.

Die Spezifikation des Sprachdialogsystems nutzt die Notation der DIN 66001 zur grafischen Darstellung des SDS in Form eines Ablaufdiagramms (engl. flowchart). Verschiedene Elemente der allgemein formulierten Norm sind dabei modifiziert, um die individuellen Anforderungen der Dialog-beschreibung zu erfüllen. Ablaufdiagramme beschreiben das Systemverhalten als eine Folge voneinander klar abgegrenzter Aktivitäten. Erst wenn eine Aktivität endet, startet die nächste. Die sequenziellen Sprachdialoge können in diesen Diagrammen daher sehr gut beschreiben werden. In Abbildung 25 sind die wichtigsten, verwendeten Elemente dargestellt.

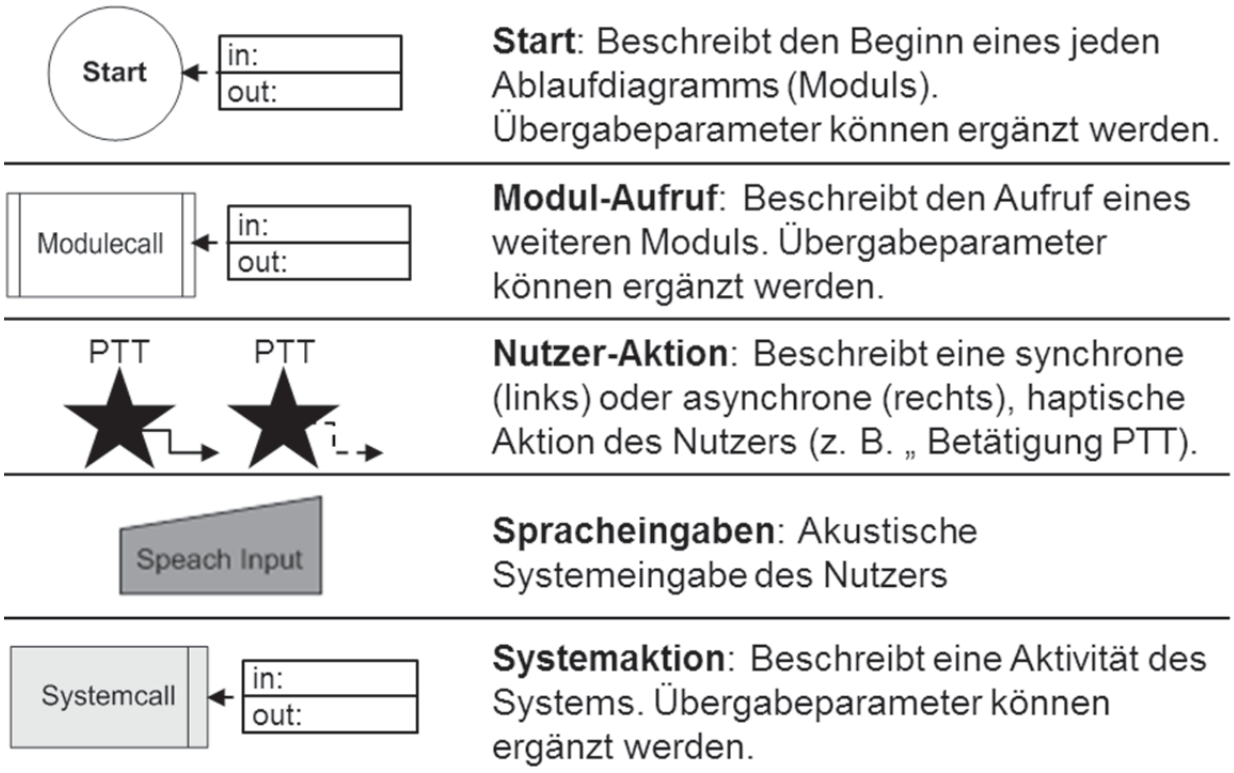


Abbildung 25: Elemente der Dialogbeschreibung im Ablaufdiagramm (Teil 1)

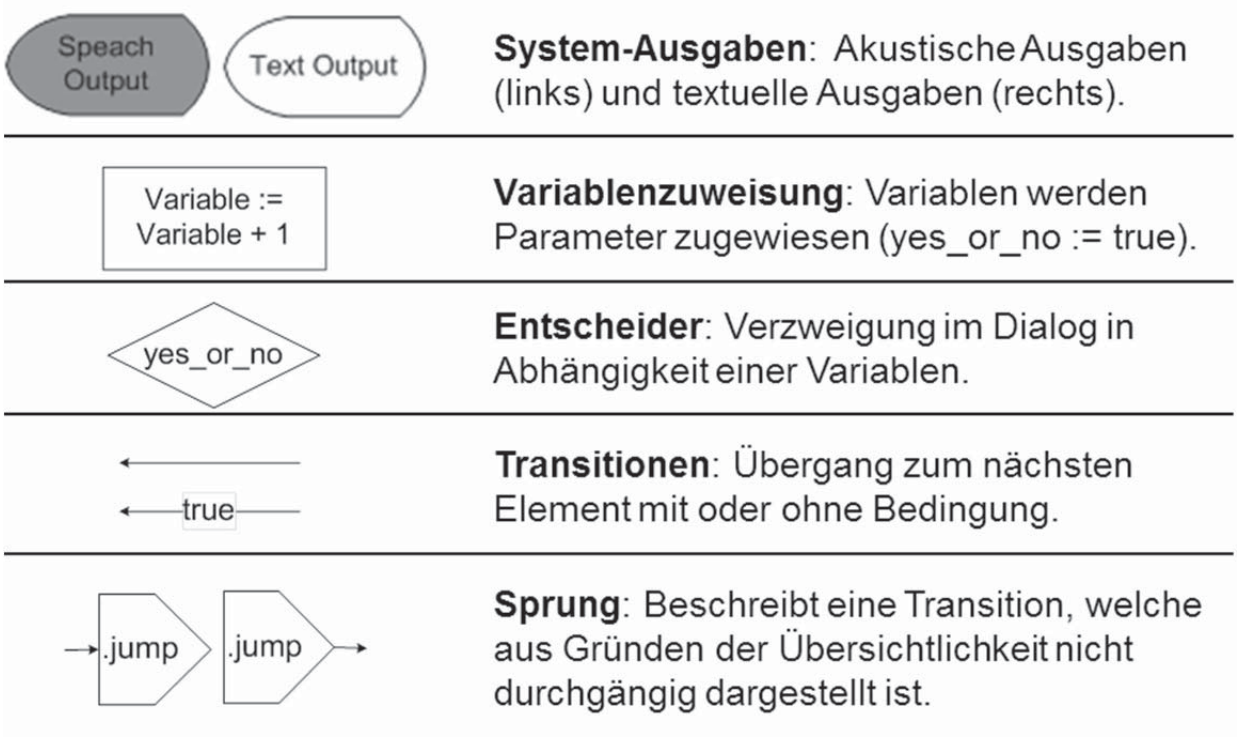


Abbildung 26: Elemente der Dialogbeschreibung im Ablaufdiagramm (Teil2)

Abbildung 27 zeigt beispielhaft, wie die beschriebenen Elemente Anwendung finden. Die Dialogbeschreibung beginnt mit Verwendung des Start-Elements, in welchem der Name des jeweiligen Moduls ergänzt ist. Es folgt die Nutzeraktion zur Betätigung des PTT, um die Sprachbedienung zu aktivieren. Daraufhin reagiert das System mit einer akustischen Rückmeldung. Nach dieser Aktivierung ist der Erkenner für Eingaben geöffnet (Systemaktion: in Recognition). Die Nutzereingabe „PIN eingeben“ führt zur nächsten Systemaktivität. In Abhängigkeit der Umgebungsbedingungen, ob ein Telefon vorhanden ist oder nicht, verzweigt sich der Dialog. Wird kein Telefon in der Systemumgebung erkannt, führt dies zu aufeinanderfolgenden, akustischen Systemausgaben („Telefon nicht verfügbar, Abbruch“) und der Dialog ist beendet. Ist ein Telefon vorhanden und befindet sich im korrekten Zustand, reagiert das System mit der akustischen Ausgabe „Bitte PIN eingeben“,

welche durch die textuelle Anzeige („PIN eingeben“) im Display unterstützt wird. Der Nutzer kann jetzt seine PIN als variable Zahlenfolge eingeben. Die erkannte Eingabe wird zur Kontrolle durch den Nutzer sowohl akustisch als auch optisch ausgegeben.

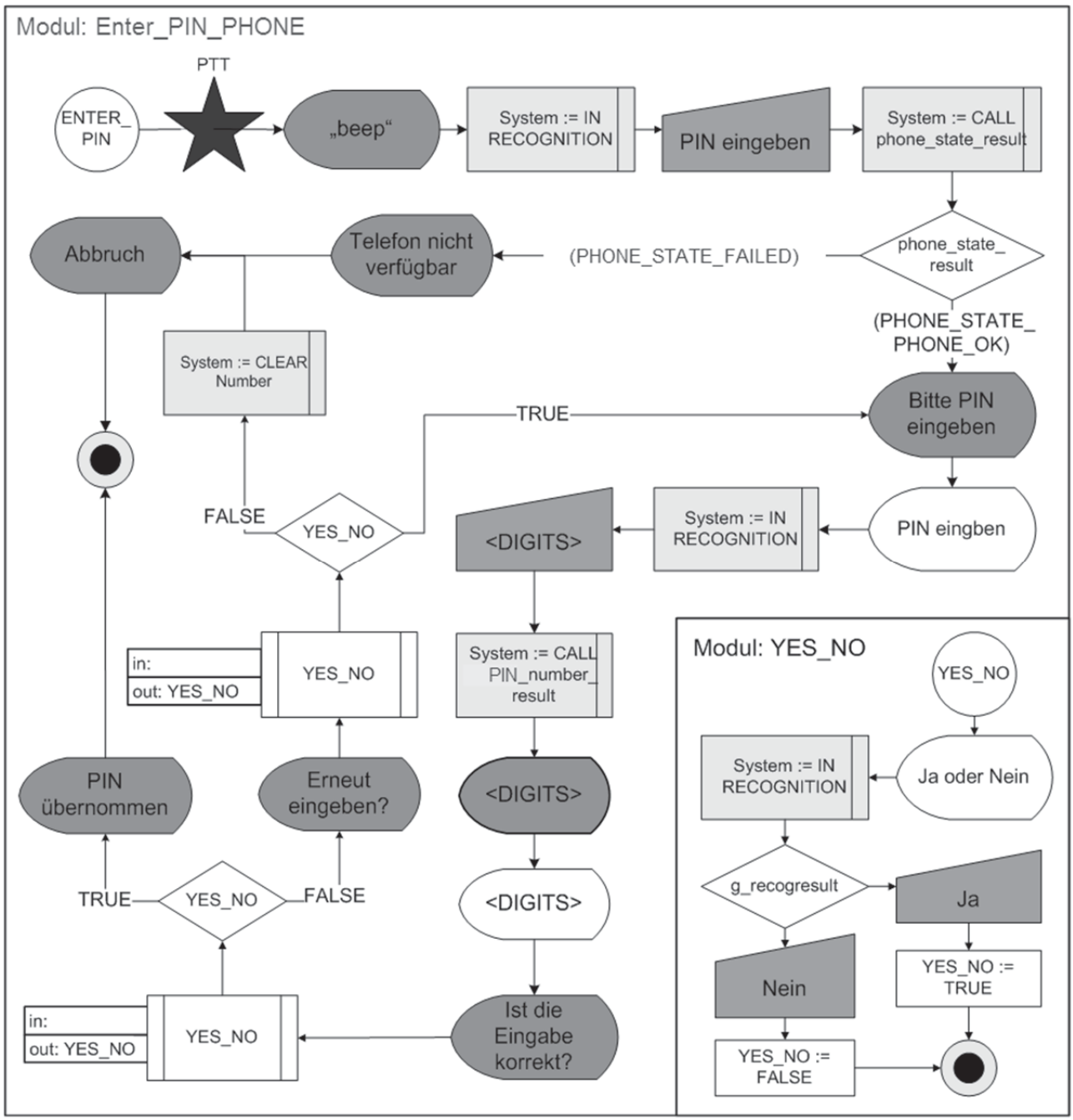


Abbildung 27: Sprachdialog im Ablaufdiagramm (Bsp.)

Nachdem das System die akustische Rückfrage zur Absicherung der Systemeingabe ausgegeben hat, wird das Modul „YES_NO“ aufgerufen, in welchem die Ja/Nein-Abfrage spezifiziert ist. An dieser Stelle wird der Vorteil einer modularen Spezifikation deutlich. Um die häufig vorkommende Ja/Nein-Abfrage nicht in jedem Dialog modellieren zu müssen, ist diese einmalig und gesondert in einem Modul spezifiziert, welches in verschiedenen Dialogen aufgerufen werden kann.

Im Modul „YES_NO“ wird, abhängig von der erkannten Benutzeräußerungen, der Booleschen Variablen „YES_NO“ der Wert „true“ für „YES“ oder „false“ für „NO“ zugewiesen. Die Variable und deren Zuweisung wird vom Modul „YES_NO“ in das Modul „Enter_PIN“ übergeben und an der folgenden Dialog-verzweigung abgefragt. Wurde mit „Ja“ geantwortet, reagiert das System mit der akustischen Rückmeldung „PIN übernommen“ und der Dialog ist beendet. Andernfalls wird dem Nutzer die akustische Rückfrage gestellt, ob er den PIN erneut eingeben will. Es folgt ein erneuter Aufruf des Moduls „YES_NO“ in Abhängigkeit dessen sich der Dialog erneut verzweigt. Möchte der Nutzer den PIN nicht erneut eingeben, wird die vorhandene Eingabe vom System gelöscht und der Dialog nach der akustischen Ausgabe „Abbruch“ beendet. Möchte der Nutzer die PIN erneut eingeben, wird der Dialog zur PIN Eingabe wiederholt.

Die vorhandene Methodik zur Spezifikation, bzw. der Systemmodellierung des SDS, ist nach kurzer Einarbeitungszeit gut lesbar und verständlich. Durch die Möglichkeit, den Sprachdialog in verschiedene Module aufzuteilen und entsprechend zu spezifizieren, gewinnen die einzelnen Ablaufdiagramme an Übersichtlichkeit. Zudem können die Module leicht von verschiedenen

Personen parallel spezifiziert werden. Kritisch ist festzuhalten, dass keine detaillierte Syntaxbeschreibung oder ein Regelwerk zur Spezifikationserstellung vorhanden ist. Um trotz dieser Umstände eine konsistente Modellierung der akustischen MMS zu realisieren, werden nach der Modellierung Syntaxchecks durchgeführt und die grundlegenden Modellierungsregeln geprüft. Diese Prüfung kann auch als statischer Test des Systemmodells verstanden werden und trägt dazu bei, die formale Qualität des Ablaufdiagramms zu erhöhen.

Die **Spezifikation der haptischen, softwarebasierten MMS** findet in unterschiedlichen Formaten statt. Ursächlich hierfür sind die grundsätzlich verschiedenen Inhalte und Eigenschaften der Bedien- und Anzeigekonzepte des Kombiinstruments (KI) und der Head Unit (HU). Die Inhalte des KI sind von geringerer Tiefe als die der HU. Die im KI hinterlegten Informationen werden über eine stringente Menüführung erreicht und abgefragt. Die Bedienelemente finden sich am Lenkrad und beschränken sich auf sechs Tasten.

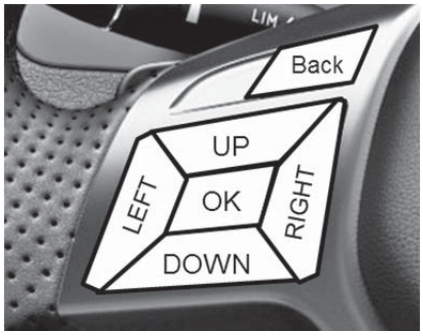


Abbildung 28: Bedienelemente des Kombiinstruments

In der Menüführung lassen sich die angezeigten Elemente über vier Richtungstasten anwählen. Bestätigt wird mit der „OK“-Taste, woraufhin das

untergeordnete Menü oder die ausgewählte Funktion aufgerufen wird. Die „BACK“-Taste ermöglicht jeweils den Rücksprung in die vorhergehende, bzw. übergeordnete Menüstruktur.

In Abbildung 29 ist beispielhaft ein Ausschnitt der Menüstruktur im hierfür verwendeten Spezifikationsformat dargestellt. Aus der Abbildung geht hervor, dass verschiedene formale Elemente verwendet werden, um die Schnittstelle zwischen Mensch und KI eindeutig zu definieren. Zum einen wird die Dialogstruktur formal, unter Verwendung UML-konformer Zustands-diagramme dargestellt (siehe auch Kapitel 5.1.2), zum anderen werden diese durch grafische Elemente erweitert[2] (grau hinterlegt). Der Dialog startet in der Listenansicht des Hauptmenüs. Die grafische Darstellung (Abbildung 29, links) veranschaulicht das ausgewählte Element im Dialog und das Listenverhalten durch Betätigung der entsprechenden Bedienelemente. In Abhängigkeit des angewählten Listeneintrags führt die Aktion „PUSH OK“ in das entsprechende Untermenü, dessen Inhalt in weiteren Zustands-diagrammen spezifiziert ist. Durch die Erweiterung der Zustandsdiagramme mit grafischen Elementen erhöht sich einerseits die Lesbarkeit, andererseits wird durch deren Verwendung die formale Durchgängigkeit (der Zustands-diagramme) unterbrochen.

[2] Die grafischen Elemente machen im Allgemeinen keine Vorgaben zur gestalterischen Umsetzung der Anzeigen (Textformat…) sondern haben beschreibenden Charakter.

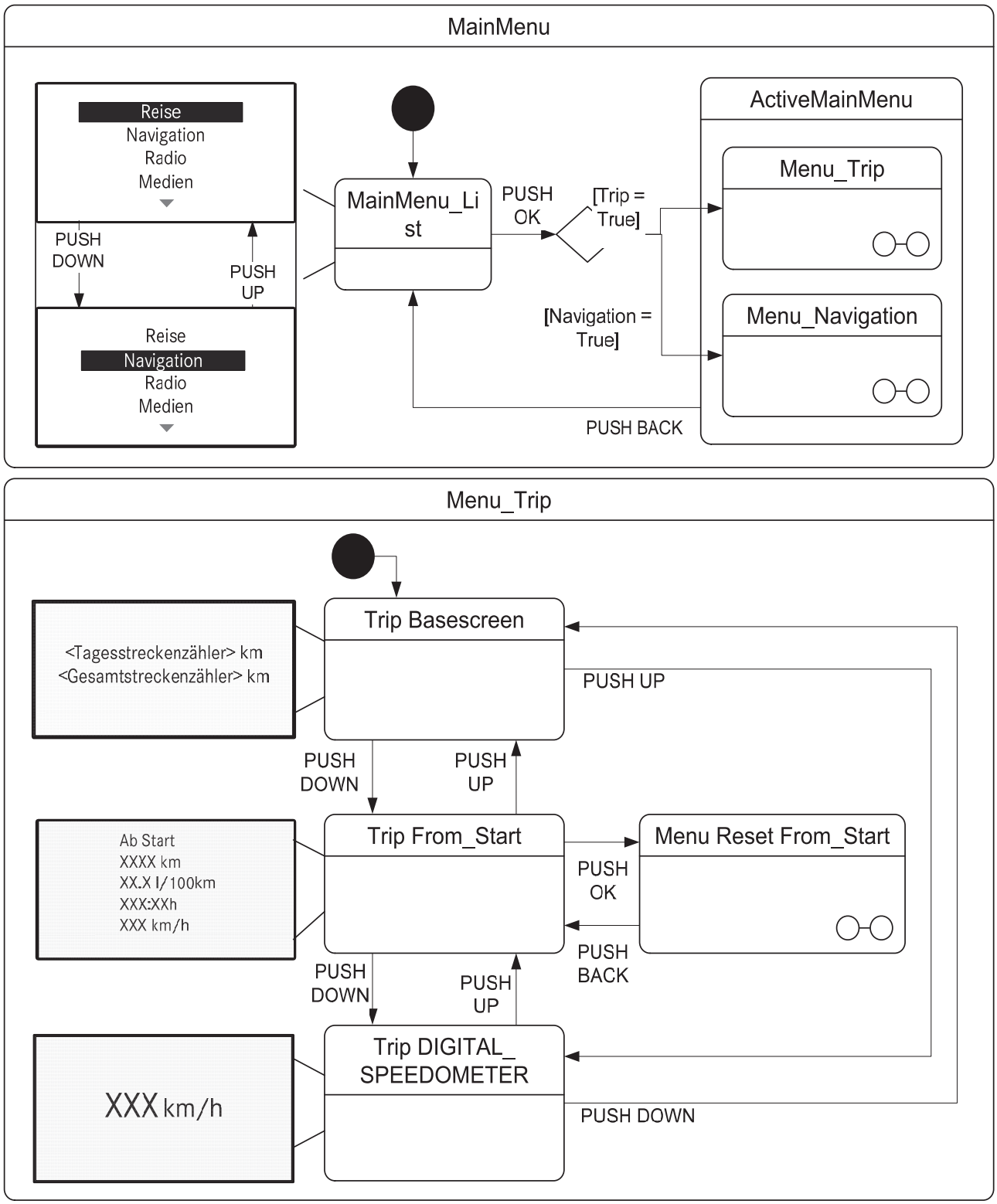


Abbildung 29: Bedienung des KI im erweiterten Zustandsdiagramm

Die Spezifikation der Bedien- und Anzeigekonzepte für die umfangreichen und komplexen Inhalte der HU wird ebenfalls mit textuellen und grafischen Spezifikationselementen realisiert. Die Beschreibung erfolgt in verschiedenen Dokumenten und Formaten. Die Formatierung der zentralen Dokumente ist in Abbildung 30 dargestellt. Das Beispiel zeigt, wie das Wellenband der Radioapplikation geändert werden kann.

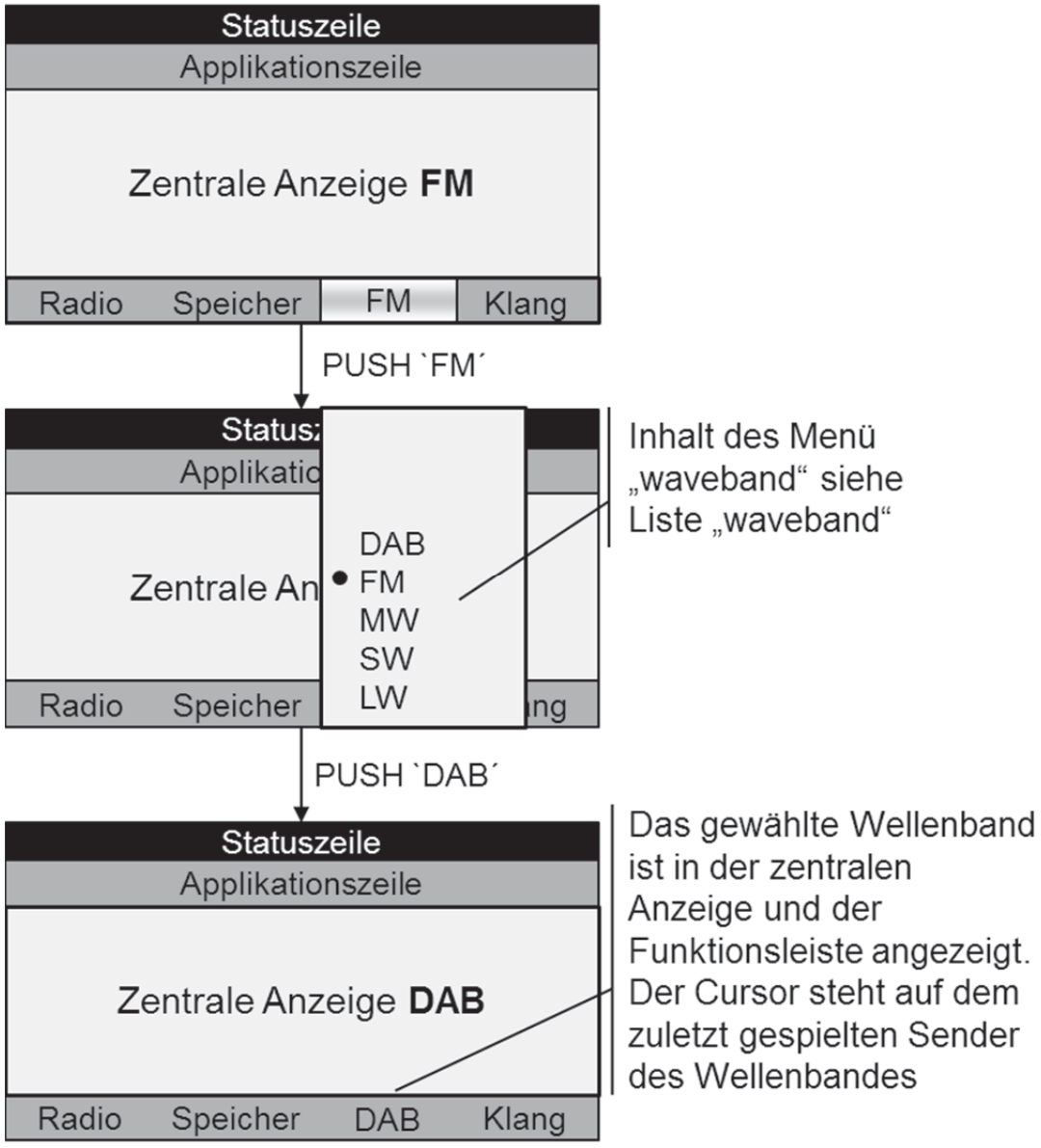


Abbildung 30: Bedienung der Head Unit in einer grafisch/textuellen Darstellung

Zu diesem Beispiel wird neben den in der Grafik eingebundenen Kommentaren textuell spezifiziert:

> *„Es gibt zwei Möglichkeiten das Wellenband zu ändern (DAB, FM, MW, SE, LW):*
> *Durch Auswahl und Bestätigung des Wellenband-Menü-Elements in der Funktionsleiste und anschließender Auswahl des gewünschten Wellenbands in der aufgerufenen Menüliste (das aktuell verwendete Wellenband ist markiert), siehe Abbildung 30. Der Tuner spielt nach Auswahl den zuletzt gespielten Sender des gewählten Wellenbands.*
>
> *Hinweis: Das Wellenband-Menü-Element gibt Rückmeldung zum aktuell aktiven Wellenband. Das Element des Wellenband-Menüs ändert sich folglich entsprechend des aktuellen Wellenbands (ist z. B. „FM“ gewählt wird das Element „FM“ angezeigt).“*

Im Gegensatz zur Spezifikation der MMS im KI und des SDS, ist die Spezifikation der HU weniger formal, dafür jedoch besser lesbar. Im beschriebenen Beispiel ist zu erkennen, dass zentrale Inhalte in der aufgezeigten Spezifikation nicht beschrieben sind. So fehlt die Information, welche haptischen Aktivitäten des Nutzers erforderlich sind, um ein bestimmtes Element in der Menüliste auszuwählen und zu bestätigen. Menülisten wird ein allgemeingültiges Verhalten zugewiesen (z.B. das Öffnen und Schließen der Liste), welches formal unter Verwendung von UML Zustandsdiagramm beschrieben wird und die Spezifikation in weiteren Dokumenten ergänzt. Des Weiteren besteht eine formale, textuelle Beschreibung der kompletten Menüstruktur im XML-Format.

Die beschriebenen Spezifikationen zur Definition der akustischen und haptischen MMS eines Infotainmentsystems zeigen, dass erhebliche Aufwänden betrieben werden, um alle Systeminhalte vollständig und eindeutig zu beschreiben. Betrachtet man die verwendeten Spezifikationen als Systemmodelle und die Inhalte nicht primär als Beschreibung einer MMS, sondern als Definition einer komplexen Software, werden verschiedene, kritische Eigenschaften der bestehenden Spezifikationsformate deutlich. Keine der verwendeten Spezifikationen hält sich durchgängig und ausschließlich an *ein* Standardformat zur Beschreibung. Hinzu kommt, dass die formalen Regeln der Spezifikation nicht in ausreichendem Umfang schriftlich festgelegt sind. Die Bearbeitung durch Spezifikateure mit unterschiedlichem Verständnis zur verwendeten Syntax führt zu erheblichen Inkonsistenzen innerhalb der Dokumente. Die beschriebene statische Syntaxprüfung ist in der Lage, grundlegende Fehler zu identifizieren, deckt jedoch bei weitem nicht alle notwendigen Formalitäten ab und kann die formale Konsistenz der Dokumente nicht ausreichend sicherstellen. In der Spezifikation der haptischen MMS werden zwar zum Teil Standardformate verwendet (wie z. B. UML), diese sind jedoch in großem Umfang durch proprietäre, grafische Elemente ergänzt. Teilweise werden diese Spezifikationen auch durch natürlichsprachliche Elemente dominiert. Die verwendeten Werkzeuge zur Spezifikation beschränken sich weitgehend auf Standardprogramme wie z. B. Microsofts „Visio“, „Word“ und Adobes „Photoshop“. Mit dem Ziel, modellbasiert zu testen und einen kommerziellen Testfallgenerator einzusetzen, entsteht die Notwendigkeit der Verwendung standardisierter Formate. Nur so kann sichergestellt werden, dass auch standardisierte Schnittstellen

zwischen den Werkzeugen zur Modellierung und Testfallgenerierung Verwendung finden können.

Formale Beschreibungstechniken lassen sich in textbasierte und grafische Ansätze unterscheiden. Bei den formalen, textbasierten Techniken ist die „Extensible Markup Language“ (XML) als eine der wichtigsten Techniken zur Spezifikation und zum Austausch hierarchisch strukturierter Inhalte und Daten zu nennen. Es bestehen verschiedene XML-Dialekte, Datenformate wie das „XML Metadata Interchange“ (XMI) der „Object Management Group“ (OMG) welches auf dem XML aufsetzt und den Datenaustausch von UML-Modellen (siehe Kapitel 5.1.2) ermöglicht. Weitere Informationen zur XML finden sich in entsprechender Fachliteratur wie [Bra08] und [Har02]. Die wichtigste formale, grafische Beschreibungstechnik ist die „Unified Modeling Languange“ (UML). Nach [Rup05] ist die UML eine Art „Weltsprache“ der Softwareentwicklung, und Roßner et. al. stellen fest, dass sich die UML „als „Lingua franca“ für die Modellierung auf breiter Basis durchgesetzt hat“ [Roß10].

5.1.2 Modellierung in UML

Die „Unified Modeling Languange” (UML) ist seit 2005 durch die ISO 19501 international standardisiert. Seitdem wurden die Inhalte der UML durch die „Object Management Group“ (OMG) kontinuierlich weiterentwickelt. Folgend wird mit Verwendung des UML Begriffs auf die seit 2011 veröffentlichte Version 2.4 der UML Bezug genommen. Die Spezifikationen der verschiedenen UML Versionen sind unter [Obj11] abrufbar.

Die UML definiert verschiedene Möglichkeiten der grafischen Darstellung eines Modells. Diese Darstellungen werden als Diagramm bezeichnet. Abbildung 31 zeigt eine Übersicht der verschiedenen Diagrammtypen. Es findet eine grundsätzliche Unterscheidung zwischen struktur- und verhaltensbeschreibenden Diagrammen statt.

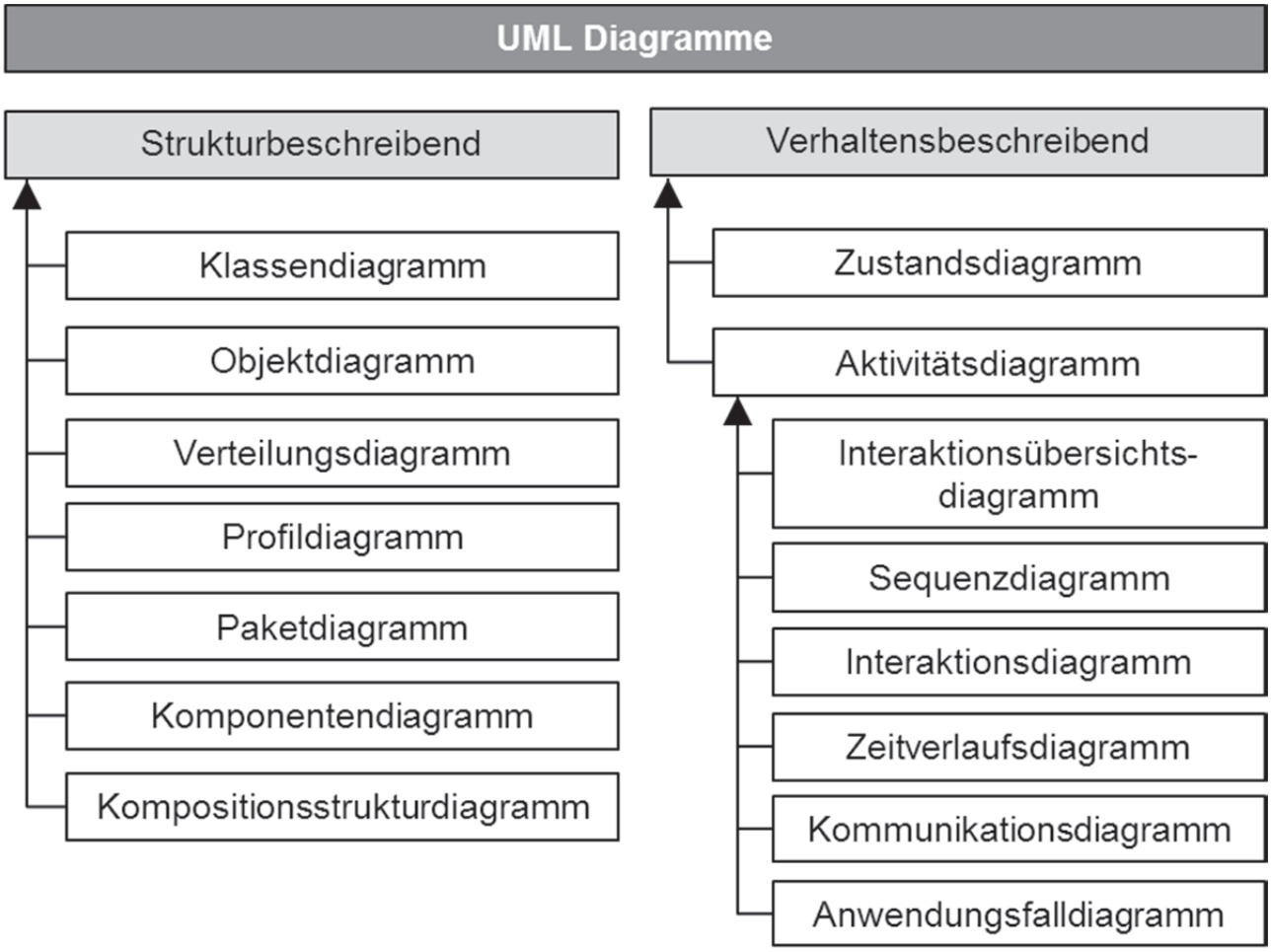


Abbildung 31: Hierarchie definierter Diagramme in UML2

Strukturbeschreibungen werden insbesondere dann verwendet, wenn der technische Aufbau eines Systems darzustellen ist. Im Klassendiagramm kann so beispielsweise die Struktur von Objekten in verschiedenen Klassen definiert werden, sowie das Zusammenspiel dieser Klassen. In verhaltens-

beschreibenden Diagrammen lassen sich hingegen dynamische (System-) Abläufe oder Prozesse beschreiben. Nachdem die Interaktion zwischen Mensch und Maschine ein dynamischer Ablauf ist, stehen die verhaltens-beschreibenden Diagramme im Fokus der weiteren Betrachtung. An dieser Stelle sei bereits auf entsprechende Fachliteratur für weiterführende Informationen wie [Boo06] und [Obj11] verwiesen.

Die Modellierung dynamischen Systemverhaltens kann in Aktivitäts- oder Zustandsdiagrammen stattfinden. Aktivitätsdiagramme beschreiben ein ablauforientiertes Systemverhalten. Aktivitäten bestehen dabei aus mehreren, miteinander verknüpften Aktionen. Es entsteht ein Graph, dessen Knoten die definierten Aktionen beschreiben und dessen Kanten den Ablauf zwischen den modellierten Aktivitäten aufzeigt. Abbildung 32 zeigt ein einfaches Aktivitätsdiagramm.

In Zustandsdiagrammen kann modelliert werden, wie ein System in Abhängigkeit seines aktuellen Zustands und seiner Randbedingungen auf Aktivitäten reagiert. Zustände sind stabil und können einzig durch das Auftreten von Aktionen oder Ereignissen verlassen werden, um in Abhängigkeit dieser in einen Zielzustand überzugehen. In Zustandsdiagramm ist folglich ein reaktives Zustandsverhalten modelliert. Ist an einem Zustandsübergang keine Aktivität oder ein Ereignis modelliert, wird dieser unmittelbar ausgeführt. Abbildung 33 zeigt ein einfaches Zustandsdiagramm, welches einen direkten Vergleich zum Aktivitätsdiagramm in Abbildung 32 zulässt. Die an den ausgehenden Transitionen der Entscheider beschriebenen Bedingungen werden auch „Guards“ oder „Guard Conditions“ bezeichnet.

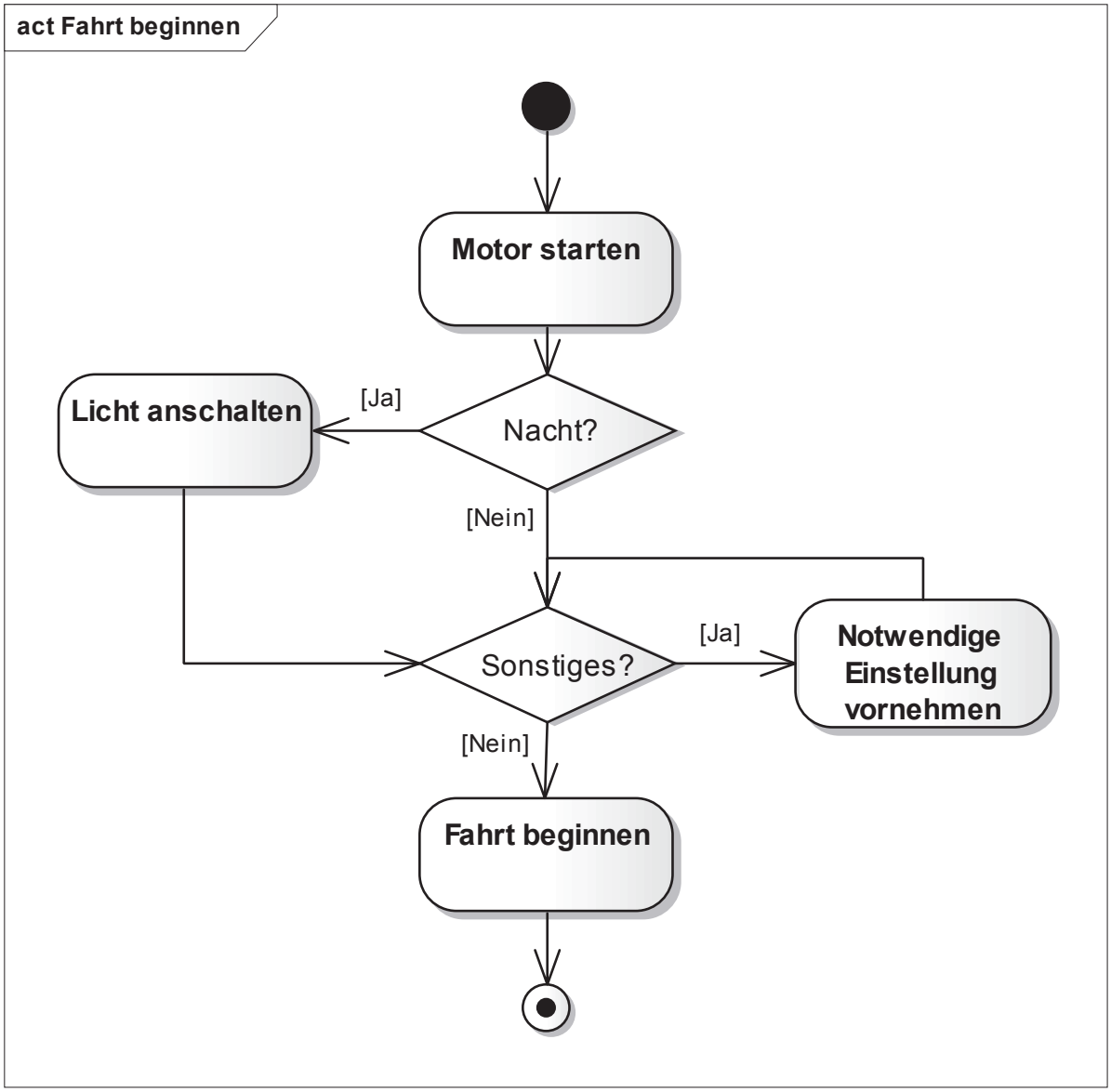


Abbildung 32: Beispiel für ein Aktivitätsdiagramm

Da die Modellierung komplexer Systeme auch in Zustandsdiagrammen schnell umfangreich und unübersichtlich werden kann, ist es sinnvoll, das Modell hierarchisch aufzuteilen. In Abbildung 33 ist der Aufruf des Diagramms „Fahrzeug in Bewegung“ modelliert. In dem aufgerufenen Diagramm kann das interne Verhalten des Zustands „Fahrzeug in Bewegung“ beschrieben werden. Durch diese modulare Verfeinerung verschiedener Zustände kann

das Modell sinnvoll und effizient strukturiert werden, wodurch die Lesbarkeit erhöht und die Bearbeitung sowie Prüfung der Modelle vereinfacht wird. Weiterführende Informationen zur Notation und Verwendung von Zustandsdiagrammen findet sich in fachspezifischer Literatur wie [Rup05].

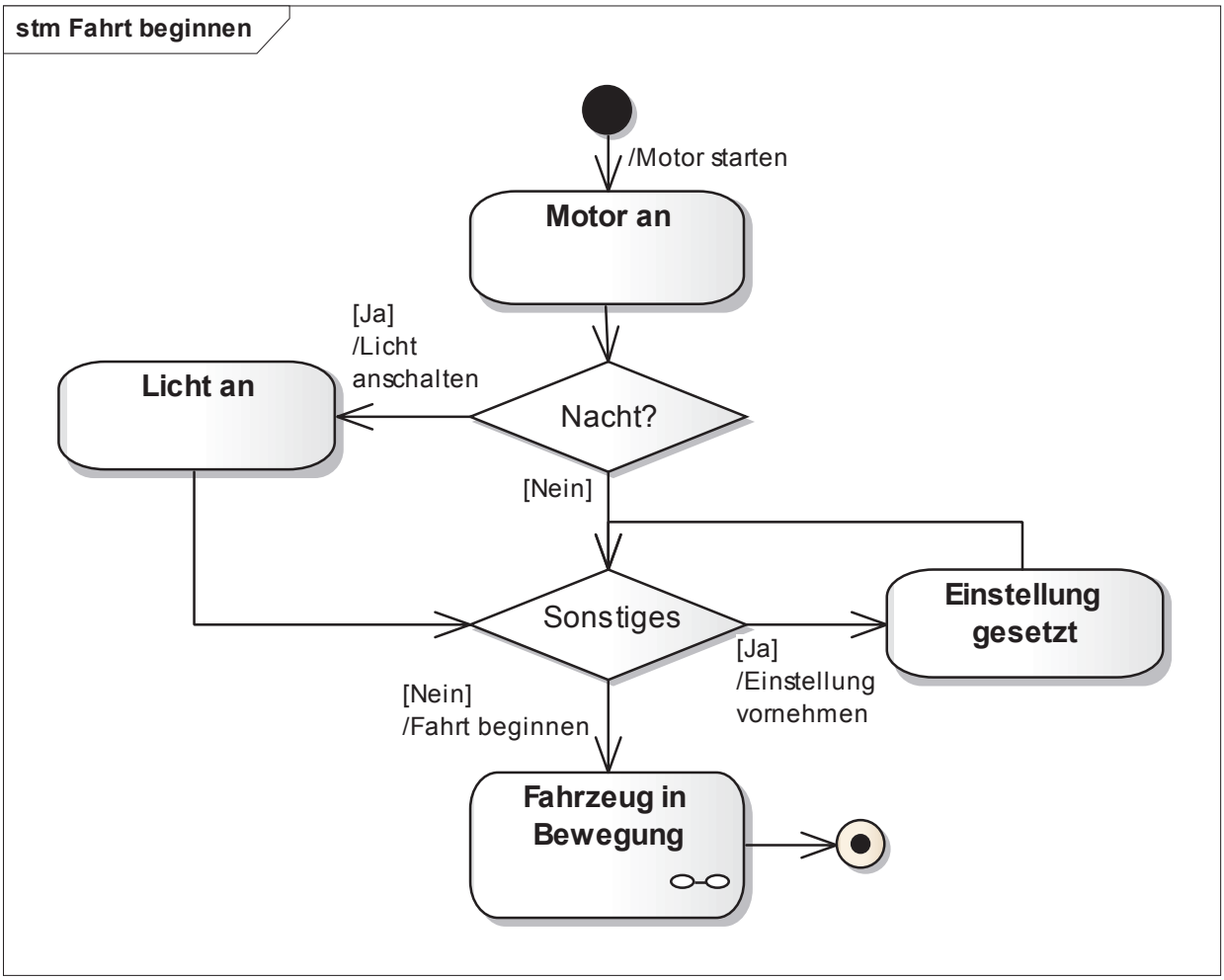


Abbildung 33: Beispiel für ein Zustandsdiagramm

In der weiten Verbreitung und Akzeptanz der UML liegt die hohe Verfügbarkeit kommerzieller Werkzeuge zur Modellierung begründet. Viele Werkzeuge ermöglichen sowohl die Modellierung in struktur- als auch verhaltensbeschrei-

benden Diagrammen. In Tabelle 8 ist eine Übersicht aktuell verfügbarer Werkzeuge in Abhängigkeit der Anschaffungskosten aufgelistet.

Tabelle 8: Werkzeuge zur Modellierung in UML

UML Werkzeug	Kosten (ab)	Klassendiagramm	Objektdiagramm	Verteilungsdiagramm	Profildiagramm	Paketdiagramm	Komponentendiagramm	Kompositionsstrukturdiagramm	Zustandsdiagramm	Aktivitätsdiagramm	Interaktionsübersichtsdiagramm	Sequenzdiagramm	Zeitverlaufsdiagramm	Kommunikationsdiagramm	Anwendungsfalldiagramm	XMI export	XMI import
UMLet	frei	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
objectiF	frei	x			x	x	x		x	x		x			x	x	x
ArgoUML	frei	x	x	x		x	x		x	x		x		x	x	x	x
Visual Classworks	50 $	x															
Astah	90 $	x	x	x		x	x	x	x	x		x		x	x	x	x
Enterprise Architect	99 €	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Sinelabore Codgen	109 €								x								x
Metamill	110 €	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Altova Umodel	149 $	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
MetaEdit+	150 €	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Modelio	150 €	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x
ModelMaker	199 €	x	x	x		x	x	x	x	x	x	x		x	x		
WinA&D	495 $															x	
EclipseUML	3950 $	x	x	x	x	x	x	x	x	x		x		x	x	x	x
AnyLogic	4.800 €	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
Poseidon for UML	48 €/Jr.	x	x	x		x	x		x	x		x			x	x	x
UML Lab	199 € /Jr.	x			x	x										x	x
SAP Sybase PowerDes.	k.A.	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x

Zudem zeigt die Tabelle auf, welche Diagrammtypen unterstützt und modelliert werden können. Um die Schnittstelle zu einem kommerziellen Testfallgenerator realisieren zu können, ist zudem betrachtet, ob erstellte Modelle im XMI-Format im- bzw. exportiert werden können. Eine erweiterte Darstellung der Tabelle findet sich im Anhang 12.2.

Grundsätzlich eignen sich mehrere der aufgezeigten Werkzeuge für die Modellierung einer Dialogstruktur in einem verhaltensbeschreibenden Diagramm. Aus Gründen der umfangreichen Modellierungsmöglichkeiten, Schnittstellen im Standardformat und Verfügbarkeit wird im Folgenden das Werkzeug „Enterprise Architect“ von „Sparx Systems“ verwendet. Es sei jedoch darauf hingewiesen, dass die Modellierung auch durch Verwendung eines anderen Werkzeugs stattfinden kann, solange dieses die entsprechenden Standardformate und -Schnittstellen der UML unterstützt.

Nachdem die Rahmenbedingungen der vorhandenen Spezifikationsmodelle und Grundlagen der UML beschrieben sind, soll im Folgenden ein formal durchgängiges Modell der softwarebasierten MMS im Kfz entwickelt werden.

5.2 Entwicklung des MMS-Modells

In Kapitel 5.1.1 sind die aktuell verwendeten Dokumente zur Spezifikation der softwarebasierten MMS beschrieben. Um den Beweis der in Kapitel 3.2 beschriebenen These A anzutreten und den MBT-Prozess umzusetzen, ist ein formal korrektes Modell der akustischen MMS notwendig. Aus diesem Modell sollen, mit Verwendung eines kommerziellen Testfallgenerators, Testfälle automatisch abgeleitet werden können. Die Anforderungen an ein solches

Modell unterscheiden sich von jenen, welche aktuell an die Spezifikationsdokumente der MMS gestellt sind.

5.2.1 Anforderungen an das Modell einer softwarebasierten MMS

Mit dem Ziel der Verwendung im MBT-Prozess muss das Modell der akustischen MMS folgenden Anforderungen entsprechen:

1) *Der Formalismus zur Modellierung muss möglichst einfach und leicht verständlich sein*.

   Der Erfolg des MBT hängt stark von dessen Akzeptanz durch die betroffenen Entwickler ab. Ein schwieriger und unverständlicher Formalismus wird kaum Akzeptanz finden. Widerstände bei dessen Einführung wären die Folge und das Potenzial der modellbasierten Vorgehensweise ist verspielt.

2) *Die Modellierung soll im Standardformat der UML stattfinden und im standardisierten Austauschformat XML/XMI exportiert werden können*.

   Mit Erfüllung dieser Anforderungen kann die Anbindung an kommerzielle Testfallgeneratoren vorbereitet werden. Ein proprietäres Modellformat würde von kommerziellen Tools nicht ohne weitere Anpassungen interpretiert werden können und die Einführung des MBT wäre gefährdet.

3) *Alle für den Test relevanten Systeminhalte müssen modelliert werden können*.

   Die Möglichkeit des Formalismus muss ausreichend mächtig sein, um die akustische MMS hinreichend zu beschreiben. Als testrelevant werden hier mit Verweis auf Kapitel 4.1.2, Abbildung 21 (S. 115) die Dialogstruktur und

Dialoginhalte definiert.[3] Zur Beschreibung von Dialogstruktur und -inhalten der MMS sind mindestens folgende Elemente zu modellieren:

a) Nutzereingaben, haptisch und akustisch.
b) Systemausgaben, optisch und akustisch.
c) Vorbedingungen (Preconditions), um einen bestimmten Systemzustand zu erreichen
d) Dialogverlauf/-struktur in Abhängigkeit der Vorbedingungen und Nutzereingaben

Die Anforderung 3 kann zu einem Interessenskonflikt mit Anforderung 1 führen. Hierin wird die Herausforderung in der Entwicklung einer möglichst einfachen und doch vollständigen Modellierung deutlich. Mit dem Grundsatz: „Modelliere so wenig wie möglich und so viel wie nötig“ soll dieser Herausforderung entgegengetreten werden.

4) *Die formalen Modellierungsregeln müssen vollständig und klar formuliert sein, sowie durchgängig Anwendung finden*.
Diese Anforderung ist Voraussetzung für eine formal konsistente und widerspruchsfreie Modellierung. Zudem müssen die Modellierungsregeln erweiterbar sein, um hinzukommende System-, Test- oder Modellierungsanforderungen einbinden zu können. Die Konsistenz der Modellierung soll durch statische Tests verifiziert werden können.

[3] Die grafische Umsetzung ist nicht im Rahmen entwickelter Bedien- und Anzeigekonzepte spezifiziert und somit nicht relevant für die funktionale Qualitätsabsicherung selbiger. Die funktionale Qualitätsabsicherung des Spracherkenners erfordert kein modellbasiertes Vorgehen (vergl. Kapitel 7.2).

5) *Eine modulare Modellierung ist zu erreichen, um die Wiederverwendbarkeit einzelner System-, bzw. Modellmodule zu ermöglichen*.
   Die Erstellung eines Testmodells ist meist zeitaufwändig und ressourcenintensiv. Eine wirtschaftliche und effiziente Verwendung von Testmodellen gelingt daher oft erst durch Wiederverwendung in aufeinanderfolgenden Entwicklungsprojekten.

Unter Berücksichtigung der formulierten Anforderungen wird verständlich, warum die vorhandenen Spezifikationen nicht oder nur bedingt in einer modellbasierten Vorgehensweise eingesetzt werden können. Die Anforderungen 1, 3 und 5 werden von den vorhandenen Spezifikationen noch weitgehend erfüllt, die Anforderung 2 und 4 jedoch nicht. Die Spezifikation der haptischen MMS nutzt zwar das Standardformat der UML, jedoch wird dieses durch proprietäre Formate erweitert. Dabei sind die formalen Regeln zur Modellierung nicht ausreichend, um eine durchgängige Verwendung ohne informelle Ergänzungen zu gewährleisten. Auch die Spezifikation der akustischen MMS liegt in keinem Standardformat vor. Positiv lässt sich hier jedoch feststellen, dass das vorliegende proprietäre Format, auf Basis von Ablaufdiagrammen, in sich schlüssig ist und keine informellen Ergänzungen notwendig macht. Die Schwierigkeiten liegen hier in Erfüllung der Anforderung 4. Im Bewusstsein der Herausforderung können zwar statische Tests zur Prüfung der formalen Korrektheit durchgeführt werden, jedoch sind diese nicht ausreichend, um die formale Konsistenz der Spezifikation sicherzustellen. Insbesondere in der Benennung und Organisation von Variablen und Parametern finden sich erhebliche Inkonsistenzen.

Mit der Entwicklung eines von der Spezifikation unabhängigen Testmodells besteht nicht nur die Möglichkeit, die formulierten Anforderungen zu erfüllen, sondern gleichzeitig auch die Chance, die vorhandenen Spezifikationen zu verbessern. Alle für das Testmodell notwendigen Informationen sind in den Spezifikationsdokumenten vorhanden. Durch die Erstellung des Testmodells aus der Spezifikation wird diese aus Perspektive des Testers betrachtet. Dabei können formale und inhaltliche Fehler in der Spezifikation bereits zu einem frühen Zeitpunkt im Entwicklungsprozess identifiziert werden. Des Weiteren können im Testmodell die für den Test relevanten Informationen explizit berücksichtigt werden und nichtrelevante Informationen entfallen. Diese Vorteile in der Erstellung eines Testmodells stehen dem hierfür erforderlichen Ressourceneinsatz gegenüber. In Anbetracht dieser zusätzlichen Aufwände ist es naheliegend, das Testmodell aus dem Systemmodell automatisiert und ressourcenschonend transformieren oder generieren zu wollen. Dabei werden jedoch die beschriebenen Vorteile der Testmodellerstellung verloren gehen. Zudem bemerken Roßner et. al., dass die Generierung von Testmodellen aus einem Systemmodell mit Veränderung der formalen Struktur oder Modellierungssprache methodisch komplex und technisch schwierig umzusetzen ist [Roß10].

Ergänzend besteht die Option, die vorhandene Spezifikation weiterzuentwickeln und eine Anbindung an kommerzielle Testfallgeneratoren zu ermöglichen. Auf diese Weise könnten Testfälle aus der Spezifikation abgeleitet werden, ein Testmodell wäre nicht länger notwendig. Diese systemmodellorientierte Vorgehensweise ist allerdings nur dann zielführend, wenn sichergestellt werden kann, dass das Systemmodell nicht zur

Codegenerierung verwendet wird. Andernfalls ist die Unabhängigkeit von SUT und Testsystem nicht gewährleistet und die Testaktivitäten werden unwirksam. Im konkreten Fall der Softwareimplementierung über einen Systemlieferanten findet keine Codegenerierung aus der Spezifikation statt, weswegen die Option einer systemmodellorientierten Vorgehensweise theoretisch bestehen bleibt. Änderungen im Spezifikationsformat werden in dieser Option jedoch nur noch unter Berücksichtigung der formalen Anforderungen des angebunden Testfallgenerators möglich. In Abbildung 34 sind die beiden Ansätze der systemmodellorientierten Vorgehensweisen (in Anlehnung an Kapitel 2.2.5, Abbildung 13 f.), mit den verschiedenen Vor- und Nachteilen gegenübergestellt.

In Anbetracht der aufgezeigten Vor- und Nachteile und unter Berücksichtigung des Umstandes, dass auf Basis der Modellierung verschiedene kommerzielle Testfallgeneratoren zu evaluieren sind, ist die Verwendung des Standardformats der UML erforderlich. Entsprechend der in Kapitel 3.1 beschriebenen Fokussierung auf die akustische MMS wird im folgenden Kapitel ein formales Testmodell einer softwarebasierten, akustischen MMS entwickelt, welches den beschriebenen Anforderungen gerecht wird und die Einführung des MBT ermöglicht. Schließlich wird gezeigt und bewertet, inwieweit die entwickelte Syntax zur Modellierung auch auf die haptische MMS übertragen werden kann.

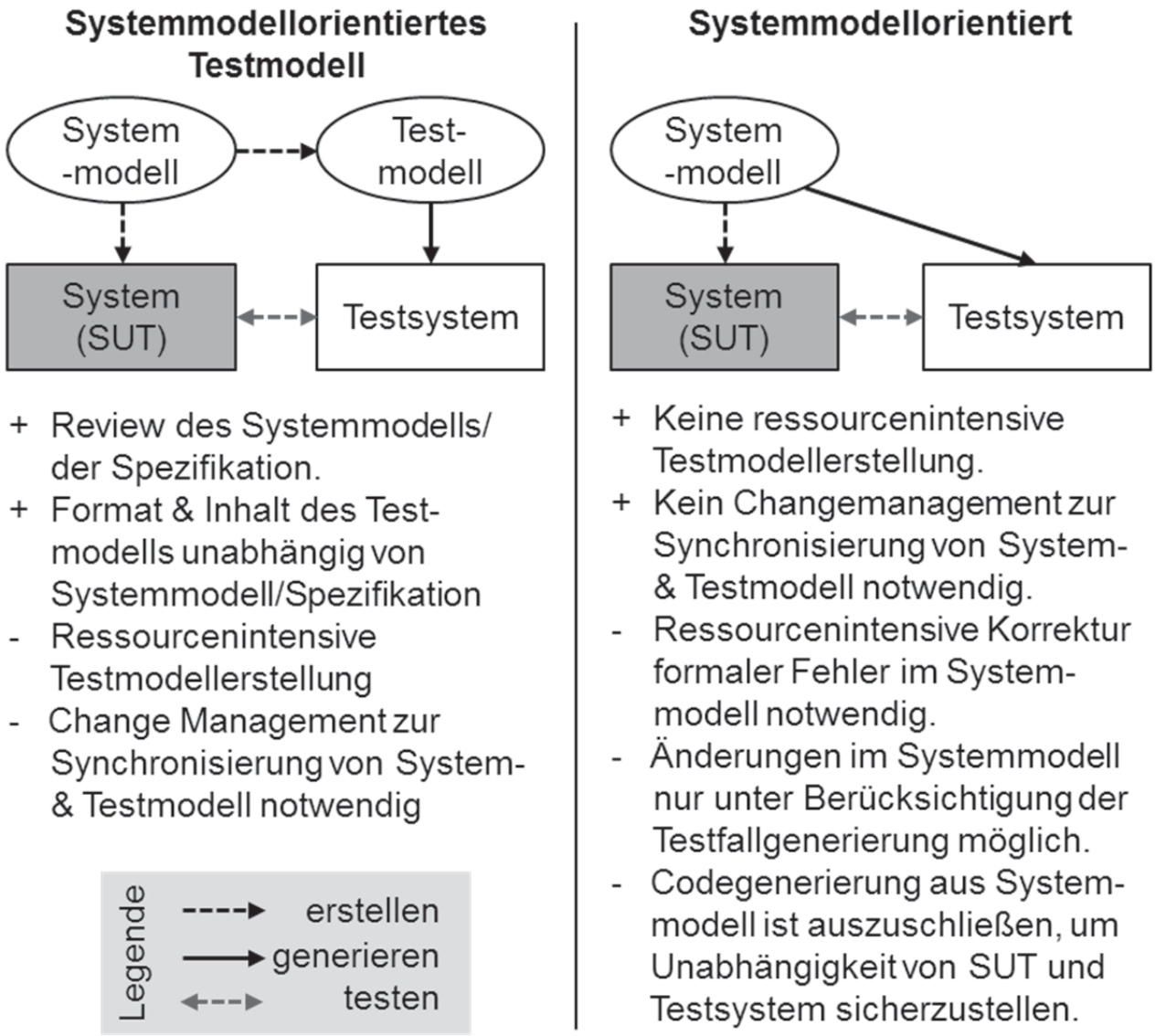


Abbildung 34: Systemmodellorientiertes Testmodell vs. Systemmodellorientiert

5.2.2 Modellierung der Sprachbedienung

Das SDS soll formal in einem Standard UML-Format beschrieben werden, um aus dem erstellten Modell Testfälle generieren zu können. Wie bereits in Kapitel 5.1.2 erläutert, beschreiben Zustandsdiagramme ein Systemverhalten, welches von verschiedenen, auf das System einwirkenden Ereignissen und Bedingungen abhängt. Nachdem die akustische MMS als reaktives System zu

verstehen ist, dessen Verhalten ausschließlich durch externe Stimuli (Nutzeraktionen) beeinflusst wird, ist eine Modellierung in Zustandsdiagrammen sinnvoll [Lüt12]. In diesen Diagrammen sind die verschiedenen Aktivitäten des Dialogs durch zu definierende Notationen der UML darzustellen (vergleiche Kapitel 5.2.1, Anforderung 3). Die folgenden Dialogelemente sind zu berücksichtigen:

- Benutzeraktionen
    - Haptische Aktivierung des Sprachdialogsystems
    - Spracheingaben
- Systemreaktionen:
    - Akustische Rückmeldung durch Sprachausgaben
    - Optische Rückmeldung durch Textausgabe
    - Systemaktivität im Hintergrund, ohne optische oder akustische Rückmeldung des SDS
- Formale Modellelemente zur Beschreibung des Dialogablaufs
    - Startpunkt und Endpunkt
    - Variablen mit Zuweisung von Werten
    - Dialogverzweigungen mit Abfrage von Zuweisungen oder Systemzuständen
    - Vorbedingungen (Preconditions)

Neben der offensichtlich notwendigen Modellierung von Benutzeraktionen und Systemreaktion sind auch formale Modellelemente hinsichtlich ihrer Syntax zu definieren. Variablen mit individuellen Wertebereichen und die Modellierung von Guard Conditions an Verzweigungen ist erforderlich, um das Durchlaufen eines bestimmten Dialogpfades in Abhängigkeit eines zuge-

wiesenen Wertes zu ermöglichen oder zu blockieren. Ebenfalls zu modellieren sind Vorbedingungen, welche ggf. Voraussetzung für das Durch-laufen eines Dialogpfades sind.

Um das SDS eines Infotainmentsystems nach den beschriebenen Anforderungen zu modellieren, sind Zustände und Ereignisse im Sprachdialog zu identifizieren. Befindet sich das System in einem Zustand, ist es in diesem (laut Definition) stabil und verlässt selbigen erst durch externe Stimuli. Diese externen Stimuli sind für ein SDS die erkannten (oder nicht korrekt erkannten) Nutzereingaben. Vor der ersten Nutzereingabe befindet sich das SDS im Wartezustand. Dieser Zustand wird nicht verlassen, bis das System auf eine Nutzeraktivität reagiert. Es lässt sich also feststellen, dass haptische und akustische Nutzeraktionen als Aktivitäten im Zustandsdiagramm und damit als Transitionen zu modellieren sind, wohingegen der Wartezustand des Systems als Zustand zu modellieren ist. Die akustischen und optischen Reaktionen des SDS werden ohne aktives Eingreifen von außen nicht unterbrochen. Folglich sind diese Systemausgaben ebenfalls als Zustand zu interpretieren. Gleiches gilt für Systemaktivitäten ohne optische oder akustische Rückmeldung des SDS. Die Tatsache, dass es sich um eine Systemaktivität handelt (analog der akustischen und optischen System-ausgabe), kann irrtümlich zu der Annahme führen, dass diese auch im Zustandsdiagramm als Aktivität zu modellieren ist. Im Sinne der Zustands-modellierung handelt es sich jedoch um einen Systemzustand, in welchem eine Aktivität ausgeführt wird. Die Abbildungen 35 bis 38 zeigen das in Kapitel 5.1, Abbildung 27 (S. 144) vorgestellte Beispieldiagramm eines SDS in der entwickelten Syntax als Zustandsdiagramm.

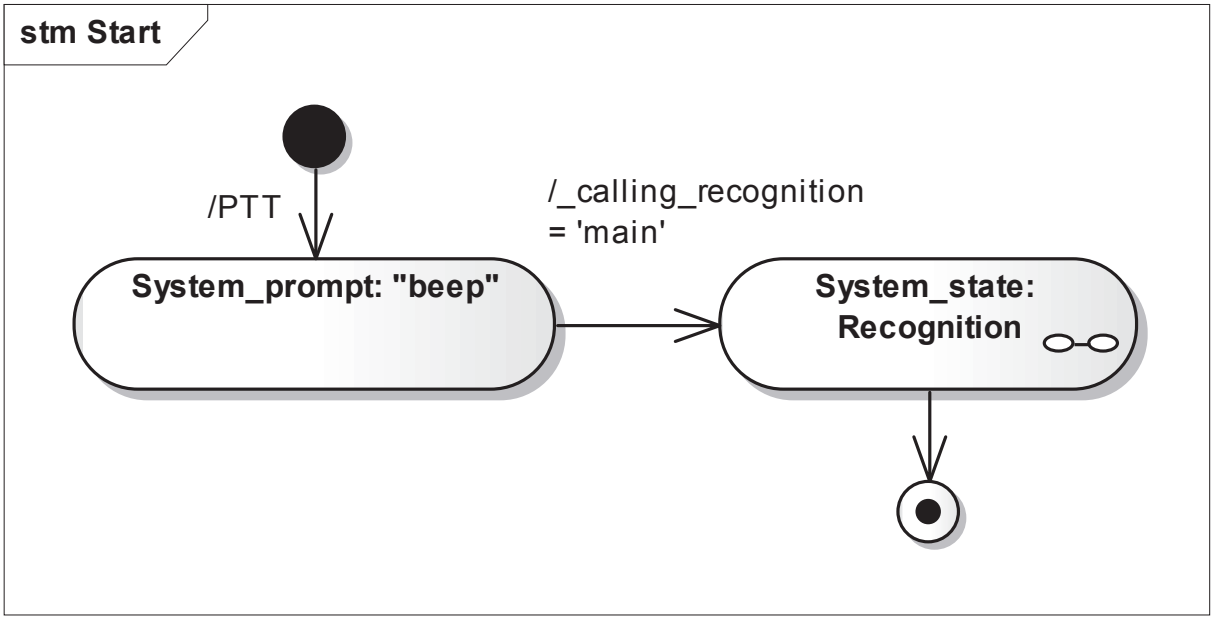


Abbildung 35: Die akustische MMS im Zustandsdiagramm (Teil 1)

Der Dialog beginnt im Zustandsdiagramm „Start“ mit der an einer Transition modellierten Aktivierung des SDS (PTT). Es folgt eine akustische Rückmeldung, bevor das System in den nächsten Zustand übergeht. Vor Aufruf des Zustandsdiagramms „System_state: Recognition“ findet eine Parameterzuweisung der Variablen „_calling_recognition“ statt.

Im aufgerufenen Zustandsdiagramm ist die Spracherkennung modelliert (Abbildung 36). Spracheingaben, welche zu jedem Zeitpunkt im Sprachdialog möglich sind (wie z. B. „Abbruch“), können ohne Guard Condition modelliert werden. Andere Spracheingaben, welche nur in einem bestimmten Kontext des Sprachdialogs gültig sind, werden durch Guard Conditions geschützt.

Nachdem die Variablenzuweisung „_calling_recognition = main“ stattgefunden hat, ist im ersten Dialogschritt entweder das Kommando „Abbruch“ oder „PIN eingeben“ möglich.

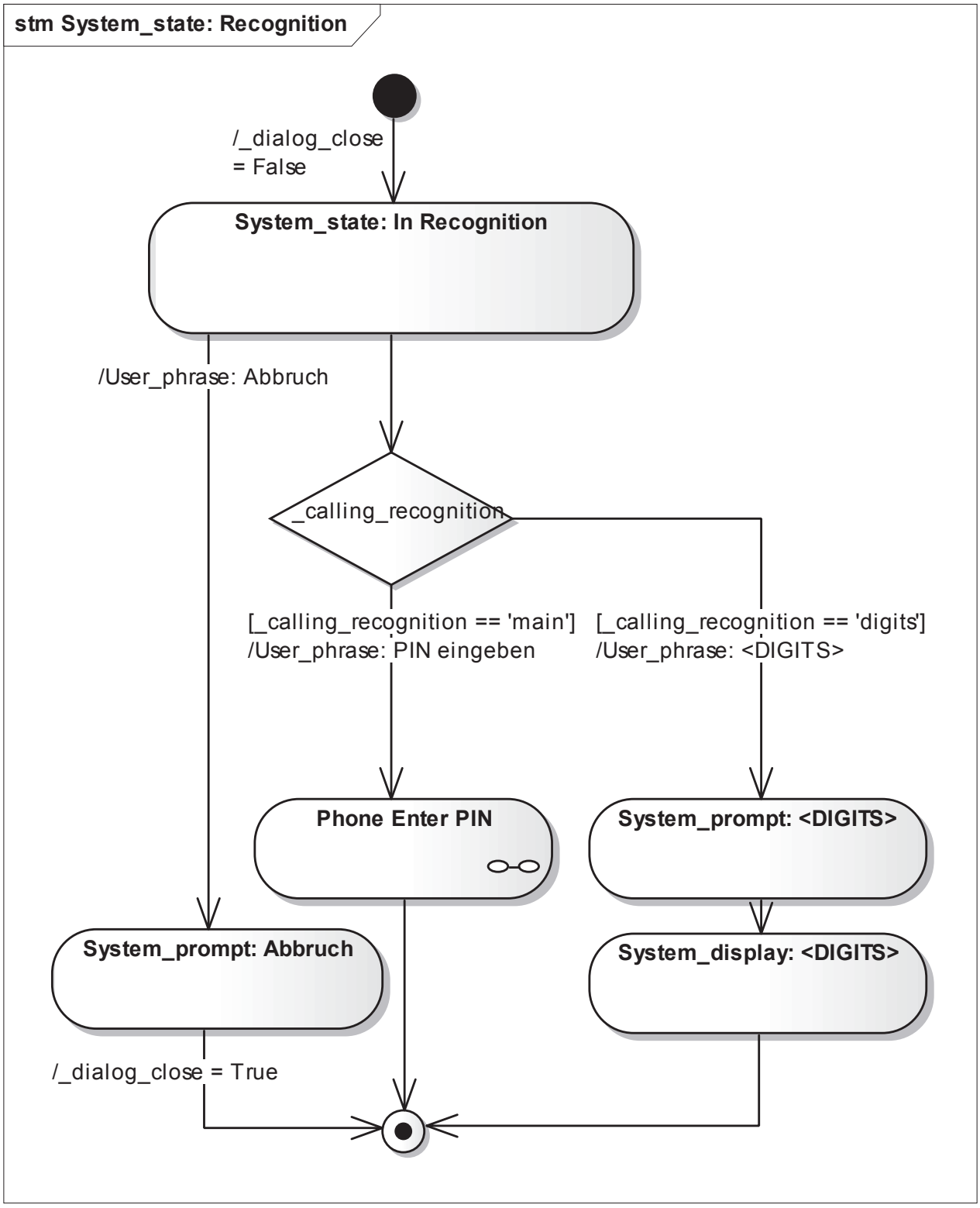


Abbildung 36: Die akustische MMS im Zustandsdiagramm (Teil 2)

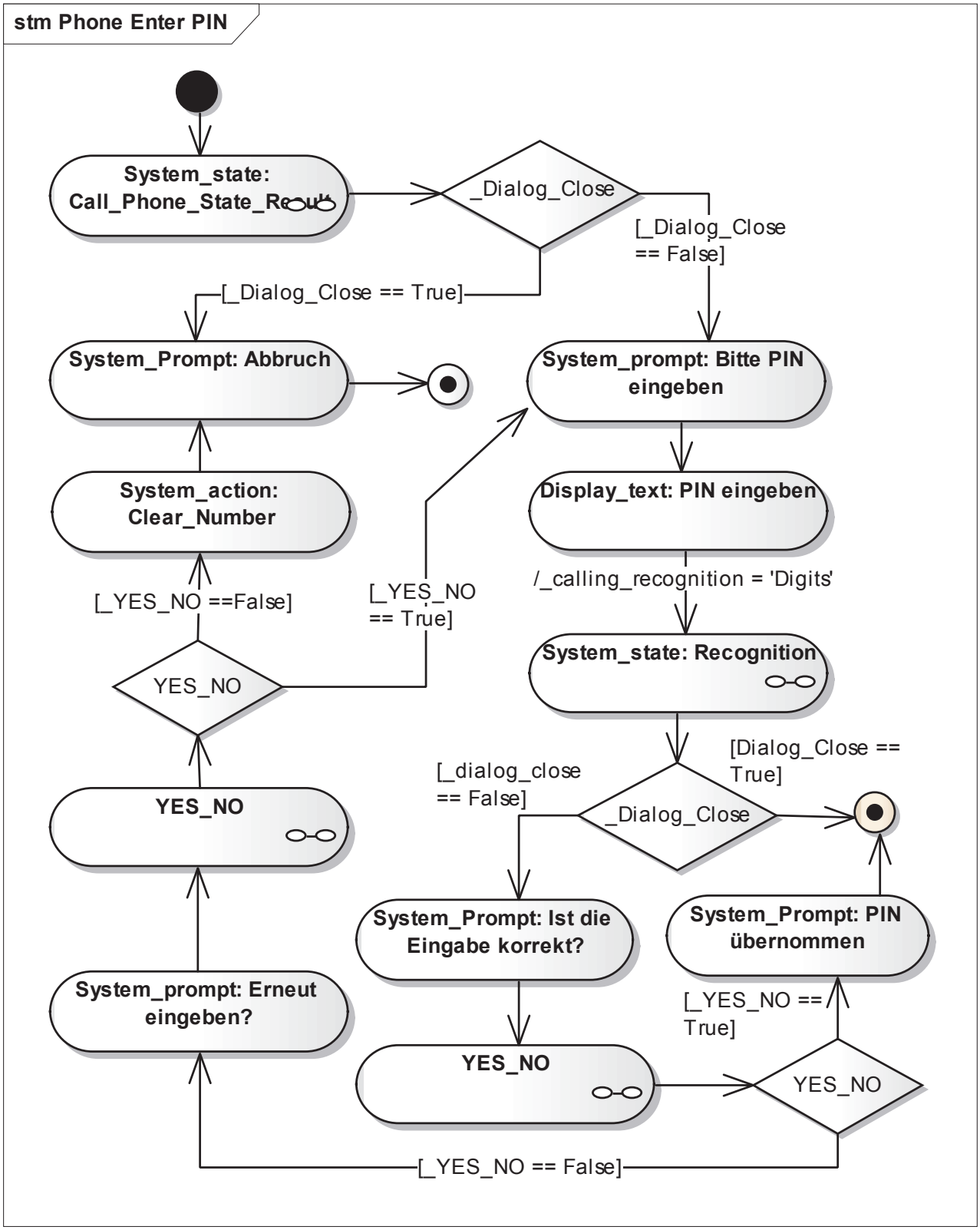


Abbildung 37: Die akustische MMS im Zustandsdiagramm (Teil 3)

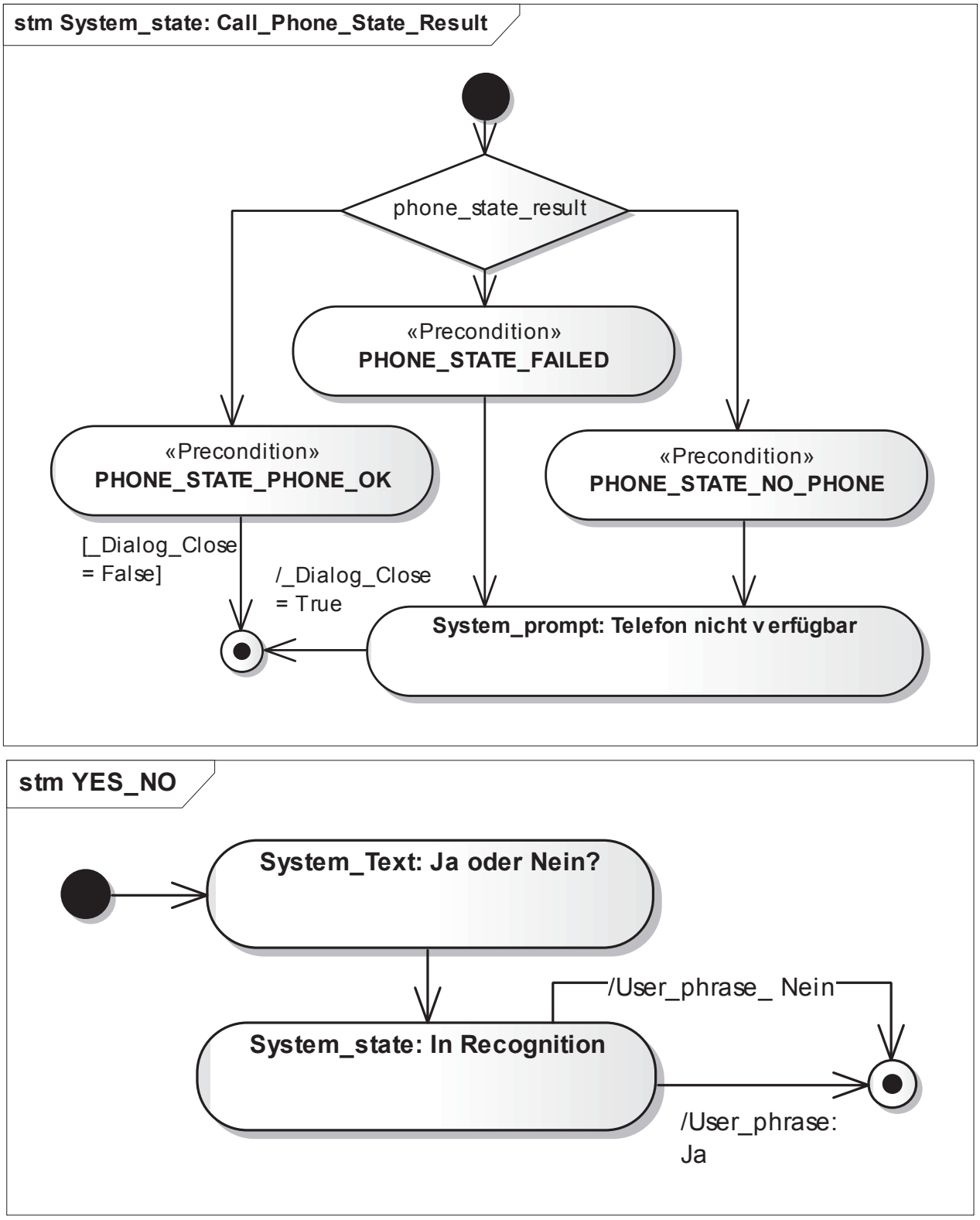


Abbildung 38: Die akustische MMS im Zustandsdiagramm (Teil 4)

Wird der Dialog mit der PIN-Eingabe fortgesetzt, folgt im Zustandsdiagramm „Phone Enter PIN“ zunächst das Diagramm „System_state: Call_Phone_ State_Result“ (Abbildung 37). In diesem werden mögliche Vorbedingungen oder Randbedingungen des Systems sowie deren Auswirkungen auf den Dialog explizit beschrieben. In Abhängigkeit dessen wird der Dialog beendet oder dessen Fortsetzung ist möglich (Abbildung 38, oben). Es folgt die Eingabe des PINs und die Abfrage, ob die erkannte Ziffernfolge korrekt ist. Hierfür wird das Diagramm „YES_NO“ aufgerufen(Abbildung 38, unten). In Abhängigkeit der Antwort wird der Dialog, analog der bisherigen Beschreibung, fortgesetzt.

Der dargestellte Beispieldialog einer akustischen MMS zeigt, dass die Modellierung in Zustandsdiagrammen, unter Berücksichtigung der in Kapitel 5.2.1 definierten Anforderungen, realisierbar ist. Schließlich sollen die entwickelten Modellierungsregeln zusammengefasst werden:

Start- und Endpunkte finden entsprechend der UML Notation für Zustandsdiagramme Verwendung.

Nutzeraktionen werden in den „Transition Properties“ an die Zustandsübergänge modelliert. Während bei haptischen Aktivitäten die verwendete haptische Taste (z. B.: PTT) angeführt wird, sind akustische Eingaben mit dem Präfix „User_phrase:“ zu versehen (z. B. User_says: Abbruch). Das Präfix muss nach Definition durchgängig verwendet werden.

Systemreaktionen jeglicher Art werden als Zustand modelliert und in den „State Properties“ beschrieben. Akustische Systemausgaben erhalten den Präfix „System_prompt:“, optische Ausgaben des SDS den Präfix

„System_text:“. Wartezustände oder Aktivitäten ohne akustische oder optische Rückmeldung des SDS können mit „System_wait“, „System_state:“ oder „System_action:“ beschrieben werden.

Vorbedingungen oder Randbedingungen zur Durchführung eines Dialogs sind als solche zu kennzeichnen. Der Vorbedingung ist ein eindeutiger Wert zuzuweisen (z. B. „phone_state_result = Phone_State_Ok“). Sowohl der Variablen im Entscheider, als auch allen Werten ist das Stereotyp „Precondition“ in den „State bzw. Choice Properties“ zuzuweisen.

Es können verschiedene Typen von **Variablen** definiert und verwendet werden. Hierzu gehören unter anderem Boolesche Variablen als auch Variablen mit Freitext. Die *Zuweisung von Variablen* findet an den Transitionen des Zustandsdiagramms statt. In den „Transition Properties“ werden hierfür „Tagged Values“ angelegt, um die Zuweisung zu formulieren. Globalen Variablen wird gegenüber lokalen Variablen ein Unterstrich vorangestellt. Die Zuweisung findet über ein Gleichheitszeichen statt. Zugewiesene freie Texte sind in Hochzeichen zu stellen, Boolesche Werte nicht (_calling = ‘main‘; _dialog_close = True). Die *Abfrage von Variablen* wird als Guard in den „Transition Properties“ modelliert. In der Rubrik „Constraints“ lässt sich die Variable mit der entsprechenden Bedingung formulieren. Soll die Variable einen bestimmten Wert besitzen, wird dies wie folgt dargestellt: _calling == ‘main‘. Soll die Variable einen bestimmten Wert *nicht* besitzen, wird dies als _calling != ‘main‘ dargestellt.

Um einen repräsentativen Ausschnitt der vorhandenen, ablauforientierten Spezifikation in Zustandsdiagrammen zu modellieren, sind insgesamt 28

zusammenhängende Module einer repräsentativen Applikation als Zustandsdiagramm umgesetzt worden. Dies entspricht etwa 5 % der gesamten Spezifikationsdokumente. Bei Erstellung dieser Testmodelle sind verschiedene Möglichkeiten sowie Herausforderungen in der Modellierung einer akustischen MMS identifiziert worden.

Das Zustandsdiagramm „System_state Recogintion“ sollte, um der Modellierungssyntax durchgängig zu folgen, immer dann aufgerufen werden, wenn die Spracherkennung aktiviert ist und auf ein Nutzerkommando wartet. In Abhängigkeit des aktuellen Kontexts (Dialog/Subdialog) ist eine finite Menge gültiger Kommandos möglich und vom System erkennbar. Zudem sind immer auch einige allgemeingültige Kommandos (globale Kommandos) erlaubt. In Abbildung 37 (S. 169) ist das globale Kommando „Abbruch“ modelliert. In den zu Grunde liegenden Spezifikationen ist die globale Verfügbarkeit dieses Kommandos nicht explizit modelliert, obwohl ebenso vorhanden. Um eine sinnvolle Modellierung globaler Kommandos umzusetzen, ist die Verwendung eines zentralen Moduls sinnvoll. Andernfalls müssten alle globalen Kommandos, bei jeder Benutzereingabe im Dialog, als alternativen modelliert werden, was zu gänzlich unlesbaren und nicht beherrschbaren Diagrammen führt.

Mit der Notwendigkeit eines zentralen Diagramms, besteht auch die Option, alle lokal verfügbaren Kommandos in diesem zu modellieren und durch entsprechende Guard Conditions zu „schützen“. Wird in einem Subdialog das Diagramm „System_state Recognition“ aufgerufen, muss zuvor der Variablen „_calling_recognition“ ein entsprechender Wert zugewiesen werden, um im Zustandsdiagramm „System_state Recognition“ die entsprechenden, lokal

gültigen Kommandos durchlaufen zu können. Vorteil dieser Modellierungssyntax ist die zentrale Organisation aller Systemeingaben (globale und lokale Kommandos) in einem Diagramm. Nachteilig ist festzustellen, dass die konsistente Modellierung der Variablen „_calling_recognition“ und deren Werte bei der großen Anzahl an Kommandos komplex und fehleranfällig würde. Schließlich besteht auch die Möglichkeit, alle lokalen Kommandos im entsprechenden Zustandsdiagramm des Subdialogs zu modellieren. So kann, wie in Abbildung 38 geschehen, die Erkennung der Ja-/Nein-Abfrage im entsprechenden Diagramm modelliert werden, ohne dass ein Aufruf des „System_state Recognition“ und die vorhergehende Variablenzuweisung notwendig ist.

Es lässt sich feststellen, dass ein zentrales Diagramm für globale und/oder häufig verwendete Kommandos notwendig ist, gleichzeitig aber auch die dezentrale Modellierung lokaler Kommandos sinnvoll erscheint. Innerhalb dieser Leitplanken gibt es für die konkrete Ausführung der Modellierung keine richtige oder falsche Lösung, sehr wohl jedoch günstige sowie unnötig komplizierte Umsetzungen. Eine günstige Lösung könnte sein, die globalen Kommandos in einem zentralen Diagramm zu modellieren und bei Bedarf aufzurufen, in selbigem jedoch einen zusätzlichen Pfad zu ergänzen, welcher direkt in das aufrufende Zustandsdiagramm zurückführt, um die dort modellierten, lokalen Kommandos zu durchlaufen (siehe Abbildung 39).

Ähnlich herausfordernd ist die Modellierung einer „Timeout“ Situation. Wird vom Spracherkenner innerhalb eines definierten Zeitraums nach Aktivierung keine Systemeingabe erkannt, wird dem Nutzer automatisch eine akustische und optische Hilfe zu möglichen Systemeingaben ausgegeben.

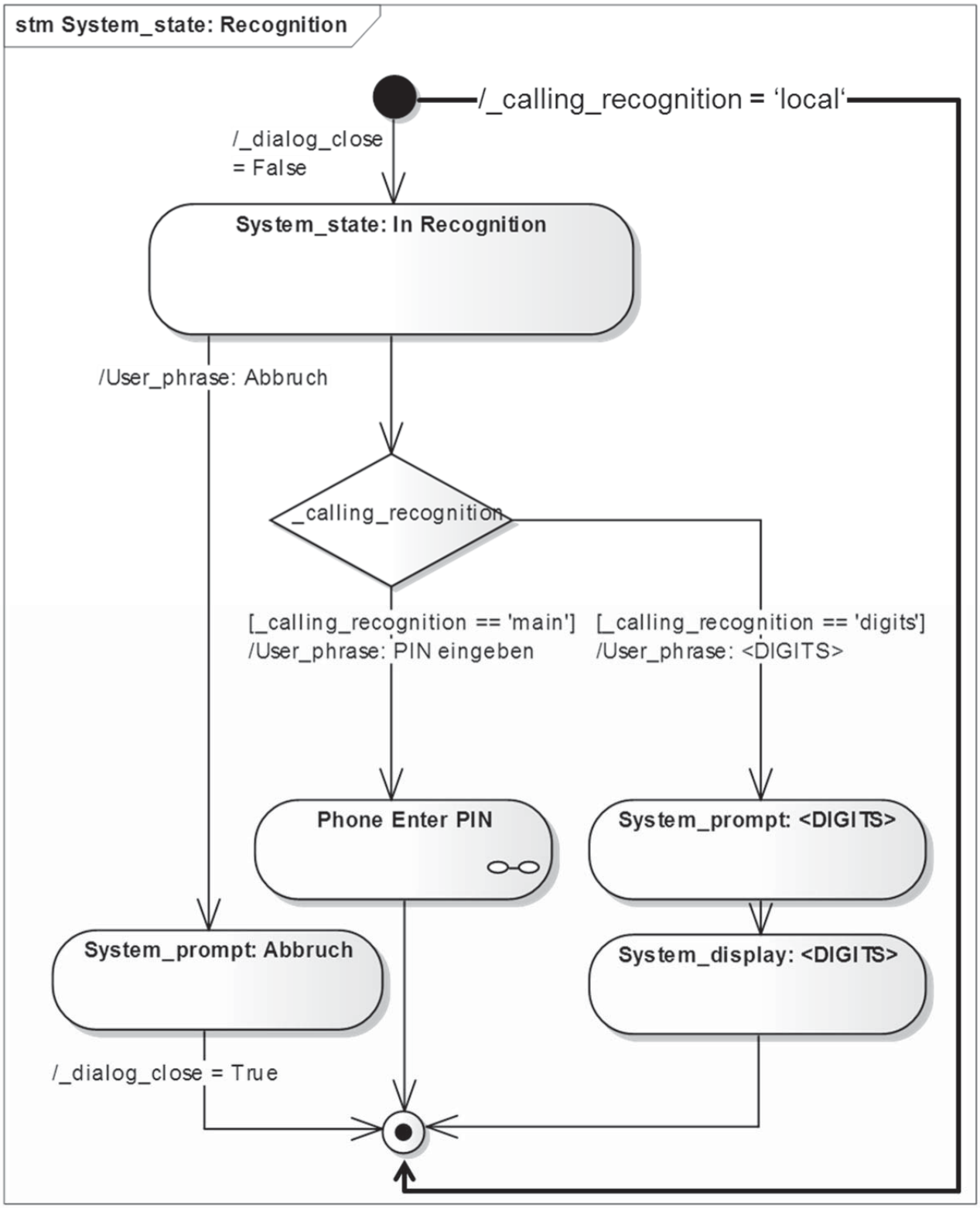


Abbildung 39: Modellierung globaler und lokaler Kommandos

Die Timeout Situation ist bei jeder Systemeingabe möglich und daher sinnvollerweise in einem zentralen Diagramm zu modellieren. Da im Falle eines Timeouts in Abhängigkeit des aktuellen Dialogs aber kontextabhängige Hilfetexte ausgegeben werden, ist eher eine dezentrale Modellierung im jeweiligen Subdialog zielführend.

Die beschriebenen Herausforderungen in der Modellierung bestehen unabhängig von der entwickelten Syntax zur Modellierung in Zustandsdiagrammen. Im bestehenden Systemmodell (der Spezifikation) wird die formale Konsistenz gegebenenfalls unterbrochen und textuell beschrieben, welche Systemeingaben als global betrachtet werden. Die Vorteile in der Erstellung eines Testmodells werden hier ersichtlich. Unabhängig von der Spezifikation kann der Inhalt des Modells schnell und flexibel an die Anforderungen der Testaktivitäten angepasst werden. Testrelevante Informationen und Formalismen können ergänzt werden, Inhalte, welche die Komplexität und Effizienz des Modells sprengen, können entfallen.

Neben diesen Erkenntnissen konnten bei Erstellung der Testmodelle verschiedene Fehler in den Spezifikationsdokumenten identifiziert werden. Damit zeigt sich, dass die Qualitätsverbesserung vorhandener Systemmodelle durch die Erstellung von Testmodellen nicht nur theoretisch möglich ist, sondern real stattfindet. In den analysierten 28 Ablaufdiagrammen des Systemmodells wurden 11 formale Fehler folgender Ausprägungen festgestellt:

- Inkonsistenzen in der Benennung von Variablen. Verschiedene Variablen haben inhaltlich die gleiche Funktion/Bedeutung.

- Inkonsistenzen in der Verwendung von Bedingungen an sich verzweigenden Pfaden. Die Alternativen werden in Klammern (True), ohne Klammern „true“, in Großbuchstaben, Kleinbuchstaben etc., modelliert. Im Vergleich von Ziffern, fehlen teilweise die Gleichheitszeichen.
- Entscheider/Verzweigungen werden falsch verwendet und haben nur einen Ausgang.
- Direkt vor Abfrage einer Variablen an einer Dialogverzweigung wird der Variablen ein Wert zugewiesen. In der Verzweigung wird folglich immer nur ein Pfad (jener mit dem zuvor zugewiesenen Wert) gültig sein.

Mit der hier dargestellten Entwicklung kann gezeigt werden, dass die Modellierung des Dialogsystems im Kraftfahrzeug als formales Zustandsdiagramm möglich ist und somit auch die Umsetzung des modellbasierten Testprozesses realisiert werden kann. Die These A aus Kapitel 3.2 ist bewiesen.

Bevor in Kapitel 6 die Beweisführung zu These B angetreten wird, soll in Kapitel 5.2.3 aufgezeigt werden, wie die entwickelte Methodik zur Modellierung der akustischen MMS auf die haptische MMS übertragen werden kann.

5.2.3 Modellierung der haptischen MMS

Die in Kapitel 5.2.1 beschriebenen Anforderungen an das Modell einer akustischen MMS gelten nicht nur für diese, sondern auch für die haptische Schnittstelle. Einzig die in Anforderung (3) beschriebenen Modellierungsinhalte sind um die akustischen Komponenten zu reduzieren. Für die Modellierung der haptische MMS soll die in Kapitel 5.2.2 definierte Notation Verwendung finden. Ergänzend sind folgende Dialogelemente der Haptik zu berücksichtigen:

- Haptische Benutzeraktionen
    - Bedienung der in Kapitel 5.1 beschriebenen Multifunktions-Lenkradtasten (MFL). Es ergeben sich sechs mögliche Benutzer-eingaben (OK, BACK, UP, DOWN, RIGHT, LEFT). Eine siebte wird durch das Halten der OK-Taste für mehrere Sekunden realisiert (Longpress_OK).
    - Bedienung des zentralen Dreh-Drück-Schiebe-Stellers (ZBE). Dessen Verwendung ermöglicht acht Systemeingaben (PUSH, UP, DOWN, RIGHT, LEFT, TURN_RIGHT, TURN_LEFT, BACK).
    - Bedienung der Hardkeys zur Ansteuerung der HU. Insgesamt sind 24 verschiedene Eingabemöglichkeiten vorhanden (NAVI, RADIO, TEL, DISC, SYS, MUTE, EJECT, CLEAR, 0-9, STAR, HASH, PHONE_START, PHONE_CLOSE, NEXT, PREV).

Im Vergleich zur Modellierung der akustischen MMS wird hier deutlich, dass viel mehr verschiedene, haptische Aktionen in der Modellierung zu berück-sichtigen sind.

Analog der Sprachbedienung lassen sich auch für die haptische Bedienung Systemzustände als Aktivitäten interpretieren, welche als externe Stimuli zu einer Änderung des Systemzustands führen. Hierzu gehören alle haptischen Nutzeraktionen, welche folglich als Transition zu modellieren sind. Die Systemzustände sind über die aktive, optische Systemanzeige darstellbar. So kann das Infotainmentsystem aktiv sein und der Cursor auf dem Navigations-Icon der Menüleiste liegen. Dieser Zustand wird sich ohne Stimuli von außen nicht ändern. Gleiches gilt für das Menü im KI. Erst die Verwendung eines haptischen Stellteils führt zum nächsten Systemzustand. Die Abbildungen 40

bis 42 zeigen das in Kapitel 5.1.1 vorgestellte Beispieldiagramm einer Menüführung im KI (Abbildung 29, S. 148).

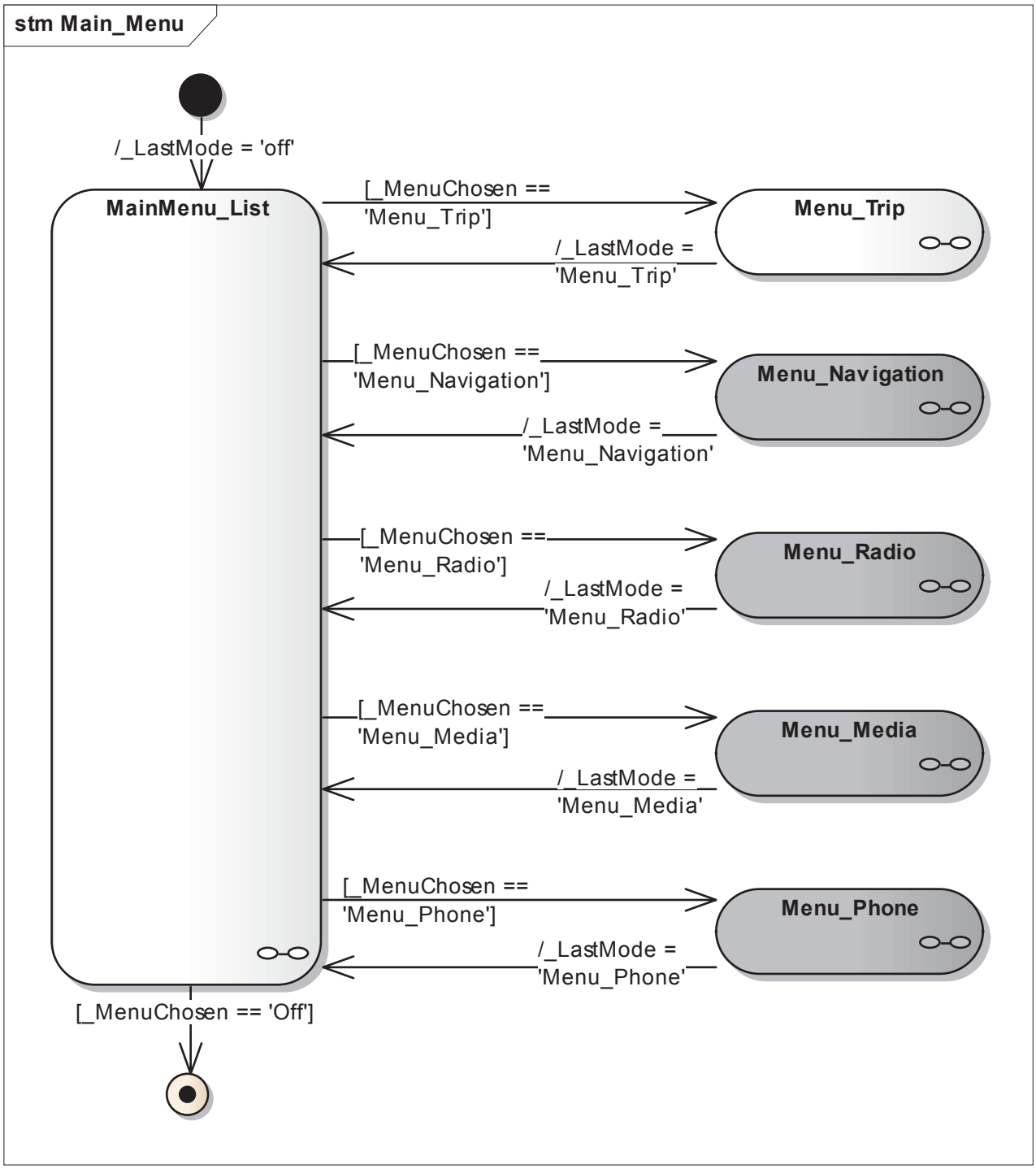


Abbildung 40: Die haptische MMS des KI im Zustandsdiagramm (Teil 1)

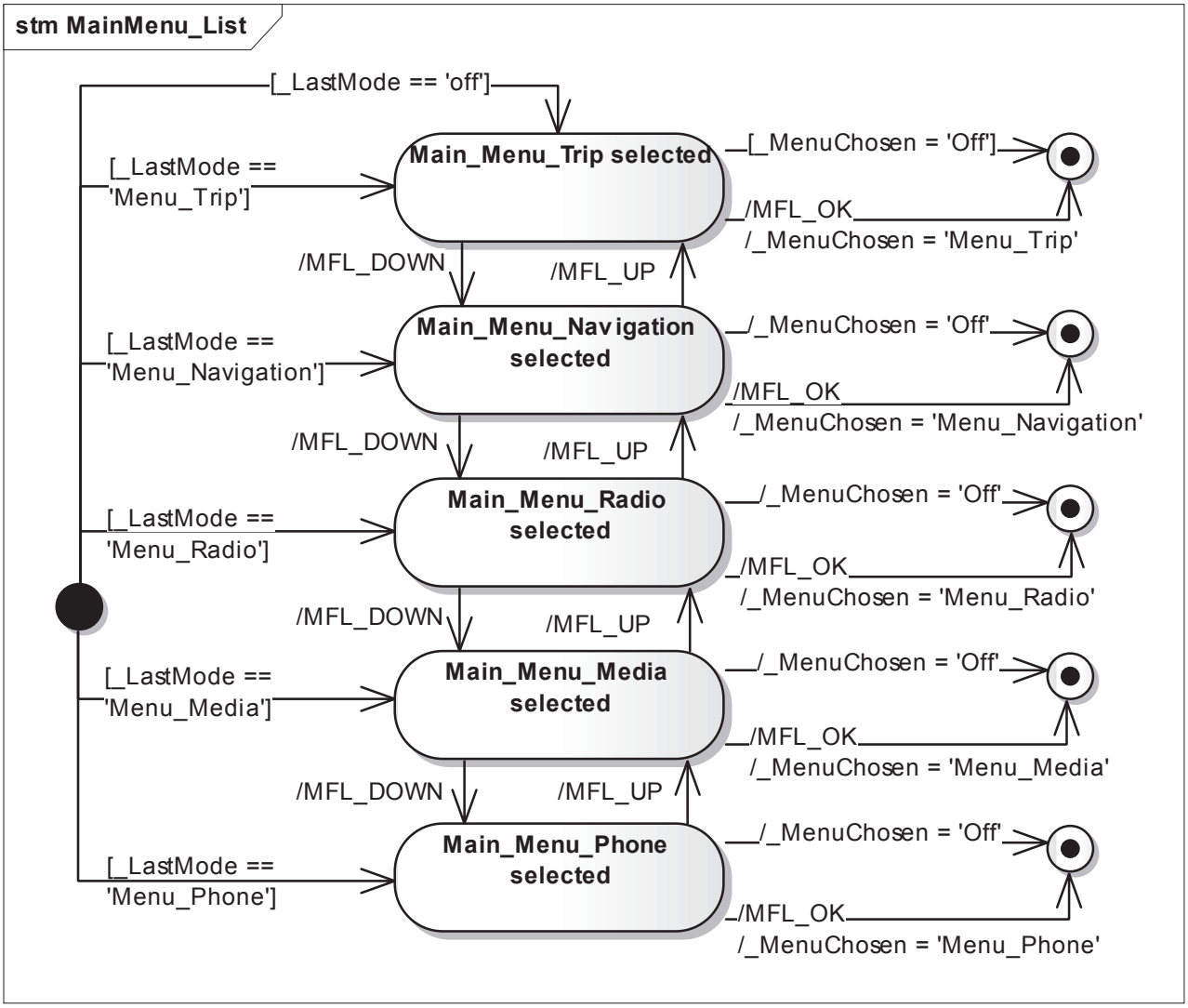


Abbildung 41: Die haptische MMS des KI im Zustandsdiagramm (Teil 2)

Es wird die in Kapitel 5.2.2 entwickelte Syntax verwendet. Ausgegraute Zustandsdiagramme sind aus Gründen der Übersichtlichkeit nicht dargestellt.

Der Dialog beginnt mit dem Einsprung in die Liste des Hauptmenüs. In Abhängigkeit des zuletzt aufgerufenen Menüeintrags ist der entsprechende Listeneintrag durch den Cursor ausgewählt. Diese Eigenschaft wird durch die variable „_LastMode“ dargestellt.

Innerhalb der Menüliste (Abbildung 41) wird durch die Nutzeraktivitäten „MFL_DOWN, -_UP, -OK“ zwischen den verschiedenen Listeneinträgen navigiert und diese ausgewählt. In Abhängigkeit des gewählten Listeneintrags erfolgt der Aufruf des entsprechenden Menüs (Abbildung 40). Im Beispiel (ab Abbildung 42) ist das Menü des Reiserechners („Menu_Trip“) modelliert.

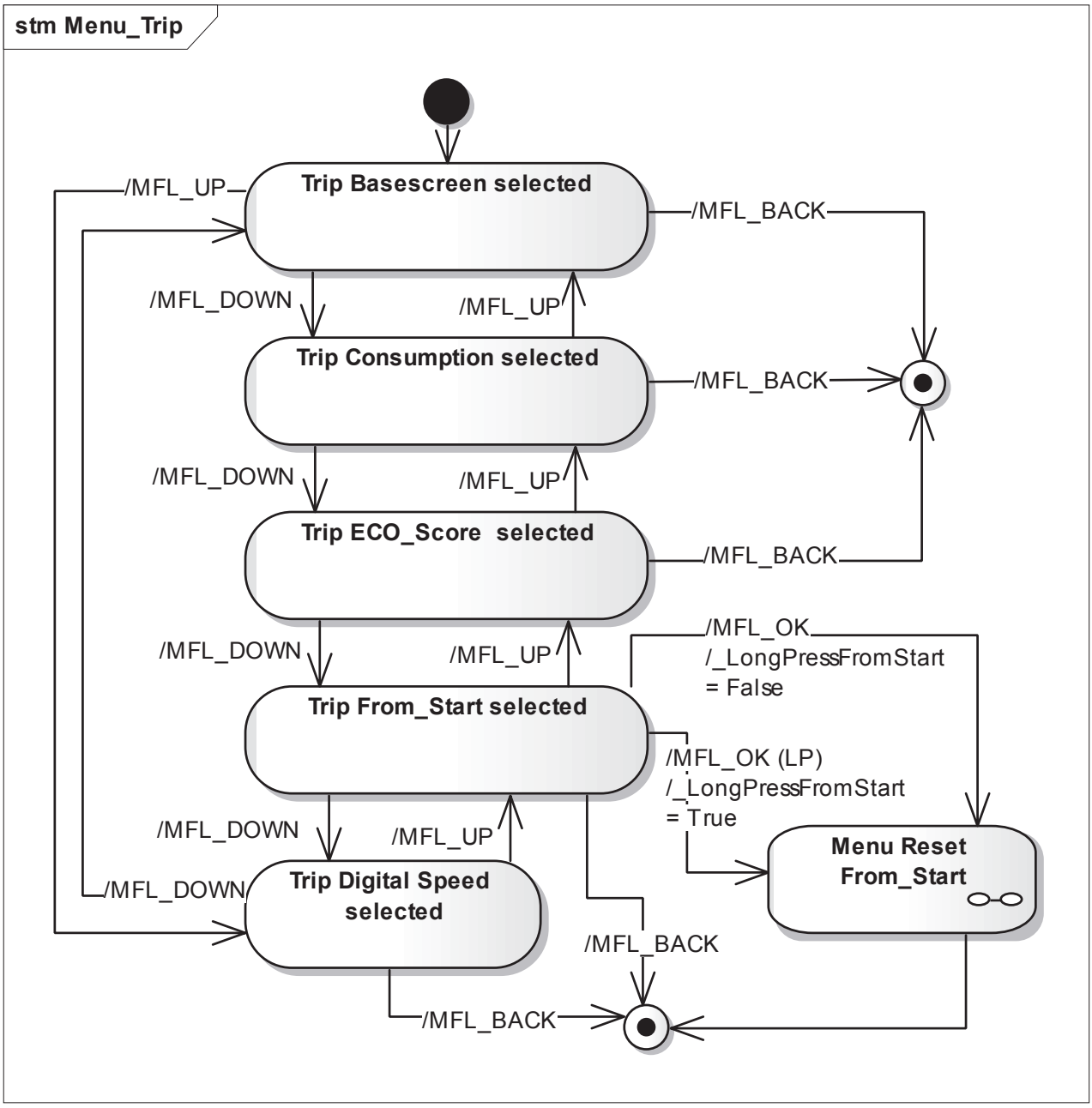


Abbildung 42: Die haptische MMS des KI im Zustandsdiagramm (Teil 3)

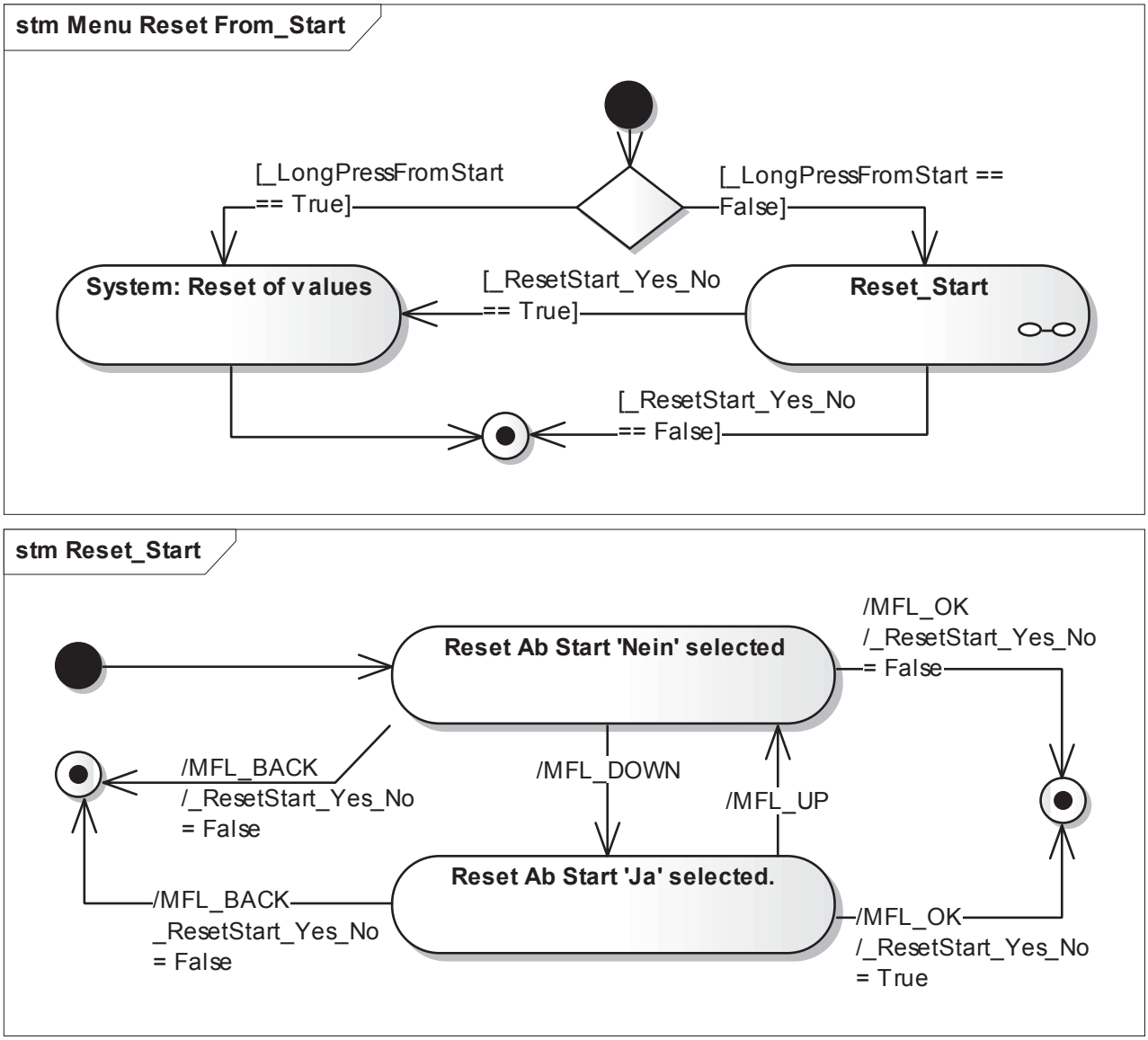


Abbildung 43: Die haptische MMS des KI im Zustandsdiagramm (Teil 3)

Die MMS des KI lässt sich aufgrund der klaren (Baum-)Struktur im aufgezeigten Format gut modellieren. Alle optischen Systemzustände können durch Beschreibung des durch den Cursor markierten Listeneintrags eindeutig beschrieben werden. Herausfordernd ist die Modellierung der in jedem Systemzustand parallel möglichen Benutzeraktionen. Ähnlich verhält es sich mit der Modellierung, wenn Listeneinträge in Abhängigkeit zuvor ausgeführter

Aktivitäten selektiert sind („Last Mode“-Abhängigkeiten). Im Vergleich zu den vorhandenen Systemmodellen ist das Testmodell formal durchgängig, jedoch weniger gut lesbar. Auch Informationen zur grafischen Darstellung gehen verloren.

Die formale Modellierung der MMS des KI hat ihre Grenzen. In jedem Systemzustand besteht die theoretische Möglichkeit, dass Warn- und Informationsmeldungen den Dialog zwischen Mensch und Maschine unterbrechen. Sollte dieses im definierten Format modelliert werden, müsste ein entsprechender Pfad an jedem Systemzustand ergänzt werden. Theoretisch ist dies zwar möglich, aufgrund des Modellierungsaufwands und der zunehmenden Komplexität praktisch jedoch nicht sinnvoll.

Im Rahmen der Analyse wurden ca. 15 % der vorhandenen Systemmodelle des KIs in Testmodelle überführt. Analog zur Bearbeitung der akustischen MMS sind bei Erstellung der Testmodelle Fehler in den Systemmodellen identifiziert worden.

- Unvollständige Modellierung von möglichen Nutzeraktivitäten
- Inkonsistenzen zwischen formaler und textueller Listenbeschreibung
- Formaler Modellierungsfehler: Zustand hat keinen Ausgang
- Logikfehler bei der Verwendung von Variablen und Guard Conditions

Zusammenfassend lässt sich feststellen, dass die Modellierung der grundlegenden Menüstrukturen des KI im Zustandsdiagramm und der beschriebenen Syntax gut möglich ist. Die Testmodelle konnten ohne großen Aufwand aus den bereits vorhandenen, ebenfalls weitgehend formalen Systemmodellen abgeleitet werden. Es ist jedoch zu beachten, dass eine

vollständig formale Beschreibung aller Systemeigenschaften und Funktionen nicht sinnvoll erscheint.

Die Abbildungen 44 bis 46 zeigen (in erweitertem Umfang) das in Kapitel 5.1.1, vorgestellte Beispieldiagramm einer Menüführung in der HU (Abbildung 30, S. 149). Es wird die in Kapitel 5.2.2 entwickelte Syntax verwendet. Ausgegraute Zustandsdiagramme sind aus Gründen der Übersichtlichkeit nicht dargestellt.

Die Modellierung der Start- (und End-) Knoten führt bereits zu einer ersten Herausforderungen. Das Infotainmentsystem kann in jedem beliebigen Systemzustand deaktiviert werden. Folglich müsste zur Abbildung der Realität an jedem Systemzustand eine Transition modelliert werden (Aktivität: „System_switch_off“). Zudem müsste einer Variablen „_Last_Mode“ ein entsprechender Wert zugewiesen werden, da bei Aktivierung des Systems die Applikation des zuletzt aktiven Systemzustandes reaktiviert wird. Aus Gründen der Übersichtlichkeit ist im Beispiel darauf verzichtet und die vereinfachte Annahme getroffen, dass bei einem Systemstart die Audio-applikation aktiv und entsprechend das Audio-Icon in der Applikationszeile angewählt ist. Beendet wird der Dialog nach Anwahl des „Phone-Icon“ in der Applikationszeile. Ausgehend von diesen vereinfachten Einstiegsbedingungen ist im Beispiel das Drop-Down-Menü (Audio App Group) der Audioapplikation modelliert, welches, wie in Abbildung 46 modelliert, durch drücken des ZBE geöffnet wird. In Abhängigkeit der hier ausgewählten Applikation wird in der zentralen Anzeige der HU die „Radio App Area“ oder „Player App Area“ angezeigt, welche durch die Nutzeraktivität „ZBE down“ erreicht wird.

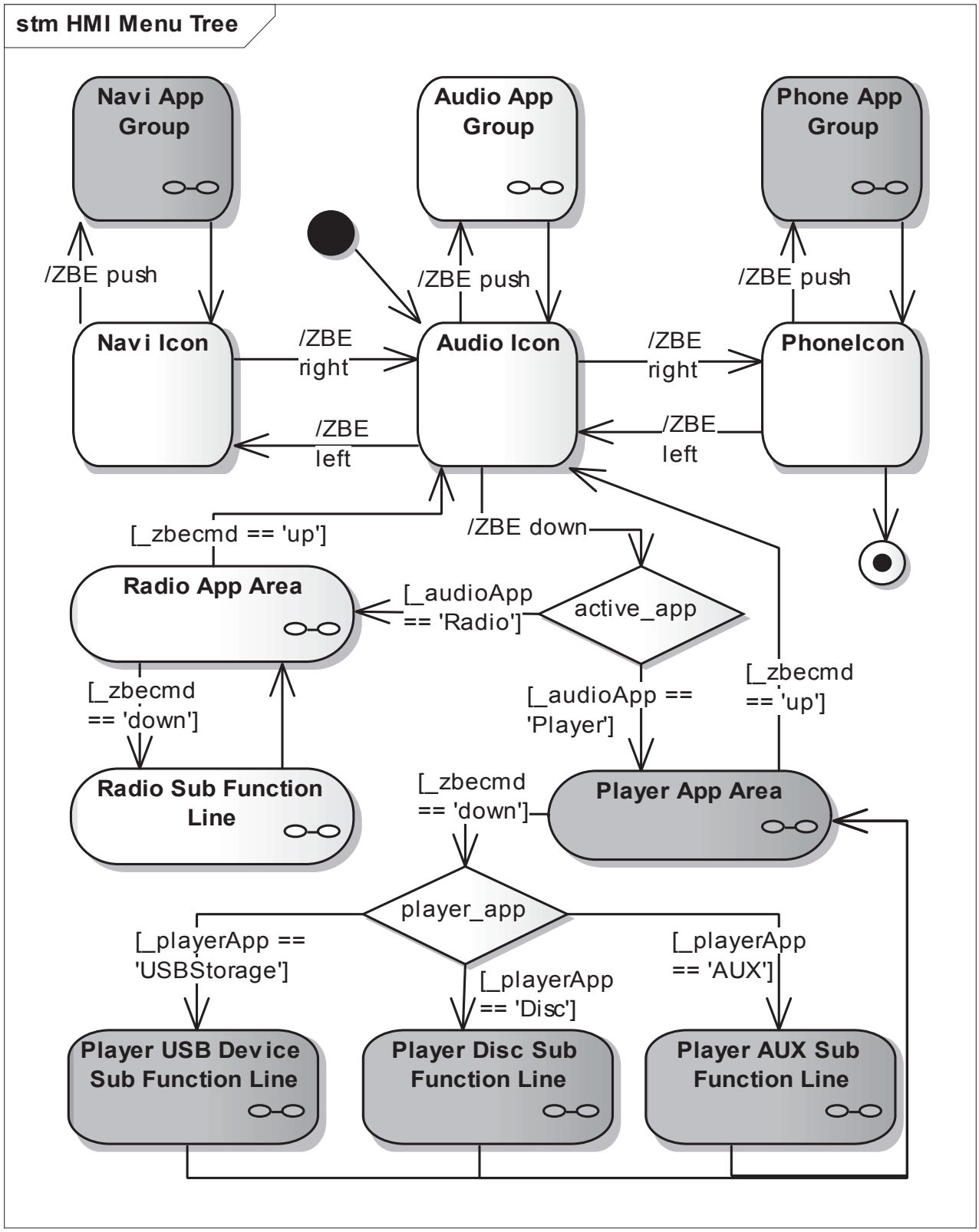


Abbildung 44: Die haptische MMS der HU im Zustandsdiagramm (Teil 1)

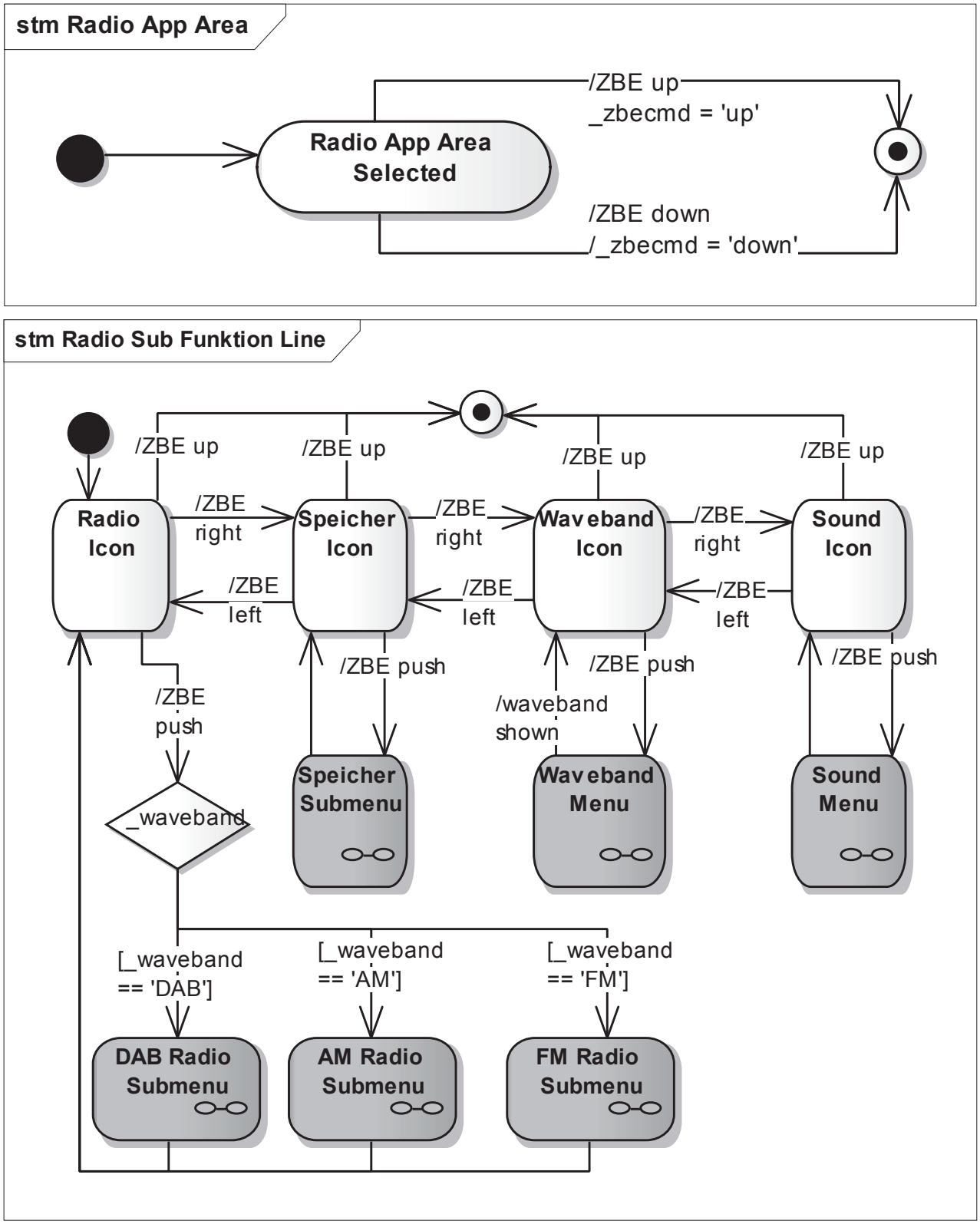


Abbildung 45: Die haptische MMS der HU im Zustandsdiagramm (Teil 2)

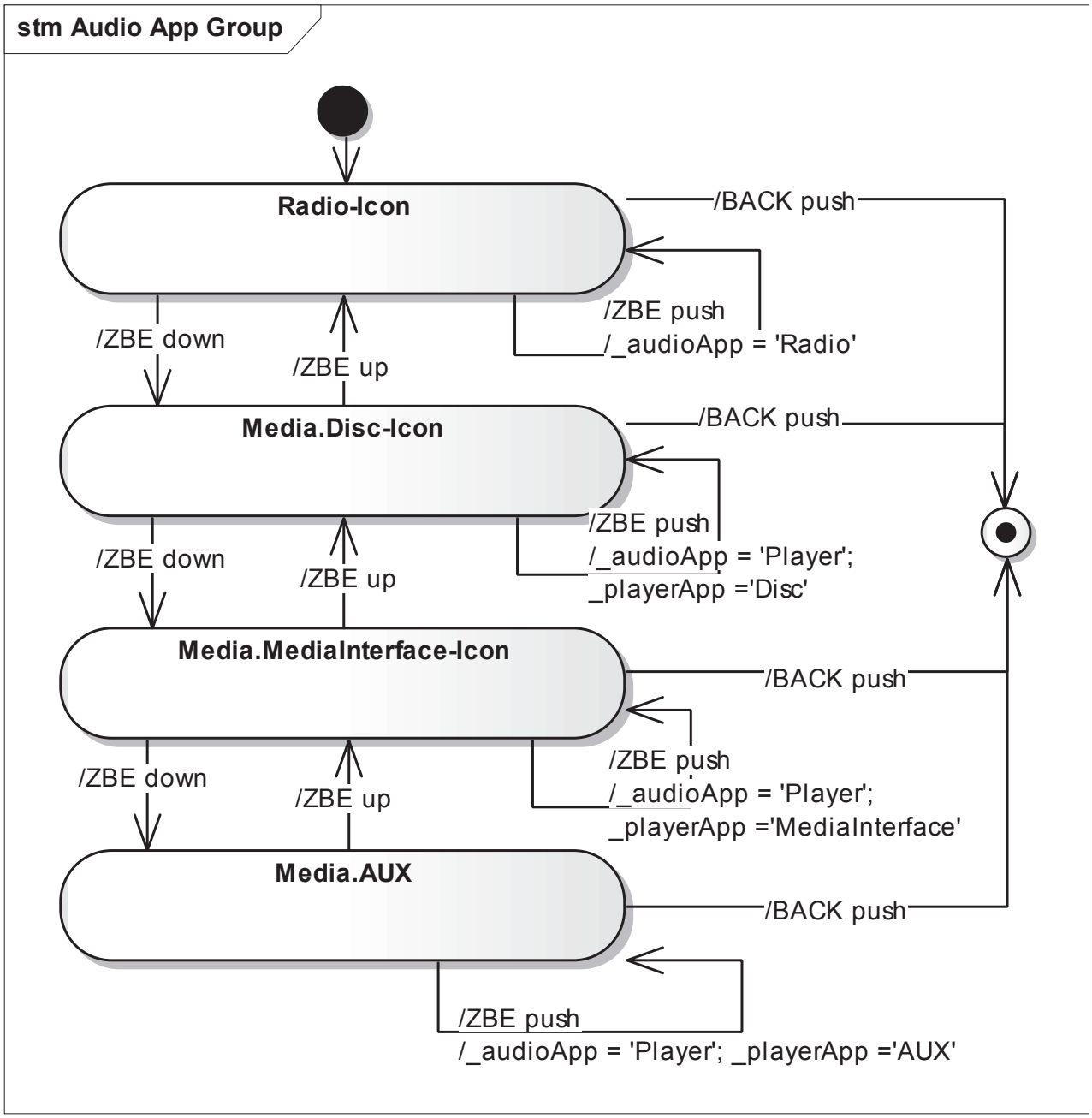


Abbildung 46: Die haptische MMS der HU im Zustandsdiagramm (Teil 3)

Durch Wiederholung derselben Nutzeraktivität wird die Funktionsleiste („Radio Sub Function Line“) erreicht, welche in einem weiteren Zustands-diagramm modelliert ist (Abbildung 45, unten).

Die Modellierung der haptischen MMS der HU ist eine Herausforderung. Erhebliche Schwierigkeiten ergeben sich durch die enorme Anzahl möglicher Nutzeraktionen. In den Beispieldiagrammen sind nur die notwendigen Nutzeraktionen dargestellt, jedoch nicht alle tatsächlich möglichen.

Um die Komplexität der Modellierung im Rahmen zu halten, wurden die Drehbewegungen des ZBE (TURN_RIGHT, TURN_LEFT), sowie alle 24 Hardkeys zur Bedienung der HU nicht berücksichtigt. Insbesondere die Hardkeys zum direkten Einsprung in verschiedene Applikationen würden den Umfang der Modellierung im vorgestellten Format sprengen, da deren Aktivierung in jedem Systemzustand möglich ist und direkt in die jeweilige Applikation führt.

Eine weitere Herausforderung ist die Modellierung von Menüeinträgen oder Listenverhalten, welche sich in Abhängigkeit des Systemzustandes verändern. Beispielhaft sei angeführt, dass die Funktionsleiste der Radioapplikation in Abhängigkeit des gewählten Wellenbandes unterschiedliche Funktionen und Inhalte besitzt. Der Inhalt des Icons zur Auswahl des Wellenbandes ist dynamisch und abhängig vom aktuell aktiven Wellenband. Zudem können über ein entsprechendes Icon in der Funktionsleiste Hintergrundinformationen zum aktuellen Sender abgerufen werden. Die Informationen sind aus technischen Gründen aber nicht in allen Wellenbändern verfügbar. Entsprechend ist das Icon in Abhängigkeit des gewählten Wellenbandes ausgegraut und wird bei haptischer Anwahl übersprungen. Solch vielschichtiges Dialogverhalten auf mehreren Ebenen, mit zahlreichen Abhängigkeiten und Eingabemöglichkeiten, führt zu einer scheinbar beliebig komplexen Modellierung. Es lässt sich zwar feststellen, dass die haptische MMS der HU grundsätzlich modelliert werden kann, dabei jedoch eine Fokussierung auf besonders

testrelevante Inhalte sinnvoll bzw. notwendig ist, um die Effizienz der modellbasierten Vorgehensweise zu gewährleisten. Ergänzend ist festzustellen, dass sich die Lesbarkeit der erstellten Testmodelle mit zunehmender Komplexität deutlich verschlechtert.

5.3 Zusammenfassung

Nachdem in Kapitel 4 die prozessualen Rahmenbedingungen für die modellbasierten Vorgehensweise zum Test der softwarebasierten MMS im Kfz analysiert und ausgearbeitet wurden, wurde in Kapitel 5 der Beweis erbracht, dass die akustische MMS des Infotainmentsystems in einem Standardformat formal modelliert werden kann (vergl. Kapitel 3.2, These A).

Um die Beweisführung anzutreten, wurden zunächst vorhandene Systemmodelle der MMS vorgestellt (Kapitel 5.1.1) und verschiedene Möglichkeiten zur formalen Modellierung aufgezeigt, um schließlich eine formale Syntax zur Modellierung der MMS zu entwickeln. Diese soll als Testbasis die Grundlage für eine folgende Testfallgenerierung und Testdurchführung schaffen. Die Anforderungen an ein solches Testmodell wurden in Kapitel 5.2.1 definiert.

Auf Basis der formulierten Anforderungen wurde in Kapitel 5.2.2 eine Modellierungssyntax hergeleitet. Die Syntax wurde praktisch angewendet, indem repräsentative Teile der akustischen MMS modelliert wurden. Hierbei wurden sowohl Herausforderungen in der Modellierung beschrieben, als auch die Potenziale der Modellierung identifiziert. So ließ sich zeigen, dass durch die Erstellung eines Testmodells nicht nur in der Theorie die Qualität des zu Grunde liegenden Systemmodells verbessert werden kann.

Es konnte bewiesen werden, dass die Modellierung der akustischen MMS des Kfz als formal durchgängiges Zustandsdiagramm möglich ist und somit auch die Umsetzung des MBT-Prozesses grundsätzlich realisiert werden kann.

In Kapitel 5.2.3 wurde die für die akustische MMS entwickelte Modellierungssyntax auf die haptische MMS übertragen, um zu zeigen, inwiefern die Vorgehensweise auch für andere Bedienmodalitäten Verwendung finden kann. Hierfür wurden Teile der MMS des KI und der HU im definierten Format modelliert. Für die Inhalte des KI ließ sich feststellen, dass die Modellierung der grundlegenden Menüstrukturen gut möglich ist. Die Testmodelle konnten ohne großen Aufwand aus dem vorhandenen Systemmodell abgeleitet werden. Einschränkend wurde festgestellt, dass für die vollständige Modellierung aller Systemfunktionen eine Weiterentwicklung der Modellierungssyntax zweckmäßig erscheint. Für die Inhalte der HU wurde festgestellt, dass das vielschichtige Dialogverhalten der MMS auf mehreren Ebenen mit zahlreichen Abhängigkeiten und Eingabemöglichkeiten zu scheinbar beliebig komplexen Modellen führt. Die haptische MMS der HU kann zwar grundsätzlich modelliert werden, dabei ist jedoch eine Fokussierung auf besonders testrelevante Inhalte notwendig, um die Effizienz der modellbasierten Vorgehensweise zu gewährleisten.

6 Testfallgenerierung zum Test der MMS

In diesem Kapitel soll gezeigt werden, dass auf Basis des in Kapitel 5 entwickelten Modells einer akustischen MMS Testfälle generiert werden können. Damit geht es um den Beweis der These B (siehe Kapitel 3.2).

6.1 Evaluation kommerzieller Testfallgeneratoren

6.1.1 Kommerzielle Testfallgeneratoren

Auf dem Markt sind verschiedene Testfallgeneratoren erhältlich. Eine umfassende Übersicht und Einordnung vorhandener Tools findet sich in [Göt09]. Die Angebote an kommerziellen Testfallgeneratoren befinden sich, aufgrund kontinuierlicher Weiterentwicklung, in einem ständigen Wandel. Modellbasiertes Testen hat Potenzial, und nicht wenige Firmen versuchen durch den Vertrieb entsprechender (Spezial-) Werkzeuge die Potenziale zu heben und wirtschaftlich für sich zu nutzen. In einer Marktanalyse konnten Werkzeuge identifiziert werden, welche grundsätzlich Potenzial zur Generierung von Testfällen für eine akustische MMS erkennen lassen. Die in Tabelle 9 aufgezeigten sechs Werkzeuge zur Testfallgenerierung werden im Weiteren analysiert und evaluiert.

Die betrachteten Werkzeuge können in zwei Kategorien unterschieden werden. Zum einen gibt es Werkzeuge, welche ausschließlich zur Testfallgenerierung verwendet werden. Hierbei wird das zu Grunde liegende Modell in einem unabhängigen Modellierungswerkzeug erstellt und anschließend über eine entsprechende Schnittstelle in den Testfallgenerator importiert

(„.getmore“, „CertifyIt“, „TCGen“). Zum anderen gibt es Werkzeuge, welche sowohl die Möglichkeit der Modellierung als auch Testfallgenerierung anbieten („Conformiq Tool Suite“, „Rational Rhapsody“, „MaTeLo Test Generator“). Dabei ist Testfallgenerierung und Modellierungssyntax innerhalb des Werkzeugs aufeinander abgestimmt. Dies schließt jedoch nicht grundsätzlich aus, dass auch Modelle aus anderen Modellierungswerkzeugen importiert werden können. Für weitere Hintergrundinformationen zu den verschiedenen Werkzeugen sei auf die in Tabelle 9 aufgeführten Quellen verwiesen.

Tabelle 9: Übersicht kommerzieller Testfallgeneratoren

Werkzeug	Vers.	Anbieter	Inhalt	Quelle (www.)
Conformiq Tool Suite	4.2	Verifysoft	Modellierung & Testfallgenerierung	http://www.verifysoft.com
Rational Rhapsody	7.5.3	IBM	Modellierung & Testfallgenerierung	http://www.ibm.com/developerworks/rational/
MaTeLo Test Generator	4.7.1	All4tec	Modellierung & Testfallgenerierung	http://www.all4tec.net/index.php/All4tec/test-cases-generator.html
"MBTsuite" (vor 2012: ".getmore")	1.3	sepp.med	Testfallgenerierung	http://www.seppmed.de/produkte/mbtsuite.html
CertifyIt (Test Designer)	4.0	Smarttesting	Testfallgenerierung	http://www.smartesting.com/index.php/cms/en/home
TCGen Test Case Generator	-	Verified	Testfallgenerierung	http://www.verified.de/en/products/tcgen

Um die Werkzeuge zu evaluieren, werden im folgenden Teilkapitel zunächst Anforderungen festgelegt. Können diese erfüllt werden, kann die Testfallgenerierung im praktischen Einsatz realisieren werden.

6.1.2 Anforderungen an das Werkzeug zur Testfallgenerierung

Mit dem Ziel, verschiedene Werkzeuge zur Testfallgenerierung zu evaluieren, sind Anforderungen an selbige zu definieren. Da kaum alle Anforderungen gleichermaßen von allen Werkzeugen abgedeckt werden, wird deren Erfüllung für jedes Werkzeug mit einem Faktor zwischen 0 und 5 bewertet.

0: Anforderung nicht erfüllt
1: Anforderung nicht ausreichend erfüllt
2: Anforderung im Ansatz erfüllt
3: Anforderung teilweise erfüllt
4: Anforderung mit geringfügiger Einschränkung erfüllt
5: Anforderung erfüllt

Des Weiteren sind nicht alle definierten Anforderungen von gleicher Priorität. Um dies zu berücksichtigen, sind die Anforderungen mit einem Faktor zwischen 1 (geringere Priorität) und 3 (höchste Priorität) gewichtet.

Die folgenden Anforderungen werden an ein Testfallgenerierungswerkzeug zum Test einer akustischen MMS gestellt. Die entsprechend zugewiesene Priorität einer Anforderung ist in Klammer ergänzt.

a) Kosten und Support

- Die Lizenzkosten für das Werkzeug sind gering, um die Wirtschaftlichkeit des MBT zu gewährleisten. (3)
- Vom Hersteller/Anbieter des Werkzeugs wird ein (leicht zugänglicher) technischer Support und kostengünstige Schulungen zum Umgang mit dem Werkzeug angeboten. (1)

b) Ressourcenbedarf und Kompatibilität

- Das Werkzeug ist zu Windows XP und Windows 7 kompatibel. (2)
- Der Ressourcenbedarf des Werkzeugs ist möglichst gering, die Stabilität des Programms im Rahmen der vorhandenen Ressourcen gewährleistet. (1)

c) Bedienung

Eine grundlegende Anforderung an den hier entwickelten MBT-Prozess ist eine möglichst einfache Vorgehensweise. Auch Tester oder Entwickler ohne entsprechende Testexpertise sollen in der Lage sein, mit dem Werkzeug Testfälle zu generieren (vergl. Kapitel 3.2).

- Der Einarbeitungsaufwand zur Verwendung des Werkzeugs muss möglichst gering sein. (3)
- Die Bedienung des Werkzeugs soll möglichst sinnfällig gestaltet sein und eine hohe die Usability aufweisen. (3)
- Das Werkzeug soll in deutscher und englischer Sprache verfügbar sein. (3)

d) Schnittstellen zu weiteren Werkzeugen

- Das Werkzeug unterstützt den Import/Export von Modellen im XMI/XML-Format. (3)
- Das Werkzeug stellt eine Schnittstelle bereit, um Werkzeuge zur Testfallorganisation und -auswertung anzubinden (z. B. HP Quality Center). (2)
- Das Werkzeug bietet weitere Schnittstellen zur Anbindung an Testwerkzeuge und unterstützt weitere Formate zum Import/Export von Modellen oder Testfällen (z. B. .xls, .xml...) (2)

e) Funktionalität der Modellierung (wenn im Werkzeug vorhanden)

- Das Werkzeug unterstützt die Testfallgenerierung aus UML-konformen Modellen. (3)
- Das Werkzeug bietet eine statische Prüfung erstellter Modelle an, um frühzeitig Fehler im Modell zu identifizieren. (3)
- Das Werkzeug verfügt über ein Versionsmanagement, um Änderungen über mehrere Versionen hinweg nachvollziehen zu können. (3)
- Werden Änderungen an einem mehrfach verwendeten Element des Modells durchgeführt, wirkt sich diese automatisch auf alle verwendeten Instanzen des Elements im Modell aus. (3)
- Das Werkzeug ermöglicht eine Anbindung an Werkzeuge zur Anforderungsdokumentation (z. B. IBM DOORS). (2)

f) Funktionalität der Testfallgenerierung

- Es besteht die Möglichkeit manuell zu definieren, aus welchen Modellteilen Testfälle generiert werden sollen. (2)
- Das Werkzeug ermöglicht die Testfallgenerierung nach verschiedenen, wählbaren Strategien. Hierzu gehört die Generierung zur vollständigen Pfadüberdeckung, Zustandsüberdeckung und Transitionsüberdeckung. (3)
- Es besteht die Möglichkeit, zufallsbasiert (randomisiert) Testfälle aus dem Modell zu generieren. (3)
- Das Werkzeug ermöglicht eine Testfallgenerierung in Abhängigkeit vorhandener Prioritäten im Modell. Voraussetzung ist, dass Prioritäten im Modell vorhanden sind (Kanten-Priorisierung). (3)
- Das Werkzeug errechnet die Modellüberdeckung und ermöglicht so die Steuerung der Generierungsstrategien über Abdeckungsgrade. (3)

- Ändert sich die Testbasis (das Modell), ermöglicht das Werkzeug eine Deltagenerierung von Testfällen. Dabei werden ausschließlich hinzukommende/veränderte Testfälle generiert. (3)
- Die Generierung der Testfälle ist reproduzierbar. Werden unabhängig voneinander Testfälle aus der gleichen Testbasis mit derselben Teststrategie generiert, sind identische Testfälle zu erwarten. (3)
- Die mit dem Werkzeug generierten Testfälle können als maschinenlesbare Testfälle generiert/exportiert werden. (3)
- Die mit dem Werkzeug generierten Testfälle können in einem für den Menschen gut lesbaren Format generiert/exportiert werden. (3)
- Werden Testfälle generiert, erstellt das Werkzeug ein Testprotokoll, in welchem grundlegende Generierungsparameter festgehalten sind. (2)

Die in Tabelle 9 vorgestellten Werkzeuge werden im Folgenden mit Bezug auf die beschriebenen Anforderungen analysiert, um festzustellen, welches Werkzeug diesen am ehesten genügt.

6.1.3 Bewertung und Auswahl eines Testfallgenerators

Für eine fundierte Bewertung wurden alle Werkzeuge einem Praxistests unterzogen. Testversionen wurden installiert, um Testfälle aus den entwickelten Testmodellen zu generieren. Dabei wurden die Werkzeuge hinsichtlich der definierten Anforderungen geprüft und bewertet. In Abbildung 47 sind die Ergebnisse der Evaluation dargestellt.

Die Datenbasis des Ergebnisdiagramms ist mit allen Teilbewertungen in der Bewertungsmatrix, Anhang 12.3 dargestellt.

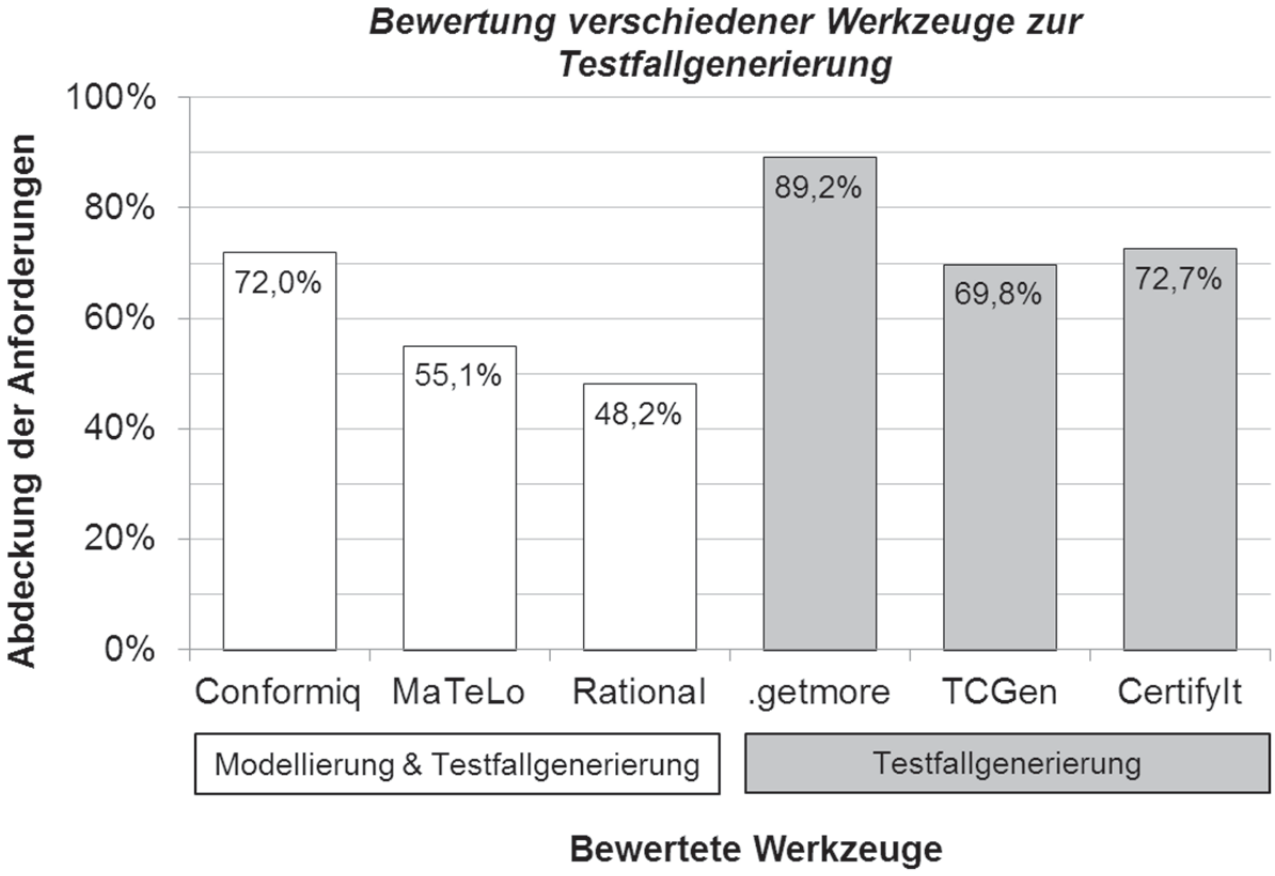


Abbildung 47: Bewertung verschiedener Testfallgeneratoren

Eine detaillierte Betrachtung der Ergebnisse führt zu verschiedenen Erkenntnissen. Es fällt auf, dass sich die Lizenzkosten teilweise erheblich unterscheiden. Das Werkzeug „TCGen“ muss dabei gesondert betrachtet werden, da es im Testbetrieb von Steuergeräten bereits verwendet wird und ohne weitere finanzielle Aufwände zur Verfügung steht. „CertifyIt“ ist zum Zeitpunkt der Analyse nicht als Einzelplatzlizenz erhältlich, sondern ausschließlich als Paket mit mindestens fünf Einzelplatzlizenzen. Damit sind die Kosten von 4000 € pro Lizenz zwar günstig, aufgrund der festgelegten Losgröße entstehen jedoch höhere Beschaffungskosten als bei anderen Werkzeugen. Teuerstes Werkzeug im Test ist die „Conformiq Tool Suite“,

deren Kosten für eine Lizenz die günstigste Alternative um das Siebenfache übersteigt. Alle Werkzeuge sind sowohl auf Windows XP als auch Windows 7 lauffähig und der Bedarf an Rechenleistung sowie weiteren Systemressourcen ist in allen Fällen akzeptabel.

In der Analyse der Bedienfreundlichkeit lassen sich wiederum Unterschiede feststellen. Während die Mehrheit der Werkzeuge mit einer überzeugenden Usability punkten kann, fallen die Werkzeuge „Rational Rapsody“ und „TCGen“ in dieser Kategorie ab. Ein ähnliches Bild ergibt sich bei der Bewertung verfügbarer Schnittstellen zu externer Software. Hier werden insbesondere die Werkzeuge „MaTeLo“ und „TCGen“ den gestellten Anforderungen nicht gerecht.

Drei der Werkzeuge bieten auch die Möglichkeit der Modellierung. In der Bewertung dieser Funktionalität kann allerdings ausschließlich das Werkzeug „Conformiq“ überzeugen. Die Werkzeuge „.getmore“, „CertifyIt“, und „TCGen“, und bieten eine entsprechende Möglichkeit zur Modellierung nicht an und sind bei der Bewertung ausgenommen.

Schließlich ist die Funktionalität der Testfallgenerierung ausführlich analysiert und bewertet. In Einzelfällen konnte kein endgültiges Urteil bezüglich Erfüllung der gestellten Anforderungen gefällt werden. Hier wurde folglich keine Bewertung vorgenommen und die Anforderung in der Ergebnisberechnung nicht berücksichtigt. Insbesondere das Werkzeug „.getmore“ konnte in der Analyse überzeugen und nahezu alle Anforderungen der Testfallgenerierung umfassend erfüllen. Mit Abstrichen, aber ebenfalls überzeugend schneidet das Werkzeug „Conformiq“ in dieser Kategorie ab.

In der abschließenden Zusammenfassung aller bewerteten Kategorien und Kriterien zeigt sich, dass das Werkzeug „.getmore“ der Firma „sepp.med“ den gestellten Anforderungen am ehesten gerecht wird. Dabei überzeugt das Werkzeug insbesondere durch seine Bedienfreundlichkeit, ausreichend Schnittstellen zu externer Software und eine überzeugende Funktionalität in der Testfallgenerierung. Zudem sind die Lizenzkosten verhältnismäßig günstig.

Das Werkzeug „.getmore“ wurde im Jahr 2012 in „MBTsuite“ umbenannt. Inhaltlich wurden dabei keine einschränkenden oder bezüglich der Anforderungen kritischen Änderungen an der Software vorgenommen. Im Folgenden wird daher der Name „MBTsuite“ weiterverwendet und kann als Synonym zu „.getmore“ betrachtet werden.

6.2 Testfallgenerierung aus dem Testmodell einer SW-MMS

Nachdem die „MBTsuite“ als Werkzeug der Wahl zur Testfallgenerierung aus dem Testmodell der akustischen MMS definiert ist, sollen die Möglichkeiten der Testfallgenerierung genauer erläutert werden.

6.2.1 Umsetzung und Herausforderungen der Testfallgenerierung

Die Vorgehensweise zur Testfallgenerierung aus den entwickelten Testmodellen ist in Abbildung 48 dargestellt. Das im Modellierungswerkzeug erstellte Testmodell wird im XMI-Standardformat exportiert und in den Testfallgenerator importiert.

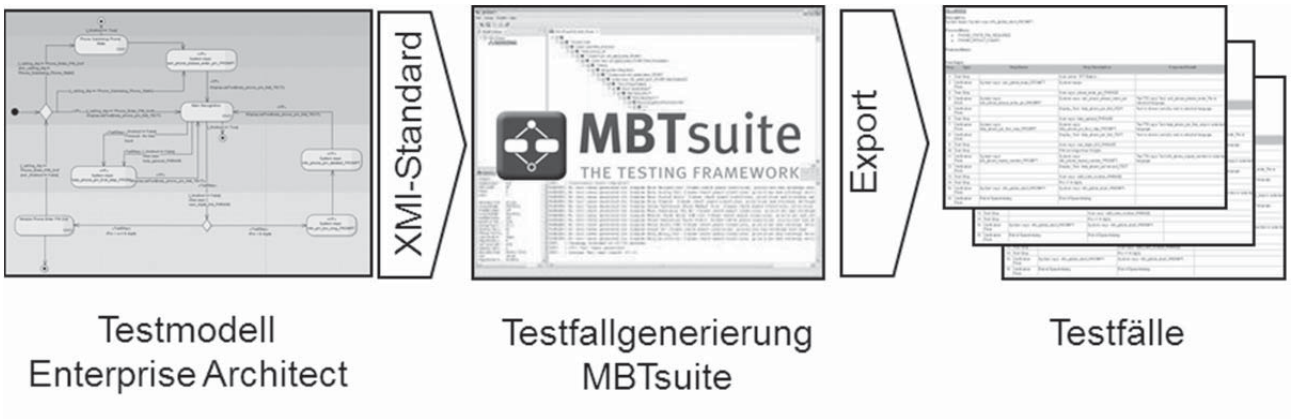


Abbildung 48: Vorgehensweise zur Testfallgenerierung „MBTsuite“

Die Testfallgenerierung kann nach verschiedenen wählbaren Strategien stattfinden. Sind Testfälle erstellt, können diese ausgewählt und in verschiedene Formate exportiert werden. Hierzu gehört neben den Formaten .doc,, .xls, html, etc. auch der Export in das HP Quality Center oder benutzerdefinierte Formate. Abbildung 49 zeigt den Aufbau der Bedienoberfläche des Werkzeugs.

Um die Testfallgenerierung mit MBTsuite aus dem entwickelten Testmodell in der Praxis umzusetzen, ist es notwendig, das Modell geringfügig zu erweitern. Der Testfallgenerator unterscheidet Testschritte (engl. „Test step“, TS) und Verifikationspunkte (engl. „Verification point“, VP). Im konkreten Fall sind die vom Menschen durchgeführten Aktivitäten als TS zu interpretieren, wohingegen die Systemreaktion als VP zu verstehen sind. In den „Eigenschaften“ der Modellelemente sind die entsprechenden Parameter (TS oder VP) zu modellieren. Des Weiteren ist es möglich, vorhandene Preconditions als solche zu parametrisieren. Zur Veranschaulichung sind die in Kapitel 5.2 dargestellten Modellbeispiele der akustischen und haptischen MMS mit den entsprechenden Erweiterungen beispielhaft umgesetzt.

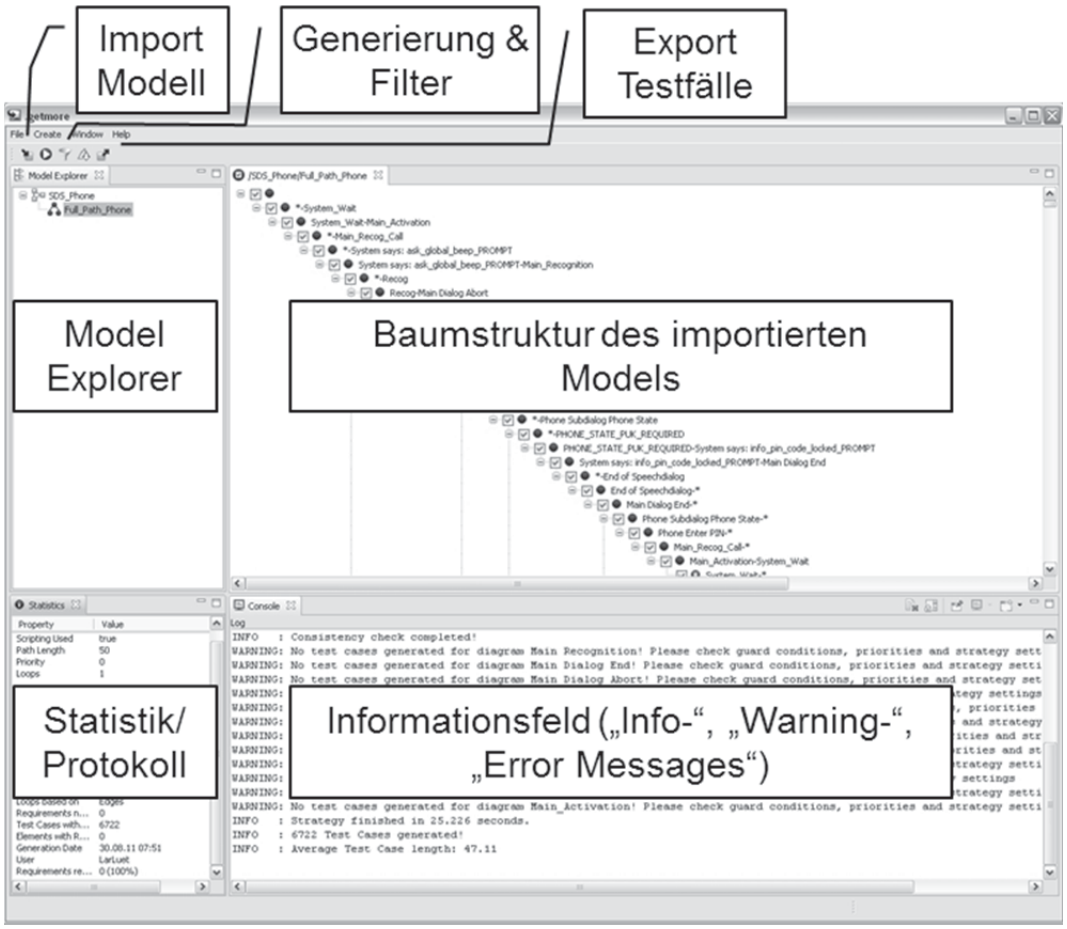


Abbildung 49: GUI der „MBTsuite“ von „sepp.med“

Die angepassten Modelle finden sich im Anhang in Abschnitt 12.3, 12.5 und 12.6. Zur Realisierung der Testfallgenerierung sind auch die repräsentativ erstellten Testmodelle der akustischen und haptischen MMS entsprechend erweitert. Für weiterführende Informationen sei auf die Modellierungsrichtlinien der Firma „sepp.med“ verwiesen [Mie11].

Beim Import von Modellen führt die MBTsuite eine statische, syntaktische Prüfung der Modelle durch, um formale Fehler noch vor der Testfallgenerierung zu identifizieren. In der praktischen Anwendung hat sich diese Funktion als wertvoll erwiesen. In den erstellten Testmodellen konnten mehrfach

formale Inkonsistenzen und Fehler korrigiert werden. Als häufigste Fehlerquelle wurden dabei fehlerhafte oder nicht modellierte Variablenzuweisungen erkannt. Ohne Korrektur dieser Fehler hätten verschiedene Pfade in der Testfallgenerierung nicht berücksichtigt werden können.

Neben diesen formalen Fehlern wurde ein weiterer, systematischer „Modellierungsfehler“ erkannt, dessen Behebung mit größerem Aufwand verbunden ist. Im Modell der akustischen MMS wurden rekursive Diagrammaufrufe identifiziert, welche bei der Modellierung aus dem Systemmodell übernommen wurden. Rekursive Aufrufe sind in der Testfallgenerierung kritisch, da sie formal als diagrammübergreifende Schleifen interpretiert werden können, welche vom Testfallgenerierungswerkzeug als solche jedoch nicht erkannt werden. Die bei der Testfallgenerierung übliche Einschränkung von Schleifendurchgängen ist in diesem Falle wirkungslos. In der Folge wird eine unendlich große Anzahl an Testfällen möglich. Aus diesem Grund ist es zwingend erforderlich, rekursive Diagrammaufrufe zu unterbinden. Auch das in Kapitel 5.2.2 vorgestellte Beispielmodell der akustischen MMS (Abbildung 37 f.) enthält einen rekursiven Aufruf. Das aus dem „Start-Diagramm“ aufgerufene Diagramm „System_state: Recognition“ ruft das Diagramm „Phone Enter PIN“ auf. In letzterem wird neben anderen Diagrammen erneut das Diagramm „System_state: Recognition“ aufgerufen. Dieser rekursive Aufruf führt zu einer diagrammübergreifenden Schleife im Modell.

Abbildung 50 (links) veranschaulicht die Beziehung zwischen den verschiedenen beschriebenen Diagrammen. In Abbildung 50 (rechts) ist zudem eine mögliche Lösung rekursiver Modelle dargestellt. Das Diagramm „System_state: Recognition“ wird hierbei aufgeteilt in eine lokale Komponente, in

welcher die kontextabhängigen Eingabekommandos modelliert sind („System_state: Phone Recognition“) und eine globale Komponente, in welcher dialogunabhängig gültige Kommandos modelliert werden können („System_state: Phone Recognition“).

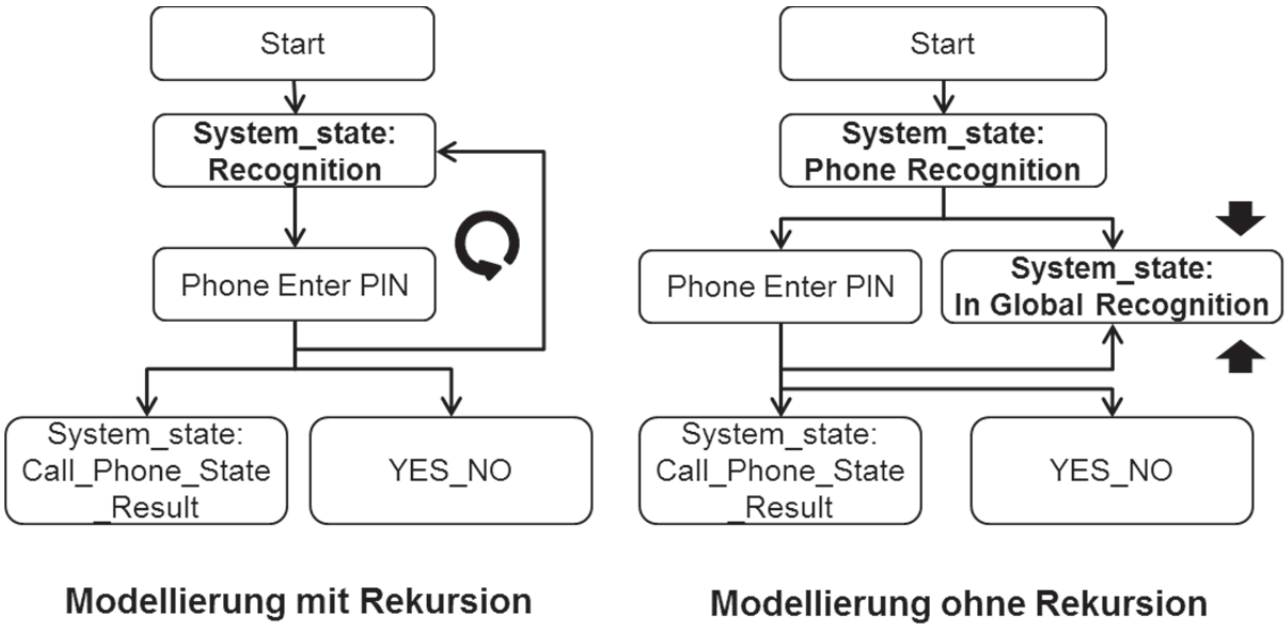


Abbildung 50: Rekursiver Aufruf im Beispielmodell der akustischen MMS

Durch diese Trennung kann das zentrale Diagramm von mehreren Diagrammen aufgerufen werden, ohne dass eine Rekursion entsteht. Diese Lösung ist im Beispieldiagramm für die akustische MMS umgesetzt und im Anhang, Abschnitt 12.3 dargestellt.

Sind die formalen Fehler und Inkonsistenzen im Testmodell korrigiert, kann die Testfallgenerierung folgen. Hierbei ergaben sich verschiedene, wertvolle Erkenntnisse hinsichtlich der anzuwendenden Teststrategien und den Grenzen der Testfallgenerierung.

6.2.2 Strategien zur Generierung von Testfällen aus dem Beispielmodell

Um die verschiedenen Möglichkeiten und Eigenschaften der Testfallgenerierung genauer zu betrachten, werden zunächst die bereits bekannten Beispiele (Anhang, Abschnitt 12.3 bis 12.6) der softwarebasierten MMS als Basis für die Generierung von Testfällen verwendet.

Es besteht die Prämisse, das SUT möglichst vollständig zu testen, um eine zweifelsfreie Aussage zu Reifegrad und funktionaler Qualität treffen zu können. Die mächtigste Teststrategie, um dieses Ziel zu erreichen, ist die vollständige Pfadüberdeckung (Kapitel 2.2.4). Die MBTsuite bietet die Möglichkeit, Testfälle nach dieser Strategie zu generieren, wobei der Anwender verschiedene Parameter der Generierung festlegen kann. Zum einen ist die maximale Anzahl der zu durchlaufenden Testschritte festzulegen. Je größer die deren Anzahl, desto tiefer dringen die Testfälle in den Dialog vor. Des Weiteren ist die maximale Anzahl an Schleifendurchläufen zu bestimmen. Im folgenden Beispiel ist diese zunächst auf eine Iteration beschränkt.

In einem ersten Durchgang werden Testfälle aus dem Beispielmodell der akustischen MMS generiert. Hierbei wird schrittweise die maximale Anzahl an Testschritten erhöht. In Abhängigkeit dieser Veränderung wird die Anzahl der generierten Testfälle und die erreichte Modellüberdeckung ausgewertet. In Abbildung 51 und Abbildung 52 ist das Ergebnis der Generierung dargestellt.

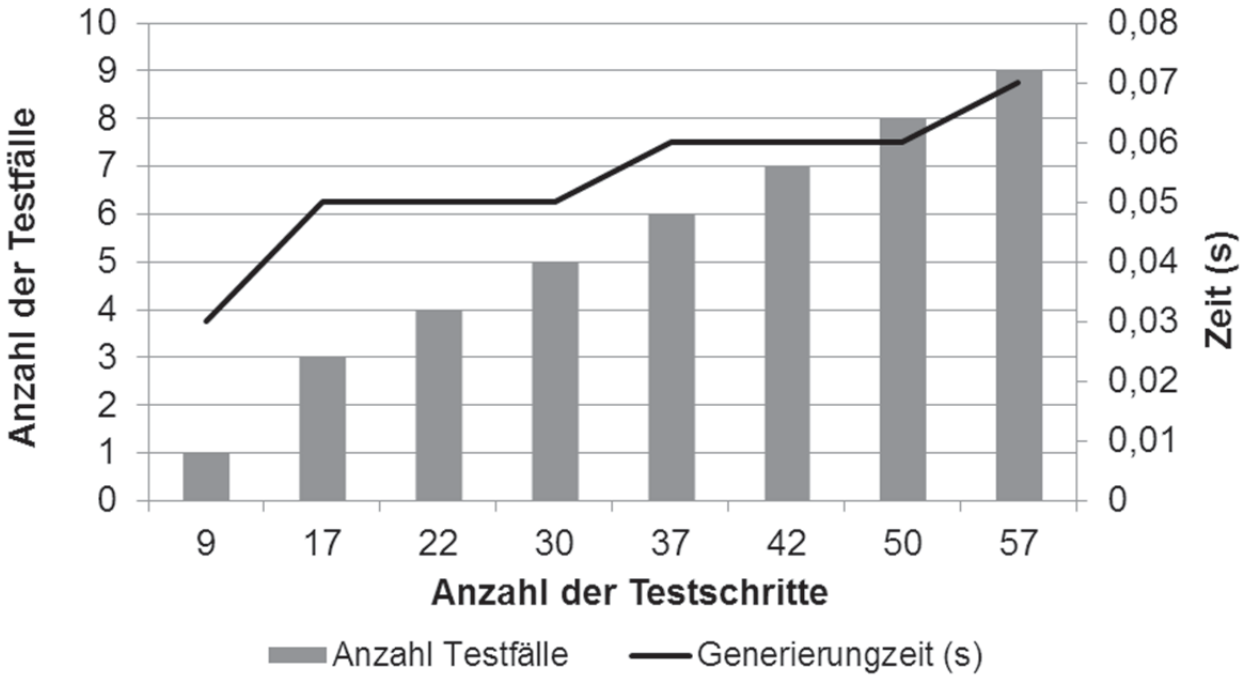


Abbildung 51: Akustische MMS: Generierungszeit und Anzahl der Testfälle in Abhängigkeit der Testtiefe (Beispielmodell)

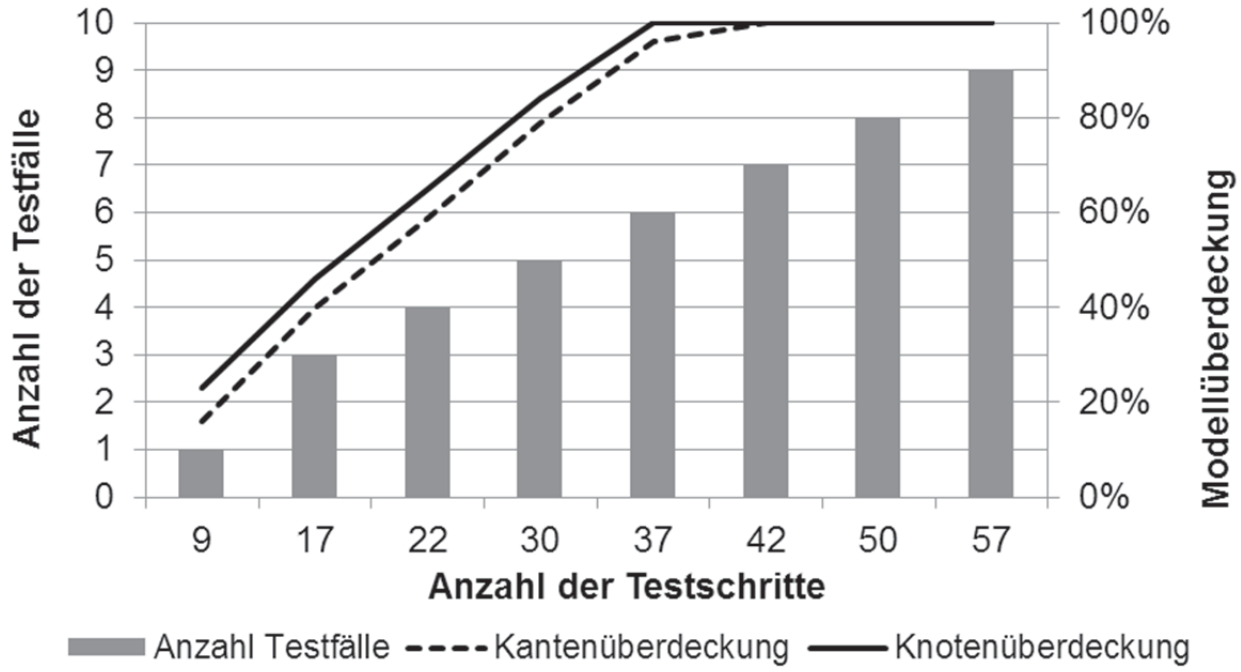


Abbildung 52: Akustische MMS: Modellüberdeckung und Anzahl der Testfälle in Abhängigkeit der Testtiefe (Beispielmodell)

Die Grafik zeigt, dass die Menge der generierten Testfälle, wie zu erwarten, mit zunehmender Testtiefe ansteigt. Gleichzeitig erhöht sich auch die Generierungszeit. Die Zusammenhänge zwischen Testtiefe, Testfallmenge und Überdeckung von Kanten bzw. Knoten im Beispielmodell der akustischen MMS sind in Abbildung 52 dargestellt. Eine vollständige Knotenüberdeckung wird mit 37 Testschritten erreicht, die Kantenüberdeckung mit 42 Testschritten.

Die Testfälle wurden nach der Generierung in das „.xls-Format“ exportiert. Einer der generierten Testfälle ist beispielhaft in Abbildung 53 dargestellt.

Test Case		**SDS Bsp0001**
Precondition(s)		phone_state-Result; PHONE_STATE_PHONE_OK
Step	**Type**	**Step Description**
1	TS	User_action: PTT
2	VP	System_prompt: "beep"
3	TS	User_phrase: PIN eingeben
4	VP	System_prompt: Bitte PIN eingeben
5	VP	Display_text: PIN eingeben
6	TS	User_phrase: <DIGITS>
7	VP	System_prompt: <DIGITS>
8	VP	System_display: <DIGITS>
9	VP	System_Prompt: Ist die Eingabe korrekt?
10	VP	System_Text: Ja oder Nein?
11	TS	User_phrase: Ja
12	VP	System_Prompt: PIN übernommen

Abbildung 53: Generierter Testfall aus dem Beispielmodell der akustischen MMS

Der dargestellte Testfall ist gut leserlich und beinhaltet alle für die Testdurchführung notwendigen Informationen. Nach Beschreibung der erforderlichen Vorbedingung sind die einzelnen Testschritte (TS) und Verifikationspunkte (VP) in der auszuführenden Reihenfolge abgebildet. Der dargestellte Testfall kann im Beispielmodell (Anhang 12.3) nachvollzogen werden.

Analog zur Testfallgenerierung aus dem Beispielmodell der akustischen MMS kann diese auch für die Modelle der haptischen MMS durchgeführt werden. Abbildung 54 zeigt, wie sich die Anzahl der generierten Testfälle mit zunehmender Testtiefe im Modell der HU- und KI-MMS entwickelt. Es ist zu beachten, dass die primäre y-Achse (Anzahl der Testfälle) logarithmisch dargestellt ist.

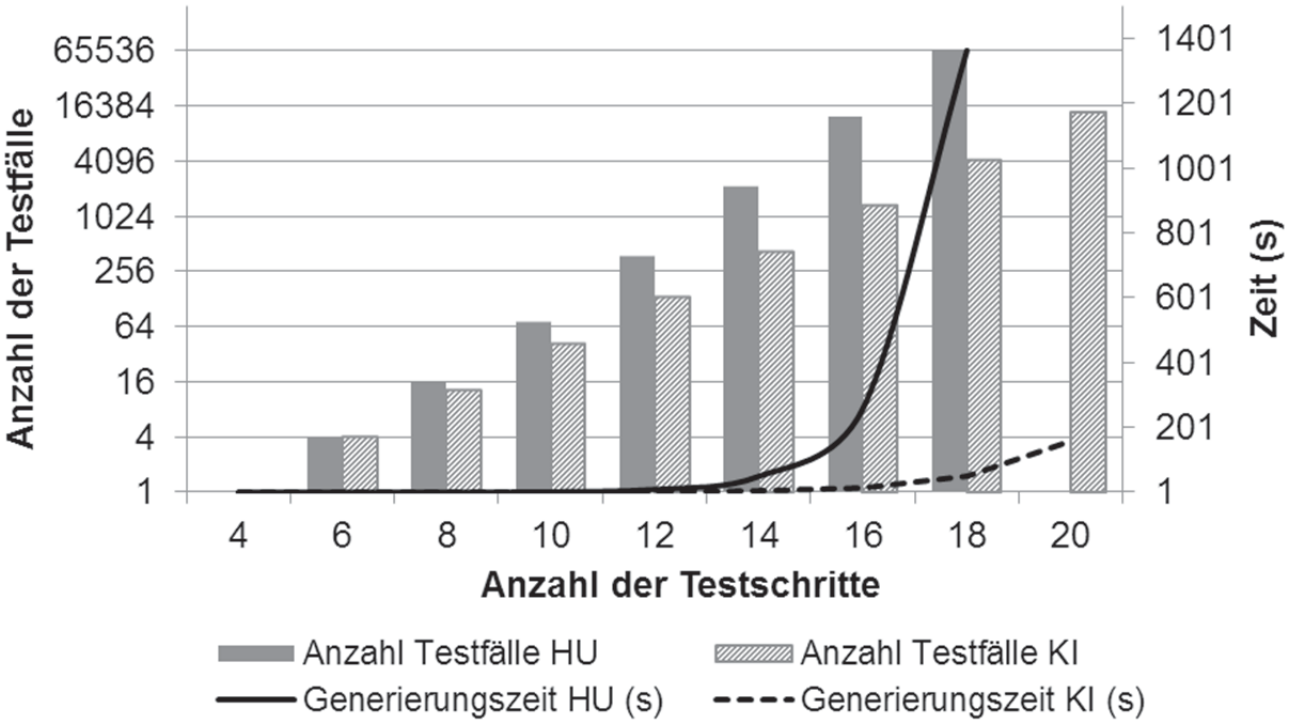


Abbildung 54: Haptische MMS; Generierungszeit und Anzahl der Testfälle in Abhägigkeit der Testtiefe (Beispielmodell)

Es fällt auf, dass die Menge der Testfälle mit zunehmender Testtiefe exponentiell ansteigt. Analog verhält sich die notwendige Generierungszeit. Dieser Sachverhalt unterscheidet sich von den Ergebnissen der akustischen MMS (Abbildung 51). Hier ist die Entwicklung der Testfallmenge weitgehend linear zur Testtiefe. Einschränkend ist festzustellen, dass die geringe Testfallmenge keine abschließende Aussage zur „Wachstumscharakteristik“ zulässt. Weitere Unterschiede werden bei Betrachtung der Modellüberdeckung deutlich. Während für eine vollständige Pfadüberdeckung des akustischen Modells neun Testfälle mit 57 Testschritten notwendig sind, werden zur Pfadüberdeckung der haptischen MMS über 260.000 Testfälle benötigt.

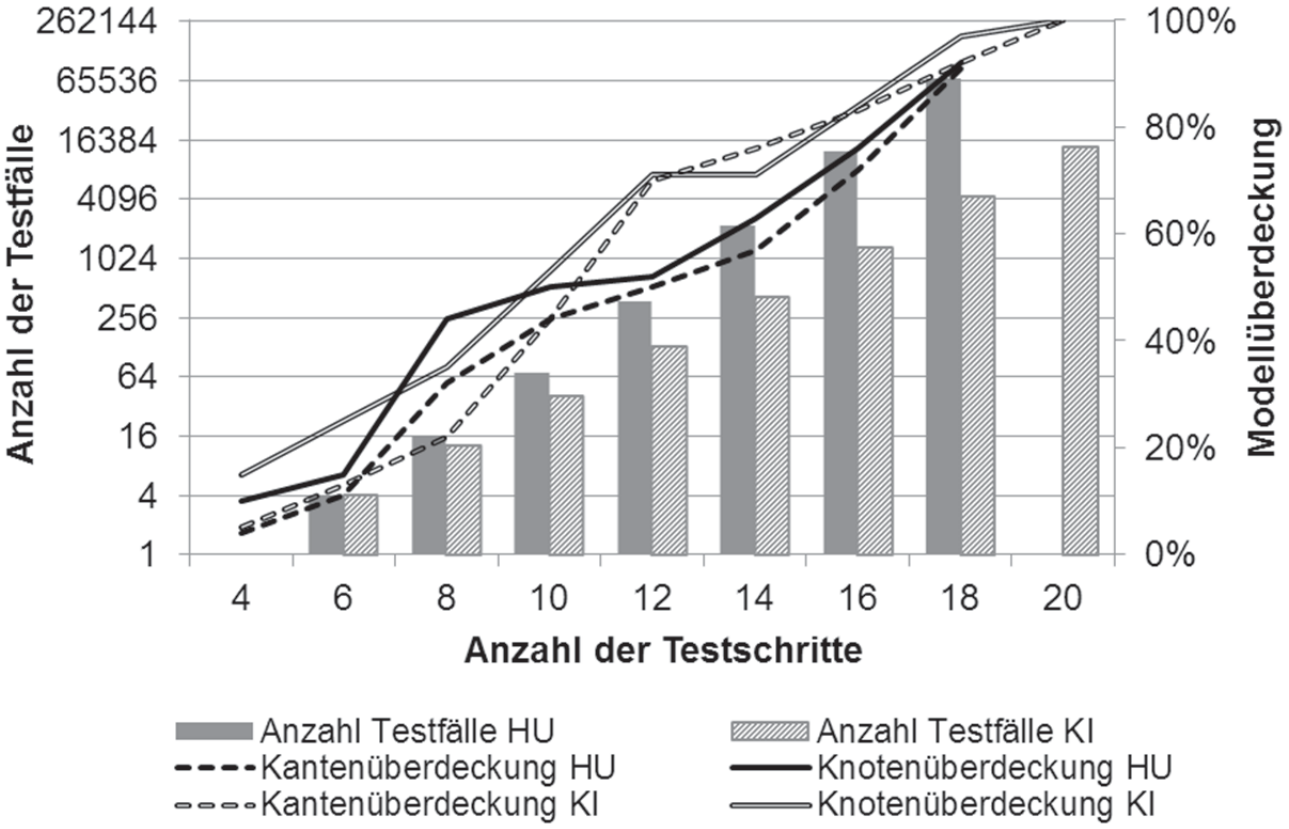


Abbildung 55: Haptische MMS; Modelüberdeckung und Anzahl der Testfälle in Abhägigkeit der Testtiefe (Beispielmodell)

Die Strategie der vollständigen Pfadüberdeckung stößt bereits in den übersichtlichen Beispielmodellen der haptischen MMS an ihre Grenzen. Praktikable Teststrategie sind hier die vollständige Kanten- oder Knotenüberdeckung. Für die vollständige Kantenüberdeckung der MMS des KI sind überschaubare 11 Testfälle mit einer durchschnittlichen Testtiefe von 20 Schritten erforderlich.

Es bleibt zu klären, warum die Testfallgenerierung in Abhängigkeit des zu Grunde liegenden Beispielmodells von so unterschiedlicher Charakteristik ist, schließlich unterscheiden sich die Modelle hinsichtlich ihrer „Größe“ nur geringfügig. Das Modell der HU-MMS besteht aus vier Diagrammen mit insgesamt 106 Modellelementen, das Modell der KI-MMS beinhaltet fünf Diagramme mit 107 Modellelementen und die akustische MMS ist in sechs Diagrammen und 103 Modellelementen umgesetzt. Die Ursache für die Unterschiede findet sich im Verhältnis zwischen Knoten (Zuständen) und Kanten (Zustandsübergänge) sowie der Modellstruktur.

- Beispielmodell der akustischen MMS: Knoten/Kanten = 1/**1,02**
- Beispielmodell der KI-MMS: Knoten/Kanten = 1/**1,74**
- Beispielmodell der HU-MMS: Knoten/Kanten = 1/**1,79**

Während das Modell der akustischen MMS nahezu gleich viele Kanten wie Knoten beinhaltet, ist dieses Verhältnis in den Modellen der haptischen MMS zu Gunsten der Kanten verschoben. Dieser Umstand begründet sich darin, dass die akustische MMS eher linear aufgebaut ist. Zwar sind durchaus Verzweigungen im Dialog vorhanden, die Abfolge der Dialogschritte in den verschiedenen Pfaden bleibt jedoch gerichtet. Anders ist dies bei der

haptischen MMS. Der Nutzer hat in jedem Systemzustand eine Vielzahl an Möglichkeiten (Kanten), um andere Systemzustände zu erreichen.

Nachfolgend werden die realen Testmodelle betrachtet, um realisierbare Generierungsstrategien zu ermitteln. Zuvor sei darauf hingewiesen, dass wie bereits erwähnt, auch die Anzahl der Schleifendurchgänge festgelegt werden kann. Im Anhang 12.8 ist dargestellt, wie sich eine Veränderung dieser Parameter auf die Anzahl der generierten Testfälle auswirkt.

6.2.3 Strategien zur Generierung von Testfällen aus dem Testmodell

Nachdem die individuelle Charakteristik eines Modells von entscheidender Bedeutung für die zu wählende Teststrategie ist, sollen im Folgenden die umfangreicheren und repräsentativ erstellten Testmodelle der akustischen und haptischen MMS im Fokus stehen.

Die erstellten Testmodelle besitzen folgende Eigenschaften:

- Testmodell der akustischen MMS: Knoten/Kanten = 1/**1,21**
- Testmodell der KI-MMS: Knoten/Kanten = 1/**1,84**
- Testmodell der HU-MMS: Knoten/Kanten = 1/**2,14**

Das Verhältnis zwischen Knoten und Kanten ist in allen erstellten Testmodellen zu Gunsten der Kanten verschoben. Es fällt auf, dass im Modell der HU-MMS zu jedem Systemzustand durchschnittlich mehr als zwei mögliche Zustandsübergänge modelliert sind. Welche Auswirkungen diese Modelleigenschaften auf die Anzahl der Testfälle bei einer vollständigen Pfad-

überdeckung und zunehmender Testtiefe mit sich bringen, wird in Abbildung 56 deutlich.

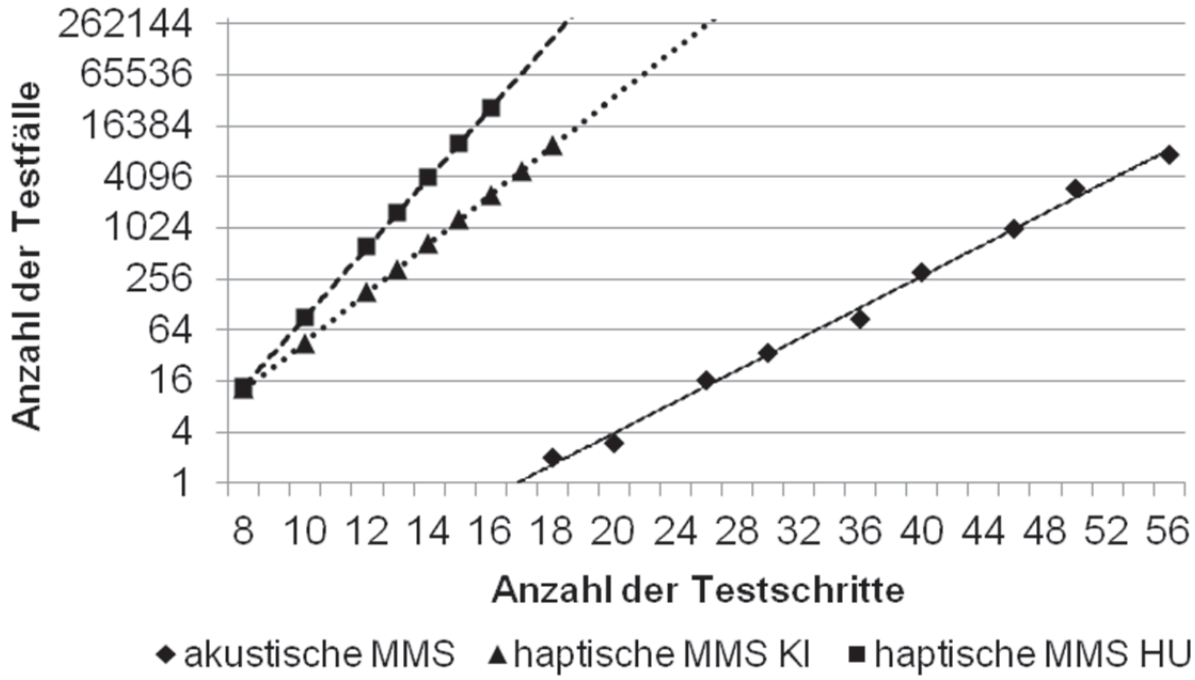


Abbildung 56: Anzahl der Testfälle in Abhängigkeit der Testtiefe (Testmodell)

In allen Fällen steigt die Anzahl der Testfälle exponentiell mit zunehmender Testtiefe. Je mehr Kanten pro Knoten vorhanden sind, umso stärker steigt die Testfallmenge bei zunehmender Testtiefe. Eine vollständige Pfadüberdeckung der modellierten Systemteile der akustischen MMS erfordert die Durchführung von über 9000 Testfällen (siehe Abbildung 57).

Unter Berücksichtigung des Aspekts, dass zum Test der gesamten akustischen MMS hochgerechnet über 320.000 Testfälle durchzuführen sind, wird deutlich, dass die Strategie der vollständigen Pfadüberdeckung nicht realisierbar ist.

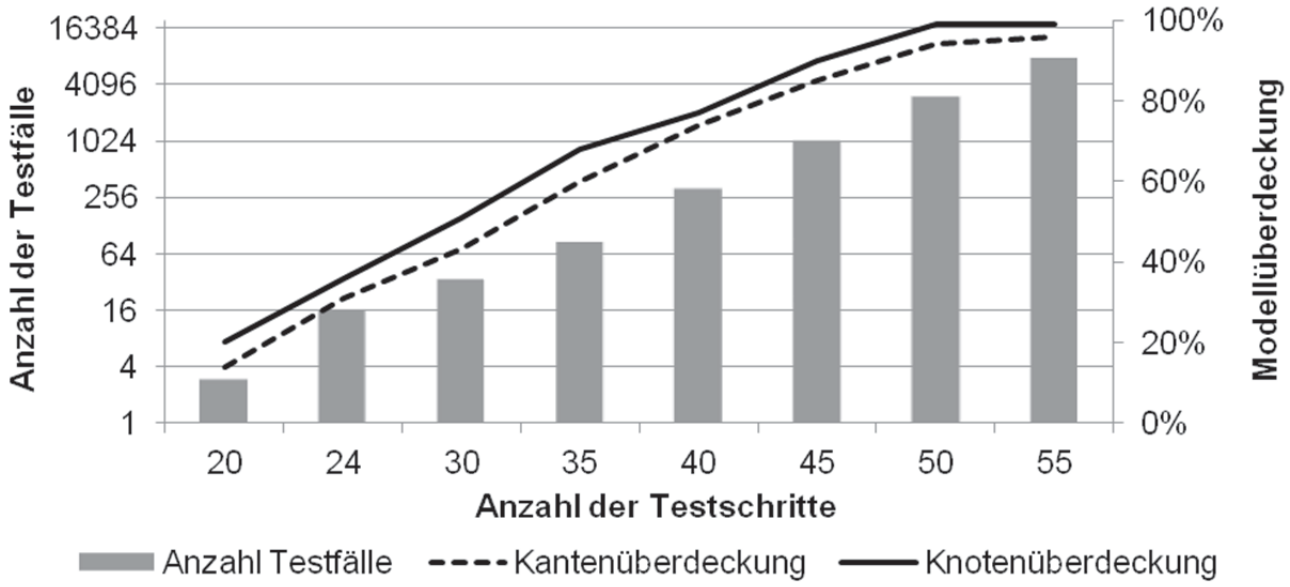


Abbildung 57: Modellüberdeckung und Anzahl der Testfälle in Abhängigkeit der Testtiefe (akustische MMS, Testmodell, Pfadüberdeckung)

Eine hinsichtlich der Testdurchführung eher realisierbare Strategie ist die vollständige Kanten- oder Knotenüberdeckung. Im Testmodell sind 31 Testfälle notwendig um alle Kanten einmal zu durchlaufen, oder 27 Testfälle, um eine Zustandsüberdeckung zu erreichen. Die durchschnittliche Testtiefe liegt zwischen 40 und 45 Testschritten. Hochgerechnet auf das gesamte Modell würden ca. 1000 Testfälle (Kantenüberdeckung) bzw. 900 Testfälle (Knotenüberdeckung) notwendig.

Neben diesen Strategien gibt es eine weitere aussichtsreiche Möglichkeit, gezielt „wichtige“ Testfälle zu generieren. Die MBTsuite ermöglicht die Testfallgenerierung unter Berücksichtigung von modellierten Prioritäten. Einer jeden Transition im Modell wird hierfür eine Priorität zugewiesen, welche auch als Übergangswahrscheinlichkeit verstanden werden kann. Transitionen beschreiben, wie in Kapitel 5.2 ausgeführt, die Aktivitäten des Menschen in

der Bedienung des Systems. Kenntnisse über das Bedienverhalten der Systemnutzer können mittels Ergänzung von Wahrscheinlichkeiten oder Prioritäten im Modell integriert werden. Basierend auf diesen Parametern besteht die Option, gezielt jene Testfälle zu generieren, welche nutzer-relevante Systemteile abdecken. Ein weiteres Kriterium für die Modellierung von Prioritäten kann die Fokussierung auf besonders fehleranfällige oder kritische Systemteile sein. Erfahrungswerte aus vorangegangenen Projekten können hierbei als Basis herangezogen werden. Dieses Vorgehen ist empfehlenswert, da die sonst große Anzahl möglicher Testfälle systematisch und individuell reduziert werden kann.

Eine weitere Strategie ist die der randomisierten Testfallgenerierung. Die MBTsuite gibt dem Anwender die Möglichkeit, diese Strategie zu steuern, indem die Anzahl der zu generierenden Testfälle sowie die maximale Testtiefe eingeschränkt werden kann. Aus dem Testmodell lassen sich auf diese Weise in 16 Sekunden 10 zufällige Testfälle mit 30 % Knotenüberdeckung und einer Testtiefe von 40 Testschritten generieren.

Es ist festzuhalten, dass die Testfallgenerierung aus dem Modell der akustischen MMS mit dem Werkzeug „MBTsuite“ umgesetzt werden kann und die aufgezeigten Strategien eine systematische Testfallgenerierung realisierbar machen. Auf deren Basis kann eine fundierte Aussage zur funktionalen Qualität des SUT getroffen werden.

Auch für die erstellten Testmodelle der haptischen MMS kann die Testfall-generierung realisiert werden. Allerdings werden dabei schnell die Grenzen des praktisch Durchführbaren erreicht. Bereits aus Abbildung 56 ist

erkennbar, dass die Anzahl der Testfälle bei einer vollständigen Pfadüberdeckung bereits für geringe Testtiefen stark ansteigt. Abbildung 58 veranschalicht, dass trotz der großen Menge an Testfällen die Abdeckung des Testmodells mit 5 % (HU-MMS) bzw. 40 % (KI-MMS) nicht ausreicht, um die funktionale Qualität der MMS umfassend zu testen. Neben der großen Anzahl an Testfällen ergeben sich, dabei, zunehmend impraktikable Generierungszeiten. Deren Entwicklung ist für die verschiedenen Testmodelle in Abbildung 59 dargestellt.

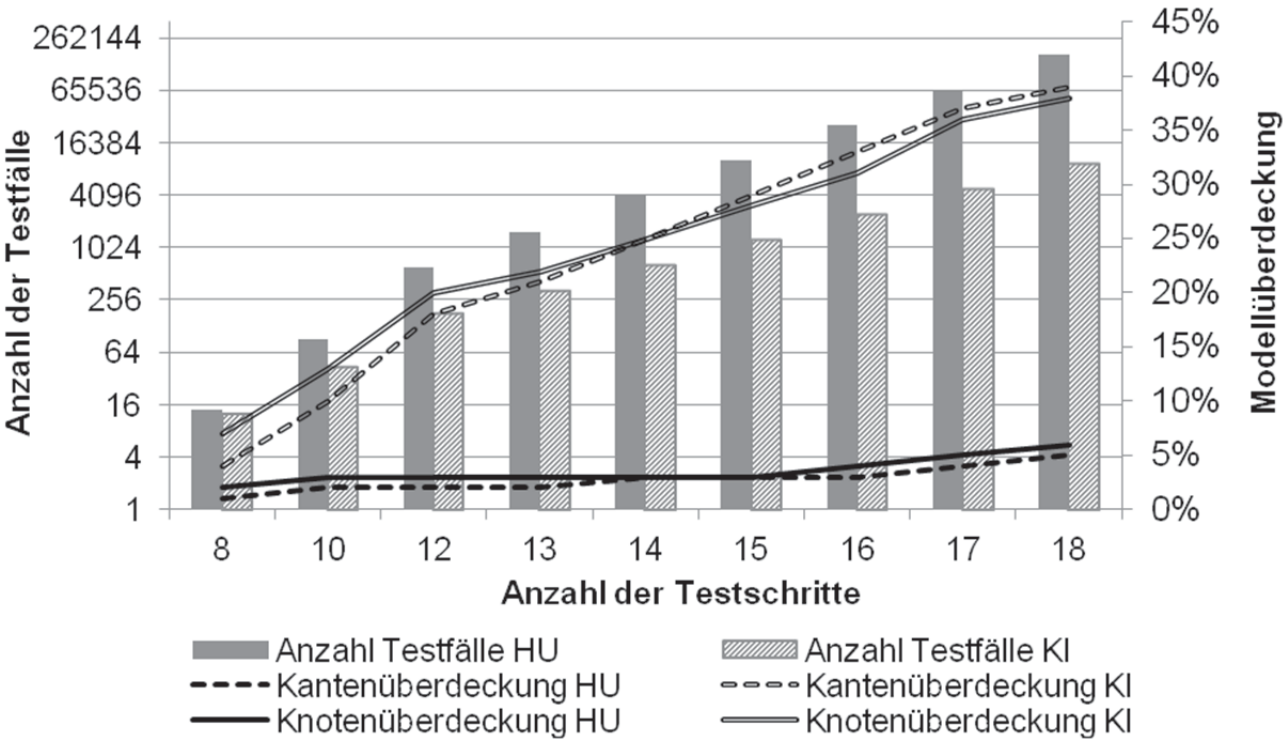


Abbildung 58: Modellüberdeckung und Anzahl der Testfälle in Abhängigkeit der Testtiefe (haptische MMS, Testmodell, Pfadüberdeckung)

Als praktikable Lösung erweist sich angesichts dieser Tatsachen die randomisierte Generierung von Testfällen. In einer Generierungszeit von 74s lassen sich 30 Testfälle mit einer durchschnittlichen Testtiefe von 34 Testschritten

generieren, welche gut 20 % der vorhandenen Knoten und Kanten der modellierten KI-MMS abdecken.

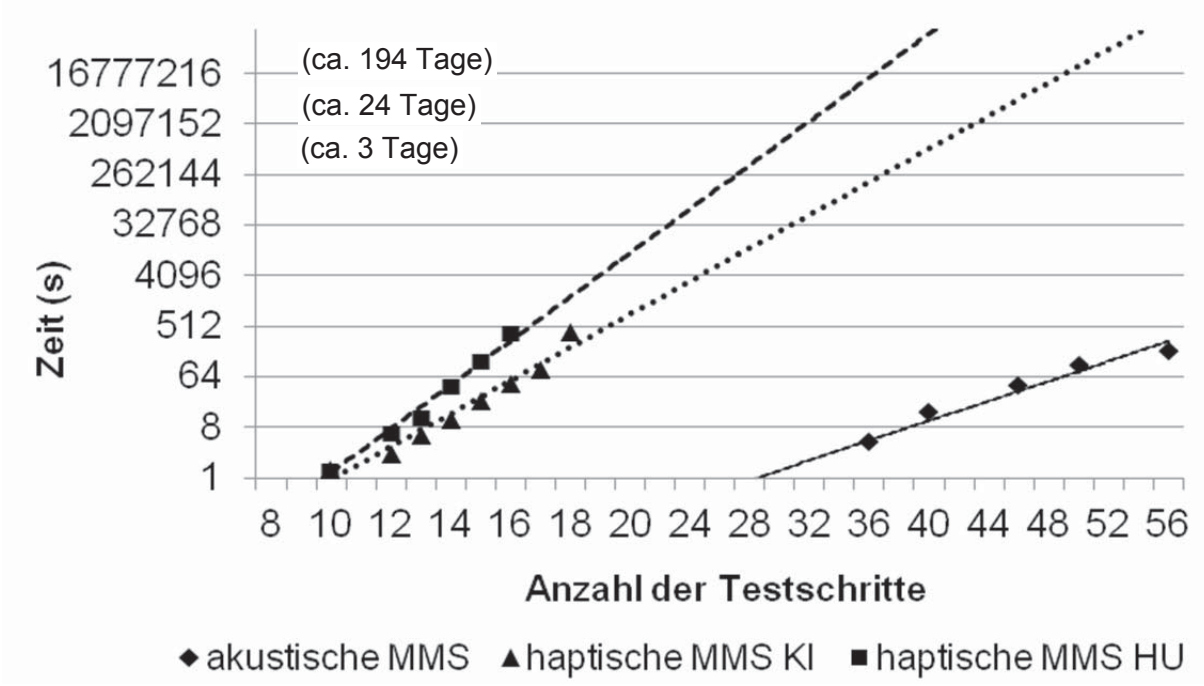


Abbildung 59: Generierungszeiten mit zunehmender Testtiefe (Testmodell)

Bei Anwendung derselben Strategie auf das Modell der HU-MMS werden 30 Testfälle mit einer durchschnittlichen Testtiefe von 70 Testschritten generiert, welche jedoch nur 5 % der vorhandenen Knoten und Kanten abdecken. Hochgerechnet auf das gesamte System würden voraussichtlich mehrere 1000 Testfälle notwendig, um alle Knoten der Testmodelle abzudecken. Es zeigt sich, dass die Testfallgenerierung aus komplexen und umfangreichen Modellen ihre Grenzen hat. Eine sinnvolle, für die HU-MMS notwendige Vorgehensweise ist hier die Priorisierung von Dialogpfaden, um gezielt besonders wichtige oder kritische Testfälle generieren zu können. Zudem ist die Modellierung möglichst einfach zu halten und auf nicht testrelevante Inhalte gezielt zu verzichten. Nur wenn diese Rahmenbedingungen berück-

sichtigt werden ist es zielführend, Testfälle aus erstellten Testmodellen der haptischen MMS zu generieren.

Nachdem bewiesen werden konnte, dass die Testfallgenerierung durch Anwendung eines kommerziellen Werkzeugs aus dem definierten Testmodell der akustischen MMS realisierbar ist, soll im anschließenden Kapitel eine weitere Möglichkeit zur Umsetzung des modellbasierten Testprozesses aufgezeigt werden: Die Testfallgenerierung aus dem vorhandenen Systemmodell der akustischen MMS.

6.3 Testfallgenerierung aus dem Systemmodell der akustischen MMS

Wie bereits in Kapitel 5.1 gezeigt werden konnte, ist das Systemmodell der akustischen MMS als formales Aktivitätsdiagramm dargestellt. Im Rahmen der durchgeführten Analyse zur Testfallgenerierung aus den Testmodellen mit dem Werkzeug „MBTsuite“ konnte festgestellt werden, dass mit Unterstützung des Werkzeuganbieters auch die Testfallgenerierung aus dem Systemmodell realisierbar ist.

Aufgrund der Tatsache, dass die Softwareimplementierung der MMS von unabhängigen Systemlieferanten durchgeführt wird (Kapitel 2.2), kann zum aktuellen Zeitpunkt eine automatisierte Codegenerierung aus der Spezifikation (= Systemmodell) ausgeschlossen werden. Demnach ist es prinzipiell denkbar, Testfälle aus dem Systemmodell zu generieren, ohne dabei zwangsläufig die Unabhängigkeit von SUT und Testsystem zu untergraben (vergl. Abbildung 34). Trotz Erfüllung dieser Bedingung birgt diese

Vorgehensweise Risiken und Nachteile gegenüber einer Generierung aus erstellten Testmodellen. Der nachweislich vorhandene Effekt einer Qualitätsverbesserung des Systemmodells durch die Erstellung von Testmodellen aus selbigem geht verloren. Ohne einen expliziten Prozess zum Review des Systemmodells besteht die Gefahr Spezifikationsfehler erst spät im Entwicklungsprozess zu erkennen. Ebenfalls kritisch zu bedenken ist der Umstand, dass die Testfallgenerierung an das vorhandene Spezifikationsformat angepasst werden muss und Änderungen dieses Formats künftig nur unter Berücksichtigung der Testfallgenerierung möglich sind. Gleichzeitig muss das Systemmodell ein detailliertes Abbild des realen, zu implementierenden Systems darstellen. Die für die Strategie der Testfallgenerierung sinnvolle, teilweise notwendige Vereinfachung der Realität, welche im Testmodell umgesetzt werden kann, entfällt bei einer Generierung aus der Spezifikation. Zudem ist das vorhandene Systemmodell, wie in Kapitel 5.1 erläutert, formal inkonsistent und fehlerhaft. Um Testfälle auf dieser Basis zu generieren, sind aufwändige Korrekturen im Systemmodell notwendig.

Gleichzeitig bestehen aber auch Aspekte, welche für die Testfallgenerierung aus dem vorhandenen Systemmodell sprechen. Während bei der Verwendung von Testmodellen ein Versions- und Changemanagement-Prozess zu implementieren ist, um die Zugehörigkeit von SUT und generierten Testfällen sicherzustellen, ist diese bei einer Testfallgenerierung aus der Spezifikation direkt gegeben. Die ressourcenintensive Testmodellerstellung wird gänzlich überflüssig. Werden diese Ressourcen eingesetzt, um durch Reviews der Spezifikation frühzeitig inhaltliche und formale Fehler zu ermitteln und um

Prioritäten oder Übergangswahrscheinlichkeiten im Systemmodell zu integrieren, so ist die Vorgehensweise zielführend.

Auf Basis dieser Überlegungen wird im Folgenden gezeigt, wie die Vorgehensweise der Testfallgenerierung aus der vorhandenen Spezifikation umgesetzt werden kann. Es wird eine alternative Vorgehensweise zum vorgestellten, Testmodellorientierten Ansatz aufgezeigt, und die Gültigkeit der in Kapitel 3.2 formulierten These B untermauert.

6.3.1 Anpassung des kommerziellen Testfallgenerators

In diesem Abschnitt soll betrachtet werden, welche Anpassungen am kommerziellen Testfallgenerator „MBTsuite“ notwendig sind, um das vorhandene Systemmodell zu importieren und daraus Testfälle zu generieren. Abbildung 60 veranschaulicht die Vorgehensweise.

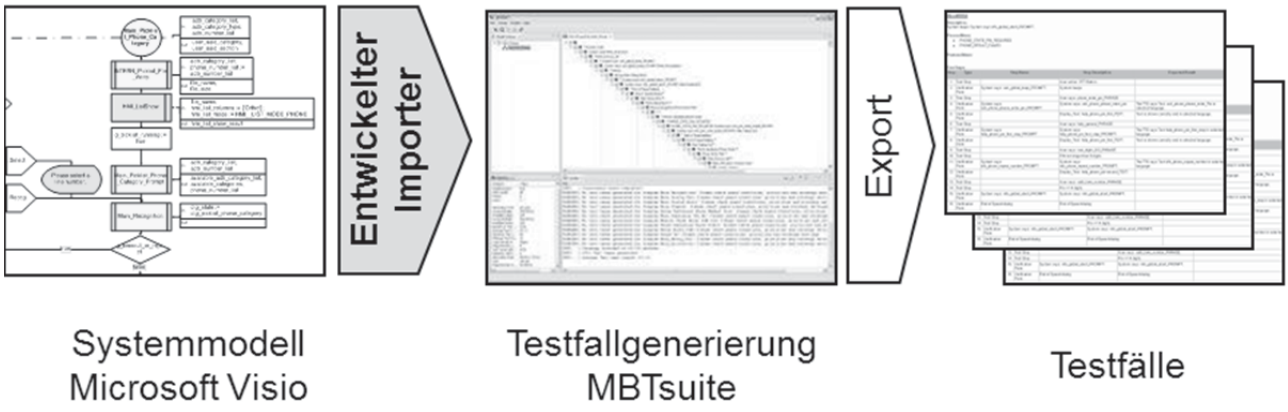
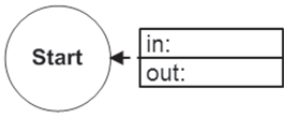
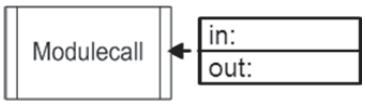


Abbildung 60: Vorgehen der Testfallgenerierung aus dem Systemmodell

Um den Import zu ermöglichen ist zu definieren, wie die verschiedenen Modellelemente (vergl. Abbildung 25, S.142) der Spezifikation von Testfallgeneratoren interpretiert werden sollen:



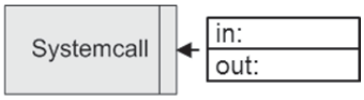
Ein Startpunkt muss in jedem Systemdiagramm vorhanden sein. Das Element hat jedoch keine sichtbare Auswirkung auf den Testfall. Inhalte müssen berücksichtigt werden, um die Verknüpfung mehrerer Module korrekt herzustellen.



Modulaufrufe werden entsprechend ihrer Funktion als Element zum Aufruf eines Unterdiagramms interpretiert. Dabei wird der Inhalt des Modulaufrufs ausgewertet, um das entsprechende Unterdiagramm zu verknüpfen. Für den Fall, dass ein entsprechendes Unterdiagramm nicht verfügbar ist, wird der Modulaufruf als Verifikationspunkt interpretiert. Auf diese Weise wird im Testfall verdeutlicht, dass sich bei Verfügbarkeit des entsprechenden Moduls weitere Dialogteile anschließen würden. Das „Nichtvorhandensein“ von Systemdiagrammen ist als „Use Case“ zu berücksichtigen, da es zielführend und notwendig sein kann, nur ausgewählte Systemteile für die Testfallgenerierung heranzuziehen. Hat ein Modulaufruf keine ausgehende Kante, wird dieser als Endpunkt interpretiert.

Benutzeraktionen werden als Knoten im Diagramm importiert und als „Teststep“ interpretiert. Die mit dem Element beschriebene Nutzeraktion (<content of Element>, hier: PTT) wird im Testfall dargestellt als „user_event_[content of Element]“.

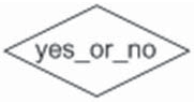
Spracheingaben werden als Knoten im Diagramm importiert und als „Teststep“ interpretiert. Die mit dem Elemente beschriebene, akustische Systemeingabe (<content of Element PHRASE>) wird im Testfall als „user_phrase_[content of Element PHRASE]“ dargestellt.



Systemaktionen werden als Knoten im Diagramm importiert und als VP interpretiert. Damit ist sichergestellt, dass Systemaktivitäten im Testfall verifizierbar sind. Die mit dem Element beschriebene Systemaktion (<content of Element SC>), wird im Testfall als „system_event_[content of Element SC]“ dargestellt.

Systemausgaben werden als Knoten im Diagramm importiert und als VP interpretiert. Inhalte der akustischen Systemausgabe (<content of Element PROMPT>) werden im Testfall als „system_prompt_ [content of Element PROMPT]“, Inhalte der optischen Systemausgabe (<content of Element TEXT>) als „system_text_ [content of Element TEXT]“ dargestellt.

Variablenzuweisungen werden als Knoten im Diagramm importiert. Der Inhalt der Variablenzuweisung wird interpretiert. Dabei werden formale Inkonsistenzen so weit möglich berücksichtigt und im Importvorgang korrigiert. Z. B. werden unterschiedliche Schreibweisen Boolescher Variablen (True, true, „true“, FALSE, False...) erkannt und entsprechend interpretiert.



Entscheider werden als Knoten im Diagramm importiert. Wenn bei der Entscheidung Variablen abgefragt werden, welche zuvor im Diagramm zugewiesen wurden, werden die Werte an den vom Entscheider ausgehenden Kanten als „Guard Conditions“ interpretiert. Ein Sonderfall ergibt sich aus der Tatsache, dass im System-modell die für die Testdurchführung notwendigen Vorbedingungen (Preconditions) nicht explizit modelliert sind. Implizit sind die Preconditions jedoch vorhanden. Auf Basis

dieser Gegebenheiten ist es möglich, folgende Regel zu definieren, um die Preconditions als solche zu erkennen: Wird eine Variable abgefragt, welche zuvor *nicht* zugewiesen wurde, ist diese (sowie die zugehörigen Werte) als „Precondition“ zu interpretieren. Die vom Entscheider ausgehenden Kanten werden in diesem Fall beliebig durchlaufen (keine „Guard Conditions“).

Sprung-Marken werden importiert und zu einer Transition verbunden. Sprünge dürfen nur innerhalb eines Diagramms stattfinden und müssen eindeutig bezeichnet sein, um eine fehlerfreie Interpretation zu gewährleisten.

Transitionen werden als Kante importiert. Vorhandene Prioritäten können interpretiert und in der Testfallgenerierung genutzt werden.

Im Systemmodell ist die Möglichkeit modelliert, dass der Nutzer kein Kommando einspricht und der Spracherkenner die zeitliche Begrenzung der Erkennung erreicht (engl. „timeout“). Für diesen Fall gibt es die Boolesche Variable „timeout“. Wird die zeitliche Begrenzung erreicht (timeout = True), erfolgt eine Systemreaktion mit einer optischen und akustische Hilfe. Für diesen speziellen Fall besteht die Notwendigkeit einer Sonderbehandlung in der Testfallgenerierung. Wird ein Entscheider und die folgende Transition „timeout = True“ durchlaufen, ist dies als „Teststep“ mit dem Inhalt „user_phrase_[timeout]“ zu interpretieren. Ohne Berücksichtigung dieser (nicht stattfindenden) Nutzeraktion würden im Testfall zwei Systemreaktionen aufeinander folgen, ohne dass die für die zweite Systemreaktion ursächliche Nutzeraktivität berücksichtigt würde.

Mit Umsetzung dieser Anforderungen konnte gezeigt werden, dass der Import vorhandener Systemmodelle in den Testfallgeneratoren realisiert werden kann.

6.3.2 Anpassung des Systemmodells der akustischen MMS

Zum Beweis der Prinziptauglichkeit des systemmodellorientierten Ansatzes finden dieselben, repräsentativ ausgewählten Systemteile der akustischen MMS Anwendung, welche bereits als Testmodell umgesetzt wurden. Neben den dabei identifizierten, formalen Fehlern (Kapitel 5.2.2) konnten durch die statische Prüfung beim Import des Modells in den Testfallgenerator weitere Fehler identifiziert werden:

- Zwei rekursive Diagrammaufrufe (vergl. Kapitel 6.2.1) konnten festgestellt werden. Um eine effiziente Testfallgenerierung zu ermöglichen, sind die betroffenen Modellelemente ohne Rekursion zu modellieren.
- Verwendete Notationselemente des Systemmodells entsprechen nicht der vorgegebenen Syntax. Verschiedene Transitionen und Systemaktivitäten sind nicht in der jeweils relevanten Notationsvorlage dargestellt, sondern unterscheiden sich formal, ohne dass dies optisch erkennbar ist.
- In vielen Fällen findet eine Abfrage von Variablen (bzw. deren Werten) statt, ohne dass diese vorher entsprechend zugewiesen wurden. Ohne die korrekte Modellierung der entsprechenden Zuweisung können fehlerhafte Testfälle generiert werden. Dabei ist zu beachten, dass die implizit modellierten Preconditions vor ihrer Abfrage nicht zugewiesen

werden (vergl. Kapitel 6.3.1). Die korrekte Wertezuweisung von Variablen ist im Systemmodell entsprechend zu korrigieren.

- Notwendige Variablen werden hinsichtlich ihrer globalen oder lokalen Gültigkeit nicht konsistent verwendet oder falsch modelliert.

Der zuletzt aufgeführte Punkt ist in Abbildung 61 dargestellt und im Folgenden erläutert.

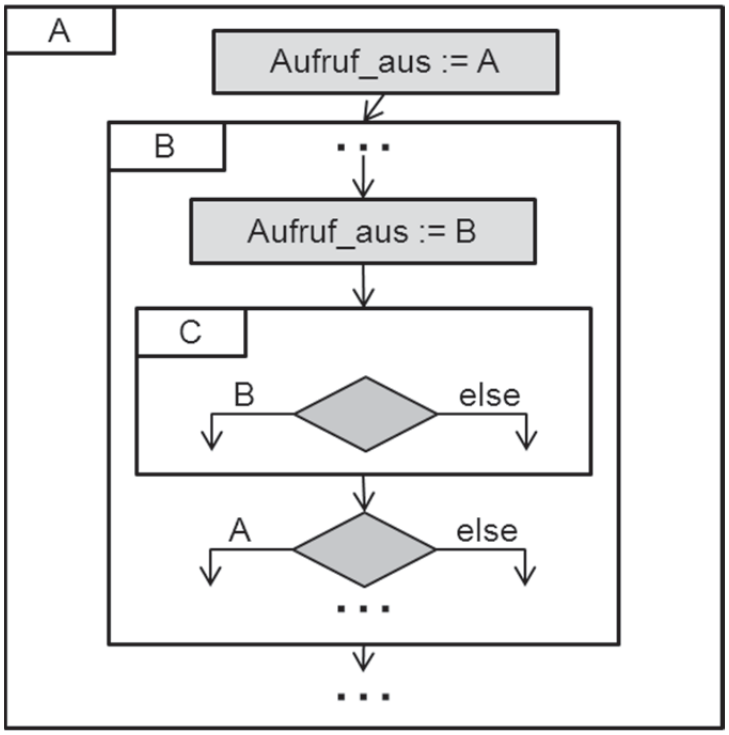


Abbildung 61: Inkonsistente Verwendung von Variablen

Vor Aufruf des Diagramms (B) erfolgt in Diagramm (A) die Variablenzuweisung „Aufruf_aus := A“. In Diagramm (B) ist, bevor die Variablenabfrage formal korrekt stattfindet, die Variablenzuweisung „Aufruf_aus := B“ modelliert, um eine entsprechende Variablenabfrage in Diagramm (C) zu ermöglichen. Dies hat zur Folge, dass die Variablenabfrage in Diagramm (B)

nicht wie beabsichtigt möglich ist. Hier verwendet der Spezifikateur die Variable lokal und berücksichtigt nicht, dass in den vorhandenen Unterdiagrammen derselben Variablen bereits ein neuer Wert zugewiesen wurde. Die Variable „Aufruf_aus“ muss aber global verwendet werden, um eine Variablenabfrage in entsprechenden Unterdiagrammen zu ermöglichen. Der menschliche Leser des Systemmodells vermag die eigentliche Absicht des Spezifikateurs korrekt zu interpretieren, in der Testfallgenerierung führt die Modellierung jedoch zu Fehlern. Es ist zu beachten, dass die Verwendung globaler (oder lokaler) Variablen, deren Zuweisungen und Abfragen formal korrekt und durchgängig zu gestalten ist.

Es zeigt sich, dass im vorhandenen Systemmodell einige Fehler und Inkonsistenzen vorhanden sind, deren Behebung notwendig ist, um eine fehlerfreie Testfallgenerierung zu ermöglichen. Die als fehlerhaft identifizierten Dialogelemente konnten zum Prinzipnachweis des Vorgehens korrigiert werden. Soll die gesamte Spezifikation der akustischen MMS entsprechend überarbeitet werden, sind hierfür ausreichend Ressourcen einzuplanen. Insbesondere die Korrektur rekursiver Aufrufe und fehlerhaft verwendeter Variablen kann nicht automatisiert durchgeführt werden sondern erfordert die kognitive Leistung und Kreativität eines Modellierers. Werden diese Aufwendungen erbracht, wird die allgemeine Qualität der Spezifikation verbessert und die Testfallgenerierung aus selbiger realisierbar.

Um weitere strategische Möglichkeiten in der Testfallgenerierung zu schaffen und eine effiziente sowie zielorientierte Vorgehensweise zu etablieren, sind Prioritäten oder Übergangswahrscheinlichkeiten im zu Grunde liegenden Modell zielführend (siehe Kapitel 6.2.2). Zur Realisierung dieser, sind

Anforderungen und Regeln an die Modellierung formuliert, welche die Ergänzung von Prioritäten an Transitionen (Kanten) im Modell ermöglichen. Dabei ist zu berücksichtigen, dass die Prioritäten bei Weitergabe der Spezifikation an Systemlieferanten nicht sichtbar sind, um entsprechende Rückschlüsse auf die verwendeten Teststrategien auszuschließen. Für den Testmanager und den Testfallgenerator müssen die Prioritäten sichtbar bzw. interpretierbar sein. Es konnte gezeigt werden, dass die entsprechenden Anforderungen realisierbar sind. Folglich kann die zum Ziel gesetzte strategische Testfallgenerierung auf Basis modellierter Prioritäten umgesetzt werden.

Nach dem zunächst bewiesen werden konnte, dass die Testfallgenerierung aus dem definierten Testmodell der akustischen MMS realisierbar ist (Kapitel 6.2), konnte ebenso bewiesen werden, dass die Testfallgenerierung aus dem vorhandenen Systemmodell (der Spezifikation) stattfinden kann (Kapitel 6.3). Diese Vorgehensweise hat Vorteile, da die ressourcenintensive Erstellung der Testmodelle entfällt, erfordert aber gleichzeitig die Investition von Ressourcen zur formalen Korrektur der verwendeten Systemmodelle. Unabhängig davon, welche der beiden möglichen Vorgehensweisen schließlich Verwendung findet, ist bewiesen, dass die Testfallgenerierung aus dem Modell einer akustischen MMS mit einem kommerziellen Werkzeug realisiert werden kann. Die Gültigkeit der in Kapitel 3.2 formulierten These B ist bewiesen und ein weiterer wichtiger Schritt zur Umsetzung des MBT vollzogen. Im Folgenden ist zu betrachten, wie die generierten Testfälle ausgeführt werden können. Hierbei gewinnt der virtuelle Tester an Gestalt.

6.4 Zusammenfassung

Nachdem in Kapitel 5 der Beweis erbracht wurde, dass die softwarebasierte MMS eines Infotainmentsystems in einem Standardformat formal modelliert werden kann, wird in Kapitel 6 der Beweis erbracht, dass auf Basis der erstellten, formalen Modelle durch Verwendung eines kommerziellen Werkzeugs Testfälle generiert werden können (vergl. Kapitel 3.2, These B).

In Kapitel 6.1 wurden insgesamt sechs verschiedene Werkzeuge zur Testfallgenerierung evaluiert. Hierfür wurden zunächst Anforderungen definiert, deren Erfüllung in einer folgenden Analyse und Bewertung ermittelt wird. Um eine aussagekräftige Evaluation zu ermöglichen, wurden alle Werkzeuge einem Praxistest unterzogen. Datenbasis waren hierbei die in Kapitel 5 erstellten Testmodelle. Das Werkzeug „MBTsuite“ der Firma „sepp.med“ wurde schließlich als jenes mit der höchsten Anforderungsüberdeckung identifiziert und ausgewählt, um die Testfallgenerierung zu realisieren.

In Kapitel 6.2 wurden die Möglichkeiten der Testfallgenerierung aus den definierten Testmodellen analysiert und mögliche Generierungsstrategien mit dem Ziel bewertet, Wege einer effizienten Vorgehensweise auszuarbeiten. Dabei wurde deutlich, dass aufgrund unterschiedlicher Modelleigenschaften individuelle Generierungsstrategien notwendig sind. Die Option der vollständigen Pfadüberdeckung führt zu einer großen Menge an Testfällen, insbesondere für die Modelle der haptischen MMS. Neben der Anzahl an Testfällen wachsen auch die Generierungszeiten mit zunehmender Testtiefe exponentiell an. Während für das Modell der akustischen MMS eine

vollständige Knoten bzw. Kantenabdeckung realisierbar ist, wird für die Testfallgenerierung aus den Modellen der haptischen MMS eine randomisierte Generierungsstrategie empfohlen. In allen Fällen empfiehlt sich die Ergänzung von Übergangswahrscheinlichkeiten oder Prioritäten im Modell, um Strategien zur zielorientierten, individuellen Testfallgenerierung realisieren zu können.

Nachdem durch Analyse und Verwendung des Werkzeugs „MBTsuite“ festgestellt werden konnte, dass auch eine Testfallgenerierung aus dem vorhandenen Systemmodell der akustischen MMS umgesetzt werden kann, wurden in Kapitel 6.3 die konkreten Vor- und Nachteile der alternativen Vorgehensweisen erläutert, bevor notwendige Anpassungen an Testfallgenerator und Systemmodell aufgezeigt wurden, um einen Import der Modelle mit anschließender Testfallgenerierung umzusetzen.

Es konnte gezeigt werden, dass die Generierung von Testfällen sowohl aus dem erstellten Testmodell, als auch aus dem vorhandenen Systemmodell der akustischen MMS durch Verwendung des kommerziellen Werkzeugs „MBTsuite“ umgesetzt werden kann. Die Qualität der generierten Testfälle basiert dabei auf der Qualität des zu Grunde liegenden Modells.

7 Automatisierte Testdurchführung – Der virtuelle Tester

In Kapitel 6 konnte bewiesen werden, dass Testfälle zur Qualitätsabsicherung der akustischen MMS automatisch generiert werden können. Diese können ohne Schwierigkeit in der manuellen Testdurchführung verwendet werden. Der Tester kann anhand des erstellten Testfalls die notwendigen Vorbedingungen für einen Test, die Testschritte und die erwartete Systemreaktion ablesen. Stimmt die tatsächliche Systemreaktion mit der beschriebenen nicht überein, ist ein Fehler identifiziert.

Die Möglichkeit der Testfallgenerierung führen trotz zielführender Teststrategien meist zu einer großen Menge an Testfällen. Aufgrund der kurzen Entwicklungszyklen ist dies eine besondere Herausforderung. Die manuelle Testdurchführung stößt hier schnell an Ihre Grenzen. In diesem Kapitel wird betrachtet, wie die Durchführung der generierten Testfälle möglichst effizient stattfinden kann. Zum Beweis der in Kapitel 3.2 formulierten These C wird eine Möglichkeit der automatisierten Testdurchführung entwickelt. Diese hat das Ziel, die Dialoge der akustischen MMS durch einen „virtuellen Tester“ zu qualifizieren. Hierfür sollen zunächst einige Rahmenbedingungen der Testdurchführung betrachtet werden.

7.1 Die Testumgebung zum automatisierten Test

Aktivitäten zum Test der softwarebasierten MMS finden insbesondere in den frühen Entwicklungsphasen, hauptsächlich im Labor und nicht im Fahrzeug

statt. Die Labore sind mit der notwendigen Infrastruktur ausgestattet, um verschiedene Hardwareelemente und Funktionen des Infotainmentsystems zu prüfen. Die Umgebung zum Test der akustischen MMS ist in Abbildung 62 schematisch dargestellt.

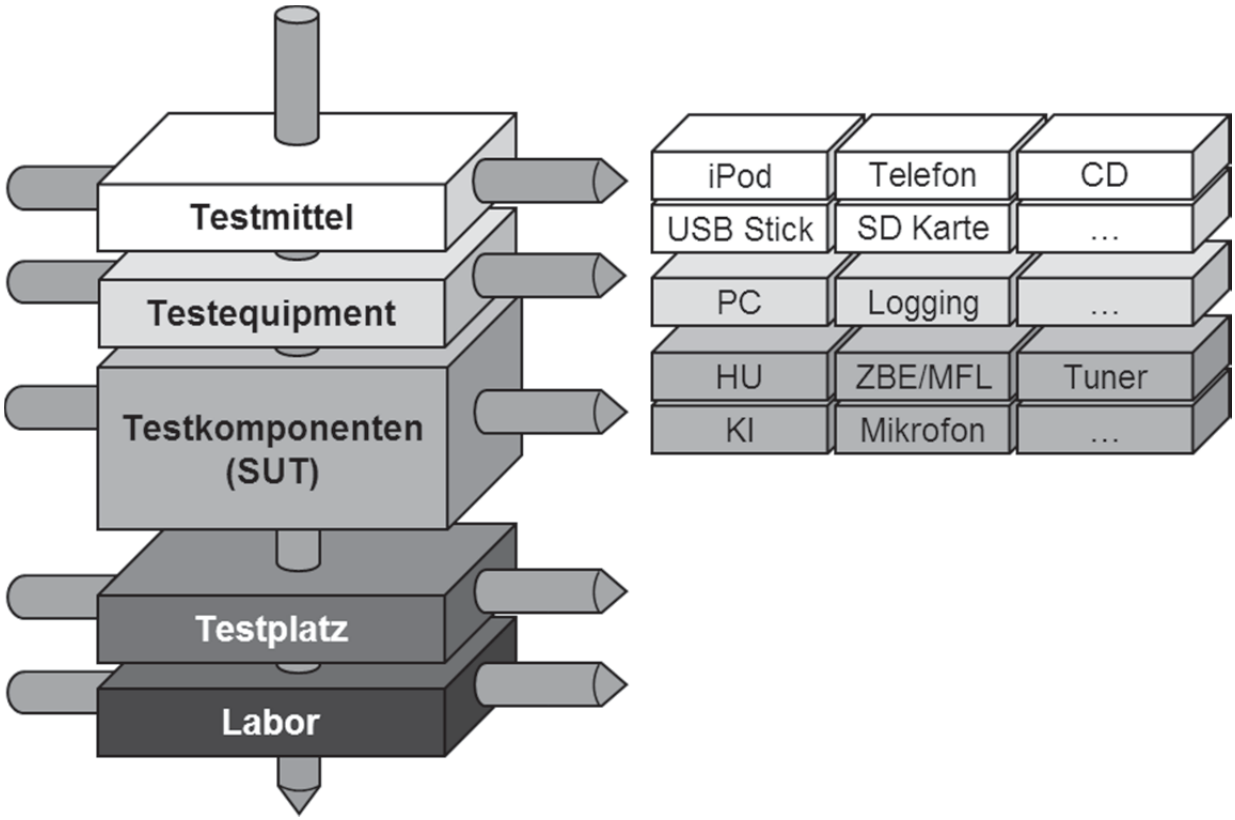


Abbildung 62: Elemente der Umgebung zum Test einer SW-MMS

Die zentralen Bausteine in der Testdurchführung sind die *Testkomponenten*. Hierzu gehört die HU mit der implementierten akustischen MMS als SUT, dass KI, die haptischen Stellteile zur Bedienung (ZBE und MFL-Tasten), das Display, Mikrofone zur Erfassung akustischer Eingaben und Lautsprecher zur Wiedergabe akustischer Signale. Auch Zündschloss, Steuergeräte, Tuner, Verstärker etc. können für den Test des Infotainmentsystems erforderlich sein und sind den Testkomponenten zuzuordnen.

Neben diesen Elementen ist *Testequipment* notwendig, um Tests effizient durchzuführen. Im einfachsten Fall werden ausschließlich Datenlogger verwendet, welche die elektronischen Signale und Daten des CAN-Bus[4] und MOST-Bus[5] aufzeichnen, um im Fehlerfall über eine Datenbasis zur Fehleranalyse zu verfügen. Für die automatische Ausführung von Testfällen sind zudem Computer zur Steuerung der Durchführung, Hardwareelemente zur Einspeisung der Steuerungssignale in CAN-Bus, MOST-Ring etc. erforderlich.

All diese Testkomponenten und -equipment werden möglichst kompakt in einer „Testbox“ zusammengefasst, an welcher der Tester das SUT wie im Fahrzeug nutzen kann (Abbildung 63).

Neben Testkomponenten und Testequipment sind für die Testdurchführung zudem oft *Testmittel* notwendig. Dabei handelt es sich um Hardware, welche variabel und optional mit dem Infotainmentsystem verbunden werden kann. Die Nutzung der implementierten Telefonfunktion ist z. B. nur in Verbindung mit einem Telefon möglich. Auch die verschiedenen Audio- und Videofunktionen erfordern den Anschluss von entsprechenden Speichermedien wie CD, DVD, SD-Karte, USB Speicher oder MP3-Player. Die verschiedenen

[4] Der CAN-Bus (Controller Area Network) ist ein standardisiertes, asynchrones, serielles Bussystem zur Vernetzung von Steuergeräten. Für weitere Informationen sei auf entsprechende Fachliteratur wie [Zim10] verwiesen.

[5] Der MOST-Bus (Media Oriented Systems Transport) ist ein Ringnetzwerk und serielles Bussystem zur Übertragung von Audio-, Video-, Sprach- und Datensignalen. Für weitere Informationen sei auf entsprechende Fachliteratur wie [Zim10] verwiesen.

Medien sind vom Tester in Abhängigkeit der Vorbedingungen für einen Testfall entsprechend einzusetzen.

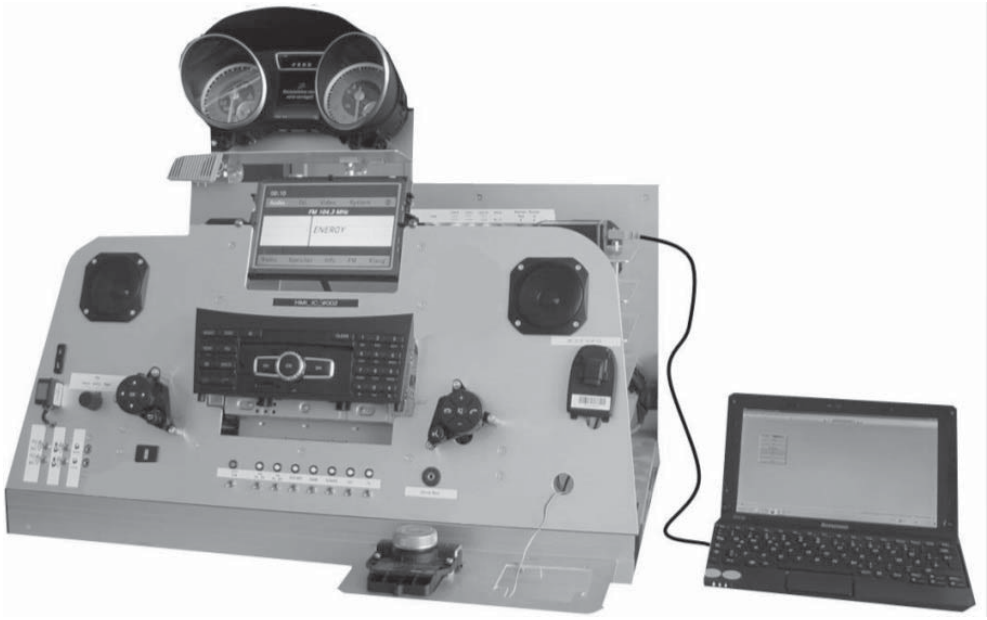


Abbildung 63: SUT in „Testbox“

In der manuellen Testdurchführung können haptische Eingaben über die vorhandenen Bedienelemente erfolgen und akustische Eingaben werden vom Mikrofon erfasst. Entsprechend werden die akustischen Systemreaktionen über die Lautsprecher und optische Systemreaktionen auf dem vorhandenen Display ausgegeben. Diese Ausgaben können vom Menschen leicht verifiziert werden. Komplexer stellt sich die Situation in der automatisierten Testdurchführung dar. Die Entwicklung und der Einsatz von mechanischen Robotern, welche automatisch vorbestimmte, haptische Aktivitäten ausführen, sind ressourcenintensiv und teuer. Aus diesem Grund bietet sich die Verwendung einer expliziten Testschnittstelle zum Infotainmentsystem an. Die Anforderungen an diese Schnittstelle werden vom Auftraggeber formuliert und vom Systemlieferanten bei der Implementierung des SUT umgesetzt.

Die Verwendung einer solchen Testschnittstelle hat verschiedene Vorteile. Es besteht die Möglichkeit, Signale an die HU zu schicken, welche eine bestimmte haptische Aktivität des Nutzers beschreiben (z. B. ZBE nach links gedrückt) und die zugehörige Systemreaktion auslösen (z. B. Cursor nach links). Des Weiteren kann für die Testschnittstelle definiert werden, dass interne Daten der HU oder Systemreaktionen, welche von außen nicht sichtbar sind, an die Testumgebung übertragen werden. Kurzum, über eine Testschnittstelle lässt sich das SUT bedienen, können Systemreaktionen erfasst werden und ein automatischer Test wird möglich. Eine solche Schnittstelle wird z. B. im automatischen Test des ASR genutzt (siehe Kapitel 7.2).

Die Implementierung einer expliziten Testschnittstelle hat aber auch Nachteile und birgt erhebliche Risiken. Zum einen ist die Schnittstelle nicht kundenrelevant und deren Implementierung (insbesondere bei erhöhtem Zeitdruck) nicht priorisiert. Dies hat zur Folge, dass die Funktionalitäten der Schnittstelle tendenziell spät geliefert werden oder möglicherweise ganz entfallen. Sind die Testaktivitäten abhängig von deren Implementierung, wird die Durchführung möglicherweise verzögert oder nicht stattfinden können, obwohl die zu testenden Systemfunktionalitäten bereits verfügbar sind. Die Einführung des MBT-Prozesses soll unter anderem aber dazu führen, Fehler im SUT möglichst früh im Entwicklungsprozess identifiziert zu können. Es besteht folglich das Risiko, dass trotz Modellierung, Testfallgenerierung und frühzeitigem Aufbau der Testumgebung die automatische Durchführung der Tests nicht stattfinden kann, weil die erforderliche Testschnittstelle nicht verfügbar ist.

Ein weiteres Risiko ergibt sich aus der Softwarearchitektur und Einbindung der Testschnittstelle im SUT. Um die Bedienung des Infotainmentsystems über die Testschnittstelle zu ermöglichen, sind die entsprechenden Anforderungen an die Funktionalität der Schnittstelle zu definieren und im SUT zu implementieren. Erhält die Testschnittstelle von der Testumgebung z. B. das Signal (X), welches definiert ist, um die Nutzeraktivität „ZBE nach links“ zu beschreiben, übersetzt die implementierte Schnittstelle das Signal, um die in der Black-Box des SUT verwendeten Parameter anzusteuern. Gleicher Vorgang gilt für die Systemausgaben. Eine interne Systemaktivität wird von der Testschnittstelle in ein definiertes Signal (Y) übersetzt und an die Testumgebung ausgegeben, welche selbiges mit einer Systemreaktion („Cursor nach links“) verknüpft. Abbildung 64 stellt das Funktionsprinzip schematisch dar.

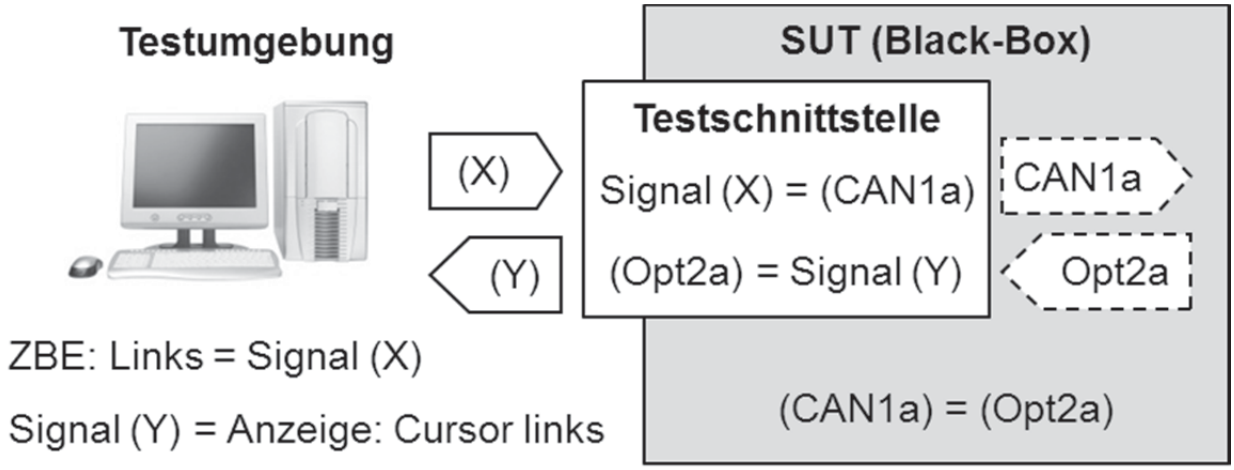


Abbildung 64: Schematisches Prinzip einer Testschnittstelle in einer „Black-Box“

Diese Schnittstellenarchitektur zum Test einer Black-Box führt zu einem prinzipiellen Problem. Werden Testaktivitäten durchgeführt und Fehler identifiziert, kann niemals eindeutig festgestellt werden, ob der Fehler

tatsächlich im SUT ursächlich ist oder die Implementierung der Testschnittstelle fehlerhaft ist.

Für die Effizienz der Testaktivitäten ist zudem kritisch, dass die Implementierung der Testschnittstelle und deren Funktionalität vom jeweiligen Systemlieferanten abhängig ist. Die Ansteuerung des SUT über die Testschnittstelle ist folglich stark projektabhängig, was eine ressourcenintensive Anpassung der Testumgebung erforderlich macht. Ziel sollte jedoch sein, projektübergreifend dieselbe Testumgebung verwenden zu können.

Bis jetzt wurden die Aktivitäten zum Test der akustischen MMS mehrheitlich über solche Testschnittstellen durchgeführt. Um die erläuterten Risiken und Nachteile in deren Verwendung zu umgehen, wurde in These C der Ansatz formuliert, bei der Testdurchführung die akustische MMS selbst als Testschnittstelle zu verwenden und einen „virtuellen Tester“ zu implementieren, ohne auf eine zusätzliche Schnittstelle zurückgreifen zu müssen (Kapitel 3.2).

Im folgenden Kapitel werden zunächst automatisierte Funktionstests der akustischen MMS betrachtet und in eine neue Testumgebung transferiert. In Kapitel 7.3 soll aufbauend auf diesen Erkenntnissen eine Testumgebung zum Beweis der These C entwickelt werden.

7.2 Automatischer Funktionstest der akustischen MMS

Im Gegensatz zum automatischen Dialogtest der akustischen MMS versteht sich der automatische Funktionstest als Test einzelner „Features“ oder Funktionen des SDS. Die Durchführung von Funktionstests basiert meist auf verhältnismäßig einfachen abstrakten Testfällen, welche iterativ durchgeführt

werden und eine große Anzahl verschiedener, konkreter Testdaten verwenden. Im Dialogtest hingegen werden viele verschiedene, teils komplexe, abstrakte Testfälle angewandt. Dabei ist eine Varianz der konkreten Eingabewerte kaum notwendig, womit auch die iterative Vorgehensweise von geringerer Bedeutung ist (Kapitel 7.3).

Beispiel für einen automatisierten, funktionalen Test ist die Qualifizierung des Spracherkenners (ASR). Der ASR interpretiert die verbalen Spracheingaben des Systemnutzers und leitet das Ergebnis an den Dialogmanager des SDS weiter (Abbildung 4, S. 36). Unabhängig von strukturellen und inhaltlichen Fehlern des Sprachdialogs, welche in einem fehlerhaften Dialogmanager begründet sind, kann es zu einer fehlerhaften Spracherkennung durch den ASR kommen, was in der Folge zu einem nicht erwarteten Dialogverhalten führt. Die Ursache für den auf diese Weise fehlgeschlagenen Dialog liegt in diesem Fall nicht im Dialogmanager begründet, sondern in der Spracherkennung.

An die Qualität der Spracherkennung sind Anforderungen gestellt, welche im Rahmen eines automatisierten Funktionstests verifiziert werden können. Der abstrakte Testfall für diesen Test kann dabei, wie in Abbildung 65 dargestellt, umgesetzt werden.

Dieser abstrakte Testfall ist von einfacher Gestalt. Dem SUT wird ein definiertes Eingabekommando (X) vorgesprochen, woraufhin der ASR das Ergebnis der Erkennung aus- bzw. an den Dialogmanager weitergibt. Die Testumgebung kann den Ausgabewert (X_1) mit dem Eingabewert (X) vergleichen und eine korrekte oder fehlerhafte Spracherkennung feststellen.

Im Anschluss kann dasselbe Kommando (X) vom nächsten Sprecher verwendet werden.

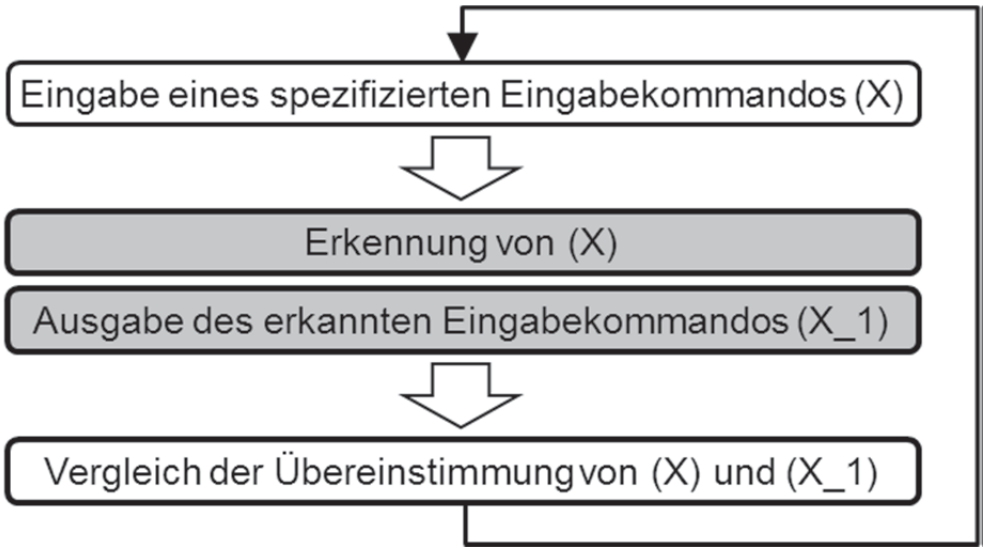


Abbildung 65: Abstrakter Testfall zum automatischen Test eines ASR

In Abbildung 66 sind die Schnittstellen zwischen Test-umgebung und ASR schematisch dargestellt.

Zur Eingabe eines Sprachkommandos kann die vorhandene akustische Schnittstelle genutzt werden, indem aufgezeichnete Sprachaufnahmen verschiedener menschlicher Sprecher abgespielt werden. Der vom ASR erkannte Eingabewert wird an den Dialogmanager weitergereicht und kann über eine zweite Schnittstelle abgegriffen werden. Über diese können die Ausgabewerte des ASR direkt ausgelesen und mit den entsprechenden Sollwerten verglichen werden. Der abstrakte Testfall kann nun mit beliebig vielen unterschiedlichen Eingabewerten (X) zyklisch und automatisch durchgeführt werden. Es ist lediglich einmalig erforderlich, ein möglichst umfangreiches „Testset“ in verschiedenen Sprachen zu erstellen.

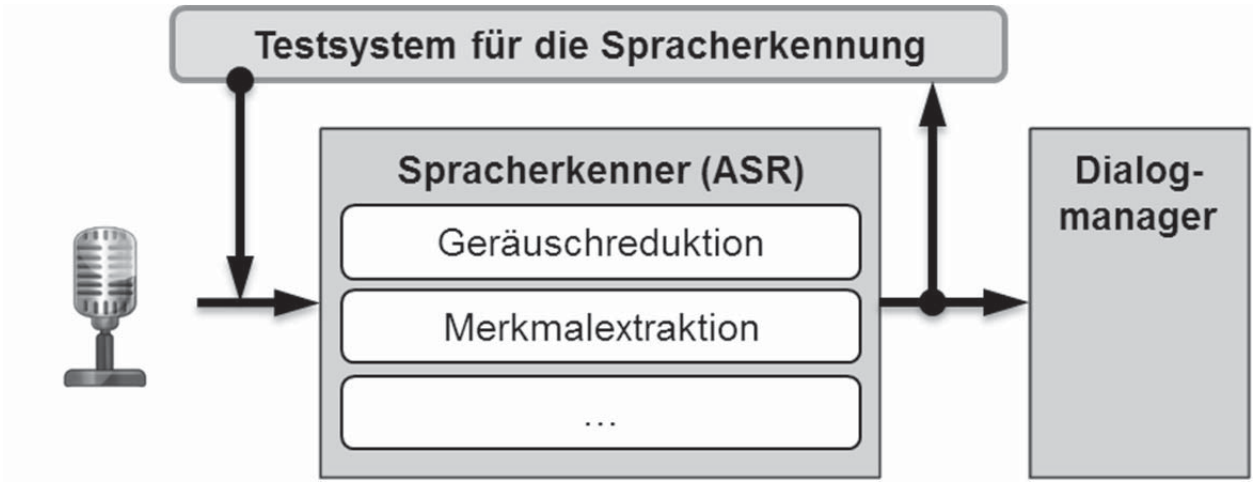


Abbildung 66: Testschnittstellen zur automatischen Verifikation eines ASR

Die hier beschriebene Vorgehensweise zum Test des ASR verwendet eine explizite Testschnittstelle (Kapitel 7.1). Wie beschrieben birgt die Verwendung einer solchen Testschnittstelle erhebliche Risiken und negative Eigenschaften. Um in einem ersten Schritt zu zeigen, dass Funktionstests der akustischen MMS auch ohne eine solche Testschnittstelle durchgeführt werden können, wird der Test einer ähnlichen Funktionalität des SDS in einer neuen Testumgebung implementiert.

Aktuelle Sprachdialogsysteme ermöglichen dem Benutzer die Zieleingabe zur Navigationsführung in wenigen Schritten. Die akustische Eingabe von Ort- und Straßennamen wird dabei nicht mehr sequenziell, sondern in einem Bedienschritt durchgeführt. Dieser „Use Case“ ist auf seine funktionale Qualität hin zu verifizieren. Dabei ähnelt der Test stark dem beschriebenen Test des ASR. Auch hier wird immer derselbe, abstrakte Testfall zur Zieleingabe iterativ durchgeführt, wobei unterschiedliche Testdaten (Ort-Straße-Kombinationen) Verwendung finden. In Abbildung 67 ist der abstrakte Testfall für den Test schematisch dargestellt.

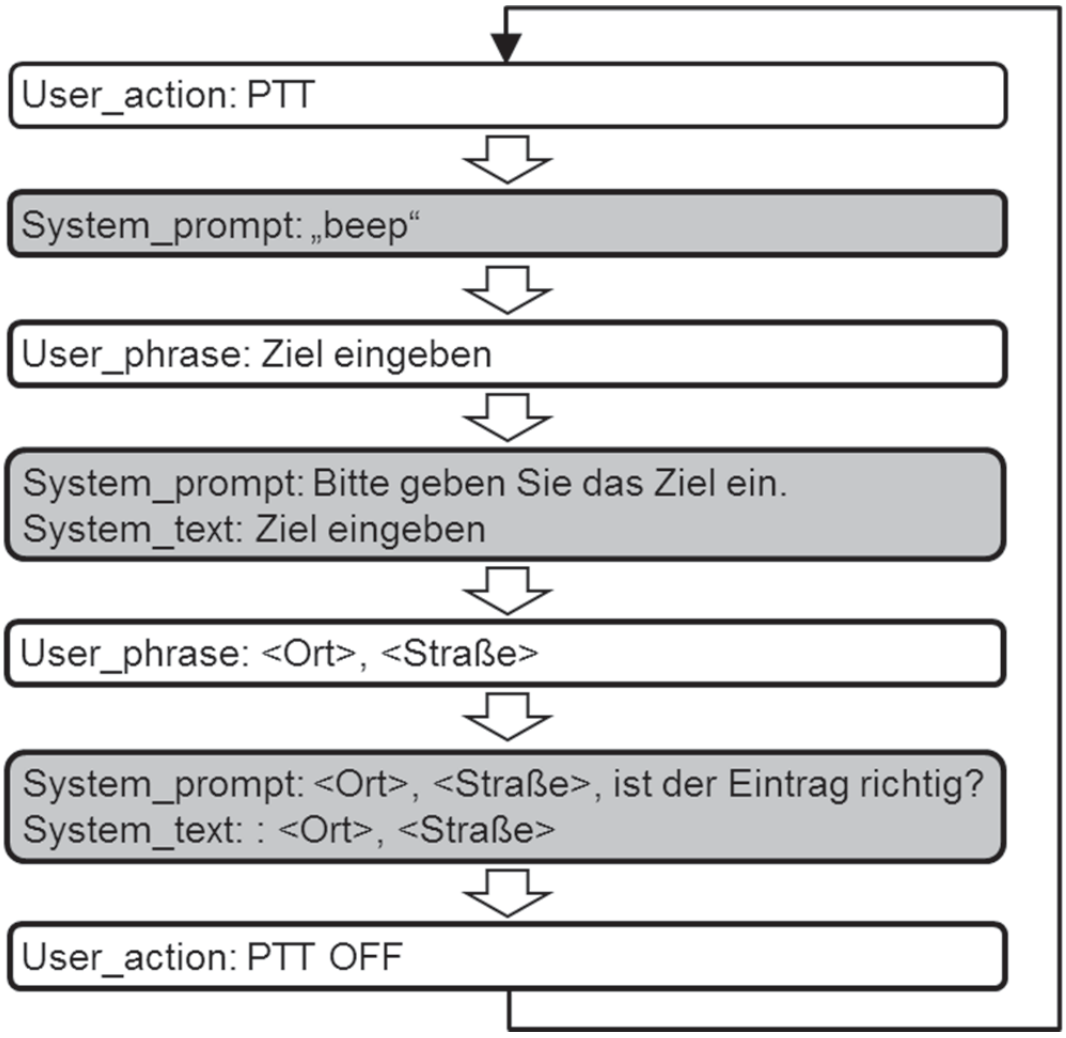


Abbildung 67: Abstrakter Testfall zum automatischen Test einer Zieleingabe

Mit dem Ziel, die automatische Testdurchführung ohne die Verwendung einer Testschnittstelle zu realisieren, bildet der dargestellte Testfall alle notwendigen Bedienschritte (TS) und Systemreaktion (VPs) ab, um das Ziel akustisch einzugeben und das Erkennungsergebnis zu verifizieren. Nach akustischer und optischer Ausgabe der erkannten Ort-Straße-Kombination wird der Testfall beendet und mit neuen Eingabedaten wiederholt.

Für verschiedene Testaktivitäten anderer Funktionalitäten steht in der Telematik eine Testumgebung zur Verfügung, welche bereits einige zentrale

Funktionen bereitstellt, um den Testfall ohne die bislang notwendige Testschnittstelle durchzuführen. Die haptischen Benutzereingaben (PTT und PTT OFF) können in dieser Umgebung als Signal auf dem CAN-Bus der HU geschickt werden. Die Testumgebung ist in der Lage, die entsprechenden Signale zu simulieren und über ein Bus-Interface in den CAN-Bus einzuspeisen.

Die akustischen Benutzereingaben bestehen aus Audioaufnahmen verschiedener Ort-Straße-Kombinationen und Sprecher. Diese Aufnahmen können, analog zum Test des ASR, von der Testumgebung abgespielt und über den Audioeingang der HU eingespeist werden.

Im Gegensatz zur Eingabe der Benutzeraktivitäten ist die Verifikation der Systemreaktion eine größere Herausforderung. Akustische Systemausgaben können von der Testumgebung bisher nicht erkannt und verifiziert werden. Für optische Systemausgaben besteht jedoch eine Lösung. Das von der HU an das Display gesendete Videosignal wird abgegriffen und von einer Software zur optischen Zeichenerkennung (engl. „Optical Character Recognition“, OCR) interpretiert. Da der Aufbau des Bildschirminhalts bekannt ist, können die entsprechend relevanten Textelemente gezielt erkannt und verwendet werden.

Wie in Abbildung 67 festgestellt werden kann, findet bei zwei der drei vorkommenden Systemreaktionen neben der akustischen auch eine optische Rückmeldung statt. Über die OCR können diese Systemrückmeldungen verifiziert werden. Die akustische Ausgabe des „beep“, nach Betätigung des PTT, ist in jedem Durchgang unverändert. Die Dauer dieser Ausgabe ist

bekannt und kann folglich berücksichtigt werden. Eine Verifikation, ob die akustische Ausgabe tatsächlich stattgefunden hat, ist bei der Zielsetzung des Testfalls nicht von zentraler Bedeutung. Vor diesem Hintergrund kann diese akustische Ausgabe angenommen werden, um den Testfall anschließend fortzusetzen. Abbildung 68 zeigt die verwendeten Schnittstellen, um den Test ohne Testschnittstelle durchzuführen.

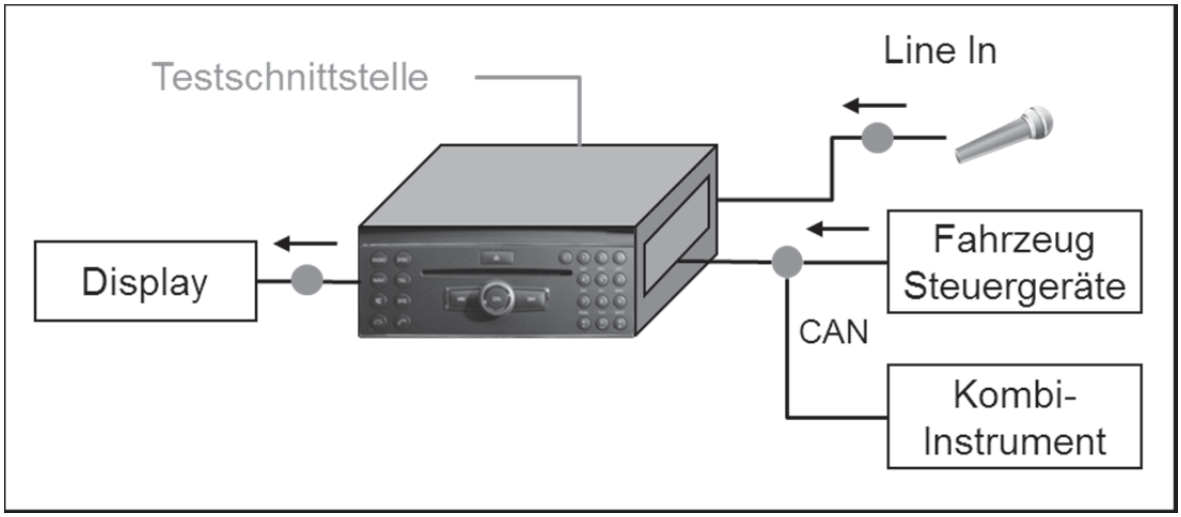


Abbildung 68: Testschnittstellen zum automatischen Test einer Zieleingabe

In einer praktischen Umsetzung konnte gezeigt werden, dass die Vorgehensweise nicht nur theoretisch möglich ist. Mit über 240 verschiedenen Testdaten wurden mehrere tausend Iterationen ausgeführt und wertvolle Erkenntnisse zum Verhalten des ASR gewonnen. Die Prüfung funktionaler Inhalte der akustischen MMS ist demzufolge auch ohne die Verwendung einer expliziten Testschnittstelle realisierbar.

Wie sieht es aber aus, wenn nicht einzelne Funktionen des SDS qualifiziert werden sollen, sondern unterschiedlichste Dialoge mit dutzenden Testschritten? Im Folgenden Kapitel wird gezeigt, wie die Testumgebung

erweitert und angepasst werden muss, um die generierten Testfälle zur Verifikation der Dialogstruktur automatisch auszuführen.

7.3 Automatischer Dialogtest der akustischen MMS

In Kapitel 6.2 und 6.3 wurde die automatische Generierung von Testfällen aus einem Testmodell oder Systemmodell der akustischen MMS entwickelt und umgesetzt. Die automatische Durchführung dieser Testfälle birgt erhebliche Vorteile. Es ist zu erwarten, dass sich die Testumgebung nach ihrer Implementierung in jedem Testdurchlauf identisch verhält. Die automatisch durchgeführten Benutzereingaben bleiben, wie das gesamte zeitliche Verhalten, auch bei mehrfacher Wiederholung unverändert. Eine Eigenschaft, die im Test durch den Menschen nicht zu erreichen ist. Hier kann der Mensch als Fehlerquelle im Test nicht ausgeschlossen werden. Zudem lassen sich Argumente aufführen, welche allgemein bei Verwendung automatisierter Systeme gelten. Zentraler Vorteil ist die Möglichkeit zum pausenlosen Einsatz der automatisierten Testdurchführung. In relativ kurzer Zeit kann so eine hohe Testabdeckung erreicht werden, um eine qualifizierte Aussage über den Reifegrad des SUT treffen zu können.

Der Fokus des Dialogtests liegt auf der Qualifizierung der Dialogstruktur und damit dem Dialogmanager (Abbildung 66, S. 237). Folgt die korrekte akustische und optische Systemrückmeldung auf eine spezifische Benutzereingabe? Ist die Eingabe kontextrelevanter Kommandos möglich und werden Systemfunktionen korrekt aufgerufen?

Der Dialogtest ist von den beschriebenen Risiken und Nachteilen bei Verwendung einer expliziten Testschnittstelle (Kapitel 7.1) ebenso betroffen wie die funktionalen Tests. Aus diesem Grund ist auch der Dialogtest ohne deren Einsatz zu realisieren (vergl. These C). Im Folgenden wird eine Testumgebung entwickelt, welche die akustische MMS nutzt, um die automatische Testdurchführung zu realisieren. Durch dieses Vorgehen entsteht ein virtueller Tester, der die Testaktivitäten des Menschen abbilden kann. Die in Kapitel 7.2 vorgestellte Testumgebung, mit den in Abbildung 68 dargestellten Schnittstellen zum SUT, bringt bereits verschiedene Voraussetzungen mit, um den virtuellen Tester zu realisieren. Hier konnte gezeigt werden, dass haptische Benutzereingaben von der Testumgebung abgebildet werden können. Zudem besteht die Möglichkeit, optische Systemausgaben zu erkennen. Um komplexe Sprachdialoge automatisch durchzuführen, sind weitere Schnittstellen zum SUT notwendig. Es wird sich zeigen, dass der virtuelle Tester in der Lage sein muss, Spracheingaben zu tätigen und die akustischen Systemrückmeldungen des SUT zu erkennen. In Abbildung 69 sind die hierfür erforderlichen Schnittstellen zum SUT dargestellt.

In den folgenden Kapiteln wird die Testumgebung weiterentwickelt um eine Möglichkeit zur automatischen Spracheingabe und zur Erkennung der akustischen Sprachausgabe des SUT zu schaffen. Der virtuelle Tester erhält eine Stimme und ein Gehör. Kann dies erfolgreich umgesetzt werden ist ferner zu zeigen, dass die generierten Testfälle zur Qualifizierung der MMS in der entwickelten Testumgebung durchgeführt werden können.

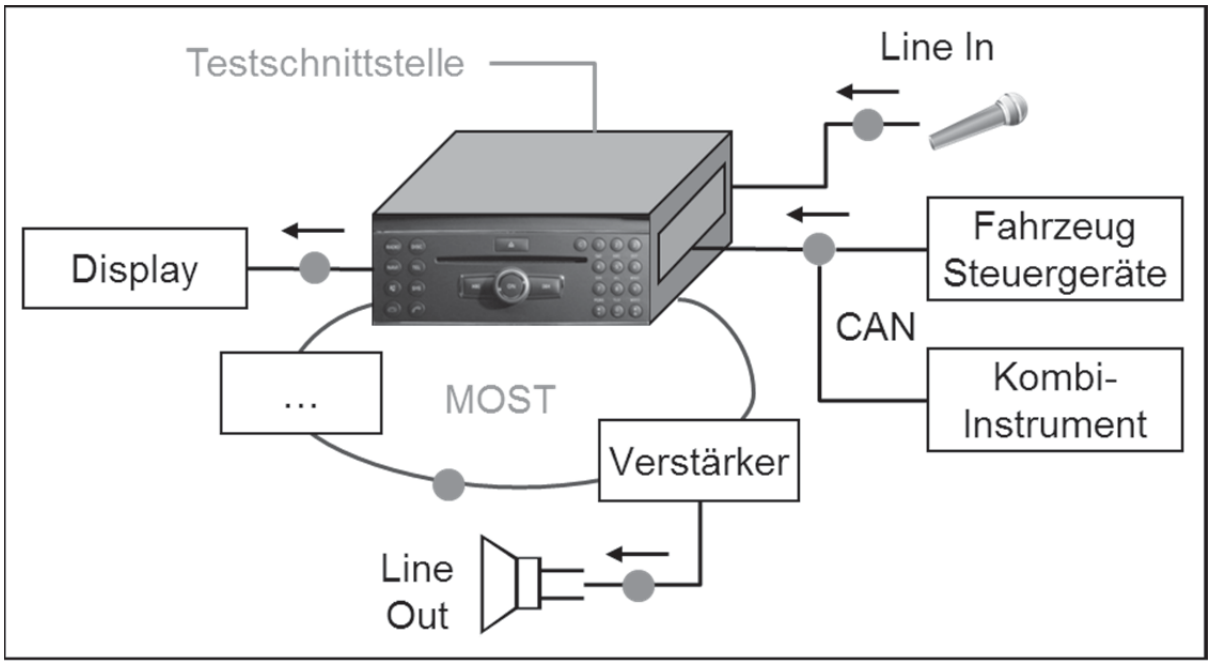


Abbildung 69: Testschnittstellen zum automatischen Dialogtest

7.3.1 Spracheingabe durch den virtuellen Tester

Neben den haptischen Benutzereingaben sind im Test der akustischen MMS die verbalen Nutzereingaben von zentraler Bedeutung. Wie in Kapitel 2.1.3 bereits gezeigt wurde, ist die Anzahl möglicher Benutzereingaben in den vergangenen Telematikgenerationen stark angestiegen und es ist zu erwarten, dass sich dieser Trend fortsetzt. Ziel ist, all diese Spracheingaben automatisch in das SUT einsprechen zu können, wenn ein generierter Testfall dies erfordert. In Abbildung 53 (S. 206) ist ein solcher Testfall beispielhaft dargestellt. Die akustischen Nutzereingaben sind durch das Präfix „User_Phrase“ beschrieben, gefolgt vom jeweiligen Kommando. Um eine automatische Ausführung ohne den Menschen zu realisieren, sind zwei Vorgehensweisen denkbar. Zum einen ist es möglich, alle Nutzereingaben von einem Menschen sprechen zu lassen und Aufnahmen zu erzeugen, welche

dem SUT bei Durchführung der Tests vorgespielt werden. Zum anderen könnte eine künstlich erzeugte Stimme Verwendung finden, welche die im Testfall textuell beschriebenen Nutzereingaben für eine Sprachsynthese verwendet.

Die Verwendung einer realen menschlichen Stimme hat den Vorteil, dass sie der Testdurchführung durch einen menschlichen Tester am nächsten kommt. Für das SUT ist nicht zu unterscheiden, ob die Nutzereingabe „Live“ oder aufgezeichnet erfolgt. Nachteilig sind die erforderlichen Ressourcen zu bewerten, um hunderte Benutzereingaben für jedes Entwicklungsprojekt von menschlichen Sprechern aufzuzeichnen, damit alle möglichen Dialoge realisieren werden können. Ändern sich Kommandos kurzfristig, müssten auch die entsprechenden Aufzeichnungen angepasst werden. Zudem muss sichergestellt sein, dass die aufgezeichneten Audiodateien eindeutig mit den im Testfall beschriebenen Nutzereingaben verknüpft werden können und Zuordnungsfehler ausgeschlossen sind. Soll der Nutzer z. B. „PIN eingeben“ sprechen, muss von der Testumgebung die korrekte Audiodatei mit dem richtigem Inhalt angezogen werden.

Angesichts der potentiellen Fehlerquellen in diesem Vorgehen ist es zweckdienlich, die Möglichkeiten einer automatisch erzeugten Spracheingabe durch Sprachsynthese zu erörtern. Die Verwendung von „Text-to-Speech-Systemen“ (TTS) wurde bereits in Kapitel 2.1.3 vorgestellt. Die Verwendung eines solchen Systems ermöglicht, einen vorhandenen Fließtext in akustische Sprachausgaben zu verwandeln. Im Beispiel könnte bei Erreichen des Testschritts „User_Phrase_[PIN eingeben]“ das beschriebene Kommando von der Testumgebung an eine TTS-Engine weitergereicht werden, welche aus

dem Text die entsprechende akustische Sprachausgabe erzeugt. Diese kann schließlich als Systemeingabe Verwendung finden. Vorteil der Vorgehensweise ist die hohe Flexibilität in der Erzeugung und Verwendung akustischer Spracheingaben. Inhaltliche Änderungen und neue Benutzereingaben sind im generierten Testfall beschrieben und würden ohne äußeres Zutun auch in der akustischen Umsetzung dargestellt. Zudem verfügt die künstliche Sprechstimme über eine unveränderliche und von Umgebungsbedingungen unabhängige Charakteristik.

Schlägt ein vom Menschen durchgeführter Dialogtest fehl, kann dies sowohl in einem fehlerhaft implementierten Dialogmanager, als auch in einer fehlgeschlagenen Spracherkennung begründet sein. Um die Dialogstruktur funktional zu testen und erkannte Fehler eindeutig dem Dialogmanager zuordnen zu können, ist es zielführend, die alternative Fehlerquelle in der Spracherkennung nach Möglichkeit auszuschließen.[6]

Die Umstände zeigen, dass die Verwendung einer synthetisch generierten Spracheingabe gegenüber der Aufzeichnung und Wiedergabe menschlicher Kommandos Vorteile besitzt. Um zu einem abschließenden Urteil zu kommen und die bestmögliche Lösung zur automatischen Spracheingabe zu ermitteln ist es notwendig, die Erkennraten synthetisch generierter Stimmen mit denen menschlicher zu vergleichen. Auf Basis der gewonnenen Erkenntnisse kann entschieden werden, ob die theoretisch bestehenden Vorteile in der Verwendung einer TTS-Engine auch in der Praxis bestehen bleiben.

[6] Die Qualität des ASR wird über den in Kapitel 7.2 beschriebenen Erkennertest abgesichert.

Auf dem Markt sind diverse kommerzielle TTS-Werkzeuge erhältlich. Verschiedene Firmen wie z. B. „Nuance“ oder „Acapela“ bieten Software zur synthetischen Generierung von Sprachausgaben an. Alle Anbieter ermöglichen dabei die Auswahl verschiedener Stimmen mit unterschiedlichen, akustischen Charakteristika. Nach einer Marktanalyse konnten acht synthetische Stimmen unterschiedlicher Anbieter für eine Evaluation ausgewählt und beschafft werden. Um die Erkennungsraten der verschiedenen Stimmen direkt vergleichen und bewerten zu können, wird ein definiertes Testset verwendet, in dem alle spezifizierten Nutzereingaben abgebildet sind. Ergänzend hierzu wurden selbe Nutzereingaben von einem menschlichen Tester gesprochen und aufgezeichnet. Mit dem Ziel, die Erkennraten des ASR in Abhängigkeit der verschiedenen Stimmen zu messen, werden diese an den im Infotainmentsystem verbauten ASR geschickt. Die Ergebnisse des Tests sind in Abbildung 70 dargestellt.

Mit Ausnahme einer TTS Stimme (Voice 3) werden alle synthetisch generierten Spracheingaben sehr gut erkannt. Dieses Ergebnis ist durchaus überraschend. Zwar ist die Annahme naheliegend, dass automatisch generierte Stimmen von einem automatischen Spracherkenner gut erkannt werden können, tatsächlich war die Qualität synthetisch generierter Stimmen aber noch bis vor kurzem bei weitem nicht ausreichend, um diese von einem Spracherkenner zu interpretieren. In dieser Evaluation zeigt sich jedoch, dass fast alle verwendeten Stimmen besser erkannt werden als die menschliche Referenzstimme. Dabei ist zudem zu berücksichtigen, dass im Evaluationstest das gesamte bestehende Wortlexikon (mit allen möglichen Nutzereingaben) Verwendung fand. In der Praxis wird in Abhängigkeit des aktuellen Dialogs

das Wortlexikon entsprechend eingeschränkt. So wird bei einer Ja/Nein Abfrage im Dialog der ASR ausschließlich auf die gültigen Kommandos „Ja“ und „Nein“ referenzieren. Je kleiner das zu Grunde liegende Wortlexikon, desto größer die Wahrscheinlichkeit einer korrekt erkannten Spracheingabe. Dies erklärt, warum die menschliche Referenzstimme zu 90 % korrekt erkannt wurde, die realen Erkennungsraten im Fahrzeug in der Regel aber 95 % übersteigen. Angesichts dieser Umstände sind die Erkennraten der synthetisch generierten Stimmen umso höher einzuschätzen. Die Stimmen 1 und 4 wurden mit 98 % am besten erkannt. Aber auch die Stimmen 2, 5, 6 und 7 können bei Erkennraten von 97 % überzeugen.

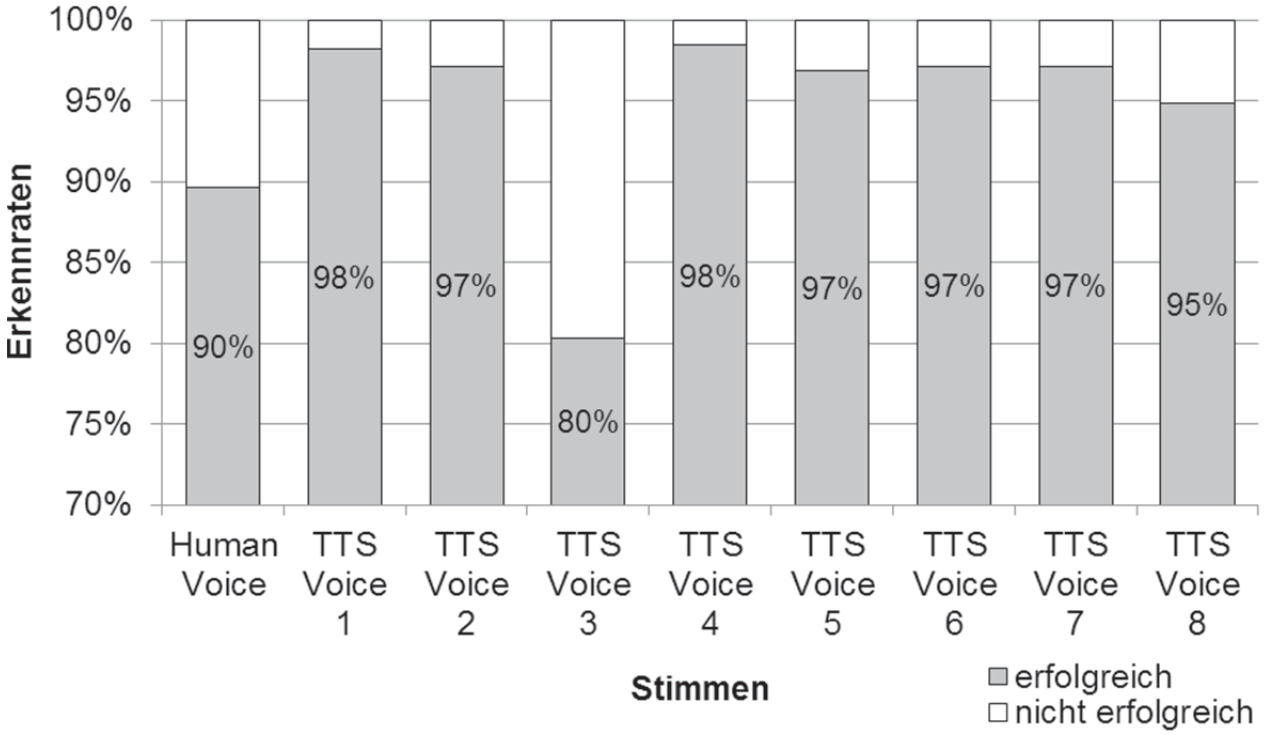


Abbildung 70: Erkennraten synthetisch generierter Stimmen (Kommandos)

Als ein erstes Ergebnis kann festgehalten werden, dass einige synthetisch generierte Stimmen vom ASR des Infotainmentsystems besser erkannt wer-

den als die menschliche Stimme. Durch die Evaluation lässt sich jedoch nicht eindeutig feststellen, welche der TTS Stimmen für das weitere Vorgehen zu wählen ist. Aus diesem Grund ist eine weitere Evaluation unter erschwerten Rahmenbedingungen sinnvoll. Anstatt die bekannten Nutzer-eingaben zu verwenden[7], sollen nun freie Eingaben differenziertere Evaluationsergebnisse herbeiführen.

Für die Testdurchführung wird der in Kapitel 7.2 beschriebene Funktionstest zur Zieleingabe verwendet. Neben den vorhandenen Testdaten menschlicher Stimmen wurden dieselben Zieleingaben synthetisch generiert. Hierbei finden die TTS Stimmen mit den höchsten Erkennraten der ersten Evaluation Verwendung. „Voice 3“ und „Voice 8“ werden nicht mehr berücksichtigt. In Abbildung 71 sind die Ergebnisse des Tests dargestellt.

Während die Erkennrate des menschlichen Sprechers in der zweiten Evaluationsrunde mit freier Spracheingabe bei 90 % liegt und sich damit auf dem gleichem Niveau wie in der ersten Evaluation bewegt, fallen die Erkenn-raten einiger synthetisch generierter Stimmen nun deutlich ab. Bei drei Stimmen (Voice 1, 5 und 6) werden weniger als 50 % der Spracheingaben korrekt interpretiert. Gut schneiden lediglich die Stimmen 4 und 7 ab, wobei insbesondere die Stimme 4 mit 90 % sehr gut abschneidet und eine noch um 14 % höhere Erkennrate aufweist als Stimme 7. Zwar ist die Erkennrate im Vergleich zur ersten Evaluationsrunde um 8 % gesunken, bewegt sich jedoch noch immer auf dem Niveau der menschlichen Stimme.

[7] Welche im Wortlexikon definiert sind und dem ASR damit „bekannt“.

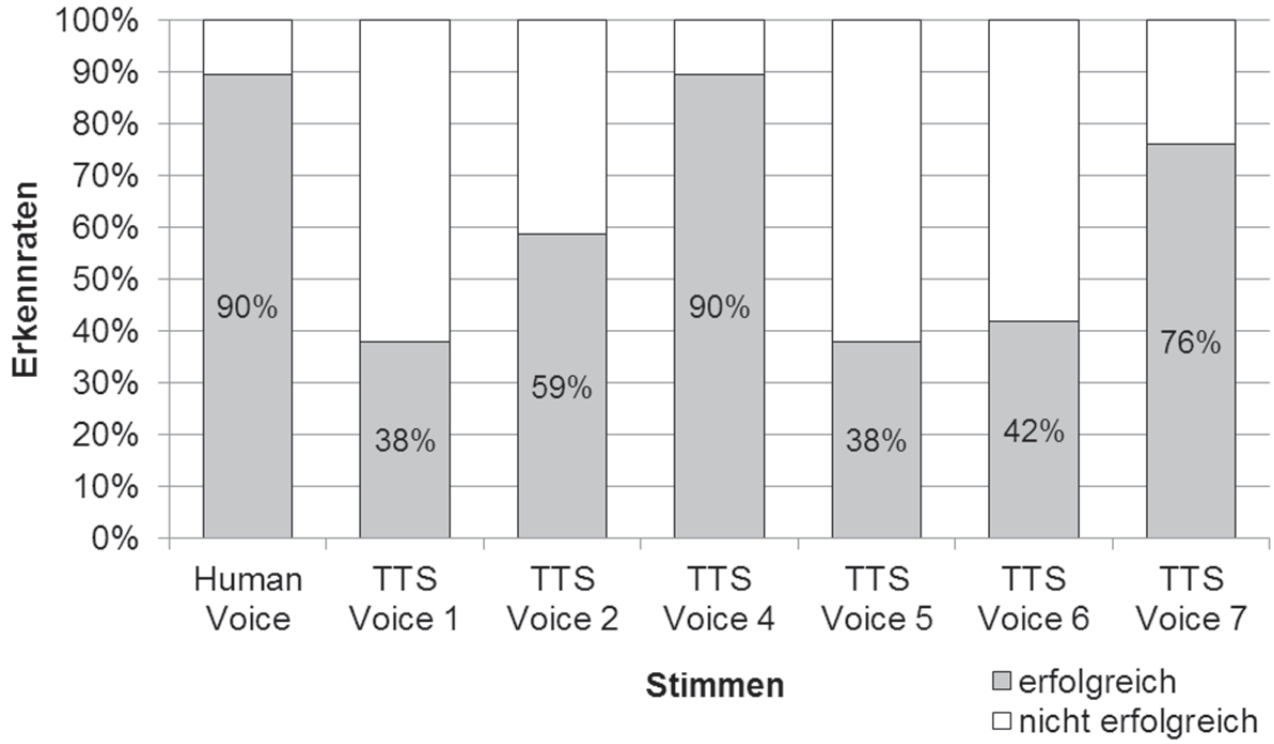


Abbildung 71: Erkennraten synthetisch generierter Stimmen (Zieleingaben)

Damit ist gezeigt, dass die Qualität synthetisch generierter Stimmen heute ein Niveau erreicht hat, welches von einem ASR ebenso gut wie eine reale, menschliche Stimme interpretiert werden kann. Die beschriebenen Vorteile in der Verwendung einer TTS basierten Vorgehensweise können folglich genutzt werden. Auf Basis der durchgeführten Evaluation wird die TTS „Voice 4“ ausgewählt, um die Spracheingabe in der automatischen Testdurchführung zu realisieren. Der virtuelle Tester erhält seine Stimme.

7.3.2 Spracherkennung durch den virtuellen Tester

Die akustischen Dialoge zwischen Mensch und Maschine sind in vielen Dialogschritten durch optische (Hilfe-)Anzeigen unterstützt. Der System-zustand kann, wie im Zieleingabetest (Kapitel 7.2) gezeigt, auf Basis dieser

Anzeigen über die Bilderkennung meist ermittelt werden. Zur automatischen Verifikation des SDS sind aber auch die akustischen Systemausgaben zu berücksichtigen. Insbesondere wenn der Sprachdialog ausschließlich durch akustische Rückmeldungen und ohne optische Unterstützung stattfindet, ist deren Erkennung zwingend erforderlich. Um die Erkennung der akustischen Systemausgaben in der Testumgebung zu realisieren, sind zwei Ansätze vielversprechend.

Die erste Möglichkeit ist die Verwendung eines „Dynamic-Time-Warping“ (DTW) Algorithmus. Dieser ermöglicht den Abgleich des vom Infotainment-system ausgegebenen Audiosignals mit einer vorgegebenen Menge möglicher Signale „Templates“. Nachdem das Infotainmentsystem eine bekannte endliche Anzahl von Sprachausgaben verwendet, ist die Anwendung der Methodik grundsätzlich möglich. Im praktischen Versuch musste jedoch festgestellt werden, dass die Erkennungsraten mit 75 % und weniger hinter den Erwartungen zurückbleiben. Für weiterführende Informationen zum DTW Algorithmus sei auf [Cul11] verwiesen.

Die zweite Möglichkeit ist die Verwendung eines kommerziellen ASR. Die Sprachausgaben des Infotainmentsystems bestehen heute mehrheitlich aus von menschlichen Sprechern vertonten Textbausteinen. Wie beschrieben ist Inhalt und Anzahl der Sprachausgaben bekannt, womit die Verwendung eines entsprechenden Wortlexikons im ASR möglich ist und die Wahrscheinlichkeit einer korrekten Erkennung erhöht werden kann. Die Vielfalt verfügbarer, kommerzieller ASR hat sich in den vergangenen Jahren mit dem Aufkauf verschiedener Anbieter durch den Marktführer Nuance reduziert. Aufgrund der Verfügbarkeit wird zunächst der Spracherkenner „VoCon“ von Nuance

evaluiert. Zur Bestimmung der Erkennraten findet ein Testset mit allen möglichen akustischen Systemausgaben des SUT Verwendung.

Die in Abbildung 72 dargestellten Ergebnisse übertreffen auch hier die Erwartungen. Mit der nahezu vollständigen und fehlerfreien Erkennung aller akustischen Systemausgaben ist bewiesen, dass auch diese von der Testumgebung interpretiert werden können, wenn ein entsprechender Spracherkenner implementiert ist. Der virtuelle Tester erhält damit sein Gehör.

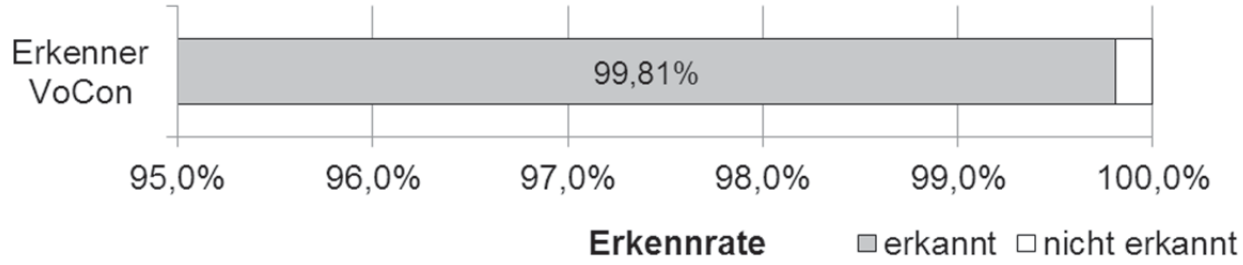


Abbildung 72: Erkennrate der akustischen Systemausgaben

Abschließend soll aufgezeigt werden, wie die generierten Testfälle in der nun erweiterten Testumgebung automatisiert durchgeführt werden können. Zunächst ist es notwendig, die Testschritte und Verifikationspunkte des Testfalls inhaltlich eindeutig identifizieren zu können. Die Grundlage hierfür wurden in Kapitel 6.3.1 geschaffen. Haptische und akustische Nutzereingaben sind ebenso eindeutig beschrieben wie optische und akustische Systemrückmeldungen oder Systemaktivitäten. Des Weiteren werden vor dem jeweiligen Testfall die notwendigen Preconditions beschrieben.

Bevor der Testfall automatisch ausgeführt werden kann, ist zu berücksichtigen, dass Nutzereingaben mit dynamischem Inhalt konkrete Werte zuzuordnen sind. Abbildung 53 (S. 206) zeigt einen Testfall, welcher die

akustische Systemeingabe <DIGITS> beinhaltet. Um den Testfall ausführen zu können, ist der abstrakte Platzhalter mit konkreten Werten zu ersetzen. Nach dem nicht der ASR, sondern die Dialogstruktur geprüft wird, genügt die Verwendung einer definierten Zahlenkombination (z. B. 7453). Um die Vielzahl generierter Testfälle automatisch mit konkreten Inhalten anzureichern, ist es zielführend, eine Datenbank aller dynamischen Nutzereingaben mit den für den Test zugewiesenen, konkreten Werten zu pflegen, um alle Platzhalter im Testfall durch Ausführung eines „Scripts“ automatisch zu ersetzen. Ein analoges Vorgehen ist auch für die Transformation in eine andere Landessprache möglich. So konnte der Prinzipnachweis erbracht werden, dass auf Basis einer Übersetzungsdatenbank alle Nutzereingaben und Systemausgaben (durch Verwendung eines entsprechenden Skripts), automatisch in die gewünschte Zielsprache „übersetzt“ werden können. Die zeitintensive Testfallgenerierung muss damit nicht für jede Landessprache wiederholt werden, sondern kann einmalig stattfinden, gefolgt von einer automatischen Übersetzung.

Während die Bearbeitung der Sprache oder der dynamischen Nutzereingaben nicht nur für die automatische Testdurchführung, sondern auch für die manuelle notwendig ist, bedürfen die Preconditions insbesondere bei der automatischen Testdurchführung weiterer Beachtung. Bei manuellen Testaktivitäten können die Umgebungsbedingungen des Infotainmentsystems und notwendige Vorbedingungen für die Testdurchführung durch den menschlichen Tester flexibel angepasst werden. Diese Möglichkeit ist im automatischen Test in der Regel nicht gegeben. Während einige Preconditions durch die Testumgebung simuliert werden können, setzen andere, wie das Einlegen

oder Entfernen eines Datenträgers, einen menschlichen Eingriff voraus. Um eine möglichst große Anzahl Testfälle über einen längeren Zeitraum ohne äußere Eingriffe durchführen zu können, ist es daher notwendig, die Testfälle entsprechend der relevanten Umgebungs-bedingungen zu sortieren. Auf diese Weise lassen sich Änderungen der Preconditions wären der Durchführung eines Testsets vermeiden. Auch dies kann durch die Verwendung eines entsprechenden Skripts oder Testwerkzeugs zur Organisation von Testfällen (wie z. B: dem HP Quality Center) bewerkstelligt werden. Sind diese Vorarbeiten abgeschlossen, können die Testfälle automatisiert durchgeführt werden. In Abbildung 73 sind die Schnittstellen zwischen Testfall, Testumgebung und SUT schematisch dargestellt.

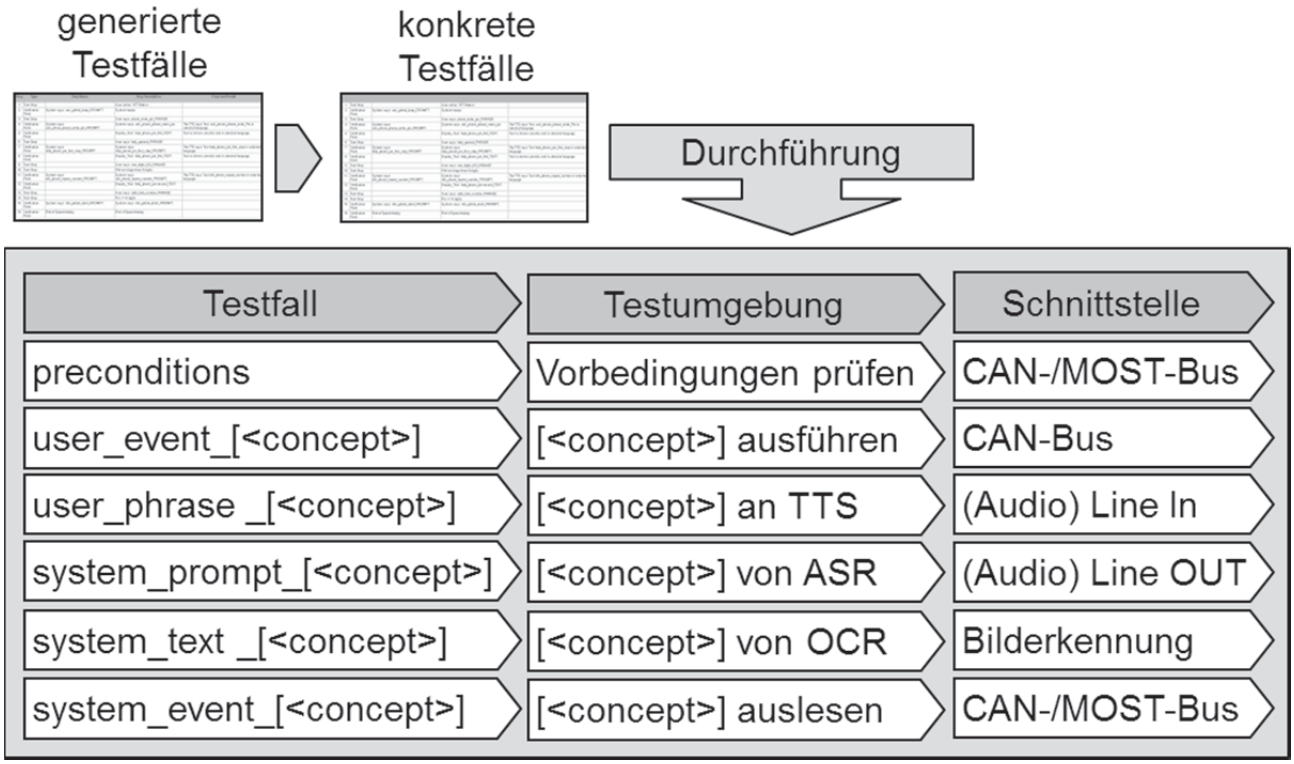


Abbildung 73: Prinzip der automatischen Testdurchführung

Die Testumgebung muss in der Lage sein, die im Testfall beschriebenen Preconditions und die zugehörigen Parameter über den CAN- bzw. MOST-Bus abzufragen. Besteht z. B. die Vorbedingung „Phone_state_ok“, kann die Testumgebung über die entsprechende Schnittstelle zum SUT auslesen, ob der Systemstatus besteht. Ist dies nicht der Fall, kann der gewünschte Systemzustand gegebenenfalls injiziert werden[8].

Haptische Nutzereingaben sind als <concept> (z. B. <PTT>, etc.) beschrieben. Die Testumgebung kann die möglichen <concept> Inhalte interpretieren und über den CAN-Bus das entsprechende Signal an das SUT senden, um die haptische Aktivität durchzuführen.

Akustische Nutzereingaben sind analog beschrieben, wobei das <concept> alle möglichen Spracheingaben beinhalten kann. Die Testumgebung wird bei Erreichen des Testschritts „user_phrase“ den Inhalt von <concept> (z. B. <Ziel eingeben>) durch die angebundene TTS-Engine in eine akustische Spracheingabe transformieren und über den Audioeingang an das SUT schicken.

Die akustischen Systemausgaben werden über den Audioausgang des SUT an die Testumgebung weitergeleitet. Diese aktiviert, angesteuert durch das Erreichen des Testschritts „system_prompt“, den angebundenen ASR, welcher die eingehende Systemausgabe interpretiert und das akustische Signal in Text transformiert. Der Text wird schließlich von der Testumgebung

[8] Vorausgesetzt, die Vorbedingung kann automatisiert hergestellt werden.

mit dem im Testfall beschriebenen Sollwert (<concept>) auf Übereinstimmung geprüft. Kann diese verifiziert werden, ist der Testschritt erfolgreich durchlaufen. Ist keine Übereinstimmung vorhanden, wird der Testfall als nicht erfolgreich gekennzeichnet.

Optische Ausgaben des SUT werden von der Bilderkennung erfasst und interpretiert. Die Ergebnisse stehen der Testumgebung in Textform zur Verfügung. Diese verwendet die Testumgebung, angesteuert durch das Erreichen des Testschritts „system_text“ und vergleicht den erkannten Text mit dem (im <concept>) beschriebenen Sollwert.

Zuletzt bleibt die Verifikation von Systemaktivitäten, welche keine akustischen oder optischen Systemrückmeldungen verursachen. In diesen Fällen bleibt die Möglichkeit, Systemaktivitäten oder -zustände über die Signalaktivitäten auf CAN- und MOST-Bus zu identifizieren. Die relevanten Signale sind von der Testumgebung zu interpretieren und mit den im Testfall beschriebenen erwarteten Systemaktivitäten zu vergleichen (system_event_[<concept>]).

Auf Basis dieser Vorgehensweise konnte der Prinzipnachweis erbracht werden, dass die automatische Testdurchführung durch Verwendung einer weiterentwickelten Testumgebung mit synthetisch generierter Sprachausgabe, Spracherkennung durch einen ASR und formale, generierte Testfälle realisiert werden kann. Damit ist auch die in Kapitel 3.2 aufgestellte These C bewiesen. Die Durchführung des Dialogtests kann ohne Verwendung einer expliziten Testschnittstelle stattfinden. In der Testdurchführung kann der Mensch durch einen virtuellen Tester ersetzt werden, welcher ausschließlich die MMS nutzt, um gegebene Testaktivitäten automatisiert durchzuführen.

7.4 Zusammenfassung

Nachdem in Kapitel 6 der Beweis erbracht wurde, dass auf Basis formaler Modelle und durch Verwendung eines kommerziellen Werkzeugs Testfälle für die akustische MMS generiert werden können, wurde in Kapitel 7 bewiesen, dass die generierten Testfälle zum Test der akustischen MMS automatisch und ohne Verwendung einer expliziten Testschnittstelle durchgeführt werden können (vergl. Kapitel 3.2, These C).

Zu Beginn des Kapitels wurden zunächst die Rahmenbedingungen für eine automatisierte Testdurchführung betrachtet. Dabei rückte die Verwendung einer expliziten Testschnittstelle zur (automatisierten) Ansteuerung des Infotainmentsystems in den Fokus. Vor- und Nachteile der Schnittstelle wurden erörtert, wobei die Risiken in deren Verwendung schließlich überwogen, worin die Ursache für den in These C beschriebenen Ansatz begründet ist.

Mit dem Ziel, die generierten Testfälle zum automatischen Dialogtest der akustischen MMS zu verwenden, wurde in Kapitel 7.2 zunächst ein automatischer Funktionstest des SDS umgesetzt. Analog zum Test der Erkennungsqualität des ASR bei Eingabekommandos wurde ein Vorgehen zum Test der Erkennungsqualität bei freien Zieleingaben umgesetzt. Dabei wurden die Testaktivitäten in eine Testumgebung transferiert, die einen automatischen Test ohne die Verwendung einer expliziten Testschnittstelle ermöglicht. Auf Basis der hierdurch erlangten Erkenntnisse wurde in Kapitel 7.3 der zum Ziel gesetzte automatische Dialogtest des SDS entwickelt. Es zeigte sich, dass die vorhandene Testumgebung um die Möglichkeiten der

akustischen Systemeingabe und der Erkennung von akustischen Systemausgaben erweitert werden muss. In Kapitel 7.3.1 sind zunächst die Möglichkeiten der akustischen Systemeingabe erörtert. Dabei wurden die Vorteile synthetisch generierter Stimmen durch TTS-Systeme deutlich.

In zwei aufeinanderfolgenden Evaluationsschritten wurden verschiedene TTS-Systeme und synthetisch generierte Stimmen bewertet. Dabei ist nicht die subjektive Sprachqualität der Stimmen von Interesse, sondern ausschließlich die Erkennraten durch den im Infotainmentsystem verbauten ASR. Schließlich konnte eine synthetische Stimme identifiziert werden, welche sich hinsichtlich der Erkennraten auf dem gleichen Niveau wie die menschliche Stimme bewegt.

In Kapitel 7.3.2 wurden Möglichkeiten zur Erkennung der akustischen Systemausgaben beschrieben. Durch einen praktischen Versuch wurde gezeigt, dass diese von einem in die Testumgebung integrierten ASR zu nahezu 100 % erkannt werden. Nachdem die erforderlichen Erweiterungen der Testumgebung entwickelt wurden konnte dargestellt werden, wie die generierten Testfälle in der Testumgebung automatisiert durchgeführt werden können. Die Zusammenhänge und Schnittstellen zwischen generiertem Testfall, Testumgebung und SUT wurden dargestellt.

Durch die entwickelte Vorgehensweise ist der automatische Dialogtest realisiert und ein virtueller Tester geschaffen, welcher ausschließlich die MMS zum Test des SDS verwendet.

8 Ergebnisse, Kostenbetrachtung und Bewertung

8.1 Ergebnisse

Die Entwicklung der softwarebasierten MMS von Infotainmentsystemen im Kfz bewegt sich aufgrund veränderter Umgebungsbedingungen zunehmend im Spannungsfeld zwischen Qualitätsanspruch, Zeit- und Kostendruck. Bei gleichzeitig immer komplexer werdenden Systeminhalten und Funktionen wird die funktionale Qualifizierung der haptischen und akustischen MMS zu einer Herausforderung (Kapitel 3.1).

Im Bewusstsein dieser Problematik wurde die zentrale These aufgestellt, dass die Entwicklung und Umsetzung einer modellbasierten sowie automatisierten Vorgehensweise zum Test der akustischen MMS einen „virtuellen Tester“ hervorbringt. Dessen Implementierung und der damit verknüpfte Einsatz neuer Prozesse, Methoden und Werkzeuge sollte die Effizienz und Qualität des Testprozesses verbessern, um das Spannungsfeld zwischen Qualität, Zeit und Kosten zu lösen. Zum Beweis dieses zentralen Ansatzes wurde selbiger in drei Thesen (A, B, C) aufgegliedert (Kapitel 3.2).

Um die Grundlage zur Umsetzung und Beweisführung herzustellen, wurden zunächst die prozessualen Rahmenbedingungen für die Einführung einer modellbasierten Vorgehensweise analysiert und ausgearbeitet. Nach der Evaluation verschiedener Prozessmodelle konnte durch Anwendung des TPI-Modells sowohl die Reife der verwendeten Testprozesse bestimmt werden, als auch die notwendigen Verbesserungsmaßnahmen, um die Voraussetzungen für das weitere Vorgehen zu schaffen (Kapitel 4).

In der Folge richtete sich der Fokus auf das zentrale Element im MBT-Prozess, die Modellierung. Nach Analyse der vorhandenen Systemspezifikationen wurde für die akustische MMS eine formale Modellierungssyntax entwickelt, welche eine Modellierung auf Basis der zuvor definierten Anforderungen ermöglichte. Repräsentative Systemteile wurden auf diese Weise modelliert. Durch die formale Beschreibung der akustischen MMS in Testmodellen wurde die Grundlage für die modellbasierte Vorgehensweise gelegt. Darüber hinaus konnte gezeigt werden, dass die Modellierungssyntax mit Einschränkungen auch auf die haptische MMS übertragen werden kann (Kapitel 5).

Das entwickelte und erstellte Testmodell der MMS findet in der anschließenden Evaluierung verschiedener Werkzeuge zur Generierung von Testfällen praktische Anwendung. Das Werkzeug „MBTsuite“ der Firma „sepp.med“ wurde aus sechs verschiedenen Prüflingen als jenes mit der höchsten Anforderungsüberdeckung identifiziert und ausgewählt, um die Testfallgenerierung zu realisieren. Verschiedene Generierungsstrategien zur Erstellung von Testfällen wurden betrachtet, um effiziente Strategien zu ermitteln. In der Analyse des ausgewählten Werkzeugs wurde erkennbar, dass dieses durch überschaubare Entwicklungsaufwendungen auch eine Testfallgenerierung aus der vorhandenen Spezifikation der akustischen MMS ermöglicht. Nach einer Erörterung der Vor- und Nachteile, welche durch eine systemmodellorientierte Testfallgenerierung entstehen, wurden die notwendigen Maßnahmen zur Realisierung dieser alternativen Vorgehensweise umgesetzt. Es wurde gezeigt, dass die Testfallgenerierung aus den erstellten Modellen der haptischen MMS grundsätzlich stattfinden kann, die Generie-

rungsstrategien aufgrund der spezifischen Modellcharakteristik jedoch schnell an ihre Grenzen stoßen (Kapitel 6).

Um die Effizienz des MBT-Prozesses auch in der Durchführung generierter Testfälle zu steigern, folgte die Weiterentwicklung der Testumgebung. Die bislang bestehende Vorgehensweise der Testdurchführung über eine im SUT implementierte Testschnittstelle birgt verschiedene Risiken und Nachteile. Folglich bestand das Ziel, die Testumgebung dahingehend zu erweitern, dass auch im automatischen Test ausschließlich die Schnittstellen zwischen Mensch und Maschine Verwendung finden kann.

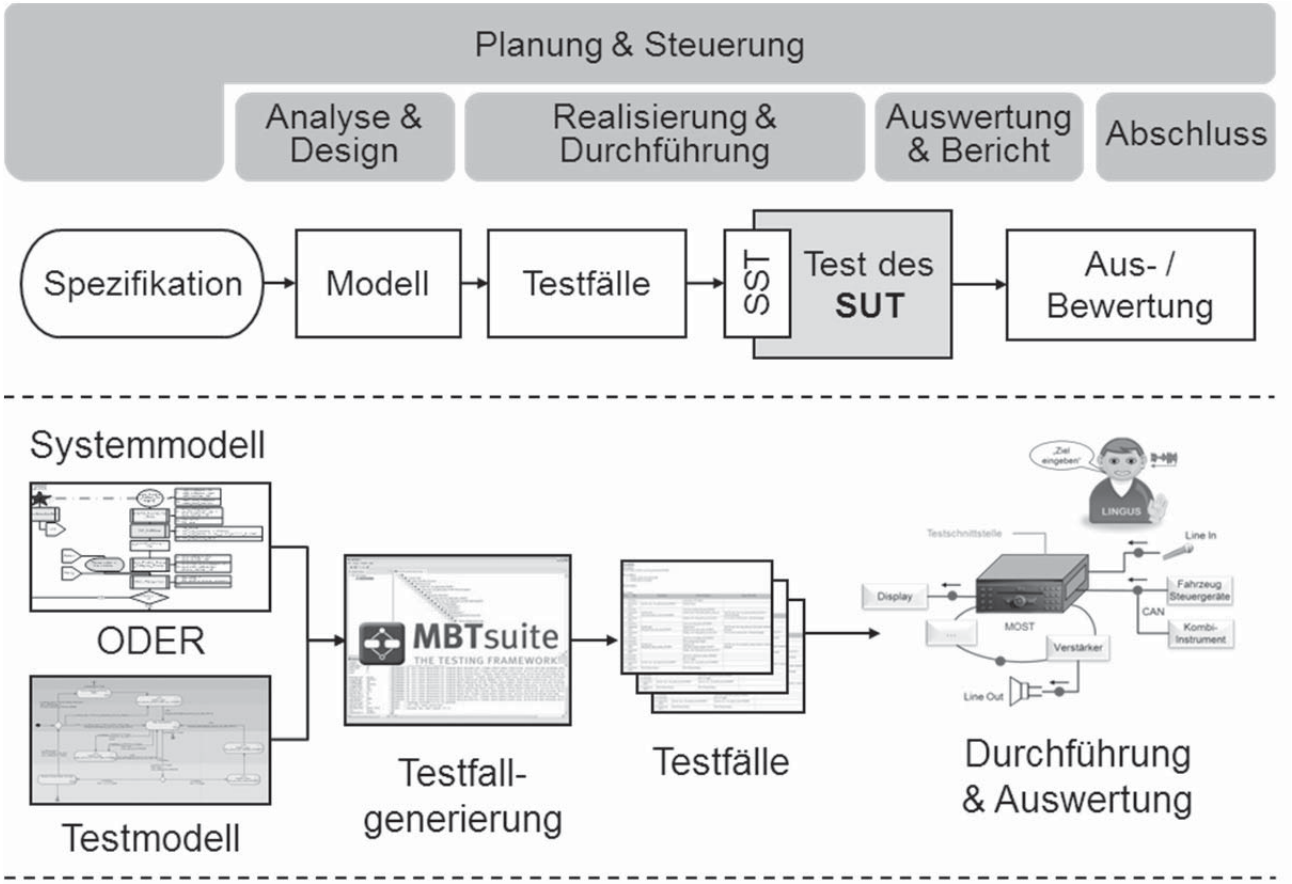


Abbildung 74: Ergebnis in der Übersicht – MBT der akustischen MMS

Durch die Evaluation und die Auswahl einer synthetisch generierten Stimme sowie die Implementierung eines ASR in der Testumgebung wurde ein virtueller Tester geschaffen, welcher den menschlichen Tester ergänzt und/oder ersetzen kann.

Die praktische Umsetzung dieser Vorgehensweise schließt den Beweis der zu Beginn formulierten Thesen und des zentralen Ansatzes ab. Die modellbasierte Vorgehensweise zum Test der akustischen MMS ist vollständig umgesetzt. In Abbildung 74 ist die entwickelte Vorgehensweise dargestellt.

Um eine abschließende Bewertung der entwickelten Methodik und deren Effizienz abgeben zu können, werden im folgenden Kapitel zunächst die wirtschaftlichen Einflussgrößen ermittelt und bewertet.

8.2 Kostenbetrachtung des MBT einer SW MMS

Die Prozesselemente der „Spezifikation und Modellierung“, „Testfallgenerierung“ sowie „Testdurchführung“ und „Auswertung“ werden im Folgenden hinsichtlich der anfallenden Kosten analysiert.

Die Erstellung und Pflege eines Testmodells ist mit Aufwendungen verbunden. Hier werden Ressourcen investiert, welche sich erst in den folgenden Prozessschritten auszahlen. Im konkreten Fall konnte jedoch eine Vorgehensweise entwickelt werden, um die vorhandene, weitgehend formale Spezifikation der akustischen MMS als Basis für die Testfallgenerierung heranzuziehen. Hierdurch entfällt einerseits die Modellierung des Testmodells, andererseits sind Investitionen notwendig, um formale Fehler in der Spezifikation zu korrigieren (Kapitel 8.3). Auf Basis der im Prinzipnachweis

durchgeführten Analysen lassen sich die zu erwartenden Aufwendungen abschätzen. In der folgenden Kostenabschätzung wird mit einem Bruttojahresgehalt von 60.000 €, bei 220 Arbeitstagen pro Jahr ausgegangen.

Im ersten Jahr eines modellbasierten Vorgehens sind umfangreiche Ressourcen für die umfassende Korrektur der gesamten Spezifikation einzuplanen. Es ist mit ca. 110 Manntagen und 30.000 € zu rechnen. Nachdem die notwendige, formale Qualität der Spezifikation erreicht ist, kann in den folgenden Jahren von deutlich geringeren Aufwendungen ausgegangen werden. Um weitere, neu identifizierte Fehler zu korrigieren, sind jährlich ca. 10 Manntage (2.727 €) zu erwarten. Die beschriebenen Kosten zur formalen Korrektur der Spezifikation fallen bei Weiterführung der bisherigen Vorgehensweise ohne MBT nicht an.

Anderes gilt für die Testfallgenerierung. Für die manuelle Erstellung der Testfälle ist viel Zeit erforderlich. Aus diesem Grund wird das Arbeitspaket von externen Fachkräften abgearbeitet, deren Stundensatz hier mit 60 €/h veranschlagt wird. Für die Initiale Testfallerstellung in einem Entwicklungsprojekt sind 200 Manntage und 96.000 € zu erwarten. Für die in den folgenden Jahren notwendigen Anpassungen der Testfälle, aufgrund geänderter Spezifikationen, sind 75 Manntage (36.000 €) einzuplanen. Hinzu kommt die Organisation der Testfälle und Korrekturen, wofür jährlich weitere 40 Manntage (19.200 €) erforderlich sind.

In der Testfallgenerierung werden die Vorteile der modellbasierten Vorgehensweise deutlich. Die Generierung neuer Testfälle und Deltagenerierung bei Änderungen der Spezifikation erfordern hier jährlich ca. 10 Manntage,

zuzüglich der für das Generierungswerkzeug anfallenden Lizenzkosten (2.727 € + 800 €). Hinzu kommt die Wartung des Generierungswerkzeugs sowie die Korrektur und Organisation der Testfälle. Im ersten Jahr ist hierbei noch mit höheren Aufwendungen von rund 30 Manntagen (8.182 €), in den folgenden Jahren mit ca. 20 Manntagen (5.455 €) zu kalkulieren.

Im nächsten Schritt ist die automatische Durchführung des Dialogtests zu betrachten. Ein erfahrener Tester kann circa 80 Testfälle pro Tag manuell abarbeiten. Für die ihm vorgelegten 800 Testfälle sind folglich ca. 10 Manntage notwendig. Werden diese drei Mal im Jahr ausgeführt, ergeben sich 30 Manntage zu 8.182 €.

Demgegenüber steht die automatische Testdurchführung. Es ist davon auszugehen, dass insbesondere im ersten Jahr des Einsatzes noch verschiedene Schwierigkeiten auftreten, welche den Einsatz von circa 30 Manntagen (8.182 €) erfordern. Mit zunehmender Erfahrung kann erwartet werden, dass sich der notwendige Betreuungsaufwand reduziert und im Weiteren jährlich 20 Manntage (5.455 €) einzuplanen sind. Ergänzend ist zu beachten, dass für eine automatische Testdurchführung die Testumgebung um Hardwarekomponenten erweitert werden muss. Bei einer Projektlaufzeit von drei Jahren sind jährlich 2.000 € für Testequipment abzuschreiben.

Die Auswertung der durchgeführten Testfälle wird sich durch die modellbasierte Vorgehensweise zunächst kaum verändern. Zwar besteht die Möglichkeit, durch die automatisierte Durchführung Fehler mit geringerem Zeitaufwand zu identifizieren, jedoch wird (zunächst) die Notwendigkeit bestehen bleiben, detaillierte Fehlerbeschreibungen manuell zu erstellen.

Aus heutiger Sicht ist zu erwarten, dass in der Auswertung durchgeführter Testaktivitäten mittelfristig keine kostenrelevanten Veränderungen stattfinden. Der beschriebene Ressourceneinsatz ist in Abhängigkeit der Vorgehensweise in Tabelle 10 dargestellt.

Tabelle 10: Kostenbetrachtung: MBT vs. manueller Test der akustischen MMS

Kostenbetrachtung manuelles Vorgehen	**Jahr 1**		**Jahr 2 bis x**	
Positionen	**Manntage**	**Kosten**	**Manntage**	**Kosten**
Spezifikation/Modell	-	-	-	-
Testfallgenerierung, Deltagenerierung, Lizenzen	200	96.000 €	75	36.000 €
Korrektur, Wartung, Organisation der Testfälle	-	-	40	19.200 €
Testdurchführung	30	8.182 €	30	8.182 €
Summe	230	104.182 €	145	63.382 €
Kostenbetrachtung MBT	**Jahr 1**		**Jahr 2 bis x**	
Positionen	**Manntage**	**Kosten**	**Manntage**	**Kosten**
Spezifikation/Modell	110	30.000 €	10	2.727 €
Testfallgenerierung, Deltagenerierung, Lizenzen	10	3.527 €	10	3.527 €
Korrektur, Wartung, Organisation der Testfälle	30	8.182 €	20	5.455 €
Testdurchführung + HW	30	10.182 €	20	7.455 €
Summe	180	51.891 €	60	19.164 €
Kostenersparnis modellbasiertes	**52.291 €**		**44.218 €**	

Aus der Tabelle geht hervor, dass durch die Anwendung des entwickelten modellbasierten Vorgehens zum Test der akustischen MMS, im ersten Jahr ein Kostenvorteil von ca. 52.000 € entsteht. Im Folgenden bleibt jährlich ein

Vorteil von ca. 44.000 € bestehen. In der Annahme einer Projektlaufzeit von drei Jahren ergibt sich schließlich ein Kostenvorteil von ca. 140.000 €. In der Übersicht wird deutlich, dass durch das modellbasierte Vorgehen zusätzliche Kosten zur Sicherstellung der formalen Qualität der Spezifikation entstehen. Gleichzeitig werden die enormen Kostenvorteile in der Testfallgenerierung ersichtlich. Die Kosten der Testdurchführung halten sich in beiden Vorgehensweisen weitgehend die Waage. Allerdings ist hier zu berücksichtigen, dass die Anzahl der durchgeführten Testfälle durch die automatisierte Vorgehensweise flexibel und mit nur geringen Mehrkosten gesteigert werden kann. Auch die Investitionen zur formalen Korrektur der Spezifikation führen zu einer Qualitätsverbesserung, welche ohne die Umsetzung des MBT-Prozesses nicht gegeben ist. In Abbildung 75 ist die zu erwartende Kostenentwicklung nach Einführung der modellbasierten und automatisierten Vorgehensweise im Test, für die verschiedenen Prozesselemente grafisch dargestellt.

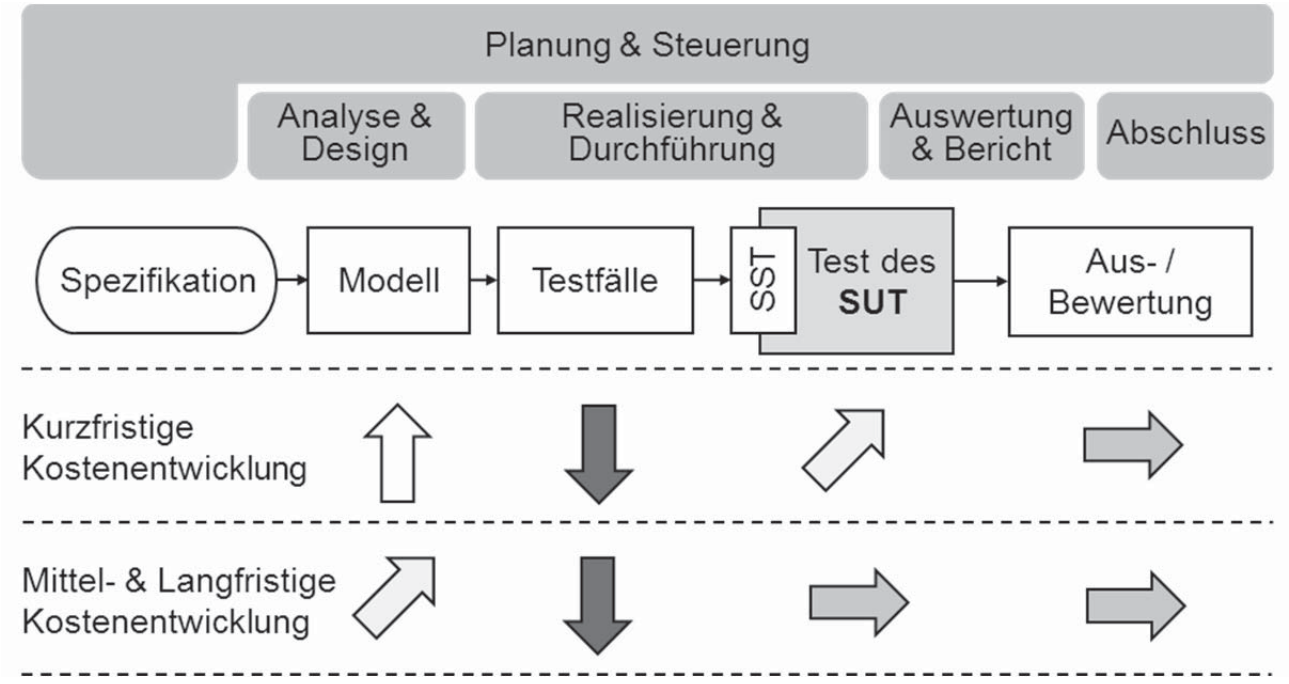


Abbildung 75: Erwartete Kostenentwicklung durch MBT

Zur Vervollständigung der Kostenbetrachtung sind auch die Entwicklungskosten von Bedeutung. Tabelle 11 gibt eine Übersicht über die verschiedenen Kostenpunkte. Gut die Hälfte der Entwicklungsaufwände entfällt auf die Personalkosten. Die weiteren Kosten verteilen sich auf die Beschaffung und Anpassung des Werkzeugs zur Testfallgenerierung, dessen Lizenz sowie die Weiterentwicklung der Testumgebung, um den automatischen Dialogtest zu realisieren. Bei Entwicklungskosten von insgesamt rund 193.000 € wird sich die Umsetzung der modellbasierten Vorgehensweise voraussichtlich im fünften Anwendungsjahr amortisieren. In der Annahme, dass ein Entwicklungsprojekt nach drei Jahren abgeschlossen ist und weitere Projekte folgen, amortisiert sich die Vorgehensweise bereits mit Beginn des zweiten Projekts.

Es zeigt sich: Die modellbasierte Vorgehensweise zum Test der akustischen MMS ist effizienter als der bisherige Testprozess.

Tabelle 11: Entwicklungskosten des MBT der akustischen MMS

Entwicklungskosten		
Positionen	**Manntage**	**Kosten**
Doktorand & Studenten	880	105.600 €
Testfallgenerierungswerkzeug und Anpassung	91	58.100 €
Lizenz Testfallgenerator	---	5.800 €
Testumgebung für automatischen Dialogtest	55	23.400 €
Summe	**1026**	**192.900 €**

Um die Wirtschaftlichkeit zu gewährleisten, ist der in Kapitel 3.2 formulierte Grundsatz zu berücksichtigen. Testprozess, Methoden und Werkzeuge zum

Test müssen ohne große Aufwendungen auf folgende Entwicklungsprojekte übertragen werden können. Ist dies gewährleistet, kann der Breakeven-Point nach 4-5 Jahren erreicht werden. Mit dieser Feststellung ist die Gültigkeit der zentralen These zur Lösung des Spannungsfeldes zwischen Qualität, Zeit und Kosten in der Entwicklung einer akustischen MMS im Kfz (Kapitel 3.2) in Gänze bewiesen.

8.3 Bewertung und Ausblick

Die Gültigkeit des in Kapitel 3.2 formulierten Ansatzes zur Lösung der Herausforderungen in der funktionalen Qualitätsabsicherung einer akustischen MMS im Kfz, konnte aufgezeigt werden. In der Beweisführung wurden verschiedene Lösungen entwickelt, welche in ihrer Gesamtheit eine modellbasierte Vorgehensweise im Test der akustischen MMS ermöglichen. Abschließend sollen die einzelnen Ergebnisse hinsichtlich ihres Potenzials, aber auch der vorhandenen Risiken beurteilt werden. Hierdurch wird auch ein realistischer Ausblick in die weitere produktive Verwendung erreicht.

Um eine Basis für die Umsetzung des MBT zu schaffen, stand zunächst der Entwicklungs- und Testprozess im Zentrum der Betrachtung. Durch eine Gegenüberstellung der konkret notwendigen Aktivitäten im Test einer softwarebasierten MMS wurde ein grundsätzliches Verständnis für die modellbasierte Vorgehensweise geschaffen. Durch die Anwendung des TPI-Modells konnten die bestehenden Testprozesse auf Basis einer anerkannten und systematischen Methodik bewertet, um notwendige Verbesserungen zielorientiert umgesetzt werden. Abgeleitete, konkrete Verbesserungsmaßnahmen wurden angestoßen, in der Praxis allerdings noch nicht vollstän-

dig umgesetzt. Es ist zu berücksichtigen, dass ein systematischer und strukturierter Testprozess nur durch den Einsatz entsprechender Ressourcen implementiert und gesteuert werden kann. Setzt sich dieses Verständnis durch, lassen sich die Ziele auf Basis der erarbeiteten Grundlagen erreichen.

In der Umsetzung des MBT fokussierten sich die Aufwendungen im Folgenden auf die Modellierung der akustischen MMS. Die entwickelte Syntax zur Modellierung erfüllt die zuvor definierten Anforderungen. Die verhältnismäßig einfache Modellierung mit definierten Regeln, die Wiederverwendbarkeit von Modellelementen durch Modularität und die Verwendung eines Standardformats schaffen die Voraussetzungen für ein praktikables Testmodell. Es zeigt sich aber auch, dass die Modellierung auf Basis der Spezifikation nicht immer trivial ist. Weitere Ressourcen sind zu investieren, um die Spezifikation korrekt zu interpretieren und im Testmodell entsprechend umzusetzen. Ein ausgeprägtes Verständnis für formale Zusammenhänge und logisches Denkvermögen ist erforderlich, um erforderliche Testmodelle zu erstellen. Bestehen bleibt der nicht zu unterschätzende Vorteil, dass durch den Ressourceneinsatz und menschliche Denkleistung die inhaltliche und formale Qualität der Spezifikation nachweislich verbessert wird. Fehler werden früh im Entwicklungsprozess identifiziert, wodurch Fehler in der Implementierung vermieden werden können.

Die Übertragung der entwickelten Modellierungssyntax zur Beschreibung der haptischen MMS zeigt die Grenzen einer „einfachen“ Modellierung auf. Während die Inhalte des KI mit den geschaffenen Möglichkeiten modelliert werden können, führt die Modellierung der HU-MMS zu komplexen und unübersichtlichen Modellen. Um eine modellbasierte Vorgehensweise für die

haptische MMS der HU zu entwickeln, sind weitere, umfangreiche Aufwendungen in die Entwicklung einer entsprechenden Modellierungssyntax zu investieren. Dabei ist aus heutiger Sicht fraglich, ob die ressourcenintensive Testmodellerstellung der gesamten HU-MMS effizient umgesetzt werden kann und wirtschaftlich zu rechtfertigen ist. Sinnvoll scheint eher die begrenzte Modellierung ausgewählter, testrelevanter Systemteile.

Mit der Auswahl und Implementierung des kommerziellen Testfallgenerators „MBTsuite“ konnte ein weiterer, zentraler Schritt zum MBT der akustischen MMS umgesetzt werden. Das Werkzeug erfüllt die zuvor definierten Anforderungen umfassend. Auf Basis des entwickelten Testmodells lassen sich nach verschiedenen Strategien zielorientiert, flexibel und systematisch Testfälle zum funktionalen Test generieren. Auch aus komplexeren Modellen können, durch Auswahl einer entsprechenden Strategie, Testfälle generiert werden. Zur Umsetzung neuer Teststrategien wird die Einführung von Prioritäten oder Übergangswahrscheinlichkeiten im Modell als sinnvoll erachtet. Die Vergabe zweckmäßiger Prioritäten im akustischen Dialog kann in weiterführenden Arbeiten vertieft werden. Neben der testmodellorientierten Vorgehensweise ist zudem eine systemmodellorientierte Vorgehensweise entwickelt und umgesetzt, in welcher die Spezifikation der akustischen MMS als Basis für die Testfallgenerierung Anwendung findet. Aufgrund der in Kapitel 6.3 erörterten Vor- und Nachteile ist davon auszugehen, dass sich in der Praxis der systemmodellorientierte Ansatz durchsetzt. Hierbei ist dringend zu berücksichtigen, dass umfangreiche Ressourcen notwendig werden, um die formale Qualität der Spezifikation zu verbessern, damit korrekte und sinnvolle Testfälle generiert werden können. Diese Investition ist zeitnah und konzentriert zu

tätigen, um eine effiziente und praxistaugliche Testfallgenerierung zu realisieren. Das Risiko in der praktischen Umsetzung des MBT liegt in der Vernachlässigung oder dem Aufschub dieser Aktivität. Mit der Qualität des für den Testprozess zentralen Modells ist die Qualität der Testfallgenerierung direkt verknüpft. Kann die erforderliche Qualität nicht innerhalb eines annehmbaren Zeitrahmens erreicht werden, besteht die Gefahr, dass die entwickelte Vorgehensweise insgesamt an Akzeptanz verliert.

Auch aus den für die haptische MMS erstellten Testmodellen können durch die „MBTsuite“ Testfälle generiert werden. In der Praxis kann dies für die KI-MMS nach Ergänzung von Prioritäten im Modell realisiert werden. Eine Testfallgenerierung aus dem Modell der HU-MMS ist zwar grundsätzlich denkbar, aufgrund der Modellcharakteristik jedoch kritisch. Ohne eine Weiterentwicklung der Modellierungssyntax ist die Testfallgenerierung nur aus stark vereinfachten Modellen realisierbar. Gegebenenfalls sind hier alternative Werkzeuge zur Testfallgenerierung zu bevorzugen.

Mit dem Ziel, die Testdurchführung der generierten Testfälle für die Verifikation der akustischen MMS effizient zu gestalten, wurde eine Test-umgebung entwickelt, welche den menschlichen Tester simuliert und die MMS zum Test des Systems verwendet. Auf eine explizite Testschnittstelle kann verzichtet werden. Um diesen automatischen Dialogtest zu realisieren, wurden zunächst funktionale Tests mit einfachen, abstrakten Testfällen in der neuen Testumgebung umgesetzt. Die schnellen und kostengünstig zu wiederholenden Testdurchläufe steigern die Effizienz im Test.

Der automatisierte Dialogtest durch einen „virtuellen Tester“ besitzt das Potenzial die Effizienz der Testdurchführung zu erhöhen. Auf dem Weg zur Praxistauglichkeit bestehen aber verschiedene Herausforderungen. Sollen viele verschiedene Dialoge automatisiert durchgeführt werden, sind Preconditions automatisiert herzustellen. Aus technischen Gründen wird ein erheblicher Anteil der Preconditions nicht automatisch herbeigeführt werden können. Mit einer sinnvollen Sortierung der Testfälle in Testsuiten wird die Problematik jedoch voraussichtlich zu lösen sein. Der Ressourcenaufwand zur Organisation der Testfälle und deren Anreicherung mit konkreten Testdaten im Falle variabler Eingabemöglichkeiten ist dabei nicht zu unterschätzen. Der Umstand, dass Testfälle automatisiert durchgeführt werden können, bedeutet nicht, dass die Testdurchführung zu einem „Selbstläufer“ wird. In der Praxis werden qualifizierte Testingenieure mit Kenntnissen zu System und Testumgebung erforderlich sein, um den automatischen Dialogtest zu betreuen. Ergänzend ist zu berücksichtigen, dass im Fehlerfall die Testumgebung zu prüfen ist, um diese als Fehlerursache auszuschließen.

Angesichts der Rahmenbedingungen bleibt abzuwarten, wie groß der Effizienzgewinn durch den automatischen Dialogtest ausfällt. Werden die durch die Generierung der Testfälle freigespielten Ressourcen in die Verbesserung des Systemmodells und die Weiterentwicklung der automatischen Testdurchführung investiert, erreicht die funktionale Verifikation der akustischen MMS eine neue Qualität. Das Spannungsfeld zwischen Qualität, Zeit und Kosten wird zwar bestehen bleiben, durch die entwickelte Methodik und die gesteigerte Effizienz im Testprozess können diese Spannungen aber merklich reduziert werden.

9 Zusammenfassung

Die softwarebasierte Mensch-Maschine-Schnittstelle (MMS) wurde in den vergangenen Jahren zu einem zentralen Bedienelemente im Kraftfahrzeug. Immer komplexere Infotainmentsysteme mit hoher Funktionsdichte erfordern eine stetige Erweiterung und Verbesserung der MMS hinsichtlich des Bedienkomforts und der Bediensicherheit. In diesem Kontext ist die akustische MMS zur Sprachbedienung der Infotainmentsysteme zunehmend in den Vordergrund gerückt. Durch die sich verkürzenden Entwicklungszyklen, steigenden Kostendruck und die hohe Funktionsdichte neuer Systeme ist die funktionale Qualität der akustischen MMS mit den bisher angewandten Prozessen und Methoden schon heute nur schwer zu gewährleisten.

Zur Lösung dieser Herausforderung ist eine These formuliert, nach welcher es möglich ist, die akustische MMS eines Infotainmentsystems im Kfz modellbasiert und automatisiert zu testen. Hierbei soll die MMS selbst als Testschnittstelle Verwendung finden, wodurch ein „virtueller Tester“ entsteht. Die These beschreibt weiter, dass Effizienz und Qualität des Testprozesses durch die Implementierung der neuen Prozesse, Methoden und Werkzeuge gesteigert wird.

Für die Beweisführung stehen zunächst die Eigenschaften und Anforderungen des Modellbasierten Tests (MBT) im Vordergrund. Durch ein modellbasiertes Vorgehen können Testaktivitäten zu einem früheren Zeitpunkt beginnen und mögliche Fehler bereits in einer frühen Projektphase identifiziert werden. Die Gegenüberstellung der Prozesse zum manuellen-, automatischen- und MBT

zeigt, dass eine Weiterentwicklung zum MBT auf nahezu alle Testaktivitäten Einfluss hat und das Verständnis zum Test ganzheitlich verändern kann. Um diese Veränderung zielorientiert umzusetzen, findet das Prozessmodell des „Test Process Improvement“ (TPI) Verwendung. Die vorhandenen Testprozesse werden auf ihre Reife hin beurteilt und notwendige Verbesserungsmaßnahmen definiert. Die Ergebnisse zeigen, dass sich die reale Prozessreife vorhandener Testprozesse auf einem zufriedenstellenden Niveau bewegt. Durch die Umsetzung der abgeleiteten, konkreten Verbesserungsvorschläge können vorhandene Defizite im Prozess abgestellt werden. Damit sind die Voraussetzungen zum MBT der akustischen MMS im Kfz erfüllt und die Modellierung rückt als zentrales Element des MBT in den Fokus der Beweisführung.

Nach einer Analyse bestehender Möglichkeiten zur Modellierung, werden Anforderungen an das Modell formuliert. Auf deren Basis wird eine Modellierungssyntax entwickelt, welche die formale Modellierung der akustischen MMS eines Infotainmentsystems ermöglicht. In der praktischen Anwendung des entwickelten Formalismus werden sowohl Herausforderungen in der Modellierung erkannt und gelöst, als auch Potenziale der Modellierung identifiziert. Es kann bewiesen werden, dass die Modellierung der akustischen MMS im Kraftfahrzeug als formales Zustandsdiagramm möglich ist. Damit ist die Basis geschaffen, um den MBT-Prozess einzuführen und den virtuellen Tester zu realisieren. Ergänzend wird geprüft, ob die entwickelte Modellierungssyntax auf die haptische MMS übertragen werden kann. Hierbei zeigt sich, dass dies für die MMS des Kombiinstruments (KI) ohne großen Aufwand realisiert werden kann. Die MMS der Head Unit (HU) erfordert aufgrund der

vielschichtigen Dialoge mit zahlreichen Abhängigkeiten und Eingabemöglichkeiten komplexere Modellierungsmöglichkeiten.

Im nächsten Schritt sind auf Basis des entwickelten Modells der akustischen MMS Testfälle zu generieren. Mit diesem Ziel werden verschiedene Werkzeuge zur Testfallgenerierung evaluiert. Nach der Bestimmung relevanter Anforderungen werden die Werkzeuge einem Praxistests unterzogen. Datenbasis sind die bereits erstellten Testmodelle. Das Werkzeug „MBTsuite“ wird schließlich als jenes mit der höchsten Anforderungsüberdeckung identifiziert. In der Folge werden verschiedene Generierungsstrategien bewertet und Wege einer effizienten Vorgehensweise ausgearbeitet. Es kann bewiesen werden, dass durch Anpassungen der „MBTsuite“ neben der testmodellorientierten Vorgehensweise auch die Testfallgenerierung aus dem vorhandenen Systemmodell der akustischen MMS umgesetzt werden kann. Die zentralen Elemente des MBT sind damit realisiert. Die entwickelte Vorgehensweise zur Testfallgenerierung wird auch auf die haptische MMS übertragen. Individuelle Modellcharakteristika erfordern hier abweichende Teststrategien. Für die komplexen Modelle der HU-MMS ist die Vorgehensweise nur eingeschränkt geeignet.

Um die Effizienz der Testdurchführung zu steigern, wird schließlich eine Testumgebung zur automatischen Ausführung der generierten Testfälle umgesetzt. In diesem Kontext rückt die Verwendung einer expliziten Testschnittstelle zur Ansteuerung des Infotainmentsystems in den Fokus. Vor- und Nachteile der Schnittstelle werden erörtert, wobei die Risiken in deren Verwendung schließlich überwiegen. Aus diesem Grund wird zunächst ein „einfacher“ Funktionstest des Sprachdialogsystems (SDS) in eine neue Test-

umgebung transferiert und so ein automatischer Test ohne die Verwendung einer expliziten Testschnittstelle realisiert. Auf Basis der hierdurch erlangten Erkenntnisse wird schließlich der zum Ziel gesetzte automatische Dialogtest des SDS entwickelt. Hierfür wird die vorhandene Testumgebung um die Möglichkeit der akustischen Systemeingabe und der Erkennung von akustischen Systemausgaben erweitert. Durch die Vorgehensweise wird der automatische Dialogtest realisiert und ein virtueller Tester entwickelt, welcher ausschließlich die MMS zum Test des SDS verwendet.

Abschließend findet eine Bewertung und Wirtschaftlichkeitsbetrachtung der entwickelten Methodik statt. Hierbei kann gezeigt werden, dass die Effizienz des Prozesses zum Test der akustischen MMS durch die entwickelte, modell-basierte Vorgehensweise gesteigert wird. Die zu Grunde liegende These ist damit bewiesen und die Herausforderungen im Spannungsfeld zwischen Qualität, Zeit und Kosten können mit dem entwickelten Ansatz entschärft werden.

10 Abstract

During the last years, the software based Human-Machine-Interface (HMI) became more and more important in operating the functions of an automobile. Complex Infotainmentsystems, packed with different applications require expanded and improved HMI concepts to improve ease of use as well as safety. Due to this trend, the acoustic HMI came to the fore. Rapid development, increasing cost pressure and various functions, implemented in new generations of Infotainmentsystems are the reason for current issues. Traditional processes and methods in testing are hardly capable to verify the functional quality in this environment.

To solve this challenge, a basic thesis is formulated. The thesis says it is possible to test the acoustic HMI of an automotive Infotainmentsystem using model-based and automated testing. At this, the HMI itself should be used as test-interface, creating a "virtual tester". The implementation of these new processes, methods and tools should improve the efficiency and quality of the test process.

Using a model-based approach, it is possible to start testing activities more early. Through this, there is a chance to identify errors at an early stage of a project. The comparison of the manual, auto-mated and model-based testing processes leads to the perception, that an implementation of MBT will have an effect on almost all testing activities and may change the understanding of testing in total. To realize the model-based approach in a target-oriented way, a process model, called "Test Process Improvement" (TPI) will be used. The existing test processes got assessed for their maturity to define

necessary activities for development. Overall, the results show a satisfying maturity level. Thereby the preconditions for model-based testing of the acoustic HMI in automobile are fulfilled and the modeling, as central component of the MBT is getting in focus.

After an analysis of existing options for modeling, requirements concerning the model are elaborated. Based on these requirements a specific syntax is developed, which allows formal modeling of the acoustic HMI of an Infotainmentsystem. In modeling practice several challenges are identified and solved as well as potentials in using the developed formalism. It can be proved that the acoustic HMI can be modeled, using a formal state diagram. Thereby, the basis to launch the MBT process and to realize the “virtual tester” is created In addition; it is analyzed whether the developed modeling syntax can be also used for the haptic HMI. It becomes apparent that this can be easily done for the HMI of the instrument cluster. Due to the complex dependencies of the existing dialogues with numerous input possibilities, the HMI of the Head Unit requires more extensive opportunities in modeling

The next step is to generate Test cases on the basis of the developed HMI model. With this objective, different tools for test case generation are evaluated. After the definition of relevant requirements, the tools are evaluated in practice, using the data of the already created test models. The tool "MBTsuite" is finally identified as the one with the highest coverage of requirements. Afterwards, various strategies for test case generation are evaluated to elaborate the path of the most efficient approach. By adapting the "MBTsuite", it can be proved; that there is not only the test model-driven approach, but also the possibility to use the existing specification of the

acoustic HMI for test case generation. With this, the essential elements of the model based testing approach are realized. The developed approach for test case generation is also transmitted to the haptic HMI. Unfortunately, specific model characteristics require different strategies in test case generation. For that reason the developed approach is of limited use for the complex models of the Head Unit HMI.

Finally a test environment is going to be implemented, which enables an automated execution of generated test cases to improve the efficiency in test execution. In this context, the use of an explicit test interface, for triggering the infotainment system, is in focus. Pros and cons of the explicit test interface are discussed, but the risks and difficulties in use prevail. To realize the automatic test execution without using an explicit test interface, a "simple" functional test case of the speech dialogue system (SDS) is transferred into a new test environment. Based on the results and experiences of this transfer, it is finally possible to develop the environment for an automated test execution of the SDS. For this, the existing test environment is extended for the ability of an acoustic system input and the recognition of the acoustic system outputs. By doing this, the automated test execution for SDS is realized, using only the HMI of the infotainment system. The “virtual tester” is created.

As a final point, the operating efficiency of the developed system is evaluated. An improved efficiency in testing, using the developed and implemented model-based approach can be shown. The basic thesis could be proved and the current challenges in development and verifying the acoustic HMI within time and cost limits can be faced.

11 Literaturverzeichnis

[Bas88] **Basili, V. R., & Perricone, B. T.** (1988). *The TAME project: Towards improvement-orientated software environments.* IEEE Transactions on Software Engineering, Vol. 14, No. 6.

[Bla03] **Black, R.** (2003). *Critical Testing Processes: Plan, Prepare, Perform, Perfect.* London: Addison-Wesley.

[Boe01] **Boer, E.** (2001). *Behavioral Entropy as a Measure of Driving Performance* (Bde. In: Proceedings of the International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design). Aspen.

[Boo06] **Booch, G., Rumbaugh, J., & Jacobson, I.** (2006). *Das UML-Benutzerhandbuch:* Version 2.0. München: Addison-Wesley.

[Bra08] **Bray, T.** (26. November 2008). W3C. Abgerufen am 05. 01 2013 von *Extensible Markup Language (XML)* 1.0 (Fifth Edition): http://www.w3.org/TR/xml/

[Bur96] **Burnstein, I., Suwannasart, T., & Carlson, C.** (1996). *Developing a Testing Maturity Model for Software test process evaluation and improvement.* International Test Conference, Test Conference, 1996. Proceedings. Altoona, PA.

[Cra02] **Craig, R., & Jaskiel, S.** (2002). *Systematic Software Testing.* Artech House Publishers.

[Cul11] **Culjat, D.** (23. Februar 1999). *DTW oder Dynamic-Time-Warping*. Abgerufen am 07. 05 2012 von: http://www.inf.fu-berlin.de/lehre/WS98/SprachSem/culjat/node4 .html

[Dan00] **Dangelmaier, M.** (2000). *Ein Verfahren zur ergonomischen Bewertung des Fahrerplatzes von Personenkraftwagen.* Stuttgart.

[DIN15008] **DIN EN ISO 15008**: *Straßenfahrzeuge - Ergonomische Aspekte von Fahrerinformations- Assistenzsystemen.* (2003).

[DIN9241] **DIN EN ISO 9241**: *Ergonomie der Mensch-System-Interaktion.* (2011).

[Dre55] **Dreyfuss, H.** (1955). *Designing for People.* New York: Allworth Press.

[Eul06] **Euler, S.** (2006). *Grundkurs Spracherkennung.* Wiesbaden: Vieweg.

[Eur10] **European Automobile Manufacturers Association, ACEA**. (2010). *The Automobile Industry - Pocket Guide.* ACEA Communications department.

[Fra03] **Frast, D.** (2003). *Testen und Testautomatisierung.* In D. Frast, Software Qualitätssicherung (Bd. Teil 2). Wien.

[Göt09] **Götz, H., Nickolaus, M., Roßner, T., Salomon, K., & Winter, M.** (2009). *iX Studie - Modellbasiertes Testen.* Heise.

[Gra96] **Graham, D., Herzlich, P., & Morelli, C.** (1996). *Computer Aided Software Testing* (Bde. The CAST-report). Cambridge Market Intelligence Limited.

[GTB12] **GTB**. German Testing Board. Abgerufen am 02. Dezember 2012 von German Testing Board: http://www.german-testing-board.info/de/downloads.shtm

[Ham09] **Hamerich, S.** (2009). *Sprachbedienung im Automobil - Teilautomatisierte Entwicklung benutzerfreundlicher Dialog-systeme.* Heidelberg: Springer.

[Hän04] **Hänsler, E.** (2004). *Acoustic Echo and Noise Control.* New York: John Wiley & Sons.

[Har02] **Harold, E.** (2002). *Die XML Bibel* (Bd. 2). mitp/bhv.

[Höh09] **Höhn, H., Sechser, B., Dussa-Zieger, K., Messnarz, R., & Hindel, B**. (2009). *Software Engineering nach Automotive SPICE: Entwicklungsprozesse in der Praxis.* dpunkt.verlag.

[Hör06] **Hörmann, K., Dittmann, L., Hindel, B., & Müller, M.** *SPICE in der Praxis: Interpretationshilfe für Anwender und Assessoren.* 2006: dpunkt.verlag.

[IEE90] **IEEE.** (1990). *IEEE Standard Glossary of Software Engineering Terminology* (Bd. IEEE 610). New York: The Institute of Electrical and Electronics Ehgineers .

[ISO04] **ISO/IEC 15004**. (2004). Information technology - Process assessment.

[Int09] **ISTQB**. (30. April 2009). *International Software Testing Qualifications Board.* Abgerufen am 24. Februar 2010 von International Software Testing Qualifications Board: http://www.istqb.org/downloads/viewcategory/20.html

[Jel98] **Jelinek, F.** (1998). *Statistical Methodes for Speech Recognition.* Cambridge, USA: MIT Press.

[Joh93] **Johannsen, G.** (1993). *Mensch-Maschine-Systeme.* Springer.

[Lac79] **Lachman R, J.L., L., & E.C., B.** (1979). *Cognitive Psychology and Information Processing: An Introduction.* Lawrence Erlbaum Assoc Inc.

[Lig09] **Liggesmeyer, P.** (2009). *Software-Qualität - Testen, Analysieren und Verifizieren von Software.* Heidelberg: Spektrum Akademischer Verlag.

[Lüt12] **Lütze, L., & Werner, S.** (2012). *Qualitätsabsicherung im Linguatronic Entwicklungsprozess - Modellbasiertes Testen auf Basis formaler Beschreibung von Sprachdialogsystemen.* Elektronische Sprachsignalverarbeitung (ESSV). Cottbus.

[Mag01] **Magnusson, K., Kroslid, D., & Bergman, B.** (2001). *Six Sigma: Die neue Qualitätsstrategie für Unternehmen.* Hanser.

[Mai12] **Maier, T., Schmid, M.** (2012). *Interface Design.* Stuttgart: Institut für Konstruktionstechnik und Technisches Design.

[Mie11] **Miethe, M.** (2011). *Modelling Guidelines for Enterprise Architect* (Bd. V1.0). sepp.med gmbh.

[Mye92] **Myers, B., & Rosson, M.** (1992). *Survey on user interface programming.* CHI'92: Proceedings of the SIGCHI conference on Human factors in computing systems. New York: ACM Press.

[Nor01] **Nordholm, S., Claesson, I., & Grbic, N.** (2001). *Adaptive Microphone Arrays for Speech Input in Automobiles.* (Bde. In: Brandstein, M. und Ward, D. (Hrsg.): Microphone Arrays) . Heidelberg: Springer.

[Obj11] **Object Management Group (OMG).** (März 2011). *Unified Modeling Language (UML),* 2.4. (OMG) Abgerufen am April 2011 von http://www.omg.org/spec/UML/

[Pal86] **Palmer, S., & Kimich, R.** (1986). *The Information Processing Approach to Cognition.* In T.J. Knapp \& L. Robertson (Eds.), Approaches to cognition: Contrasts and Controversies. Erlbaum.

[Pol02] **Pol, M., Koomen, T., & Spillner, A.** (2002). *Management und Optimierung des Testprozesses.* dpunkt.verlag.

[Ras86] **Rasmussen, J.** (1986). *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering.* North-Holland.

[Roß10] **Roßner, T., Brandes, C., Götz, H., & Winter, M.** (2010). *Basiswissen modellbasierter Test.* Heidelberg: dpunkt.Verlag.

[Rüh90] **Rühmann, H.-P., & Schmidtke, H.** (1990). *Gestaltung der Schnittstelle Mensch-Maschine.* Hogrefe Verlag für Psychologie.

[Rup09] **Rupp, C., & Spohisten.** (2009). *Requirements Engineering und - Management.* Nürnberg: Hanser Verlag.

[Rup05] **Rupp, C., Hahn, J., Queins, S., Jeckle, M., & Zengler, B.** (2005). *UML2 Glasklar. Praxiswissen für die UML-Modellierung* (Bd. 2 Auflage). München: Carl Hanser Verlag.

[Sch07] **Schieferdecker, I.** (2007). *Modellbasiertes Testen.* OBJEKTspektrum (03/07, pp.39-45).

[Sog09] **Sogeti Deutschland GmbH.** (2009). *TPI - Business Dirven Test Process Improvement.* Abgerufen am 28. April 2010 von TPI Automotive: http://www.tpiautomotive.de

[Spa11] **Spath, D., Weisbecker, A., J., B., Peissner, M., Hermann, F., & Hipp, C.** (2011). *Usability und Human-Machine-Interfeces in der Produktion*. Stuttgart: Fraunhofer Verlag.

[Spi03] **Spillner, A.** (2003). *Das W-Modell, Vorteile der agilen Prozesse in einem konservativen Umfeld nutzen.* Bremen/Oldenburg: Hochschule Bremen.

[Spi08] **Spillner, A., Roßner, T., Winter, M., & Linz, T.** (2008). *Praxiswissen Softwaretest - Testmanagement.* dpunkt.verlag.

[Sub10] **Subaru.** (2010). Subaru Deutschland. Abgerufen am 15. Oktober 2010 von Unternehmenskommunikation: www.subaru-presse.de/symmetrical-awd.html

[Tou09] **Toutenburg, H., & Knofel, P.** (2009). *Six Sigma - Methoden und Statistik für die Praxis.* Springer.

[Utt07] **Utting, M., & Legeard, B.** (2007). *Practical Model-Based Testing.* Elesvier.

[Utt06] **Utting, M., Pretschner, A., & Legeard, B.** (2006). *A Taxonomy of Model-Based Testing* (Bd. Working Paper Series). Hamilton, New Zealand.

[Var98] **Vary, P., Heute, U., & Hess, W.** (1998). *Digitale Sprachsignal-verarbeitung.* Stuttgart: Teubner Verlag.

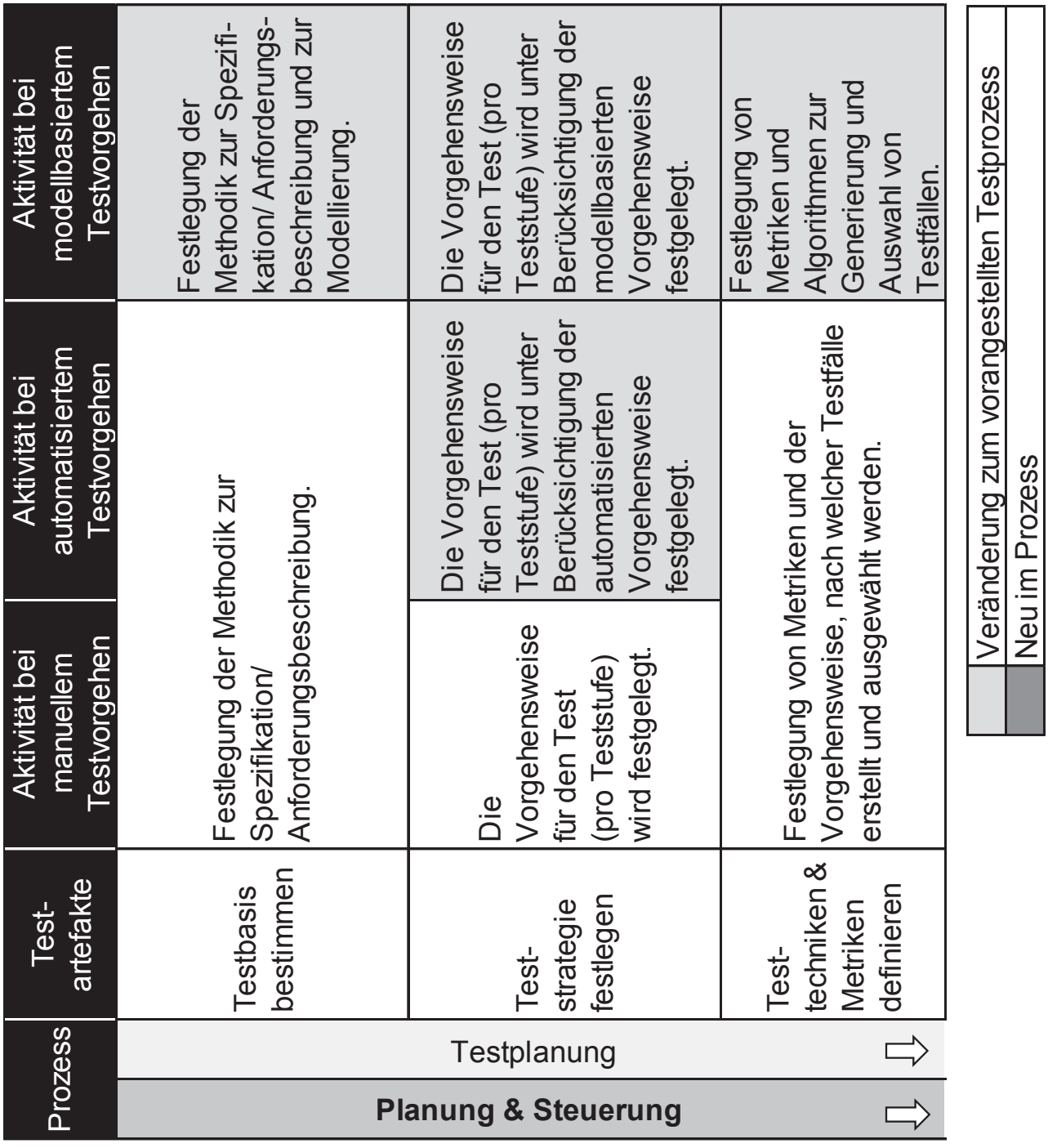
[Wes02] **Westkämper, E., & Warnecke, H.-J.** (2002). *Einführung in die Fertigungstechnik,* S.41. Stuttgart/Leipzig/Wiesbaden: B.G. Teubner.

[Wir04] **Wirsing, M.** (2004). *Black-Box-Test und White-Box-Test. In Methoden des Software Engineering.* München.

[Zim10] **Zimmermann, W., & Schmidgall, R.** (2010). *Bussysteme in der Fahrzeugtechnik – Protokolle, Standards und Software-architektur* (Bd. 4. Auflage). Vieweg+Teubner.

12 Anhang

12.1 Aktivitäten des manuellen, automatischen und modellbasierten Tests



Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Planung & Steuerung				
Testplanung	Testendkriterien bestimmen	Festlegung von Bedingungen, welche den Abschluss des Testprozesses definieren (z.B. die Testabdeckung bestimmter Anforderungen).		Festlegung von Bedingungen, welche den Abschluss des Testprozesses definieren (z.B. Modell-Überdeckungs-kriterien).
	Zeitliche Planung	Aktivitäten, Meilensteine, Abhängigkeiten, Start- und Enddaten werden festgelegt.	Aktivitäten, Meilensteine, Abhängigkeiten, Start- und Enddaten werden unter Berücksichtigung der automatisierten Tests festgelegt.	Aktivitäten, Meilensteine, Abhängigkeiten, Start- und Enddaten werden unter Berücksichtigung der Modellierung festgelegt.
	Kosten-voranschlag erstellen	Es wird ein Kostenvoranschlag für die Testaktivitäten erstellt.	Es wird ein Kostenvoranschlag für die Testaktivitäten erstellt. Dabei wird die Automatisierung berücksichtigt.	Es wird ein Kostenvoranschlag für die Testaktivitäten erstellt. Dabei wird die Modellierung berücksichtigt.

Veränderung zum vorangestellten Testprozess

Neu im Prozess

Prozess		Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Planung & Steuerung ⇩	Testplanung	Infrastruktur und Werkzeugunterstützung definieren	Die Testumgebung, Büroeinrichtung und Tools werden bestimmt.	Die Testumgebung, Büroeinrichtung und Tools zur Testfallerstellung und automatisierten Testdurchführung werden bestimmt.	Die Testumgebung, Büroeinrichtung und Tools zur Modellierung, Testfallgenerierung und Testdurchführung werden bestimmt.
		Verantwortlichkeiten festlegen	Zuweisung der Aufgaben, Kompetenzen und Verantwortung, sowie Festlegung der Hierarchie zur Berichterstattung.	Zuweisung der Aufgaben, Kompetenzen und Verantwortung, sowie Festlegung der Hierarchie zur Berichterstattung unter Berücksichtigung der automatisierten Vorgehensweise.	Zuweisung der Aufgaben, Kompetenzen und Verantwortung, sowie Festlegung der Hierarchie zur Berichterstattung unter Berücksichtigung der modellbasierten Vorgehensweise.
		Dokumentation der Planung	Die Ergebnisse der Testplanung werden im Testkonzept dokumentiert.		

	Veränderung zum vorangestellten Testprozess
	Neu im Prozess

Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Planung & Steuerung	Teststeuerung	Organisation der Testmittel	Ein Testmanagement zur Überwachung und Anpassung der Testmittel und -aktivitäten wird eingerichtet, sowie Change- und Versionsmanagement für Testbasis & Testfälle.	Ein Testmanagement zur Überwachung und Anpassung der Testmittel und -aktivitäten wird eingerichtet, sowie Change- und Versionsmanagement für Testbasis, Modell & Testfälle.
		Bericht-erstattung	Es findet eine zyklische Berichterstattung unter Verwendung der festgelegten Metriken statt.	
Analyse & Design ⇨	Infrastruktur	Aufbau der Testinfrastruktur	Realisierung der Infrastruktur zur Testdurchführung (z.B. Testmittel, Büroräume, usw.).	Realisierung der Infrastruktur zur durchführung automatischer Tests (z.B. Testmittel, -werkzeuge, Büroräume, usw.).
				Realisierung der Infrastruktur zur durchführung modellbasierter Tests (z.B. Testmittel & -werkzeuge, Modellierungswerkzeug e, Büroräume, Verfahren usw.)

Veränderung zum vorangestellten Testprozess

Neu im Prozess

Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Analyse & Design				
Modellierung	Testmodell/ Testbasis erstellen			Das als Testbasis vorgesehene Modell (Test- oder Systemmodell) zur Generierung der Testfälle wird erstellt.
Prüfung der Testbasis	Statische Tests durchführen	Prüfung der Spezifikation und Anforderungsdokumente ohne Ausführung der Software.		Prüfung der Spezifikation, Anforderungsdokumente und Modelle ohne Ausführung der Software.
Testfälle ⇨	Generierung abstrakter oder konkreter Testfälle			Generierung von Testfällen aus dem Modell mit oder ohne konkrete Ein- und Ausgabewerte. (In Abh. Der Verfügbarkeit).

	Veränderung zum vorangestellten Testprozess
	Neu im Prozess

Prozess
Testartefakte
Aktivität bei manuellem Testvorgehen
Aktivität bei automatisiertem Testvorgehen
Aktivität bei modellbasiertem Testvorgehen
Analyse & Design
Testfälle
Erstellung abstrakter oder konkreter Testfälle
Manuelle Erstellung konkreter oder abstrakter Testfälle auf Basis der Anforderungen oder explorativer und erfahrungsbasierter Kenntnisse.
Manuelle Erstellung konkreter oder abstrakter Testfälle oder Testskripte auf Basis der Anforderungen oder explorativer und erfahrungsbasierter Kenntnisse.
Eine zusätzliche, manuelle Erstellung konkreter oder abstrakter Testfälle oder Testskripte auf Basis des Modells oder der Anforderungen ist optional möglich.
Vor- und Nach-bedingungen erfassen
Der für die Testausführung eines bestimmten Testfalls notwendige Zustand des SUT und/oder dessen Umgebung, wird aus der Testbasis ermittelt. (Nachbedingungen analog).
Der für die Testausführung eines bestimmten Testfalls notwendige Zustand des SUT und/oder dessen Umgebung, wird aus der Testbasis/ dem Modell ermittelt. (Nachbedingungen analog).
Veränderung zum vorangestellten Testprozess
Neu im Prozess

Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Analyse & Design				
Testfälle ⇨	Reproduzierbarkeit der Testfälle sicherstellen			Die Generierung der Testfälle soll reproduzierbar sein. Der Algorithmus zur Generierung erfüllt diese Anforderung.
	Rückverfolgbarkeit sicherstellen	Es soll möglich sein, zusammengehörige Teile von Dokumentation und Software zu identifizieren, insbesondere Anforderungen mit den zugehörigen Testfällen.	Es soll möglich sein, zusammengehörige Teile von Dokumentation und Software zu identifizieren, insbesondere Anforderungen mit den zugehörigen Testfällen und/oder Testskripten.	Es soll möglich sein, zusammengehörige Teile von Dokumentation und Software zu identifizieren, insbesondere Anforderungen, zugehörign Testfälle und Modelle.

	Veränderung zum vorangestellten Testprozess
	Neu im Prozess

Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Analyse & Design	Sollwerte/ Testorakel aufbauen	Das festgelegte (Soll) oder vorausgesagte (Orakel) Verhalten eines Systems ist im Testfall, der Testbasis oder einer Datenbank zu hinterlegen.	Das festgelegte (Soll) oder vorausgesagte (Orakel) Verhalten eines Systems ist im Testfall, Testskript oder einer Datenbank zu hinterlegen.	Das festgelegte (Soll) oder vorausgesagte (Orakel) Verhalten eines Systems ist im Modell, Testfall, Testskript oder einer Datenbank zu hinterlegen.
Testfälle	Dokumentation der Testfallerstellung	Die Erstellung der Testfälle ist zu dokumentieren		Die Generierung der Testfälle ist zu dokumentieren
Realisierung ⇨	Konkrete Testfälle realisieren	Aus abstrakten Testfällen werden konkrete, ausführbare Testfälle erstellt.	Aus abstrakten und/oder konkreten Testfällen werden konkrete, ausführbare Testskripte erstellt oder generiert.	Aus abstrakten und/oder konkreten Testfällen werden konkrete, ausführbare Testfälle generiert. (falls nicht bereits vorhanden).
Realisierung Testfälle				

	Veränderung zum vorangestellten Testprozess
	Neu im Prozess

Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Realisierung	Testsequenzen / Testsuiten zusammenstellen	Testfälle werden zu Testsequenzen/ Testsuiten zusammengestellt. Dies kann in Abhängigkeit von für die Testdurchführung relevanten Bedingungen erfolgen.		
Organisation der Testfälle	Priorisierung der Tests	Die Testfälle/Testsuiten werden priorisiert und eine def. Reihenfolge zur Durchführung festgelegt		Die Testfälle / Testsuiten werden automatisch* priorisiert und eine def. Reihenfolge zur Durchführung festgelegt. (*Wenn das Modell Daten zu Prioritäten beinhaltet).
	Change-/Versions-management	Anforderungskonforme Testfälle werden sichergestellt. Die Beziehungen zwischen Testbasis, Testfall und SUT lassen sich über mehrere Versionen hinweg nachvollziehen.	Anforderungskonforme Testfälle werden sichergestellt. Die Beziehungen zwischen Testbasis, Testfall, Testskript und SUT lassen sich über mehrere Versionen hinweg nachvollziehen.	Anforderungskonforme Testfälle werden sichergestellt. Die Beziehungen zwischen Testbasis, Modell, Testfall und SUT lassen sich über mehrere Versionen hinweg nachvollziehen.

Veränderung zum vorangestellten Testprozess

Neu im Prozess

Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Realisierung	Prüfung der Testumgebung	Die Korrektheit der Testumgebung wird überprüft.		
Testausführung	Testfälle ausführen	Testfälle und freie Testaktivitäten werden manuell durchgeführt.	Testfälle werden durch entsprechende Tools automatisch ausgeführt.	
	Fortsetzung im Fehlerfall		Die Fortsetzung des Testdurchlaufs im Fehlerfall wird für den automatischen Test sichergestellt.	
	Exploratives/ Erfahrungs-basiertes Testen		Neben den Systematischen Tests werden explorative und erfahrungsbasierte Tests durchgeführt.	
	Soll-Ist Vergleich	Der Abgleich der Ist- und Soll-Werte findet parallel zur Testdurchführung oder im Anschluss statt.	Es findet ein Abgleich der Testergebnisse (Ist-Werte) mit den Sollwerten statt. Der Abgleich kann automatisch stattfinden.	

	Veränderung zum vorangestellten Testprozess
	Neu im Prozess

Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Realisierung				
Testausführung	Prüfung im Fehlerfall	Im Fehlerfall wird sichergestellt, dass der betroffene Testfall korrekt ist und als Fehlerquelle ausgeschlossen werden kann.	Im Fehlerfall wird sichergestellt, dass der betroffene Testfall und das Testskript korrekt sind und als Fehlerquelle ausgeschlossen werden können.	Im Fehlerfall wird sichergestellt, dass der betroffene Testfall, sowie das Modell korrekt sind und als Fehlerquelle ausgeschlossen werden können.
	Regressionstest	Erneutes Testen nach einer Modifikation. Ein Regressionstest wird		
Bericht-erstattung	Testprotokollierung	Protokollierung der durchgeführten Testfälle, inklusive aller Rahmenbedingungen und Einzelheiten der Testausführung.		
	Abweichungs-management	Prozess der Dokumentation, Analyse, Bearbeitung und Abschluss eines aufgedeckten Fehlerzustands. Es sind Werkzeuge zur Organisation und Synchronisation der aufgedeckten Fehler zu verwenden.		
Auswertung				
Verwendung Metriken ⇨	Produktmetriken einsetzen	Produktmetriken finden Verwendung.		Produktmetriken finden Verwendung, erweitert um Metriken zur Modellierung und Testfallgenerierung.

Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Auswertung	Prozessmetriken einsetzen	Prozessmetriken finden Verwendung.	Prozessmetriken finden Verwendung, erweitert um Metriken zur automatischen Testdurchführung.	Prozessmetriken finden Verwendung, erweitert um Metriken zur Modellierung und Testfallgenerierung.
Verwendung Metriken	Testabschluss-bericht weiterleiten	Testaktivitäten und -ergebnisse sind zusammenfasst. Der Bericht enthält eine Bewertung der durchgeführten Tests mit Bezug auf die definierten Testendekriterien.		
Abschluss	Lessons Learned	Review/Walkthrough des Testprozesses/der Testaktivitäten. Erfahrungen aus dem Projekt und Verbesserungspotential e fließen in folgende Projekte ein.	Review/Walkthrough des Testprozesses/ der Testaktivitäten. Erfahrungen aus dem Projekt und Verbesserungspotential e fließen in folgende Projekte ein. Dabei ist explizit auch die automatisierte Durchführung zu betrachten.	Review/Walkthrough des Testprozesses/ der Testaktivitäten. Erfahrungen aus dem Projekt und Verbesserungspotential e fließen in folgende Projekte ein. Dabei ist explizit auch die Modellierung, und Testfallgenerierung zu betrachten.
Abschluss der Aktivitäten ⇨				
⇨				

Veränderung zum vorangestellten Testprozess

Neu im Prozess

Prozess	Testartefakte	Aktivität bei manuellem Testvorgehen	Aktivität bei automatisiertem Testvorgehen	Aktivität bei modellbasiertem Testvorgehen
Abschluss	Archivierung der Testmittel	Die Testmittel werden archiviert, um Testbedingungen reproduzierbar und die Wiederverwendung möglich zu machen.	Die Testmittel (inkl. Tools zur automatischen Testdurchführung) werden archiviert um Testbedingungen reproduzierbar und die Wiederverwen-dung möglich zu machen.	Die Testmittel (inkl. Tools zur Model-lierung & Testfall-generierung) werden archiviert um Tes-tbedingungen reproduzierbar und die Wiederverwen-dung möglich zu machen.
Abschluss der Aktivitäten	Wieder-verwendung der Testmittel	Die Testmittel werden auf zukünftige Testprozesse übertragen.	Die Testmittel (inkl. Tools zur automatischen Testdurchführung) werden auf zukünftige Entwicklungsprojekte übertragen.	Die Testmittel (inkl. Tools zur Modellierung & Testfallgenerierung) werden auf zukünftige Entwicklungsprojekte übertragen.
	Abschlussbericht	Dokument zur abschließenden Zusammenfassung des gesamten		

Veränderung zum vorangestellten Testprozess

Neu im Prozess

12.2 Werkzeuge zur Modellierung in UML (Übersicht)

UML Werkzeug	Anbieter	Quelle/ Webseite	Version	Klassendiagramm	Objektdiagramm	Verteilungsdiagramm	Profildiagramm	Paketdiagramm	Komponentendiagramm	Kompositionsstrukturdiagramm	Zustandsdiagramm	Aktivitätsdiagramm	Interaktionsübersichtsdiagramm	Sequenzdiagramm	Zeitverlaufsdiagramm	Kommunikationsdiagramm	Anwendungsfalldiagramm	XMI export	XMI import
Altova Umodel	Altova Inc.	altova.com	2012.2	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
AnyLogic	XJ Technologies	anylogic.com	6.8.0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	
ArgoUML	Open Source	argouml. tigris.org	0,34	×	×	×		×	×		×	×		×		×	×	×	×
Astah	Change Vision, Inc.	astah.net	6.6.3	×	×	×		×	×	×	×	×		×		×	×	×	×
EclipseUML	Omondo	ejb3.org	3,8	×	×	×	×	×	×	×	×	×		×		×	×	×	×
Enterprise Architect	Sparx Systems	sparx systems.eu	9.3	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
MetaEdit+	MetaCase	metacase .com	5.0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Metamill	Metamill Software	metamill.com	6	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Modelio	Modeliosoft	mode liosoft.com	2.2	×	×	×	×	×	×	×	×	×	×	×		×	×	×	×
ModelMaker	ModelMaker Tools	modelmaker tools.com	11.2	×	×	×		×	×	×	×	×	×	×		×	×		

UML Werkzeug	Anbieter	Quelle/ Webseite	Version	Klassendiagramm	Objektdiagramm	Verteilungsdiagramm	Profildiagramm	Paketdiagramm	Komponentendiagramm	Kompositionsstrukturdiagramm	Zustandsdiagramm	Aktivitätsdiagramm	Interaktionsübersichtsdiagramm	Sequenzdiagramm	Zeitverlaufsdiagramm	Kommunikationsdiagramm	Anwendungsfalldiagramm	XMI export	XMI import
objectiF	microTOOL	objectiF.de	7.1	x			x	x	x		x	x		x			x	x	x
Poseidon for UML	Gentleware GmbH	gentle ware.com	8.0	x	x	x		x	x		x	x		x			x	x	x
RTDS	PragmaDev	pragmadev.com	V4.31	x		x				x	x		x	x					x
SAP Sybase	Sybase GmbH	mypower designer.com	16. Jan	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x
Sinelabore Codgen	Sinelabore.com	sine labore.com	08/02								x								x
UML Lab	Yatta Solutions GmbH	uml-lab.com	1.4.3	x			x	x										x	x
UMLet	UMLet Team	umlet.com	11.5.1	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
Visual Paradigm	Visual Paradigm	visual-paradigm.com	9	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Visual Classworks	Step Ahead W	visual classworks.com	7.5.1.3	x															
WinA&D	Excel Software	excel software.com																x	

12.3 Tools zur Testfallgenerierung – Bewertungsmatrix

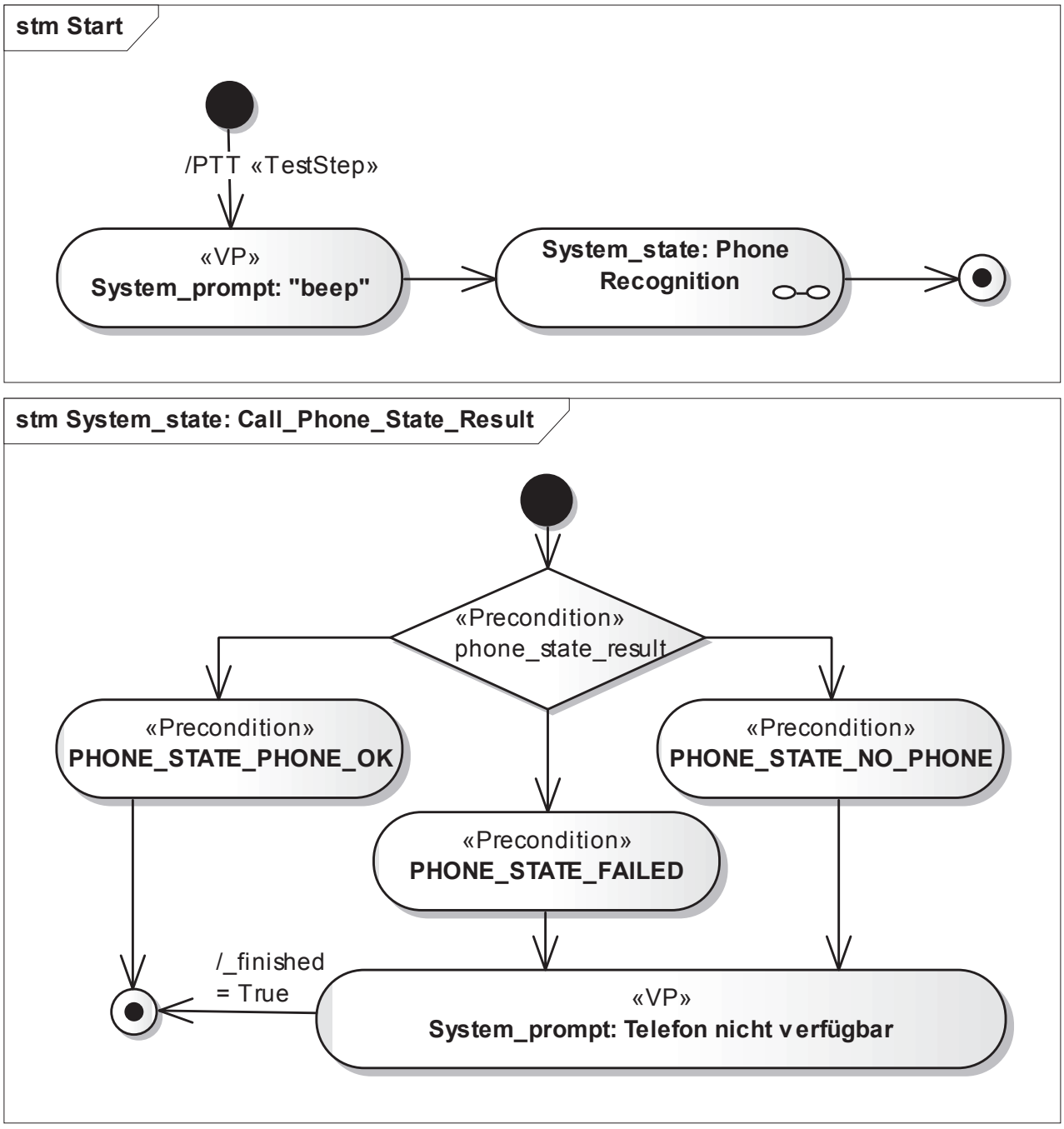
		Conformiq	MaTeLo	Rational	.getmore	TCGen	Certifyit
Kosten (€) & Support		28000	21000	20000	9000	verfügbar	20000*
Preis/Lizenz	3	1	1	2	4	5	4
Schulung und Support	*1*	4	3	4	5	5	3
Wertung		**35,0%**	**30,0%**	**50,0%**	**85,0%**	**100,0%**	**75,0%**
Ressourcenbedarf/ Kompatibilität							
Komp. zu WIN XP/ WIN7	2	5	5	5	5	5	5
Ressourcenbedarf und Stabilität	*1*	3	5	4	4	3	3
Wertung		**86,7%**	**100,0%**	**93,3%**	**93,3%**	**86,7%**	**86,7%**
Usability							
Einarbeitungsaufwand	3	3	3	2	4	2	4
Usability	3	3	4	1	4	3	4
Sprachen: GER & ENG	3	3	3	3	3	3	3
Wertung		**60,0%**	**66,7%**	**40,0%**	**73,3%**	**53,3%**	**73,3%**
Schnittstellen an externe Software/ Standards							
Import/Export XMI/XML	3	4	2	4	4	4	3
Schnittstellen zu externer SW	2	4	1	3	5	4	3
Anbindung an Werkzeug für Testfallmanagement	2	5	3	3	5	0	5
Wertung		**85,7%**	**40,0%**	**68,6%**	**91,4%**	**57,1%**	**71,4%**

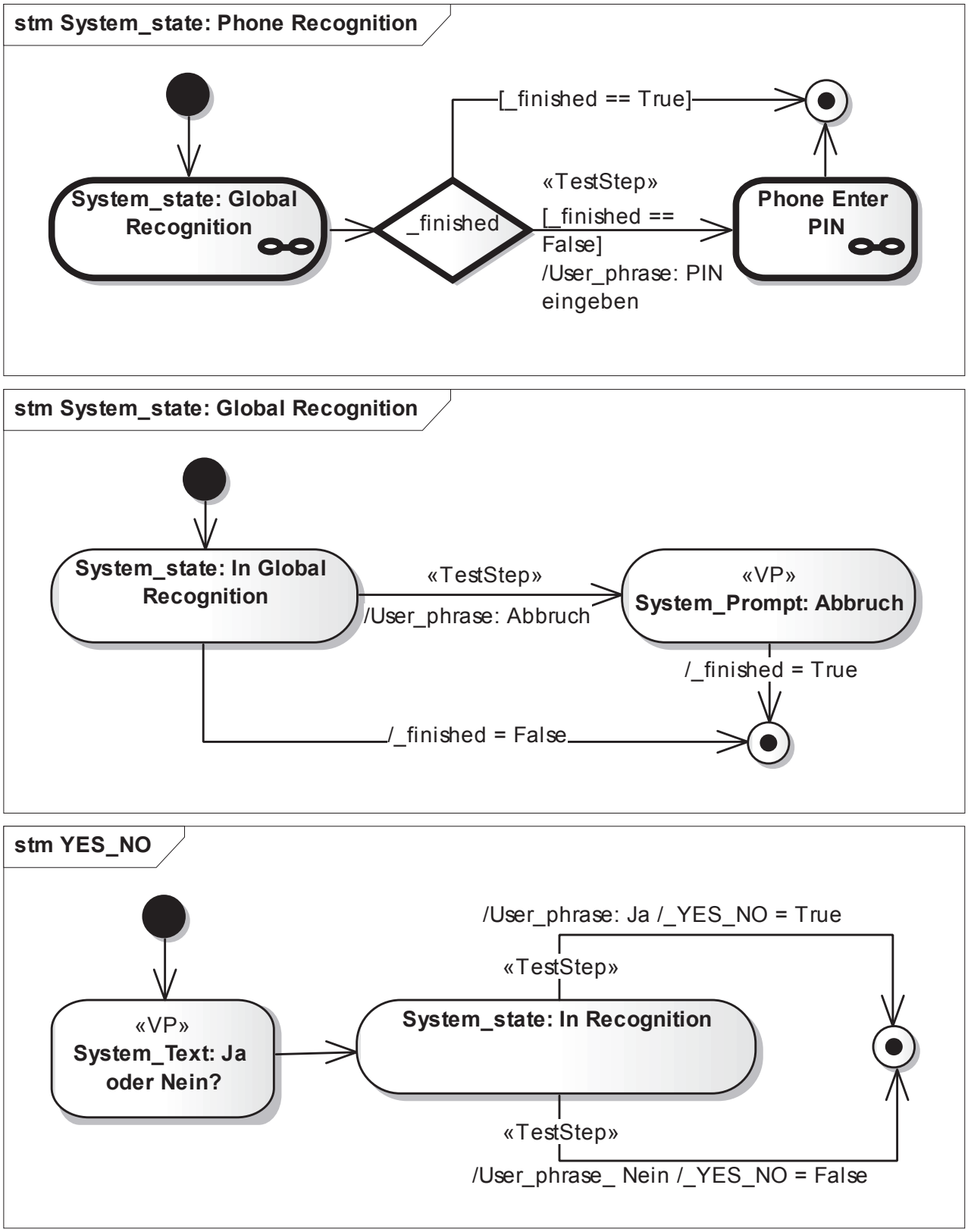
Fortsetzung nächste Seite

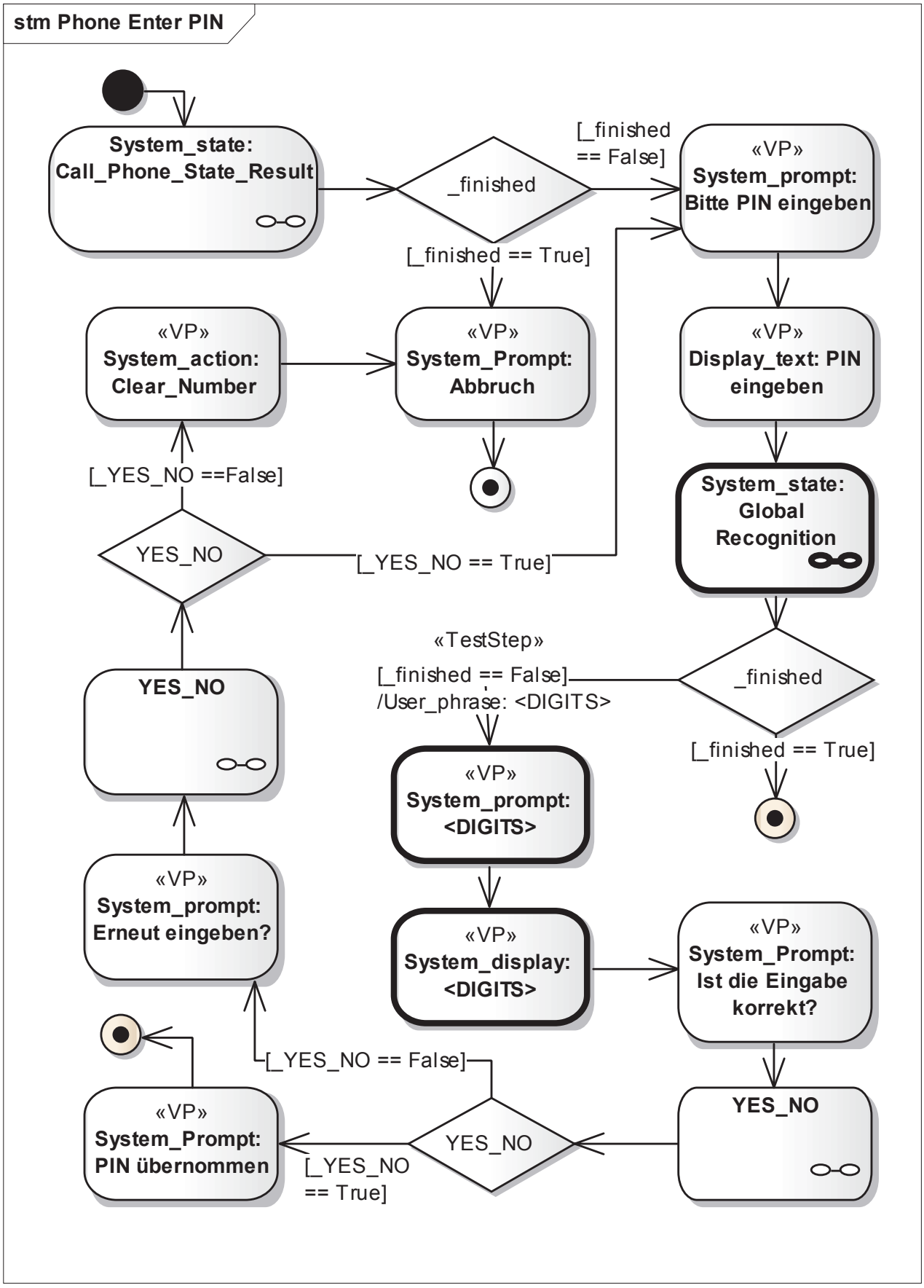
		Conformiq	MaTeLo	Rational	.getmore	TCGen	Certifylt
Funktionalität Modellierung							
Unterstützung von UML	*3*	5	2	5			
Statische Prüfung	*3*	3	2	0			
Versionsmanagement	*3*	3	1	0			
Änderungen wirken sich auf alle Instanzen aus	*3*	3	3	3			
Anbindung IBM DOORS	*2*	3	3	5			
Wertung		**80,0%**	**50,0%**	**56,7%**			
Funktionalität Testfallgenerierung							
Manuelle Definition zu erstellender Testfälle	*3*		1	4	4	2	4
Strategien wählbar (Knoten-, Kanten-, Pfad-)	*3*	5	2	2	5	5	4
Zufallsbasiert	*3*		5	5	5	5	0
Nach Kantenpriorisierung	*3*	2	4		5	1	
Steuerbar durch Modellüberdeckung	*3*	4	3	2	4	1	4
Deltagenerierung bei Änderung der Testbasis	*3*		5	0	4	5	4
Testfälle reproduzierbar	*3*	5	0	0	5	4	5
Testfälle als Skript	*3*	3	3	2	5	5	4
Lesbarkeit der Testfälle	*3*	4			5		3
Testprotokoll	*2*	5	5		5	5	4
Wertung		**79,0%**	**60,8%**	**42,9%**	**93,8%**	**72,3%**	**70,8%**
Gesamtwertung		**72,0%**	**55,1%**	**48,2%**	**89,2%**	**69,8%**	**72,7%**

12.4 Modellbeispiel des SDS zur Testfallgenerierung

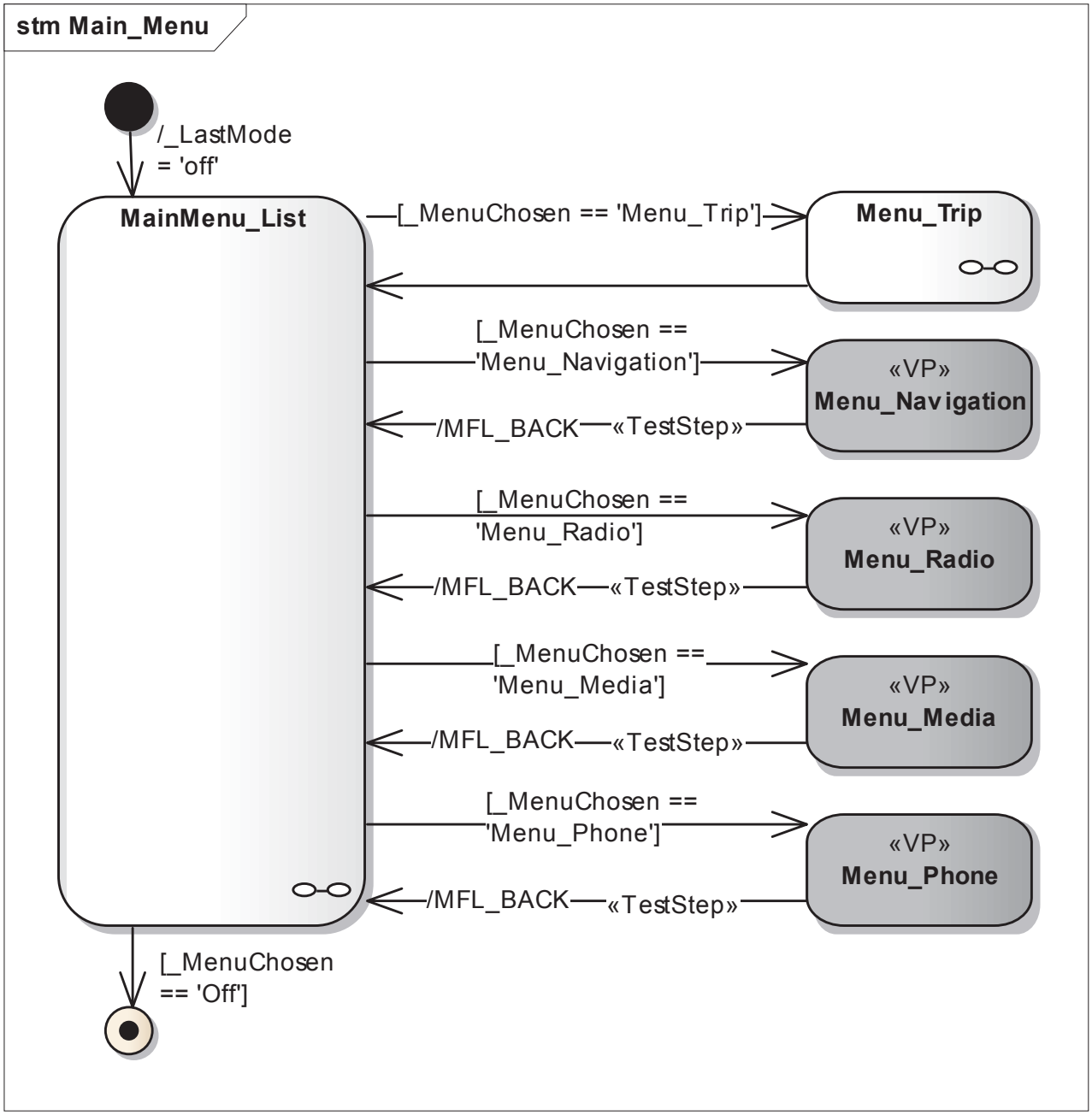
Zustände, welche mit einem fetten Rahmen modelliert sind, wurden aufgrund identifizierter Rekursionen angepasst.

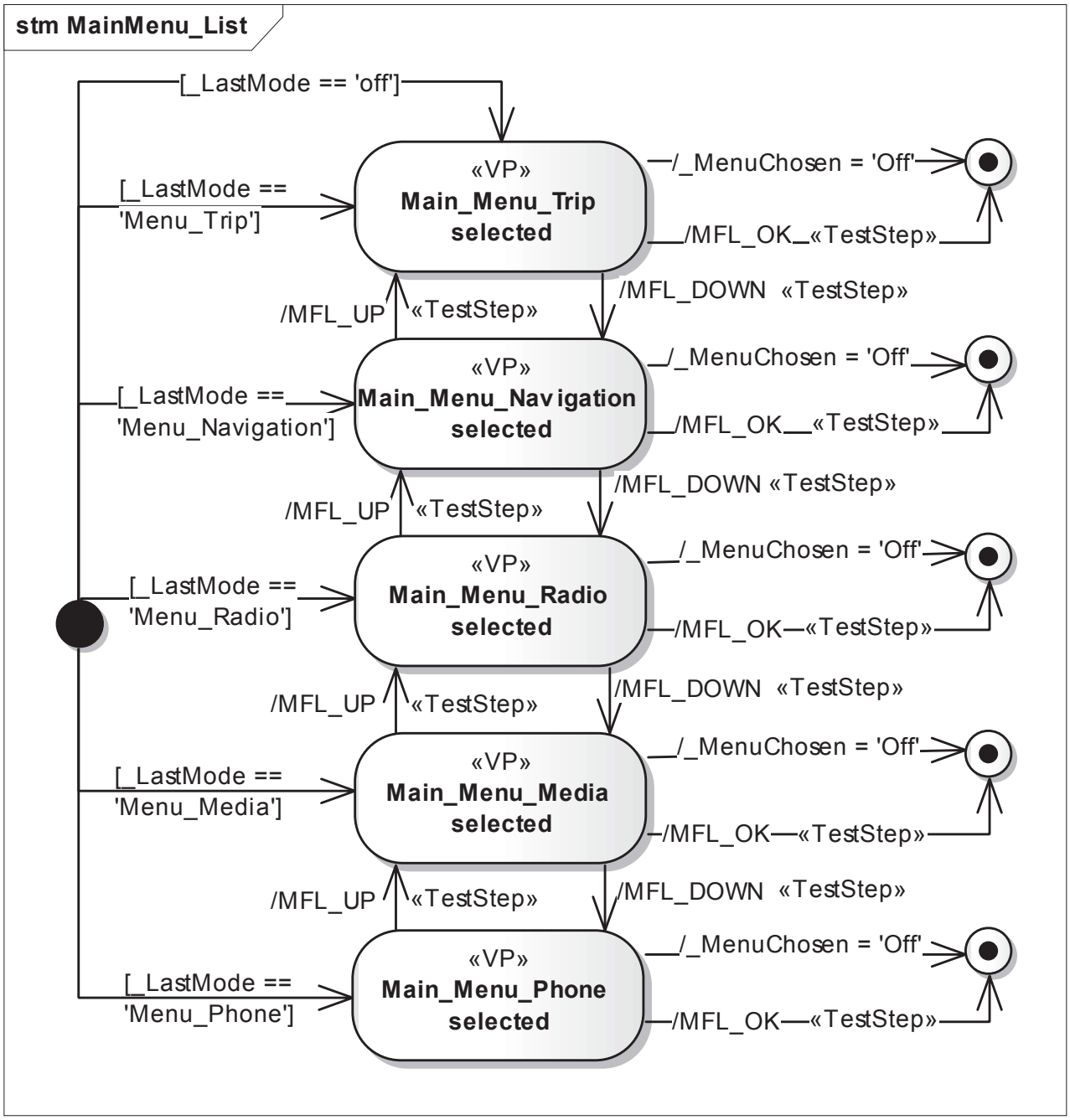


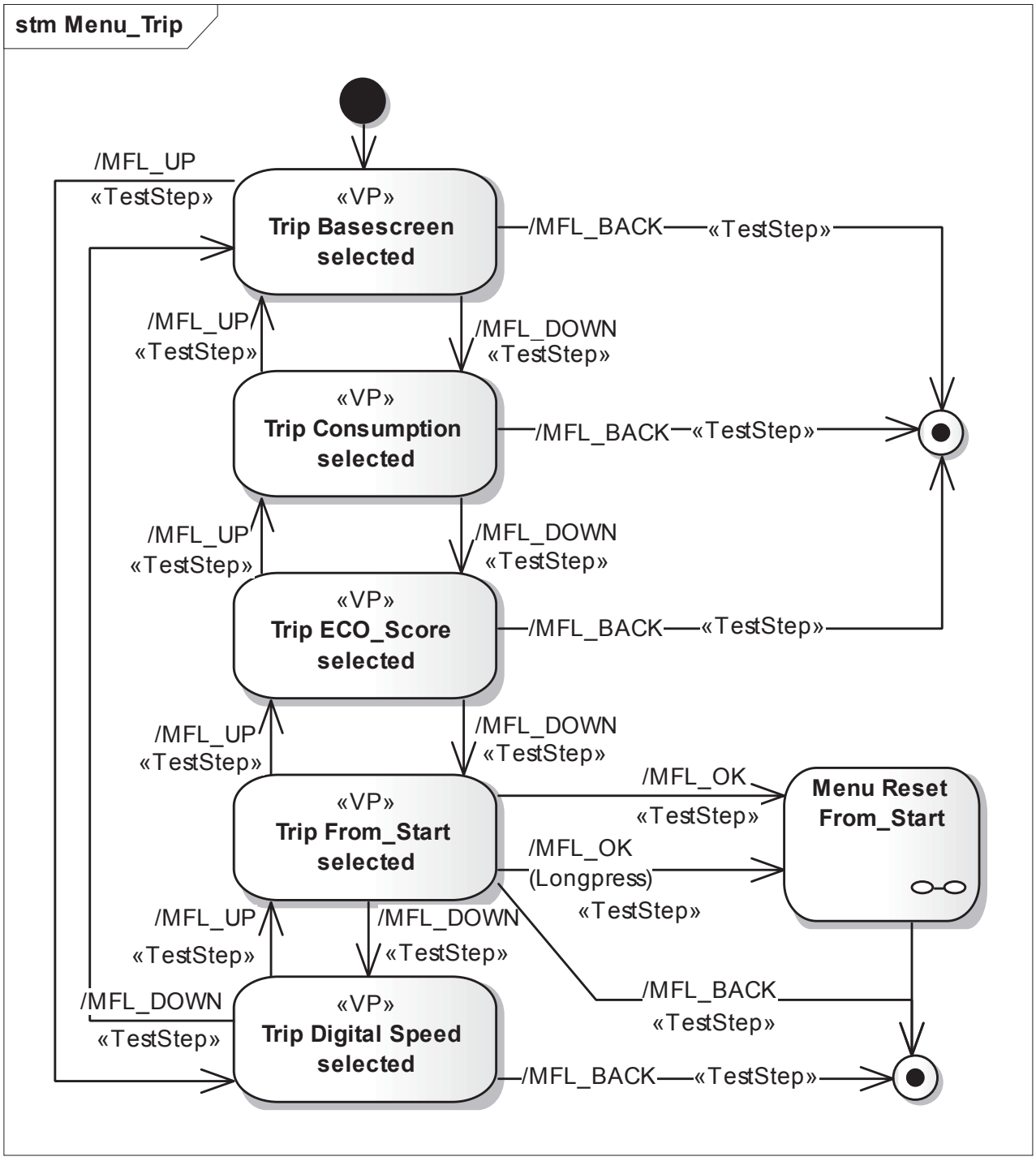

stm System_state: Phone Recognition
System_state: Global Recognition
_finished
[_finished == True]
«TestStep»
[_finished == False]
/User_phrase: PIN eingeben
Phone Enter PIN
stm System_state: Global Recognition
System_state: In Global Recognition
«TestStep»
/User_phrase: Abbruch
«VP»
System_Prompt: Abbruch
/_finished = True
/_finished = False
stm YES_NO
«VP»
System_Text: Ja oder Nein?
/User_phrase: Ja /_YES_NO = True
«TestStep»
System_state: In Recognition
«TestStep»
/User_phrase_ Nein /_YES_NO = False

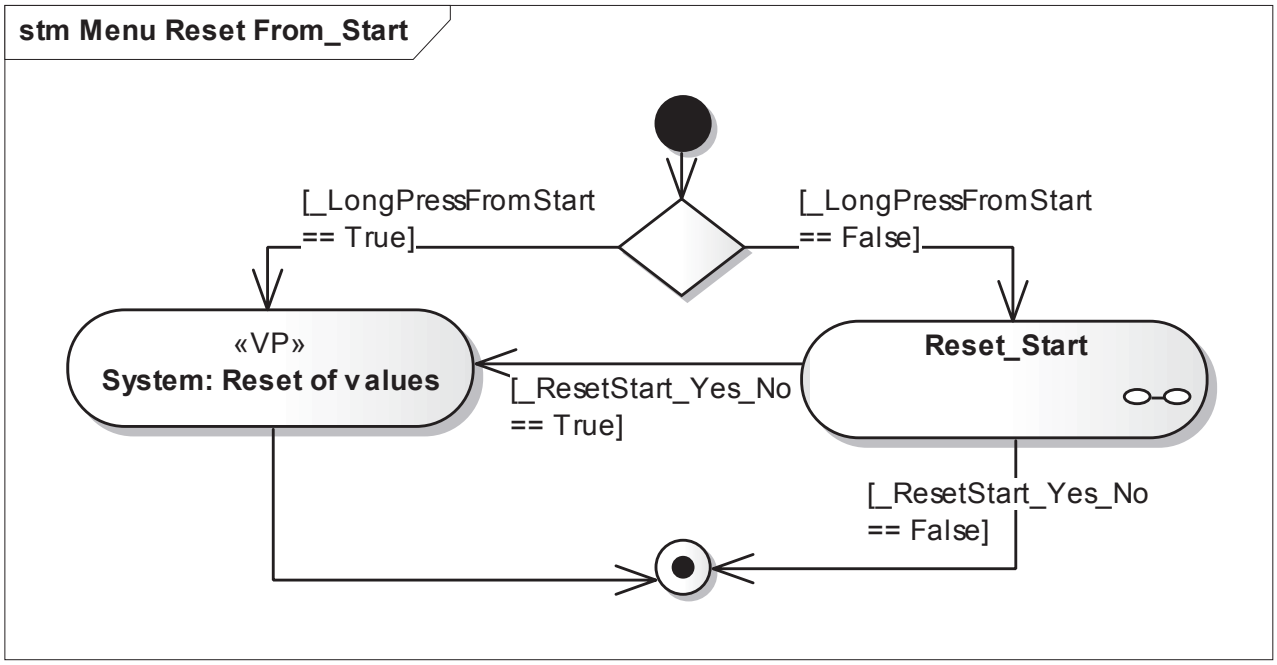

stm Phone Enter PIN
System_state:
Call_Phone_State_Result
_finished
[_finished
== False]
«VP»
System_prompt:
Bitte PIN eingeben
[_finished == True]
«VP»
System_action:
Clear_Number
«VP»
System_Prompt:
Abbruch
«VP»
Display_text: PIN
eingeben
[_YES_NO ==False]
System_state:
Global
Recognition
YES_NO
[_YES_NO == True]
«TestStep»
[_finished == False]
/User_phrase: <DIGITS>
_finished
[_finished == True]
YES_NO
«VP»
System_prompt:
<DIGITS>
«VP»
System_prompt:
Erneut eingeben?
«VP»
System_display:
<DIGITS>
«VP»
System_Prompt:
Ist die Eingabe
korrekt?
[_YES_NO == False]
«VP»
System_Prompt:
PIN übernommen
YES_NO
YES_NO
[_YES_NO
== True]

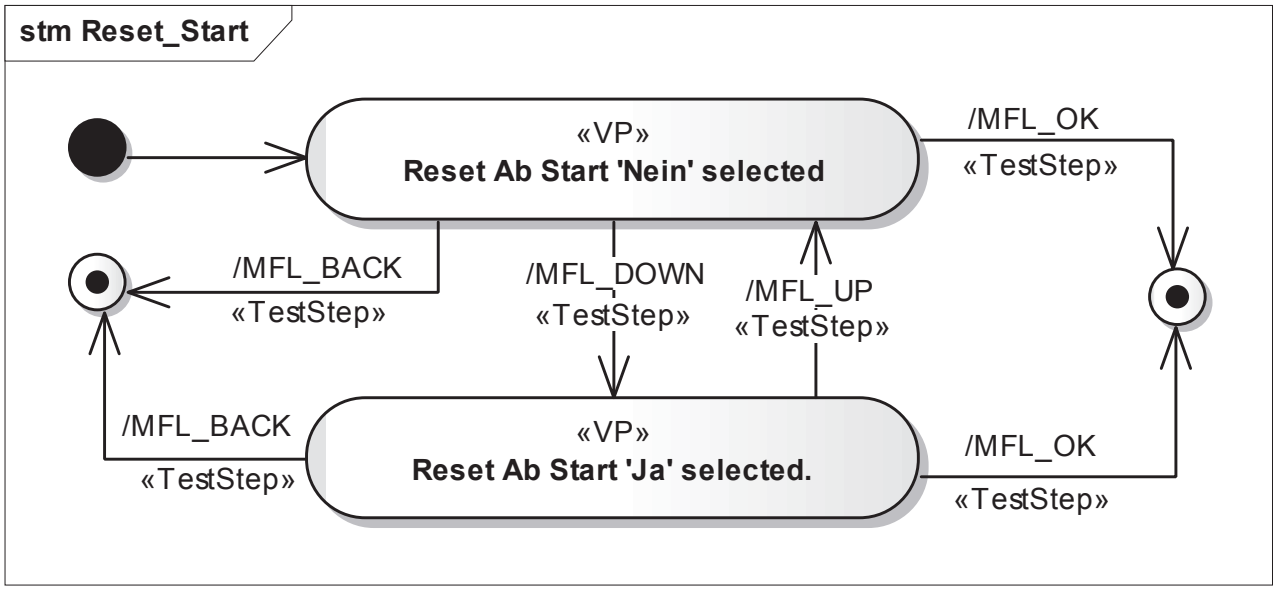
12.5 Modellbeispiel des KI zur Testfallgenerierung



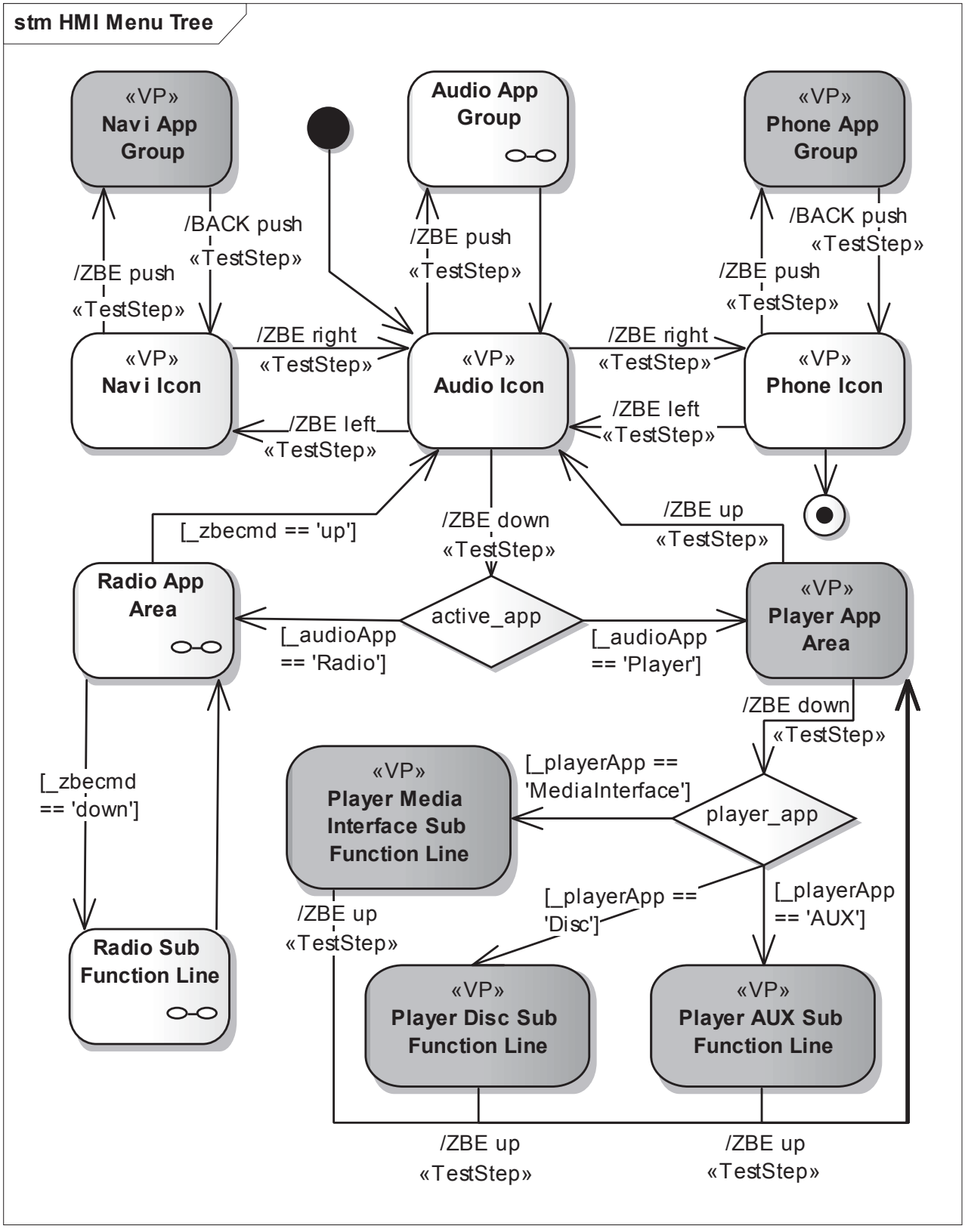

stm MainMenu_List
[_LastMode == 'off']
«VP»
Main_Menu_Trip
selected
[_LastMode ==
'Menu_Trip']
/_MenuChosen = 'Off'
/MFL_OK «TestStep»
/MFL_DOWN «TestStep»
/MFL_UP
«TestStep»
«VP»
Main_Menu_Navigation
selected
[_LastMode ==
'Menu_Navigation']
/_MenuChosen = 'Off'
/MFL_OK «TestStep»
/MFL_DOWN «TestStep»
/MFL_UP
«TestStep»
«VP»
Main_Menu_Radio
selected
[_LastMode ==
'Menu_Radio']
/_MenuChosen = 'Off'
/MFL_OK «TestStep»
/MFL_DOWN «TestStep»
/MFL_UP
«TestStep»
«VP»
Main_Menu_Media
selected
[_LastMode ==
'Menu_Media']
/_MenuChosen = 'Off'
/MFL_OK «TestStep»
/MFL_DOWN «TestStep»
/MFL_UP
«TestStep»
«VP»
Main_Menu_Phone
selected
[_LastMode ==
'Menu_Phone']
/_MenuChosen = 'Off'
/MFL_OK «TestStep»

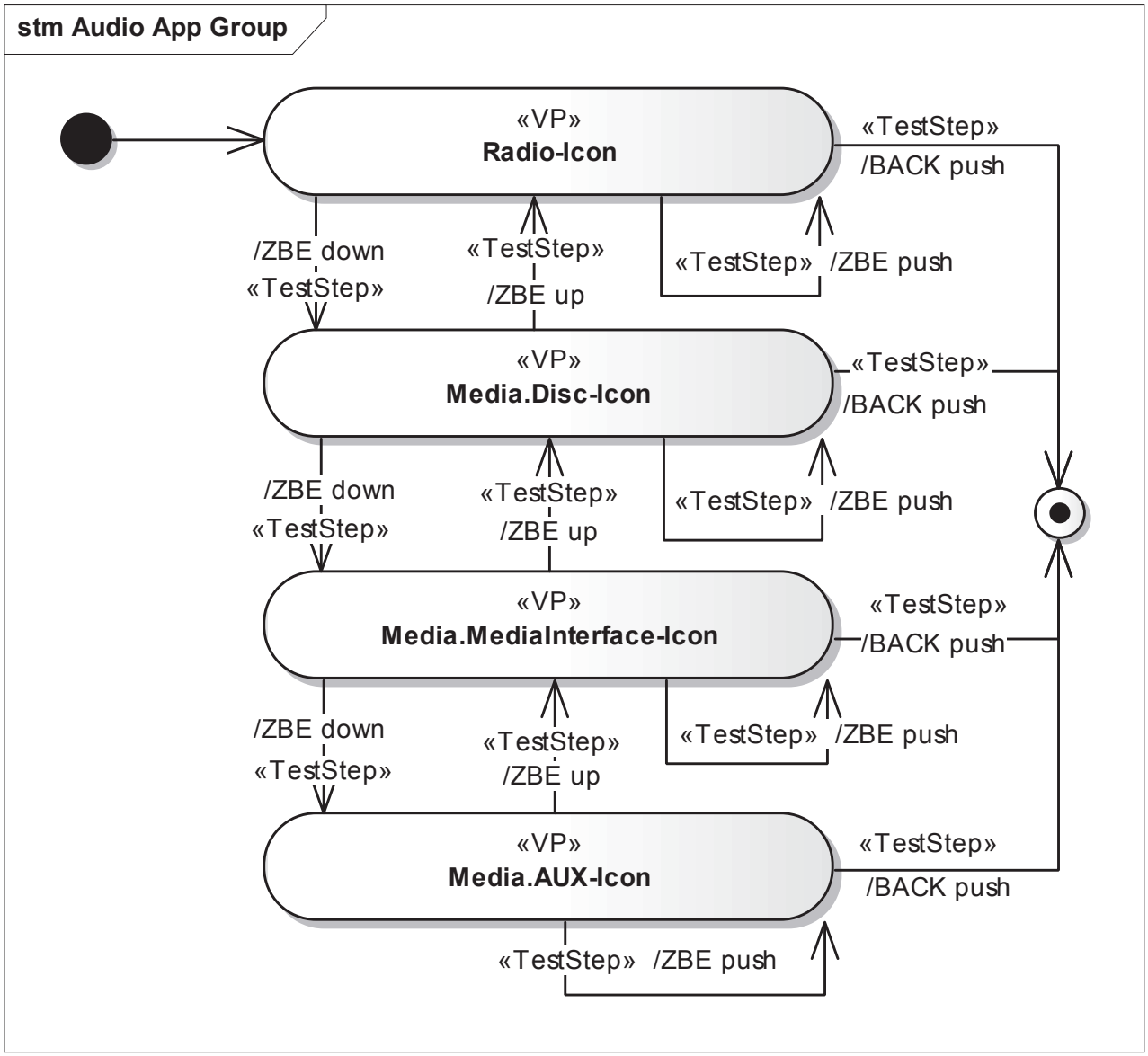

stm Menu_Trip
/MFL_UP
«TestStep»
«VP»
Trip Basescreen selected
/MFL_BACK
«TestStep»
/MFL_UP
«TestStep»
/MFL_DOWN
«TestStep»
«VP»
Trip Consumption selected
/MFL_BACK
«TestStep»
/MFL_UP
«TestStep»
/MFL_DOWN
«TestStep»
«VP»
Trip ECO_Score selected
/MFL_BACK
«TestStep»
/MFL_UP
«TestStep»
/MFL_DOWN
«TestStep»
«VP»
Trip From_Start selected
/MFL_OK
«TestStep»
Menu Reset From_Start
/MFL_OK
(Longpress)
«TestStep»
/MFL_UP
«TestStep»
/MFL_DOWN
«TestStep»
/MFL_BACK
«TestStep»
/MFL_DOWN
«TestStep»
«VP»
Trip Digital Speed selected
/MFL_BACK
«TestStep»

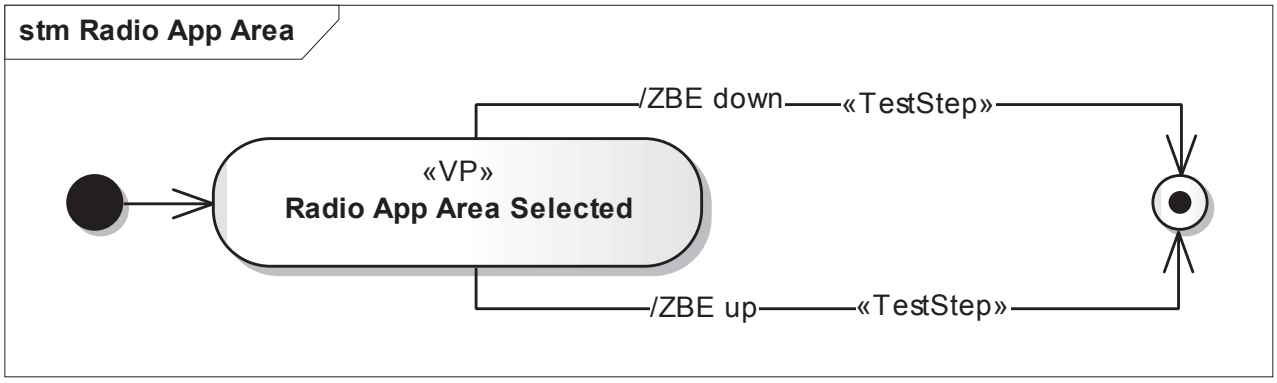

stm Menu Reset From_Start
[_LongPressFromStart
== True]
[_LongPressFromStart
== False]
«VP»
System: Reset of values
Reset_Start
[_ResetStart_Yes_No
== True]
[_ResetStart_Yes_No
== False]

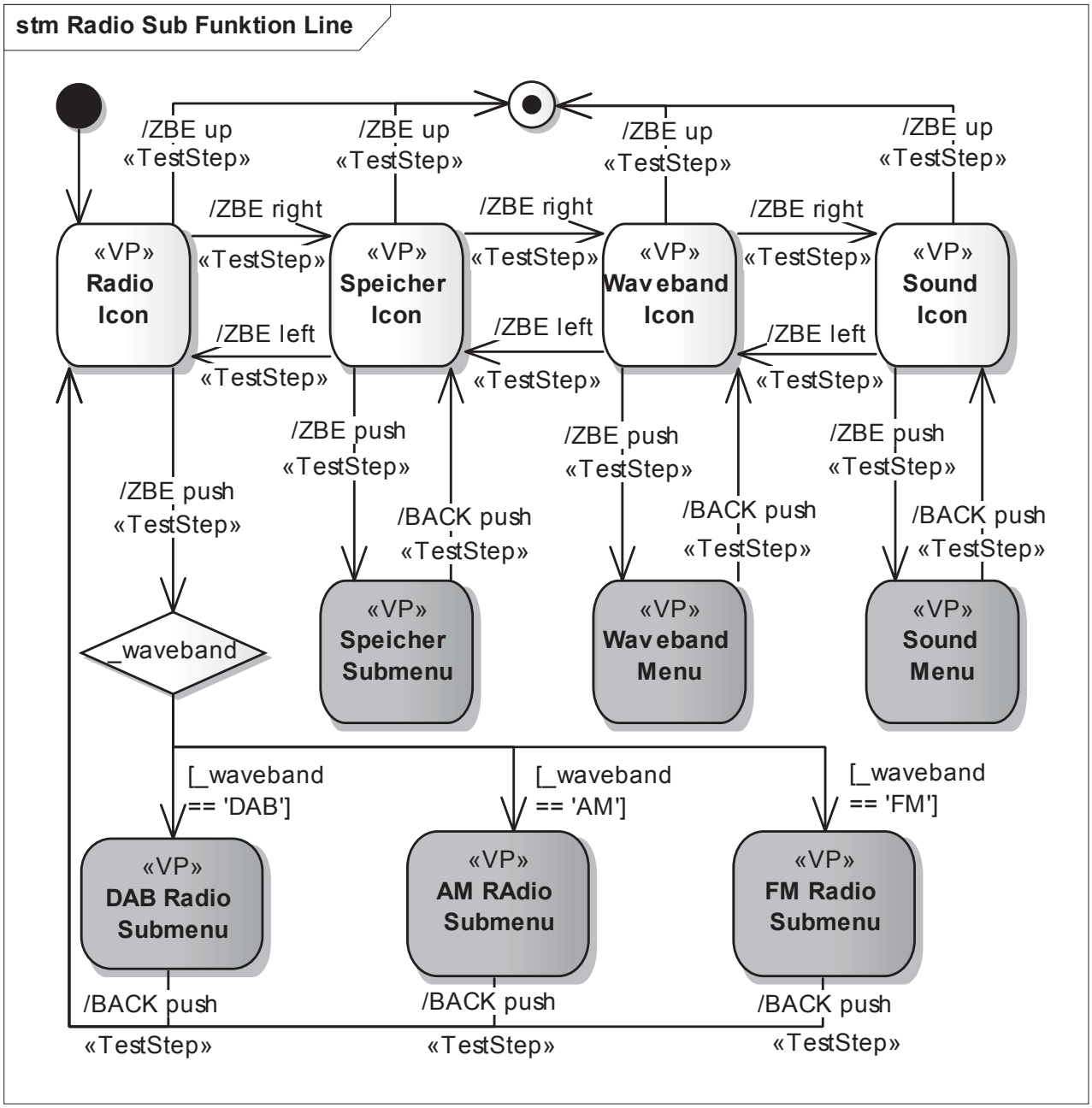

stm Reset_Start
«VP»
Reset Ab Start 'Nein' selected
/MFL_OK
«TestStep»
/MFL_BACK
«TestStep»
/MFL_DOWN
«TestStep»
/MFL_UP
«TestStep»
/MFL_BACK
«TestStep»
«VP»
Reset Ab Start 'Ja' selected.
/MFL_OK
«TestStep»

12.6 Modellbeispiel der HU zur Testfallgenerierung




stm Audio App Group
«VP»
Radio-Icon
«TestStep»
/BACK push
/ZBE down
«TestStep»
«TestStep»
/ZBE up
«TestStep»
/ZBE push
«VP»
Media.Disc-Icon
«TestStep»
/BACK push
/ZBE down
«TestStep»
«TestStep»
/ZBE up
«TestStep»
/ZBE push
«VP»
Media.MediaInterface-Icon
«TestStep»
/BACK push
/ZBE down
«TestStep»
«TestStep»
/ZBE up
«TestStep»
/ZBE push
«VP»
Media.AUX-Icon
«TestStep»
/BACK push
«TestStep»
/ZBE push


stm Radio App Area
/ZBE down
«TestStep»
«VP»
Radio App Area Selected
/ZBE up
«TestStep»


stm Radio Sub Funktion Line
/ZBE up «TestStep»
/ZBE up «TestStep»
/ZBE up «TestStep»
/ZBE up «TestStep»
/ZBE right «TestStep»
/ZBE right «TestStep»
/ZBE right «TestStep»
«VP» Radio Icon
«VP» Speicher Icon
«VP» Waveband Icon
«VP» Sound Icon
/ZBE left «TestStep»
/ZBE left «TestStep»
/ZBE left «TestStep»
/ZBE push «TestStep»
/ZBE push «TestStep»
/ZBE push «TestStep»
/ZBE push «TestStep»
/BACK push «TestStep»
/BACK push «TestStep»
/BACK push «TestStep»
«VP» Speicher Submenu
«VP» Waveband Menu
«VP» Sound Menu
_waveband
[_waveband == 'DAB']
[_waveband == 'AM']
[_waveband == 'FM']
«VP» DAB Radio Submenu
«VP» AM RAdio Submenu
«VP» FM Radio Submenu
/BACK push «TestStep»
/BACK push «TestStep»
/BACK push «TestStep»

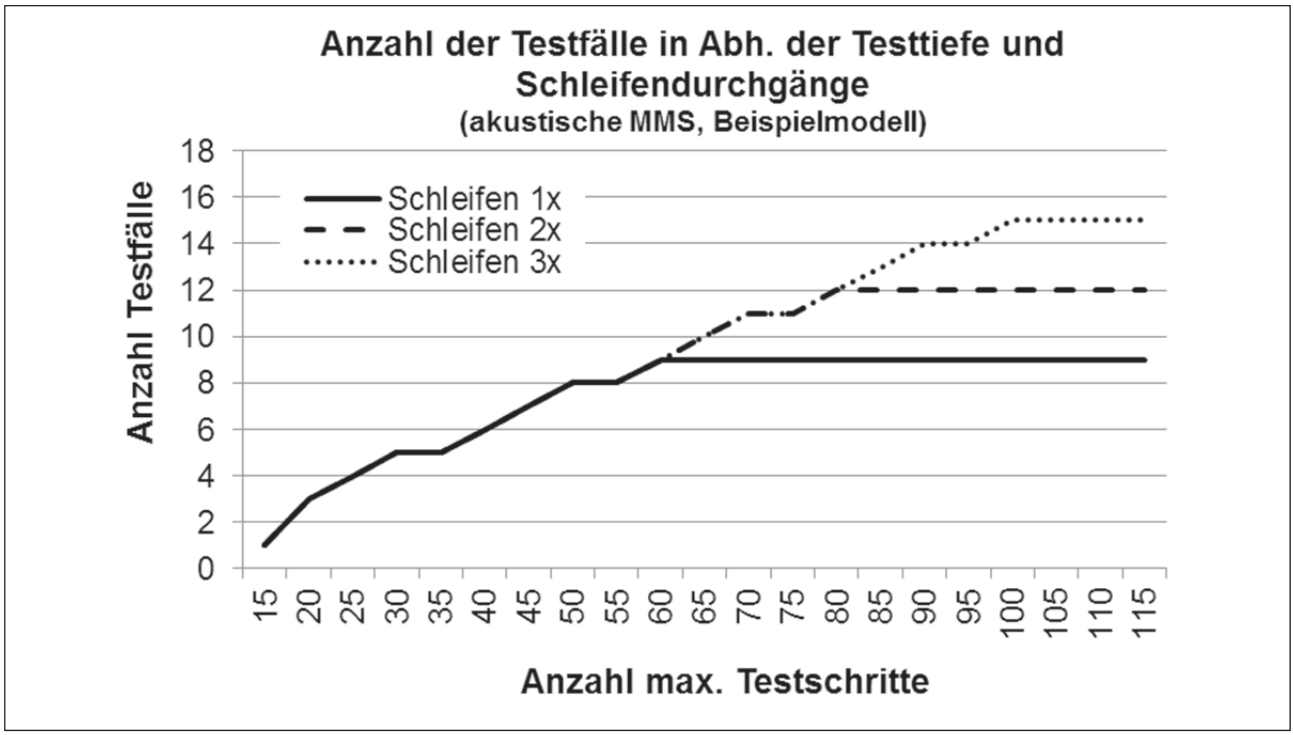
12.7 Beispiel für generierte Testfälle

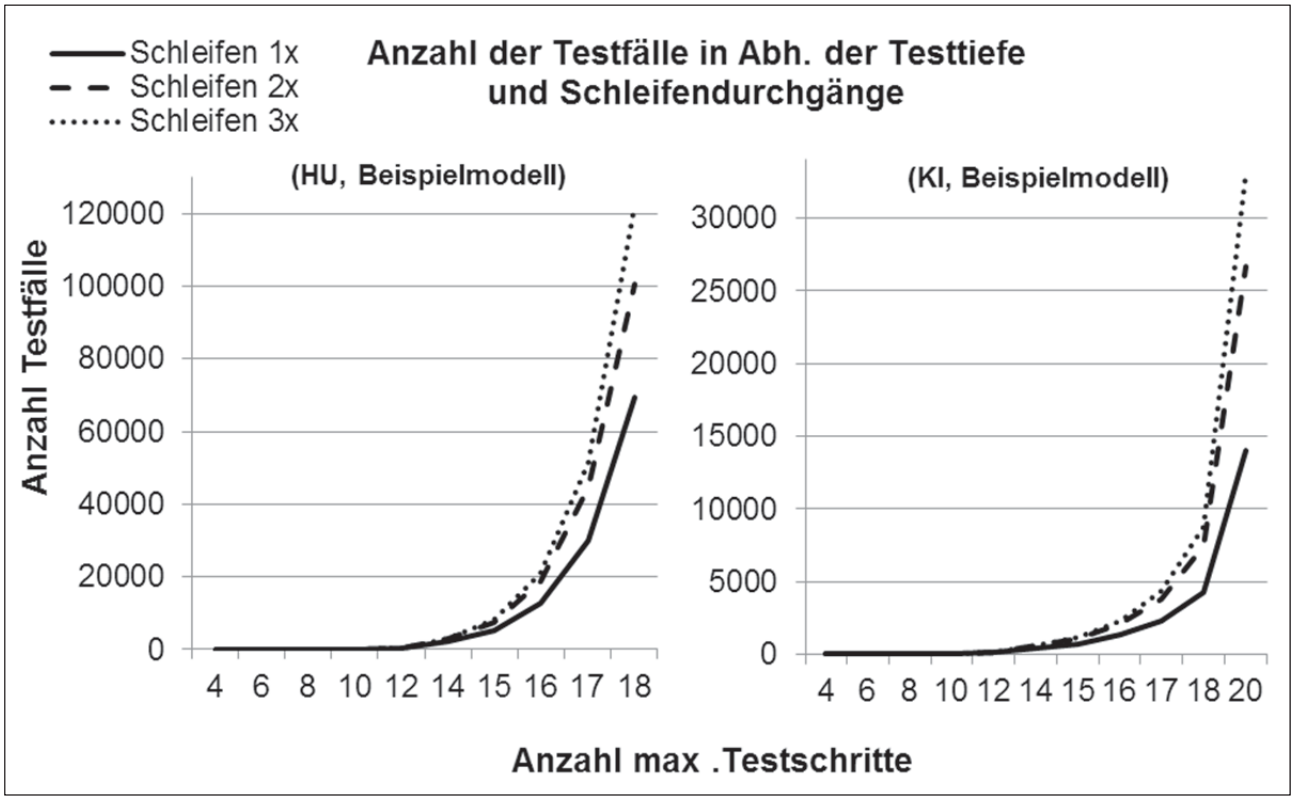
Generierte Testfälle aus den Beispielmodellen der haptischen MMS (siehe Anhang 12.5 und 12.6)

Test		KI Bsp0002
Step	Type	Step Description
1	VP	Main_Menu_Trip selected
2	TS	User action: MFL_OK
3	VP	Trip Basescreen selected
4	TS	User action: MFL_UP
5	VP	Trip Digital Speed selected
6	TS	User action: MFL_UP
7	VP	Trip From_Start selected
8	TS	User action: MFL_OK
9	VP	Reset Ab Start 'Nein' selected
10	TS	User action: MFL_DOWN
11	VP	Reset Ab Start 'Ja' selected.
12	TS	User action: MFL_BACK
13	VP	Main_Menu_Trip selected
14	TS	User action: MFL_DOWN
15	VP	Main_Menu_Navigation selected

Test		HU Bsp0003
Step	Type	Step Description
1	VP	Audio Icon selected by cursor
2	TS	User action: ZBE down
3	VP	Radio App Area selected by cursor
4	TS	User action: ZBE down
5	VP	Radio Icon is selected by cursor
6	TS	User action: ZBE push
7	VP	FM Radio Submenu selected
8	TS	User action: BACK push
9	VP	Radio Icon is selected by cursor
10	TS	User action: ZBE right
11	VP	Speicher Icon is selected by cursor
12	TS	User action: ZBE up
13	VP	Radio App Area selected by cursor
14	TS	User action: ZBE up
15	VP	Audio Icon selected by cursor
16	TS	User action: ZBE right
17	VP	Phone Icon selected by cursor

12.8 Testfallgenerierung: Einfluss mehrfacher Schleifendurchgänge





In dieser »Schriftenreihe zu Arbeitswissenschaft und Technologiemanagement« werden die Dissertationen, die im Rahmen von Forschungs- und Entwicklungsarbeiten am Institut für Arbeitswissenschaft und Technologiemanagement IAT der Universität Stuttgart und am Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO entstanden sind, veröffentlicht.

Die beiden Institute verknüpfen universitäre Grundlagenforschung mit angewandter Auftragsforschung und setzen diese erfolgreich in zahlreichen Projekten praxisgerecht um.

Technologiemanagement umfasst dabei die integrierte Planung, Gestaltung, Optimierung, Bewertung und den Einsatz von technischen Produkten und Prozessen aus der Perspektive von Mensch, Organisation, Technik und Umwelt. Dabei werden neue anthropozentrische Konzepte für die Arbeitsorganisation und -gestaltung erforscht und erprobt. Die Arbeitswissenschaft mit ihrer Systematik der Analyse, Ordnung und Gestaltung der technischen, organisatorischen und sozialen Bedingungen von Arbeitsprozessen sowie ihren humanen und wirtschaftlichen Zielen ist dabei zentral in die Aufgabe des Technologiemanagements eingebunden.

ISBN 978-3-8396-0837-1

9 783839 608371

ISSN 2195-3414

Fraunhofer Verlag