Freie Universität Berlin

Qualitätskriterien und Kooperationsaspekte in einem durchgängigen Entwicklungsprozess von Mensch-Maschine-Schnittstellen für Infotainmentsysteme

Diplomarbeit zur Erlangung des akademischen Grades
Diplominformatiker

Eingereicht bei: Prof. Dr. Lutz Prechelt

am: 13. Februar 2006

von: Olaf Hecht,
geboren am 20. Dezember 1979 in der FU Berlin,
olafhecht@web.de,
Matrikelnummer 3517447

Betreuer: Dipl. Math. Stephan Salinger
Dipl. Inf. Gunnar Wegner

Eidesstattliche Erklärung

Ich erkläre an Eides Statt, dass ich die Diplomarbeit mit dem Titel „Qualitätskriterien und Kooperationsaspekte in einem durchgängigen Entwicklungsprozess von Mensch-Maschine-Schnittstellen für Infotainmentsysteme“ selbständig verfasst habe. Alle Stellen der Arbeit, die anderen Werken wörtlich oder sinngemäß entnommen sind, habe ich unter Angabe der Quelle als Entlehnung kenntlich gemacht.

Berlin, den 13. Februar 2006

Gott sei Dank.

Ich bedanke mich bei meinen Betreuern für ihre freundliche und hervorragende Unterstützung. Ich danke Herrn Salinger, dessen Kritiken und ständige Emailbeantwortungsbereitschaft erheblich zur Qualität dieser Arbeit beigetragen haben. Bei Herrn Wegner bedanke ich mich vor allem für die grenzenlose Forschungsfreiheit in seiner Abteilung und für seine Förderung und Forderung in den letzten Jahren.

Für ihre freundliche Unterstützung und Offenheit will ich mich hiermit auch bei allen IAV-Mitarbeiter bedanken, ohne die ein Großteil dieser Arbeit nicht hätte entstehen können.

Ein ganz besonderer Dank geht an meine Familie - vor allem für die allsonntägliche Entspannung von der Arbeit. Auch bei meinen Freunden will ich mich für die Unterstützung bedanken - vor allem für das Verständnis dafür, dass ich in den letzten Wochen etwas weniger Zeit mit ihnen verbringen konnte. Besonderer Dank sei hier an Julius und Lotte gerichtet, die mir des Öfteren bei Fragen zur deutschen Sprache weiterhelfen konnten.

Bei meinen Freunden und meiner Familie bedanke mich außerdem für die ständige Bereitschaft, mir nicht-fachspezifische Fragen zu beantworten.

Auch beim heimatlichen Wetter möchte ich mich für die Unterstützung in den letzten Wochen bedanken. Außerdem danke ich dem Schicksal der frühen Geburt, dass es mir ermöglichte, auf Diplom studieren zu können.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Zur Einleitung in das Thema dieser Diplomarbeit werde ich nachfolgend die Ingenieurgesellschaft Auto und Verkehr, die Aufgabenstellung, die in dieser Arbeit untersuchten Projekte und die Gliederung der Arbeit vorstellen.

1.1 Ingenieurgesellschaft Auto und Verkehr

Diese Diplomarbeit entstand in Kooperation mit der Ingenieurgesellschaft Auto und Verkehr - im Folgenden kurz IAV genannt. Die IAV stellt sich in ihrer Informationsbroschüre über das Firmenprofil wie folgt selbst dar: „Als international aufgestelltes Engineering-Unternehmen bietet die IAV der Automobilbranche eine Vielzahl von Entwicklungsdienstleistungen für zukünftige Fahrzeuggenerationen. Engineering-Kompetenzen für das ganze Fahrzeug und interdisziplinäres Arbeiten sind unsere besonderen Stärken, denn nur der Blick auf die Zusammenhänge garantiert serientaugliche Lösungen. [...] Erfindungsgeist, Enthusiasmus und der Anspruch, in puncto Technik etwas zu bewegen, sind die Triebfedern von über 2.500 IAV-Mitarbeiterinnen und Mitarbeitern im In- und Ausland.“ [IAV05]

1.2 Aufgabenstellung

In der Mitte des Jahres 2005 konzipierten meine Betreuer und ich folgende Aufgabenstellung für diese Diplomarbeit:

„Moderne Infotainmentsysteme der Automobilindustrie zeichnen sich durch die Integration verschiedener Dienste (Radio, CD, CD-Wechsler, MP3, DVD, TV, Navigation, Verkehrsinformationen, Telefon etc.) in einem Gerät aus. Dabei spielt ein einheitliches Erscheinungsbild der graphischen Benutzeroberfläche und ein durchgehendes Bedienkonzept eine entscheidende Rolle, definieren sie doch die Alleinstellungsmerkmale der einzelnen Systeme der verschiedenen Fahrzeughersteller.

Die Entwicklung dieser Infotainmentsysteme [s. Kapitel 2.1] gestaltet sich heutzutage meist noch relativ ineffizient. Das Erscheinungsbild und das damit einhergehende Bedienkonzept wird mit konventionellen Windows-Tools (MacroMedia Director, PhotoShop, StateChart Editoren, etc.) entworfen, ebenso wie die zugehörige Simulation PC-basiert, z.B. mit Hilfe einer Flash-Animation, entwickelt wird. In den meisten Fällen wird für jedes Gerät eine neue Simulation gebaut. Die eigentliche Gerätesoftware wird zusätzlich zu den Entwicklungsschritten im Designprozess nochmals entwickelt.

Motiviert dadurch, die gleichartigen Entwicklungsschritte in diesem Prozess zusammenzufassen und den Prozess damit effizienter zu gestalten, hat die IAV GmbH einen durchgehenden Entwicklungsprozess definiert, in welchem in der Designphase eine allgemeingültige Datenbasis mit allen das HMI [Mensch-Maschine-Schnittstelle (s. Kapitel 2.2)] eines Infotainmentsystems betreffenden Informationen (Struktur, Aussehen, Verhalten) entsteht. Außerdem wird die Funktionsschnittstelle zu den einzelnen Gerätekomponenten spezifiziert. Aus der entstehenden Datenbasis generiert eine Laufzeitumgebung eine Simulation, die unter Verwendung von Simulationskomponenten einer realen Gerätesimulation sehr nahe kommt. Gleichzeitig dient diese Datenbasis aber auch als Input für die Implementierung der finalen Gerätesoftware. Dazu ist es notwendig, alle benötigten Informationen aus der Datenbasis zu extrahieren und in ein der Zielplattform entsprechendes, performantes Format zu überführen.

Eines der Ziele dieser Arbeit soll eine qualitative Bewertung dieses durchgehenden Entwicklungsprozesses bezüglich seiner Teilschritte sein. Es ist eine detaillierte Beschreibung und Analyse des Prozesses notwendig. Dazu ist er in seine Teilschritte zu

zerlegen. Sowohl für den Gesamtprozess als auch für die Teilschritte sind formale [gemeint waren informale] Prozessbeschreibungen zu erarbeiten. Da an der Entwicklung von Infotainmentsystemen verschiedene Rollen, wie z.B. Designer, technische Experten oder Tester mitwirken, ist es auch wichtig zu betrachten, welche Rollen jeweils bei der Bearbeitung der Teilschritte beteiligt sind. Um zu einer qualitativen Bewertung zu gelangen, sollen für die einzelnen Teilprozesse verschiedenartige Bewertungskriterien definiert werden. Damit die Qualität später auch meßbar ist, sollen für die Kriterien Metriken aufgestellt werden.

Ein weiteres Ziel dieser Arbeit ist es herauszufinden, welche Möglichkeiten der Kooperation es zwischen den verschiedenen Rollen während des Entwicklungsprozesses gibt und welche Effekte diese erzielen würden. Eine Frage, die sich dabei stellt, ist, welche Zusammensetzungen von Kooperationsteams in Bezug auf die Qualität des Prozesses sinnvoll sind. Außerdem ist zu betrachten, in welchen Teilschritten und zu welchen Zeitpunkten diese generell möglich sind und gegebenenfalls qualitätssteigernd sein können. Dafür sollte auch gegebenenfalls empirisch ermittelt werden, auf welche der zuvor definierten Bewertungskriterien eine Kooperation Auswirkungen haben kann.“

Der hier erwähnte Entwicklungsprozess nennt sich TeleDrive® VISION und wird im Folgenden nur noch kurz TeleDrive VISION genannt. Der Prozess ist ca. drei Jahre alt. Es handelt sich also um einen sehr jungen Prozess. Die dieser Diplomarbeit zugrunde liegenden Untersuchungen fanden anhand einer Betrachtung von zwei großen Projekten statt. Diese Projekte waren die ersten, die den TeleDrive VISION Prozess in seiner vollen Breite einsetzten.

Ein Ziel der Arbeit war es, eine Analyse der Schwächen und Stärken des bestehenden Prozesses durchzuführen und daraufhin Vorschläge zur Prozessverbesserung auszuarbeiten. Die Vorschläge sollten dabei auch berücksichtigen, in wie weit eine Prozessverbesserung durch verstärkte Kooperation der Prozessbeteiligten entstehen kann. Außerdem sollte die Prozessqualität durch das Aufstellen verschiedener Maße quantitativ bestimmt werden.

„Jede Software-Entwicklung soll in einem festgelegten organisatorischen Rahmen erfolgen. Ein Prozess-Modell – auch Vorgehensmodell [kurz V-Modell] genannt – beschreibt einen solchen Rahmen. In ihm wird festgelegt, welche Aktivitäten in welcher Reihenfolge von welchen Personen [bzw. Rollen] erledigt werden und welche Ergebnisse [...] dabei entstehen und wie diese in der Qualitätssicherung überprüft werden.“ [Bal96, S. 54] Der TeleDrive VISION Prozess wurde zum Zeitpunkt der Untersuchungen zwar schon in Projekten eingesetzt, ein detailliertes Prozessmodell in schriftlicher Form existierte aber noch nicht.

Um einen Prozess verbessern zu können, bedarf es jedoch als erstes einer Analyse des momentanen Zustands des eingesetzten Prozessmodells. Oder wie Watts S. Humphrey es in [Hum89] formulierte: “If you don’t know where you are, a map won’t help.“ Ein erstes Ziel der Diplomarbeit war es deshalb, mit Hilfe einer empirischen Untersuchung ein Prozessmodell aufzustellen.

Dadurch, dass vorher noch nicht zu erahnen war, welche Problembereiche ich im Prozess identifizieren würde, waren die Anforderungen an diese Arbeit nicht konkret festlegbar. Die Entstehung der Arbeit hatte daher einen stark iterativen Charakter.

1.3 Untersuchte Projekte

Es folgt ein Zitat aus einer Informationsbroschüre der IAV: „In der Automobilbranche entscheidet technologischer Fortschritt über die Marktanteile der Zukunft. Und dieser Fortschritt muss vor den Blicken anderer geschützt werden. Deshalb ist Geheimhaltung eine unserer „offensichtlichsten“ Eigenschaften.“ [IAV05]

Eine Schwierigkeit bei der Ausarbeitung dieser Arbeit war es, Projektdetails geheim zu halten, obwohl die Untersuchungen in Projekten stattfanden. Um dem Leser jedoch trotzdem einen ausreichenden Einblick über den Kontext dieser Arbeit zu verschaffen, folgen hier einige Eckdaten zu den untersuchten Projekten.

Die Projektstruktur sah in beiden Projekten folgendermaßen aus: Ein Fahrzeughersteller beauftragte mehrere Zuliefererfirmen für den Bau von in den Fahrzeugen zu integrierenden Infotainmentsystemen. Für die Erstellung der Produktspezifikation beauftragte der Fahrzeughersteller die IAV-Abteilung „Elektronik Infotainment - Produkte“. In dieser Abteilung fanden meine Untersuchungen hauptsächlich statt. Auch die Gerätetests wurden von einer IAV-Abteilung vorgenommen.

1.3.1 Projekt A

Das Projekt A hatte es zum Ziel, ein Infotainmentsystem der unteren Leistungsklasse zu entwickeln, was unter anderem bedeutet, dass das darin enthaltene Navigationssystem keine Landkartendarstellung anbietet. Es sollten zwei Gerätevarianten entstehen, deren Mensch-Maschine-Schnittstellen sich nach Aussage eines IAV-Mitarbeiters um mehr als 20 Prozent unterschieden. Zum Ende des Projekts hin umfassten die Geräte jeweils über 300 Menüs. Zum Zeitpunkt der Untersuchung lief das Projekt seit ca. zweieinhalb Jahren und war soweit fortgeschritten, dass die Geräte bereits (nur noch) getestet und defektbereinigt wurden. An dem Projekt waren über 80 Personen weltweit beteiligt.

1.3.2 Projekt B

Ziel des Projekts B war es, ein Infotainmentsystem der mittleren Leistungsklasse zu entwickeln, was unter anderem bedeutet, dass das darin enthaltene Navigationssystem eine Landkartendarstellung anbietet. Es sollten über 30 Gerätevarianten entstehen, die auf einer Hauptvariante basierten. Die Hauptvariante umfasste zu einem Zeitpunkt der Untersuchung über 600 Menüs. Zu Beginn meiner Diplomarbeit lief das Projekt bereits seit mehr als einem Jahr. Am Ende der Diplomarbeit hieß es, dass das Projekt noch mehr als ein Jahr lang weiterlaufen solle. An dem Projekt waren über 100 Personen weltweit beteiligt.

1.4 Aufbau der Arbeit

Nach diesem einleitenden Kapitel werde ich zunächst im 2. Kapitel Grundlagen zum Untersuchungsobjekt und in Kapitel 3 Grundlagen der Untersuchungsmethodik erläutern. Weitere Grundlagen werden jeweils in den betreffenden Kapiteln erklärt.

Um das erste Teilziel dieser Arbeit zu erreichen, nahm ich den momentanen Zustand des Prozessmodells mit Hilfe von leitfadenorientierten Interviews mit IAV-Mitarbeitern auf. Die Methodik dieser Untersuchung beschreibe ich in Kapitel 4.

Aufgrund des Umfangs habe ich dem von mir erstellten Prozessmodell, also dem Ergebnis der im vierten Kapitel beschriebenen empirischen Untersuchung, zwei Kapitel gewidmet: In Kapitel 5 ist ein Prozessmodell des gesamten TeleDrive VISION Prozesses zu finden. Dem folgt in Kapitel 6 ein detaillierteres Modell des Spezifizierungsprozesses. Die beiden Kapitel enthalten auch die von mir aufgedeckten Stärken des Prozesses.

Die identifizierten Schwächen bzw. Probleme werden im 7. Kapitel erläutert. Um einigen der Schwächen zu begegnen, habe ich eine auf den Prozess zugeschnittene rollenkooperative Form der Durchsicht (engl. *review*) konzipiert. Diese wird in Kapitel 8 beschrieben. Um die Durchsichtsform daraufhin zu überprüfen, ob sie auch in der Praxis im TeleDrive VISION Prozess einsetzbar ist, führte ich eine empirische Studie innerhalb des Projekts B durch. Für die Bewertung der Studie benötigte ich ein Umfangsmaß für die in den Projekten eingesetzte Spezifikationssprache *Infotainment Markup Language* (IML) (s. Kapitel 2.4). Das Maß definiere ich in Kapitel 9. Die Studie ist im 10. Kapitel beschrieben. Das 11. Kapitel schlisst diese Arbeit ab.

2 Grundlagen zum Untersuchungsobjekt

Um dem Leser das für die folgenden Kapitel notwendige Hintergrundwissen zu vermitteln, werde ich in diesem Kapitel zentrale Begriffe der Arbeit erläutern. Zunächst werde ich den Kontext der Arbeit mit einer Erklärung der auch schon im Arbeitstitel vorkommenden branchen- und fachspezifischen Begriffe Infotainmentsystem und Mensch-Maschine-Schnittstelle erläutern. Für die Spezifizierung von Mensch-Maschine-Schnittstellen hat die IAV mit der *Infotainment Markup Language* eine eigene Sprache entwickelt, die ich am Ende des Kapitels kurz vorstellen werde.

2.1 Infotainmentsystem

Der Begriff Infotainment ist ein aus den englischen Wörtern *information* und *entertainment* zusammengezogenes Kunst- und Kofferwort. Ursprünglich wurde es von dem Kommunikationswissenschaftler und „Medienökologen“ Neil Postman für die Verbindung von Informationen mit Unterhaltungswerten in modernen Medien geprägt [Thi05] [Wik05a] und wird in der Kommunikationswissenschaft oft im negativen Sinne verwendet [Hol97, S. 273]. Die Automobilindustrie übernahm diesen Begriff für im Fahrzeug integrierte Systeme, die den Fahrzeuginsassen Informationen und Unterhaltung bieten: den Infotainmentsystemen.

Seit dem Ende des 20. Jahrhunderts drängen Informations-, Kommunikations- und Unterhaltungstechnologien vermehrt ins Fahrzeug. Das herkömmliche Autoradio wurde im Laufe der Zeit durch immer mehr Dienste erweitert. Darunter befinden sich neben Diensten neuerer Hörfunk-Technologien, wie DAB (Initialwort für engl. *Digital Audio Broadcast*, digitaler Hörfunk), auch Infotainmentdienste, wie CD-, DVD- und MP3-Spieler, digitales Fernsehen, Internet, Email und mobiles Telefonieren. Ein weiteres Subsystem eines modernen Infotainmentsystems ist zumeist ein Navigationssystem. Neben den dafür herkömmlichen Anwendungen, wie Routenplanung und Zielführung, bieten diese Systeme zumeist weitere Dienste an, wie z.B. das Anzeigen von und Leiten zu an der Fahrstrecke gelegenen speziellen Orten, welche die Fahrzeuginsassen möglicherweise interessieren, sogenannte POI (Initialwort für *Points Of Interest*), wie z.B. Tankstellen, Restaurants, Hotels oder Flughäfen.

Abbildung 1: Beispiel für ein Infotainmentsystem [IAV06]

Ein Unterscheidungsmerkmal von Infotainmentsystemen zu modernen Mobilfunkgeräten oder PDA (Initialwort für *Personal Digital Assistant*) ist, dass die Systeme in Fahrzeugen eingebaut sind. Deshalb bieten die Systeme oft auch fahrzeugspezifische Funktionalitäten an, wie z.B. das Verstellen einer Sitzposition oder die Steuerung der Heizung oder Klimaanlage.

Für die Fahrzeughersteller ist der Einbau von Infotainmentsystemen bereits heute ein „umsatzfördernder Differenzierungsfaktor“ [UFS03] für die Ausstattung von Kraftfahrzeugen. Ähnlich wie auf dem Markt für Mobilfunkgeräte integrieren die Hersteller immer mehr Funktionalitäten in die Geräte, um sich von der Konkurrenz abzuheben und ihre Fahrzeuge somit besser vermarkten zu können.

2.2 Mensch-Maschine-Schnittstelle (HMI)

Mit der Mensch-Maschine-Schnittstelle (engl.: *Human-Machine Interface* (HMI)) bezeichnet man die Schnittstelle zwischen einem technischen Gerät und einem menschlichen Benutzer. Ein älterer, nicht geschlechtsneutraler Begriff aus dem Englischen hierfür ist *Man-Machine Interface* (MMI). Im weiteren Verlauf dieser Arbeit kürze ich den Begriff Mensch-Maschine-Schnittstelle mit dem auch in der deutschen Literatur häufig verwendeten englischsprachigen Initialwort HMI ab.

In der Informatik wird im Zusammenhang mit HMI auch oft von der Benutzerschnittstelle (engl.: *User Interface* (UI)) gesprochen. Ein Bestandteil des HMI eines Computers ist zumeist eine grafische Benutzerschnittstelle (engl.: *Graphical User Interface* (GUI)). Balzert definiert sie als ein „Grafikbildschirm bestehend aus einer [...] Arbeitsoberfläche und [...] Fenstern, über die der Benutzer mit der Anwendungssoftware interagiert und kommuniziert.“ [Bal96] Weitere Bestandteile des HMI können durch Hardware realisiert sein. So sind z.B. für einen persönlichen Computer (PC) ein Bildschirm, eine Tastatur und eine Maus übliche Bestandteile der Mensch-Maschine-Schnittstelle.

Ausschlaggebend für die Effektivität und die Akzeptanz beim Benutzer einer Maschine ist nicht nur die zur Verfügung gestellte Funktionalität. Vielmehr ist es wichtig, dass dem Benutzer die Funktionalität auf eine intuitive und ansprechende Weise zur Verfügung gestellt wird. Aus diesem Grund hat sich in den letzten Jahren die Mensch-Computer-Interaktion (engl.: *Human-Computer Interaction* (HCI)) als eine Teildisziplin der Informatik herausgebildet. Sie beschäftigt sich mit der „benutzergerechten Gestaltung von interaktiven Systemen“ [Wik05b]. Das Fach HCI behandelt neben softwaretechnischen Aspekten auch Aspekte der Psychologie, Mathematik, bildenden Kunst, Soziologie und der künstlichen Intelligenz [Joh92]. Auch linguistische Aspekte können bei der Erstellung eines HMI eine Rolle spielen. Bei der HCI handelt es sich also um ein stark interdisziplinäres Fach.

Durch diesen hohen Grad an Interdisziplinarität sind an der Entwicklung eines HMI zumeist Personen aus verschiedensten Berufsgruppen beteiligt. Die an der Soft- und Hardwareentwicklung eines Geräts beteiligten Ingenieure sind für die Beachtung von ergonomischen und designtechnischen Gesichtspunkten des HMI zumeist weniger geeignet, da sie die Dinge zu sehr vom technischen Standpunkt aus betrachten. In größeren Projekten sind deshalb häufig Grafikdesigner und Ergonomen für diese Aufgaben zuständig. Diese Interdisziplinarität erfordert eine verstärkte Kooperation zwischen Ingenieuren, Ergonomen und Grafikdesignern bei der Entwicklung eines HMI, nicht nur im Bereich von Infotainmentsystemen.

2.3 HMI eines Infotainmentsystems

Das HMI eines Infotainmentsystems besteht meistens aus der GUI der Gerätesoftware und verschiedensten Bedienelementen. Zur Ausgabe der Informationen dient in modernen Fahrzeugen im Allgemeinen ein hochauflösendes Farbdisplay. Außerdem werden Informationen des Navigationssystems den Fahrzeuginsassen in vielen Fällen durch Sprachausgabe vermittelt.

Die Straßenverkehrsordnung schreibt bezüglich der Bedienung eines Infotainmentsystems im Gegensatz zu Mobilfunkgeräten nach Aussage eines IAV-Mitarbeiters (bisher noch) nichts vor. Die Benutzung einer Standard-PC-Tastatur oder einer Maus in einem Automobil wäre jedoch vor allem für den Fahrer unpraktisch und zu stark vom Fahren ablenkend. Auch würde dies nicht der Norm DIN-EN 614-1 („Sicherheit von Maschinen – Ergonomische Gestaltungsgrundsätze“) [DIN95] entsprechen. Deshalb hat die Automobilindustrie in den letzten Jahren verschiedene Konzepte für die Bedienung von Infotainmentsystemen entwickelt.

Zum einen befinden sich oftmals Tasten für die Steuerung des Systems direkt am Gerät, wie es auch bei einem Autoradio der Fall ist. Zum anderen integrieren viele Fahrzeughersteller grundlegende Bedienelemente, wie z.B. Tasten für die Lautstärkereglung, zusätzlich noch im Lenkrad. Bei dem *iDrive*-Konzept von BMW [BMW05] wurde eine Idee des HMI eines Schaltwagens für die Bedienung von Infotainmentsystemen wiederverwendet. Wie in Abbildung 2 zu sehen ist, befindet sich dafür bei dem *iDrive*-Konzept ein sogenannter Drehdrücksteller in der Mittelkonsole zwischen dem Fahrer- und dem Beifahrersitz.

Ein weiterer Ansatz für ein Bedienkonzept eines Infotainmentsystems ist die Verwendung eines Sensorbildschirms (engl.: *touchscreen*), der es einem Benutzer ermöglicht, das Gerät durch Berühren des Bildschirms zu bedienen. Für zukünftige HMI wird auch an anderen Interaktionstechniken geforscht, wie Sprachsteuerung oder Steuerung durch Gestiken oder Blickbewegungen des Fahrers.

Abbildung 2: HMI eines Automobils der 7er Serie von BMW [MS05]

Ziel der Automobilindustrie für die Planung und Entwicklung eines HMI für ein Infotainmentsystem sollte es meiner Meinung nach sein, den Fahrzeuginsassen eine einheitliche, ansprechende und anwenderfreundliche Benutzerschnittstelle zu bieten. Dafür ist zu beachten, dass ihre Zielgruppe ein breites Spektrum der Gesellschaft abdeckt. Bei der Entwicklung des HMI müssen deshalb sowohl die Ansprüche und Fähigkeiten eines mit Pferdekutsche und Röhrenradio aufgewachsenen Urgroßvaters berücksichtigt werden, als auch die seiner technikaffinen Enkelin.

2.4 Infotainment Markup Language (IML)

In den letzten Jahren etablierten sich für die Beschreibung grafischer Benutzerschnittstellen mehrere auf der *Extensible Markup Language* (XML) basierende Formate mit ähnlich klingenden Namen, wie z.B. die *Extensible Interface Markup Language* (XIML), die *User Interface Markup Language* (UIML) oder auch die für das nächste Betriebssystem von Microsoft (Vista) entwickelte Sprache *Extensible Application Markup Language* (XAML). Eine kurze Vorstellung dieser Sprachen mit Verweisen auf weiterführende Literatur ist in [Jud04] zu finden. Die Auszeichnungssprache XML ist in [W3C02] spezifiziert.

Die *Infotainment Markup Language* (IML) ist eine von der IAV entwickelte Sprache für die Spezifizierung von HMI für Infotainmentsysteme. Auch sie basiert auf XML. Ein Vorteil des Verfassens von Dokumenten im XML-Format liegt darin, dass die so erzeugten Dokumente sowohl maschinen- als auch menschenlesbar sind. Die Maschinenlesbarkeit ist dabei nicht auf bestimmte Systeme begrenzt, was einen weiteren Vorteil von XML-Formaten ausmacht: die Plattformunabhängigkeit. Spezifikationen werden heutzutage oft noch in semiformalen oder gar informalen Sprachen geschrieben. Auf der Sprache XML basierende Spezifikationsformate haben den Vorteil, dass die Informationen damit formal und folglich konsistent, also widerspruchsfrei, festgehalten werden können. Außerdem wird so ein automatisches Generieren von Quellcode direkt aus der Spezifikation realisierbar.

Es handelt sich bei IML um eine domänenspezifische, anwendungsbezogene, deskriptive Sprache der vierten Generation. Der Begriff der Sprachen der vierten Generation (engl.: *fourth Generation Language* (4GL)) wurde laut [Wik05d] erstmals von James Martin in seinem Buch „*Applications Development Without Programmers*“ verwendet. Dieser Buchtitel trifft auch auf die Intention von IML zu. Es soll der Industrie mit der Sprache und darauf ausgelegten Werkzeugen in der Zukunft ermöglicht werden, das HMI von Anwendungen nicht mehr von Programmieren erstellen zu lassen, sondern von entsprechend ausgebildeten Ergonomen und Grafikdesignern.

IML dient zum einen der einheitlichen Spezifizierung des Aussehens und Verhaltens der grafischen Benutzeroberfläche eines Infotainmentsystems. Zum anderen lassen sich damit auch andere für ein HMI spezifische Eigenschaften beschreiben, wie z.B. Merkmale einer Sprachsteuerung. Die bei der Spezifizierung eines Systems mit Hilfe von IML entstehenden Dateien bilden zusammen die IML-Datenbasis.

2.4.1 IML-Datenbasis

Die IML-Datenbasis enthält die für eine formale Spezifikation eines HMI erforderlichen Dateien. Dies sind:

- XML-Schemata, welche die Strukturen der IML-Dateien festlegen
- IML-Dateien
  - Spezifikation von (zumeist grafischen) Elementen des HMI, sogenannten IML-Widgets
  - HMI-API: Spezifikation der von den Gerätekomponenten bereitzustellenden Funktionen, die in den IML-Widgets verwendet werden
  - *DataDictionary.xml*: Verzeichnis mit Schlüssel-Wert-Abbildungen unter anderem für Werte von im HMI verwendeten Farben und Positionen
  - Diverse andere Dateien zur Konfigurierung des Systems
- Ressourcen-Dateien
  - Grafikdateien für im HMI angezeigte Bilder
  - Audiodateien für durch das HMI abgespielte Töne oder Tonmengen
  - Sprachdateien für Übersetzungen von im HMI dargestellten Texten z.B. ins Deutsche, Englische etc.

2.4.2 IML-Widget

Der Begriff Widget bezeichnet ein nicht spezifiziertes technisches Produkt, das (noch) keinen Namen hat. Laut [Wik06e] und [Wat06] beruht die häufig gegebene Erklärung, dass es sich bei dem Begriff um ein aus den englischen Wörtern *window* und *gadget* (zu dt. etwa: Ding, Gerät) gebildetes Kofferwort handelt, auf einer Pseudo-Etymologie, da der Begriff bereits in den zwanziger Jahren des zwanzigsten Jahrhunderts, also noch vor dem ersten Computer, gebraucht wurde.

In der Informatik bezeichnet ein Widget eine Komponente einer GUI, wie z.B. ein Textfeld oder ein Auswahlkästchen (engl. *checkbox*). In IML werden „sämtliche Elemente (wie z. B. grafische Objekte, Funktionen sowie Strukturbeschreibungen) eines HMI“ [IML05] als einzelne IML-Widgets betrachtet. Auch grundlegende Verhaltensmuster, wie Schaltuhren (engl. *timer*) oder Animationen werden in Form von IML-Widgets spezifiziert. Jedes IML-Widget wird dabei in einer eigenen IML-Datei beschrieben.

Wie es bei Widgets üblich ist [Bal96, S. 552 ff.], lassen sich IML-Widgets nach einer Art Baukastenprinzip durch Objektkomposition aus anderen IML-Widgets zusammenstellen. Die Evolution eines HMI beginnt in der Basis, bei den sogenannten Basis-Widgets (IML: BaseWidget). Hierzu zählen unter anderem Textfelder, Linien und Grafiken. Aus Kombinationen von Basis-Widgets lassen sich elementare IML-Widgets erstellen. Diese lassen sich wiederum in einer beliebig tiefen Hierarchie zu Menüs kombinieren. Menüs werden in der Regel in Menünetzen zusammengefasst und diese bilden das System(-Widget). Vereinfacht ist dies anhand eines Beispiels in Abbildung 3 dargestellt. Durch das Baukastenprinzip kann sichergestellt werden, dass die Menüs HMI-weit aus sowohl vom Verhalten als auch vom Aussehen her einheitlichen IML-Widgets bestehen.

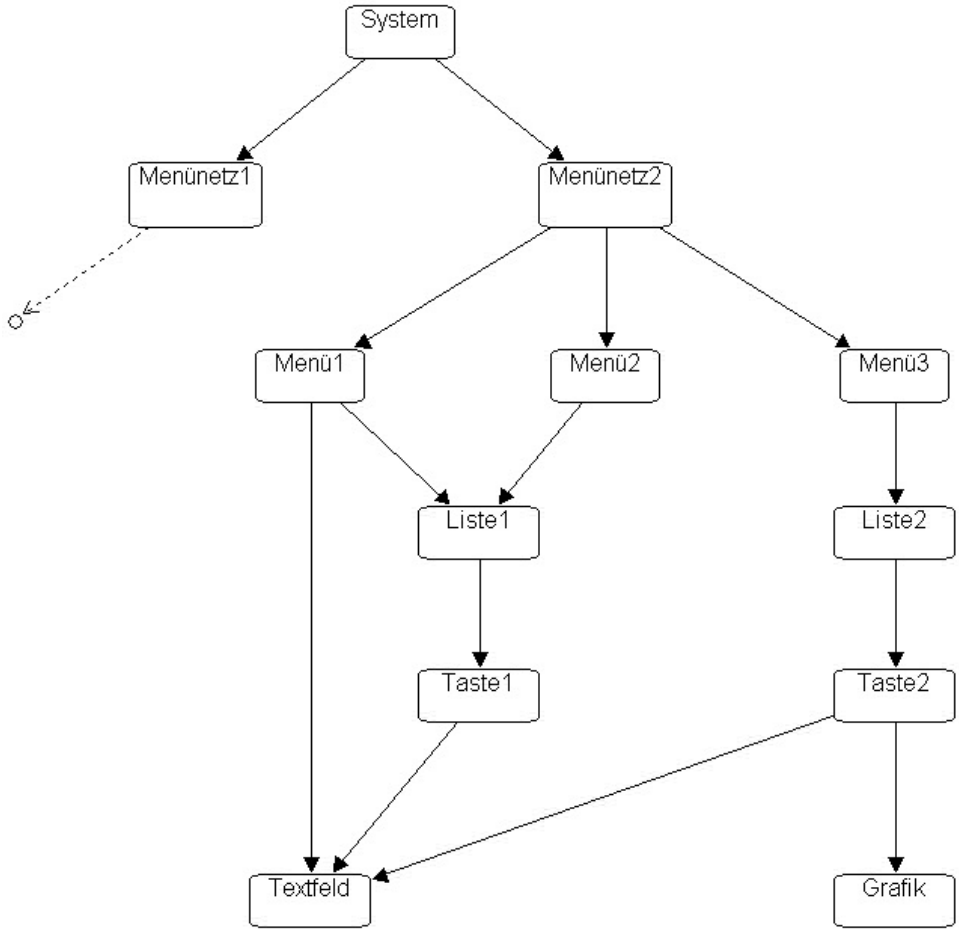

Abbildung 3: Beispielhafte IML-Widget-Hierarchie

Basis-Widgets unterscheiden sich von anderen IML-Widgets dadurch, dass sie nur aus Eigenschaften (IML: *Properties*) und nicht aus anderen IML-Widgets bestehen können. Die Nicht-Basis-Widgets werden im Folgenden „zusammengesetzte IML-Widgets“ genannt. Bei den Eigenschaften kann es sich z.B. um die Eigenschaft „Inhalt“ eines Textfelds handeln. Neben dem Namen wird zusätzlich immer noch der Datentyp der Eigenschaft, wie z.B. eine Zeichenkette oder eine Ganzzahl, spezifiziert.

Die Basis-Widgets besitzen außerdem spezifische Eigenschaften, wie z.B. eine mit dem Namen „ImplementationProperty“. Darin lässt sich eine externe, z.B. in Java geschriebene, Implementierung spezifizieren, die das Basis-Widget zur Anzeige bringen soll. Die Basis-Widgets werden ansonsten implementierungs- und damit auch plattformunabhängig beschrieben.

Jedes IML-Widget unterliegt einem XML-Schema, nach dem es aufgebaut sein muss. Dieses ist in Abbildung 35 (im Anhang 12.1, Seite 113) und in einer groben Fassung in Abbildung 4 grafisch dargestellt.

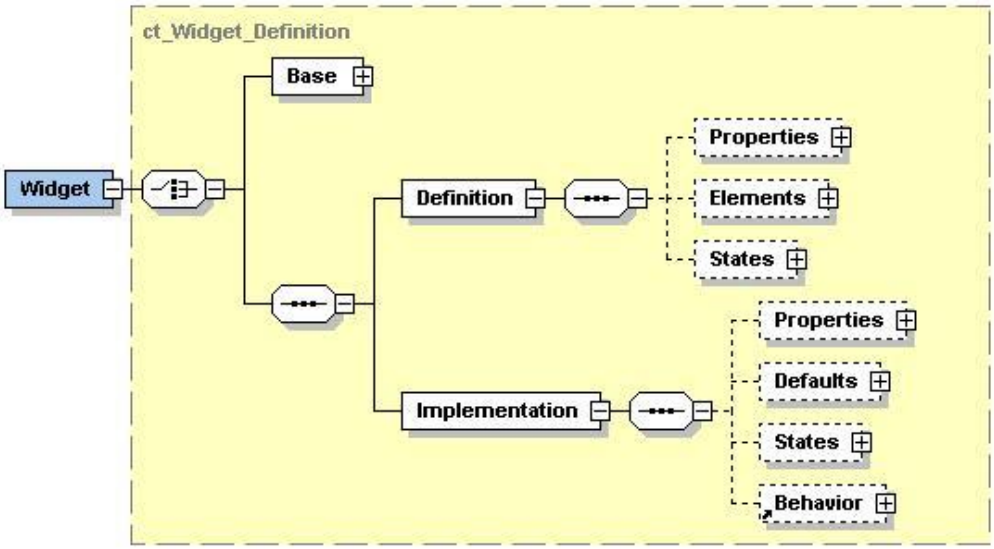

Abbildung 4: Grobes XML-Schema für IML-Widgets

Da es in der Regel nicht ausreicht, ein HMI nur aus Basis-Widgets zusammenzustellen, lassen sich die zusammengesetzten IML-Widgets durch weitere Eigenschaften im „Definition/Properties"-Zweig erweitern. Außerdem lassen sich für das IML-Widget jeweils noch Zustände (IML: *States*) definieren, um z.B. für ein Widget „Taste" die Zustände „gedrückt" und „unberührt" spezifizieren zu können.

Jedes zusammengesetzte IML-Widget besitzt zusätzlich zu dem im letzten Absatz beschriebenen Definitionsteil einen Implementierungsteil. Darin lassen sich zum einen die im Definitionsteil lediglich deklarierten widget-eigenen Eigenschaften mit konkreten Werten belegen. Zum anderen lassen sich im „Implementation/Defaults"-Zweig Werte von Eigenschaften der im Definitionsteil importierten IML-Widgets überschreiben. Dabei handelt es sich um Überschreibungen des Standardzustands (engl. *default state*) des IML-Widgets. Im „Implementation/States"-Zweig lassen sich außerdem Werte von Eigenschaften in anderen Zuständen der IML-Widgets angeben. In beiden Fällen ist es auch möglich, die Werte von in eingebundenen IML-Widgets eingebundener IML-Widgets rekursiv bis zu den Basis-Widgets zu überschreiben.

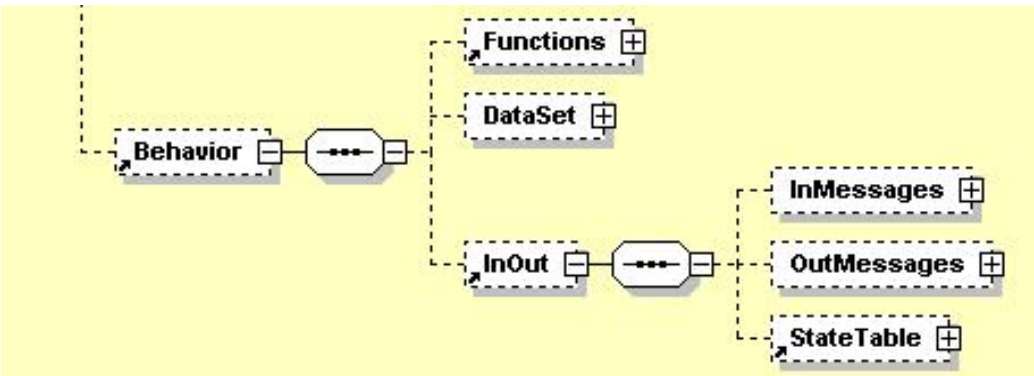

Abbildung 5: Ausschnitt aus dem Verhaltensteil des XML-Schemas für IML-Widgets

Der Implementierungsteil eines IML-Widgets umfasst außerdem noch einen Verhaltensteil, dargestellt in Abbildung 5. Darin lassen sich zum einen Funktionen der HMI-API (s. Kapitel 2.4.1) deklarieren bzw. importieren. Zum anderen können dort von anderen IML-Widgets eingehende - und an andere gerichtete - Nachrichten deklariert werden. Für eingehende Nachrichten kann eine Reaktion in einer Zustandstabelle (IML:

StateTable) spezifiziert werden. Eine Reaktion kann ein Zustandswechsel des IML-Widgets, ein Funktionsaufruf und/oder der Versand einer Nachricht sein.

Neben des dem Widget-Schema zugrundeliegenden Formats umfasst das IML-Format noch sogenannte Punktnotationen. Diese werden unter anderem verwendet, um Referenzen auf bereits definierte Eigenschaften von eingebundenen IML-Widgets herzustellen. In Abbildung 6 ist ein Basis-Widget „Bitmap“, welches eine Grafik spezifiziert, in XML dargestellt.

```
<?xml version="1.0" encoding="UTF-8"?>
<Widget d1p1:noNamespaceSchemaLocation=".\..\..\..\XSD\Widget_V20.xsd"
     Name="Bitmap" Type="WIDGET TYPE BASE"
     xmlns:d1p1="http://www.w3.org/2001/XMLSchema-instance">
  <Base Name="Bitmap">
    <Properties>
      <Property Name="BitmapProperties" ValueType="DATA TYPE_STRUCT">
        <Property Name="File" ValueType="DATA_TYPE_STRING">
          <Values>
            <FixedValue>
              <Value Value=".DICT{.STR{'Name','Einer','Datei'}}"/>
            </FixedValue>
          </Values>
        </Property>
        <Property Name="JPG" ValueType="DATA_TYPE_STRING">
          <Values>
            <FixedValue>
              <Value Value="jpg"/>
            </FixedValue>
          </Values>
        </Property>
        <Property Name="Extension" ValueType="DATA_TYPE_STRING">
          <Values>
            <FixedValue>
              <Value Value=".LINK{.PROPERTY{JPG}}"/>
            </FixedValue>
          </Values>
        </Property>
        <Property Name="Path" ValueType="DATA_TYPE_STRING">
          <Values>
            <ConditionalValue>
              <Condition Name="Path_Condition">
                <Expression>
                  <Term Term=".LINK{.PROPERTY{Extension}"/>
                  <BinaryOperator Operator="EQUAL"/>
                  <Term Term=".LINK{.PROPERTY{JPG}}"/>
                </Expression>
              </Condition>
              <Value Value="./jpg_files/"/>
            </ConditionalValue>
            <DefaultValue>
              <Value Value="./"/>
            </DefaultValue>
          </Values>
        </Property>
      </Property>
    </Properties>
  </Base>
</Widget>
```

Abbildung 6: Bsp.: Basis-Widget "Bitmap" in IML

Die Abbildung 6 zeigt (in grüner Farbe) beispielhaft vier Punktnotationen in den Eigenschaften (IML: Property) namens „File“ und „Extension“. Mit Hilfe der .STR-Anweisung werden die drei Zeichenketten „Name“, „Einer“ und „Datei“ zu „NameEi-

nerDatei“ zusammengezogen. Die .DICT-Anweisung löst den Schlüssel „NameEinerDatei“ danach mit Hilfe des DataDictionary zu einem dort hinterlegten Wert auf. Mit der in der Eigenschaft „Extension“ spezifizierten .LINK- / .PROPERTY-Anweisung wird ein Verweis auf die Eigenschaft „JPG“ erzeugt, so dass die Eigenschaft „Extension“ in diesem Fall die gleiche Zeichenkette zugewiesen bekommt, wie die Eigenschaft „JPG“.

Insgesamt existierten zum Zeitpunkt der Untersuchung nach einer Emailbefragung aller – mir bekannter - weltweit aktiven IML-Entwickler mindestens 16 verschiedene Punktnotationen. Erwähnt seien an dieser Stelle jedoch nur noch die .LANG- und die .FUNCTION-Anweisungen. Ähnlich wie bei der .DICT-Anweisung erfolgt bei der .LANG-Anweisung eine Auflösung eines Schlüssels, nur dass in diesem Fall das Schlüssel-Wert-Paar in der Sprachdatei der aktiven Benutzeroberflächen-Sprache (z.B. Englisch oder Spanisch) gesucht wird. Damit wird die Spezifizierung eines mehrsprachigen HMI erleichtert. Mit Hilfe der Punktnotationen wird auch auf die im Verhaltensteil eines IML-Widgets spezifizierten Funktionen zugegriffen. Eine .FUNCTION-Anweisung führt eine Funktion aus und liefert falls vorhanden einen Rückgabewert zurück.

In der Beschreibung der Eigenschaft „Path“ ist beispielhaft eine IML-Bedingung angegeben. Hierbei handelt es sich um eine bedingte Wertzuweisung. Bedingungen können in IML jedoch auch zu anderen Zwecken eingesetzt werden, z.B. für einen bedingten Nachrichtenversand oder einen bedingten Funktionsaufruf.

Die hier gegebene Erläuterung zum Thema IML soll dem Leser lediglich einen ersten Einblick in die Sprache gewähren und dient vor allem dazu, ihm eine Grundlage für das Verständnis der nachfolgenden Kapitel zu schaffen. Eine mehr als 100 Seiten lange und damit umfassendere Beschreibung der Sprache ([IML05]) kann bei der IAV erfragt werden.

Im nun folgenden Text bezeichnet der Begriff Widget ein IML-Widget.

2.4.3 Kritiken am IML-Format

Mit der Sprache IML schlägt die IAV ein Modell vor, welches das Aussehen und das Verhalten von Widgets nicht voneinander trennt. Beides wird zusammen in einer Datei beschrieben. Als „defacto Standard für den Grobentwurf aller komplexen Softwaresysteme“ [Wik06f] gilt das Beobachtermuster Model-View-Controller (kurz MVC) (s. [GHJV96, S. 5 ff.]). Darin werden das Datenmodell, die Darstellungsschicht und die Steuerungsschicht (in der Theorie streng) von einander getrennt. „In den Zeiten vor MVC tendierten Entwürfe von Benutzungsschnittstellen dazu, diese Objekte in einem einzigen Objekt zusammenzuführen. Das MVC-Paradigma entkoppelt sie, um die Flexibilität und Wiederverwendbarkeit zu erhöhen.“ [GHJV96, S. 5]

Das hat unter anderem auch den Vorteil, dass die für die Entwicklung einer GUI anfallenden Arbeiten auf Grafikdesigner und Softwareentwickler aufgeteilt werden können. Das Speichern von Informationen über Aussehen und Verhalten in einer Datei hingegen erschwert das parallele Arbeiten von Grafikdesignern und Softwareentwicklern, da zu einem Zeitpunkt nur eine Person die aktuelle Version der Datei besitzen und bearbeiten kann.

Die IAV hat sich laut eines Mitarbeiters aus technischen Gründen dazu entschieden, keine diesbezügliche Trennung vorzunehmen. Das Aussehen eines Widgets hängt oft von dessen Zustand ab. Der Zustand wird mitunter dynamisch (zur Laufzeit) in der Gerätesoftware festgelegt und in IML per Funktionsausruf abgefragt. So wird z.B. eine Taste in einer GUI oft ausgegraut dargestellt, wenn die durch die Taste aktivierbare

Funktionalität zur Benutzungszeit nicht zur Verfügung steht. In Abbildung 7 ist z.B. die Funktionalität „Einzahlung“ ausgegraut dargestellt.

Abbildung 7: Mensch-Geldmaschine-Schnittstelle einer deutschen Bank[1]

In einer Beschreibungssprache für GUI-Widgets muss es also möglich sein, sich bei der Beschreibung des Aussehens auf verhaltensbedingte Zustände beziehen zu können. Wenn Entwickler Verhalten und Aussehen jeweils in einer gesonderten XML-Dateien beschreiben würden, müsste ein neues Sprachkonstrukt eingeführt werden, um diese Beziehungen herstellen zu können. Dieses Konstrukt ließe sich jedoch nicht mit einem XML-Schema überprüfen. Es ließe sich damit nicht mehr prüfen, ob eine Eigenschaft des Verhaltens, auf die sich in der Aussehensbeschreibung bezogen wird, überhaupt existiert. Eine Lösung, auf die ein IAV-Mitarbeiter in einer diesbezüglichen Diskussion mit mir kam, könnte folgende sein: Die Struktur der Widgets sollte zwar weiterhin im XML-Format spezifiziert werden, jedoch im XML-Schema-Format, da es damit möglich ist, andere Dateien in einer Datei zu inkludieren und sich auf deren Elemente und Eigenschaften zu beziehen.

Am Rande bemerkt: Auch Microsoft trennt in der neuen GUI-Spezifikationssprache XAML im Gegensatz zu früher (vgl. MFC [Kru06]) das Aussehen und Verhalten nicht mehr von einander: „XAML has very intimate ties with the Avalon class library: every element type you can use in XAML is actually a class and, specifically, a descendent of the UIElement or ContentElement classes declared in the MSAvalon.Windows namespace. Among the descendents of UIElement is Control, from which are descended all the common user-interface controls such as buttons, scroll bars, list boxes, edit fields, and so forth.” [MS06] (siehe Abbildung 8) Dies soll hier jedoch kein Pro-Argument für die Ehe von Aussehen und Verhalten sein.

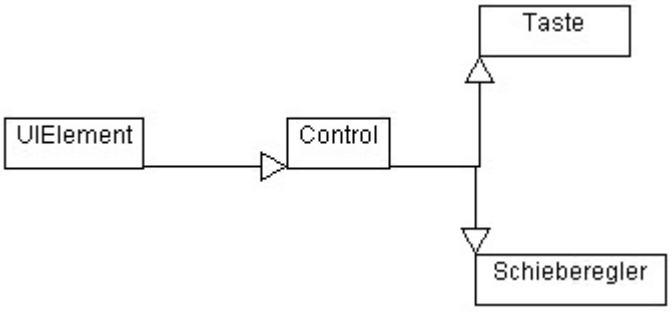

Abbildung 8: Ausschnitt aus der Klassenhierarchie von Avalon

[1] Keine demokratische Wahl: Da nur eine Antwort möglich ist, kann sich die Maschine in diesem Fall auch die Frage sparen.

Ein weiterer Kritikpunkt, auf den ich beim Erarbeiten dieses Kapitel stieß, sind die oben kurz beschriebenen Punktnotationen. Diese sind in meinen Augen ein Bruch in der Sprache, da eine schemabasierte Prüfung der Punktnotationen wie sie zur Zeit vorgenommen werden, nicht möglich ist. Es ließen sich zwar reguläre Ausdrücke für die Überprüfung verwenden, jedoch kann damit nicht alles abgedeckt werden, da wie in Abbildung 6 zu sehen ist, die Punktnotationen auch in geschachtelten Ausdrücken verwendet werden können. Es war für die Prüfung der Korrektheit in den untersuchten Projekten deshalb erforderlich ein extra Testwerkzeug zu entwickeln. Auch hier wäre eine schemabasierte Spezifikation möglicherweise eine Lösung. Ein anderer Lösungsweg wäre es, das XML-Schema-Format um kontextfreie Grammatiken zu erweitern.

3 Grundlagen zur Untersuchungsmethodik

Ein Ziel der Arbeit war es, die Prozessqualität quantitativ zu bestimmen. Nachfolgend erläutere ich deshalb die Begriffe Qualitätskriterien und Maße und gebe eine kurze Einführung in die *Goal, Question, Metric*-Methode, die ich in der Arbeit für Messungen angewendet habe.

3.1 Qualitätskriterien im Software-Ingenieurwesen

Der Begriff Qualität wird nach DIN 55350 wie folgt definiert: „Unter Qualität versteht man die Gesamtheit von Eigenschaften und Merkmalen eines Produkts oder einer Tätigkeit, die sich auf deren Eignung zur Erfüllung gegebener Erfordernisse bezieht.“ [SoK06a] Diese Eigenschaften und Merkmale werden auch als Qualitätskriterien bezeichnet. Nach der Norm ISO/IEC 9126 sind sechs Qualitätskriterien für Softwareprodukte standardisiert, die in folgender Grafik dargestellt sind.

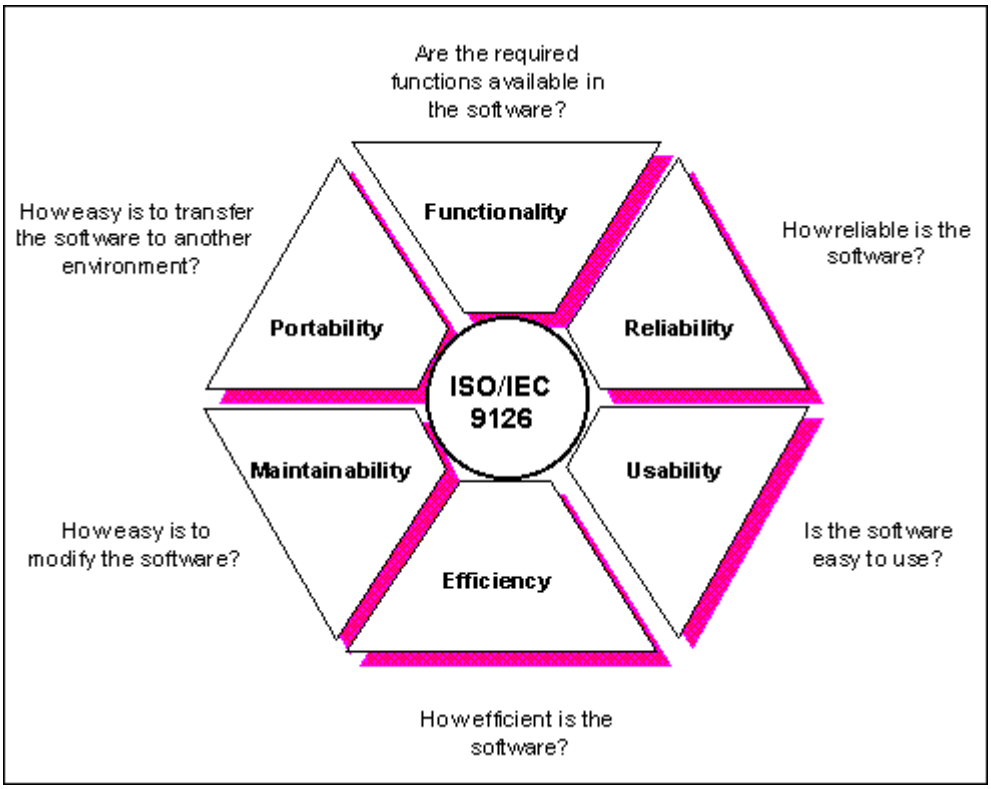

Abbildung 9: Qualitätskriterien für Softwareprodukte nach ISO/IEC 9126 [CSE06]

Diese sechs Kriterien lassen sich wie in folgendem Ursache-Wirkungsdiagramm geschehen in weitere Subkriterien aufschlüsseln. Definitionen zu den Subkriterien sind unter anderem auf [CSE06] zu finden.

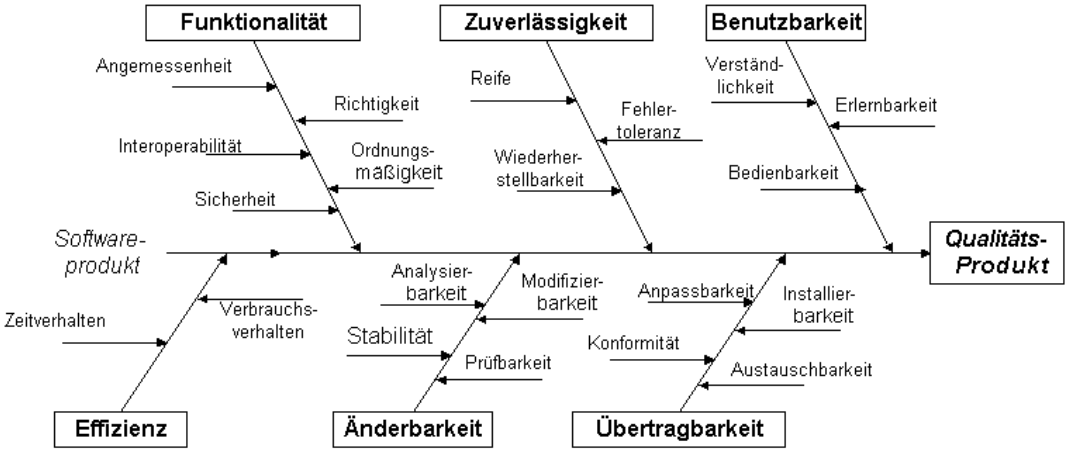

Abbildung 10: Ursache-Wirkungsdiagramm für Qualitätskriterien [Dum06]

Da die Automobilhersteller sich in den letzten Jahren vermehrt über die mangelnde Effizienz der Automobilentwicklungsprozesse in Deutschland beklagen, scheint die Effizienz auch für Softwareentwicklungsprozesse in der Automobilbranche ein wichtiges Qualitätskriterium zu sein. Eine genauere Betrachtung dieses Prozessmerkmals lässt erkennen, dass es allumfassend ist.

Nach der Norm ISO 9000 ist Effizienz definiert als das „Verhältnis zwischen dem erreichten Ergebnis und den eingesetzten Ressourcen“ [DIN00, S. 22], also als der Kehrwert des Aufwand-Leistungs-Verhältnisses. In einem Softwareentwicklungsprozess lässt sich die Leistung anhand der Qualität des erzeugten (Zwischen-)Produkts unter Berücksichtigung des Produktumfangs bestimmen. Die Qualität wird mit den oben stehenden Produktqualitätskriterien gemessen. Der Aufwand bestimmt sich aus den Kosten, die durch die Ausübung des Prozesses entstehen. Im Software-Ingenieurswesen ist der entscheidende Faktor hierfür die Zeit, die von den Entwicklern für die Produkterstellung benötigt wird, da materielle Aufwände hier vernachlässigbar gering sind. Auch die Zeit, die zum Erlernen des Prozesses benötigt wird, kann den Aufwand beeinflussen.

Die Qualitätskriterien Aufwand und Leistung können je nach Einsatzort und Rahmenbedingungen unterschiedlich gewichtet sein. Z.B. ist bei der Entwicklung einer Software für die Steuerung einer ESA-Rakete (höchstwahrscheinlich) die Zuverlässigkeit und damit auch die Leistung höhergewichtet als die Kosten der Entwicklung.

Auswirkungen auf die Effizienz eines Softwareentwicklungsprozesses haben unter anderem folgende Prozessqualitätskriterien:

- der Reifegrad: z.B. gemessen mit Hilfe des *Capability Maturity Models* (CMM, s. [Hum89])
- die Anpassbarkeit des Prozesses: Wie flexibel kann auf Änderungswünsche und neue Anforderungen reagiert werden?
- der erforderliche Kommunikationsaufwand der Prozessbeteiligten untereinander
- Unterstützung bzw. Grad der Ermöglichung von oben stehenden Produktqualitätskriterien (wie z.B. Wiederverwendung)

Zur Ermittlung der Ausprägungen von Qualitätskriterien, bietet es sich an, Messungen vorzunehmen.

3.2 Maße für die Softwareentwicklung

Ein Maß ist eine Abbildung von empirischen Merkmalen von Objekten der realen Welt auf formale Objekte, also Zahlen oder Symbole [Zus91, S. 29] [KoS05, F. 4]. In der Informatikliteratur wird in diesem Zusammenhang auch oft der Begriff der Metrik synonym verwendet. Da dieser Begriff in einer Mutterwissenschaft der Informatik, der Mathematik, jedoch (ungeschickter Weise[1]) ein spezielles Maß, nämlich ein Abstandsmaß bezeichnet [Zus91, S. 29], und damit, objektorientiert gedacht, eine Verwendung des Begriffs Metrik für etwas weniger spezielles in der Informatik einem Typfehler gleichkäme, bleibe ich hier bei dem „korrekten“ [Zus91, S. 28] Begriff Maß.

Die Ausprägung eines Maßes lässt sich durch Messung bestimmen. Der Zweck einer Messung ist es, das Messobjekt analysieren, bewerten oder verbessern zu können oder aber Vorhersagen über ein Objekt treffen zu können.

Maße lassen sich in direkte und indirekte Maße unterscheiden. Direkte Maße sind Maße, welche unmittelbar aus einer Messung resultieren, wie z.B. die Dauer eines Erstellungsprozesses. Dies muss nicht immer der Fall sein. So lassen sich Maße auch durch Verknüpfung bereits vorhandener Maße definieren. Ein auf diese Weise gewonnenes Maß wird dann als indirektes Maß bezeichnet. Zur Messung von Qualitätskriterien im Software-Ingenieurwesen werden zumeist indirekte Maße herangezogen. Ein Beispiel dafür ist die Summe der in einem Entwicklungsprozess durch Testen und Durchsichten aufgedeckten Defektanzahlen.

Im Ingenieurwesen lassen sich Maße außerdem noch nach Produkt- und Prozessmaßen differenzieren (vgl. [Pre99]). Ein Prozessmaß charakterisiert dabei ein Merkmal eines Entwicklungsprozesses. Typische Beispiele für Softwareentwicklungsprozessmaße sind:

- Zeitaufwand für einen (Teil-)Prozess,
- Kosten für die Ausführung einen (Teil-)Prozess,
- Zeit-/Planabweichungen,
- Anzahl der Änderungen an Dokumenten und Zwischenprodukten,
- Anzahl von Besprechungen oder Durchsichten,
- Anzahl aufgedeckter Defekte, getrennt nach Phasen des Defekteinbaus.

Ein Beispiel für ein grob granuliertes Prozessmaß ist das von Humphrey in [Hum89] vorgestellte Prozessbewertungsrahmenwerk CMM [Pfl97].

Produktmaße beschreiben Eigenschaften eines Produkts und lassen sich in interne und externe Maße unterscheiden. Ein Produkt kann dabei sowohl das Endprodukt eines Prozesses sein als auch ein Zwischenprodukt, wie z.B. ein Spezifikationsdokument. Als internes Produktmaß werden Maße bezeichnet, die sich unmittelbar auf das Produkt beziehen [BeR04, S. 36]. Beispiele für interne Maße für Software sind dessen Komplexität, Defektanzahl oder auch die Anzahl der Quelltextzeilen. Externe Produktmaße wiederum “beschreiben Merkmale des Produkts, die von außen sichtbar sind“ [Pre99]. Sie setzen sich zumeist indirekt aus internen Produktmaßen zusammen. Maße, welche die Benutzbarkeit einer Software beschreiben, sind z.B. externe Produktmaße.

Eine disjunkte Trennung zwischen Prozess- und Produktmaßen ist nicht immer möglich. So kann die Anzahl von in einem erstellten Produkt gefundenen Defekten dazu dienen, Aussagen sowohl über die Prozess- als auch die Produktqualität zu treffen. In der Literatur (vgl. z.B. [Fen91], [Pre99]) wird teilweise außerdem noch in Ressourcenmaße unterschieden. Darunter fallen Maße, welche die in einem Entwicklungsprozess zur Verfügung stehenden Ressourcen, wie z.B. das Entwicklungsteam, charakterisieren.

[1] Eine Übersetzung aus dem Griechischen des Begriffs Metrik ergibt Zählung oder Messung [Wik05c].

Auch hier ist eine scharfe Trennung nicht immer möglich. So kann z.B. der Zeitaufwand für ein Softwareentwicklungsprojekt einerseits als Prozess- und andererseits auch als Ressourcenmaß eingeordnet werden. Ressourcenmaße bilden aber auch keine Teilmenge der Prozessmaße, da z.B. die Größe und Fähigkeiten eines Entwicklungsteams nicht als Prozessmaß zu bezeichnen sind.

3.2.1 Skalen

Maße lassen sich nach den ihnen zugrundeliegenden Skalen unterteilen. „Eine Skala ist eine Zuordnungsvorschrift, die die Regeln angibt, nach der empirischen Beobachtungen (Merkmalsausprägungen) Zahlen [oder Symbole] zugeordnet werden." [Jai05] In der Literatur ([Zus91], [Pre99], [Rei02], [Jai05]) wird zumeist in die, in der folgenden Tabelle zusammengefassten, fünf Skalenniveaus unterschieden.

Skalentyp	Charakteristika	Merkmalsart	Erlaubte math. Operationen	Erlaubte Transformationen	Beispiele
Nominalskala	ungeordnete Klassifizierung, Äquivalenzrelation	qualitativ	=, ≠	g: eineindeutig	Programmiersprache, Postleitzahl, Geschlecht
Ordinalskala	Rangordnung, Ordnungsrelation		=, ≠, <, >	g: streng monoton steigend	Prozessreifegrad nach CMM, Platzierung beim Sport, Windstärke in Beaufort
Intervallskala	Abstände zwischen Merkmalsausprägungen exakt bestimmbar, diskret quantitativ	quantitativ	=, ≠, <, >, +, -	g(x) = a x + b mit a > 0	Komplexität nach McCabe, Temperatur in Celsiusgraden, Intelligenzquotient
Verhältnisskala (auch Rationalskala genannt)	Abstände zwischen Merkmalsausprägungen sind gleich groß, absoluter Nullpunkt, kontinuierlich quantitativ		=, ≠, <, >, +, -, *, /	g(x) = a x mit a >0	Projektdauer, Projektkosten, Temperatur in Kelvingraden, nach Behebungskosten gewichtete Anzahl von bei Durchsichten aufgedeckter Defekte
Absolutskala	Wie die Verhältnisskala, jedoch sind die Merkmalsausprägungen ausschließlich absolute Werte			g(x) = x	Gemessene Anzahlen, wie die Anzahl von bei Durchsichten aufgedeckter Defekte

Tabelle 1: Übersicht über Skalentypen

Je tiefer in der Tabelle 1 abgestiegen wird, desto genauer sind die Daten und folglich größer ist der Informationsgehalt. Dementsprechend sind Maße umso besser für Statistiken verwendbar, je niedriger das Skalenniveaus ist. So ist z.B. die Berechnung eines Mittelwerts nicht für Maße mit zugrundeliegender Ordinalskala möglich, sondern erst ab der Intervallskala. Die höheren Skalen sind jeweils Spezialisierungen der darunter liegenden Skalen. Das heißt eine Ordinalskala ist z.B. eine Nominalskala, die um die Eigenschaft der vollständigen Ordnung erweitert ist. In Abbildung 11 ist diese Hierarchie der beschriebenen Skalen dargestellt.

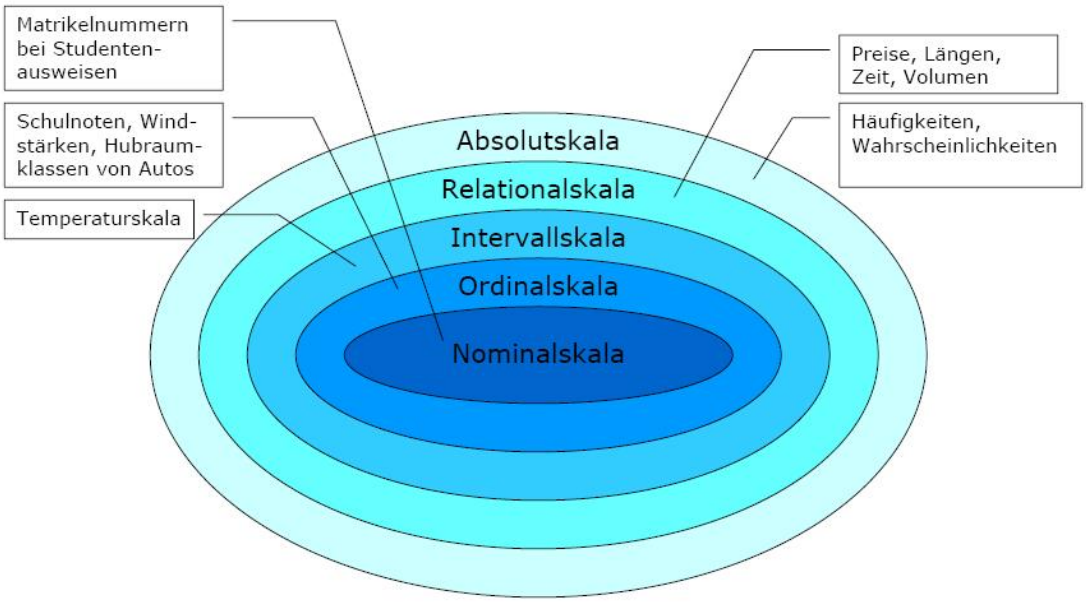

Abbildung 11: Skalenhierarchie[1] [Fäh02]

3.2.2 Kriterien an ein ideales Maß

Ideale Maße sollten nach Mills [Mil88] folgende Eigenschaften erfüllen:

- einfach und präzise definierbar, so dass eine spätere Datenanalyse (auch durch einen Dritten) einfach möglich ist
- objektiv, so dass das Maß aussagekräftig ist
- einfach zu erheben, so dass der für das Messprogramm anfallende Aufwand noch in angemessener Relation mit dem daraus resultierenden Nutzen steht
- validiert, so dass sichergestellt ist, dass das verwendete Maß das misst, was beabsichtigt wird, das es misst
- robust, so dass nicht signifikante Veränderungen des Messobjekts keine größeren Auswirkungen auf das Maß haben

Diese Kriterien können aber nicht immer vollständig erfüllt werden. So ist es z.B. schwierig, ein hundertprozentig objektives Maß für die Benutzbarkeit einer Software zu definieren, da sie oft auch von den subjektiven Eindrücken und Erwartungen von Benutzern abhängt. Bei der Aufstellung eines Maßes sollte jedoch trotzdem darauf geachtet werden, die Kriterien, soweit es der Kontext der Untersuchung erlaubt, zu erfüllen.

Ein Maß sollte auch konsistent, das heißt frei von Widersprüchen zu anderen Maßen sein, um die Aussagekraft des Maßes zu gewährleisten. Die Konsistenz schließt dabei

[1] Die hier mit Relationalskala betitelte Skala entspricht der oben beschriebenen Rationalskala.

auch mit ein, dass es für einen Dritten möglich sein muss, zu den gleichen Ergebnissen zu gelangen. Dies ist z.B. für das Aufwandsabschätzungsmaß der *Function Points* nicht immer der Fall [Pres97]. Das heißt aber eben nicht, dass dieses Maß unbrauchbar ist, sondern lediglich, dass es nicht perfekt ist. Ein perfektes Schätzmaß zu suchen, widerspricht sich jedoch auch schon in sich.

Ein nicht zwingend erforderliches aber wünschenswertes Kriterium an ein gutes Maß ist, dass es eine automatische Erhebung erlaubt. Zum einen ist dies im Zweifel billiger und zum anderen hat eine manuelle Datenerhebung oft den Nachteil, dass an der Entwicklung beteiligte Personen von ihrer Arbeit abgehalten werden. In einem Projekt zur Untersuchung des *Personal Software Process* stellte Phillip Johnson fest, dass für viele Softwareentwickler schon alleine das Drücken einer Taste als zu großer Mehraufwand wahrgenommen wird, wenn es darum geht, Daten über ihre Aktivitäten zu erheben [John01].

„Der Zweck von Metriken liegt nicht im blindwütigen Erfassen von Daten“ [Tha93, S. 262]. Ein weiteres Kriterium an ein gutes Maß ist deshalb, dass es zielgerichtet sein sollte. Das heißt ein Maß sollte immer auf Grund eines übergeordneten Messziels definiert werden, da es ansonsten schnell passieren kann, dass ein Messprogramm zum Selbstzweck wird.

3.3 Goal, Question, Metric (GQM) -Paradigma

Messprogramme sind im Software-Ingenieurwesen zumeist dadurch motiviert, den Entwicklungsprozess verbessern zu wollen. Messen an sich verbessert eine Software oder einen Prozess jedoch noch nicht. Es kann aber mit Hilfe einer durchdachten Vorgehensweise geeignet sein, eine Grundlage für Verbesserungen zu schaffen.

Die Wissenschaftler, die sich mit der Vermessung von Software und Softwareentwicklungsprozessen beschäftigen, stimmen größtenteils darin überein, dass für eine effektive Vermessung eine Zielsetzung gegeben sein muss. Oftmals lag das Scheitern von in Firmen eingeführten Programmen zur Vermessung daran, dass sich zu stark auf das Erheben von Daten konzentriert wurde und weniger darauf, was mit den Messungen in Erfahrung gebracht werden sollte [Kit96]. Um nützliche Daten zu sammeln und dabei effektiv vorzugehen, sollten deshalb, bevor mit einer Datenerhebung begonnen wird, Ziele für die Untersuchung aufgestellt werden.

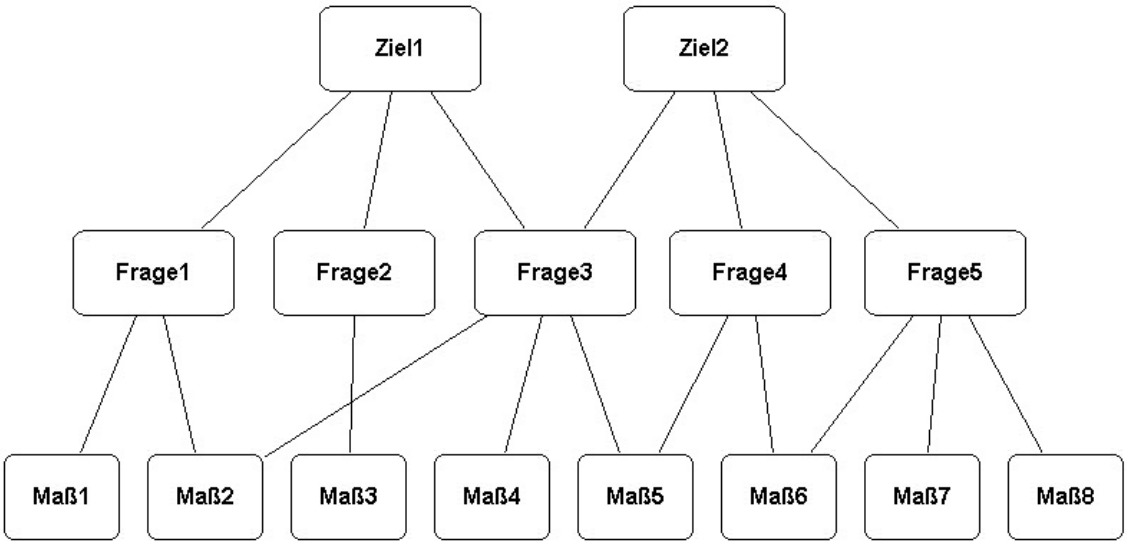

Abbildung 12: Hierarchische Struktur eines beispielhaften GQM-Plans

Ein auf diesem Prinzip der Zielorientierung basierender Ansatz für Messvorhaben ist das Goal, Question, Metric (GQM) -Paradigma. Es wurde 1984 von Basili und Weiss an der University of Maryland im Rahmen eines NASA-Projekts [BW84] als Resultat von praktischer Erfahrung und akademischer Forschung eingeführt und später unter anderem von Rombach [BR88] weiter entwickelt.

Die Planung des Messprogramms erfolgt beim GQM-Paradigma nach einem absteigenden Ansatz (s. Abbildung 12). Zuerst wird ein Auswertungsziel (*Goal*) definiert, das bestimmt, was die Messung bezwecken soll. Für das Ziel werden Fragestellungen (*Question*) abgeleitet, die Aufschluss über den Fortschritt der Realisierung des Ziels geben sollen. Erst auf Grundlage der Fragen werden die Maße (*Metric*) aufgestellt. Sie dienen einer quantitativen Beantwortung der Fragen.

Messpläne werden dadurch zielgerichtet abgeleitet. Es werden keine überflüssigen Messungen vorgenommen. Auch wird damit schon vorab ein Rahmen für die Datenanalyse bestimmt. Nach der Datenerhebung werden die gewonnenen Daten aufsteigend interpretiert. Sie werden zur Beantwortung der aufgestellten Fragen herangezogen, um entscheiden zu können, in wie weit das definierte Ziel erreicht ist.

Ein Auswertungsziel wird zumeist ausgehend von einem Geschäftsziel, das vom Management vorgegeben wird, bestimmt. Ein Geschäftsziel kann unter anderem auch aus einem vorherigen Messprogramm resultieren. Durch eine Messung können Problembereiche des untersuchten Objekts aufgedeckt werden, die bei der Planung der Messung noch nicht bekannt waren. Ein Geschäftsziel für ein neues GQM-Programm kann es dann sein, diese entdeckten Probleme zu beheben.

Für die Definition eines Messziels wird in der Literatur (vgl. [SoB99], [SoK06b]) zumeist vorgeschlagen, sich an der in Tabelle 2 dargestellten Vorlage zu orientieren.

Analysiere das	<Objekt>
zum Zwecke des	<Zweck>
im Hinblick auf	<Fokus>
aus der Perspektive von	<Perspektive>
im Kontext von	<Kontext>

Tabelle 2: Vorlage für die Definition eines Messziels

Das *Objekt* der Analyse bzw. der Untersuchungsgegenstand kann bei einer Messung z.B. ein (Teil-)Prozess, ein Projekt, ein Produkt oder eine Ressource sein. Eine Messung sollte wie bereits erläutert immer zu einem bestimmten *Zweck* erfolgen. Dieser kann z.B. sein, Verständnis über das Objekt der Analyse zu erlangen, Vorhersagen darüber treffen zu können, es zu bewerten, zu steuern oder es zu verbessern. Um eine Eingrenzung der zu untersuchenden Eigenschaften des Objekts vorzunehmen, wird der *Fokus* der Untersuchung bestimmt. Hier stellt sich die Frage, welche Qualitätskriterien fokussiert werden sollen.

Die *Perspektive* gibt an, aus wessen Blickwinkel die Daten gesammelt werden sollen. Mögliche Perspektiven sind die eines Projektteams, einzelner Rollen, eines Projektleiters, des Managements oder auch die eines (End-)Benutzers einer entwickelten Software. Außerdem wird in der Vorlage der Messzieldefinition noch der *Kontext* der Untersuchung mit angegeben. Hier kann es sich beispielsweise um ein bestimmtes Projekt oder auch um eine Abteilung einer Firma, in der die Messungen vorgenommen werden sollen, handeln.

4 Empirische Untersuchung des TeleDrive VISION Prozesses

Dieses Kapitel beschreibt die Zielsetzung und Vorgehensweise, mit der ich die empirische Untersuchung des TeleDrive VISION Prozesses vornahm. Um den Prozessablauf aufzunehmen, zu analysieren und Stärken und Schwächen des Prozesses aufzuspüren, führte ich Interviews mit am Prozess beteiligten Mitarbeitern durch. Die Methodik der Studie ist in diesem Kapitel beschrieben. Die Ergebnisse sind in den nachfolgenden zwei Kapiteln festgehalten.

4.1 Ausgangssituation

Zum Zeitpunkt meiner Untersuchung war der Prozess noch relativ jung. Es wurde bis dahin noch kein Projekt vollständig damit durchgeführt. Die zwei untersuchten Projekte (s. Kapitel 1.3) waren laut Herrn Wegner die einzigen, die den Prozess bisher in seiner vollen Breite, das heißt mit Spezifikation in IML, durchlebten. Die Projekte waren beide noch nicht vollständig abgeschlossen. Projekt A stand jedoch bereits kurz vor dem Abschluss. Dadurch waren Daten über bereits beendete Projekte nicht verfügbar. Zudem waren Messungen zu Beginn dieser Arbeit noch kein Bestandteil des Prozesses, wodurch während der Projekte kaum quantitative Daten über den Prozess und dessen (Zwischen-) Produkte gesammelt wurden.

Nach Aussage von Herrn Wegner existiert in der IAV zwar ein SPICE-Prozess (s. z.B. [Sok06d]) für die Softwareentwicklung im Allgemeinen, jedoch wird dieser lediglich bei Eigenentwicklungsprojekten eingesetzt. Bei Dienstleistungsprojekten, wie den hier untersuchten, passt sich die IAV an die Prozesse des jeweiligen Auftraggebers an.

Detaillierte Informationen über den Ablauf des TeleDrive VISION Prozesses in schriftlicher Form waren noch nicht vorhanden. Vielmehr waren die Informationen verteilt über die Köpfe der Mitarbeiter mehrerer Abteilungen. Der Prozess wurde also bereits gelebt jedoch nicht schriftlich festgehalten.

4.2 Problemstellung und Zielsetzung

Eine Prozessbeschreibung in schriftlicher Form hat den Vorteil, dass sich damit den beteiligten Parteien zum Start eines Projekts der Prozess besser erläutern lässt. Außerdem bietet es neuen Mitarbeitern die Möglichkeit, sich in das Themengebiet auch ohne (formlose) Interviews einzuarbeiten und den Hintergrund ihrer Tätigkeit in Erfahrung zu bringen.

Ein Ziel der Studie war es deshalb, ein detailliertes Prozessmodell zu erstellen. Der Prozess sollte in seine Teilprozesse zerlegt werden und es sollten die beteiligten Rollen bestimmt und charakterisiert werden. Für die Rollenbeschreibung lag ein Fokus auf deren Kooperation. Es sollte untersucht werden, in welchen Teilprozessen die Rollen mit anderen am Prozess beteiligten Rollen kooperieren.

Außerdem sollte mir die Erstellung der Prozessdokumentation notwendige Einsichten für nachfolgende Untersuchungen verschaffen. Ein weiteres Ziel war es dabei, Stärken und Schwächen des Prozesses aufzuzeigen. Mit der Untersuchung sollten Problembereiche im Prozess aufgedeckt werden, um Ansatzpunkte für Prozessverbesserungsvorschläge zu erlangen.

Da der Spezifizierungsprozess den innovativen Part und Kern des TeleDrive VISION (Gesamt-)Prozesses darstellt, entschied ich mich dazu, diesen in einer tiefergehenden Betrachtung zu analysieren. Eine ähnlich detaillierte Betrachtung des Gesamtprozesses hätte aufgrund seines Umfangs, wie er auch schon aus den unter 1.3 stehenden Projekt-

beschreibungen hervorgeht, außerhalb des Rahmens einer Diplomarbeit gelegen. Der Gesamtprozess sollte deshalb nur grob eingeteilt und analysiert werden.

Der TeleDrive VISION Prozess befand sich zum Zeitpunkt dieser Arbeit durch die Einführung eines von der IAV entwickelten Werkzeugs, des HMIStudios (s. Kapitel 5.4), im Umbruch. In den von mir untersuchten Projekten wurde es jedoch zu Beginn der Untersuchung noch nicht eingesetzt. Deshalb lag die Konzentration der Studie vor allem auf den Schnittstellen zwischen den einzelnen Teilprozessen, also den Prozesszwischenprodukten, und weniger auf den konkret auszuführenden Aktivitäten. Die tatsächlichen Kleinstprozesse wurden also nicht betrachtet, da davon auszugehen war, dass sie sich bei Einsatz des HMIStudios von den vorherigen unterscheiden werden.

4.3 Fragestellung

Die von mir durchgeführte Studie sollte für die Analyse des gesamten TeleDrive VISION Prozesses folgende Fragen beantworten:

1. Was ist die Motivation und die Grundidee für den Prozess?
2. Aus welchen Teilprozessen setzt sich der Prozess zusammen?
3. Wie lassen sich die Teilprozesse charakterisieren?
4. Wie sind die Teilprozesse zueinander angeordnet?

Auch bei der Untersuchung des Spezifizierungsprozesses war es erforderlich, zu analysieren, aus welchen Teilprozessen er besteht. Aufgrund des Entschlusses die Spezifizierung der Infotainmentsysteme konzentrierter zu betrachten, war hier die Zielsetzung, eine feinere Zerlegung zu erhalten. Zur Charakterisierung der Teilprozesse sollten detaillierte Informationen über folgende Bereiche erhoben werden:

- Ziele des Teilprozesses
- Vorbedingungen
- Eingaben
- Beteiligte Rollen
- Aktivitäten
- Eingesetzte Werkzeuge
- Verwendete Methoden, Richtlinien und Konventionen
- Erforderliche Kommunikation und Kooperation mit anderen Rollen
- Ergebnisse
- Nachbedingungen

4.4 Wahl der Methodik

Die Daten, die ich erheben konnte, waren ausschließlich qualitativer Natur. Ich habe mich deswegen dazu entschieden, für die Datenerhebung qualitative Einzelinterviews mit den Prozessbeteiligten durchzuführen. Die dabei angewandte Technik orientierte sich an der des problemzentrierten Interviews, wie es unter anderem in [Wit00] beschrieben wird.

Das problemzentrierte Interview ist eine abgewandelte Form des narrativen Interviews. Letzteres hat das Ziel, den Befragten über ein bestimmtes Themengebiet erzählen zu lassen. Dabei lebt das Interview von der Offenheit der Befragung, das heißt der Befragte soll möglichst von sich aus berichten und der Interviewer muss vor allem darauf acht geben, dass die Erzählung nicht ins Stocken gerät und dass sie nicht vom Thema abschweift. Ansonsten bleibt er passiv. Diese Interviewform wird auch als „weiches Interview“ [Lam05, S.343] bezeichnet

Beim problemzentrierten Interview werden die Befragten als Experten ihrer Handlungen begriffen [Wit00]. Ein Instrument dieser Befragungstechnik ist der Leitfaden. Er enthält die in Erfahrung zu bringenden Themen stichpunktartig oder als bereits vorformulierten Fragen. Dem Interviewer dient er als Gedächtnisstütze und Orientierungshilfe für das Interview.

Ein zentrales Prinzip der qualitativen Sozialforschung ist die Offenheit [Lam05, S. 21]. Deshalb sollen Fragen hier offen gestellt werden, das heißt den Befragten sollen keine Antwortmöglichkeiten vorgegeben werden. Der Vorteil von qualitativen Interviews gegenüber quantitativen Forschungsmethoden liegt vor allem in der Flexibilität. Dem Interviewer ist es möglich, auf das Gesagte des Befragten einzugehen und daraufhin weitere Fragen zu stellen. Manchmal werden auch wichtige Details von einem Befragten als so selbstverständlich betrachtet, dass er sie gar nicht erst erwähnt. Dem kann ein Interviewer, sofern er es bemerkt, mit suggestiven Zusatzfragen entgegenwirken.

Allerdings ist ein qualitatives Interview in der Regel auch zeitaufwändiger in der Durchführung als ein standardisiertes Interview, wie z.B. ein Fragebogen. Da die Anzahl der von mir zu befragenden Personen jedoch relativ gering war, wirkte sich dieser Nachteil nicht auf die Wahl meiner Methodik aus.

4.5 Durchführung

Um Informationen über die Grundidee, die Motivation und den groben Ablauf des Prozesses zu ermitteln, führte ich Interviews mit vier leitenden Mitarbeitern der untersuchten Projekte, wobei zwei den Spezifizierungsprozess leiteten und die anderen beiden an der Leitung des Testprozesses beteiligt waren. Mit der gleichen Intention befragte ich in mehreren Sitzungen Herrn Wegner. Er ist Miterfinder des Prozesses und Leiter der IAV-Abteilung, die für die in den untersuchten Projekten vorgenommenen HMI-Spezifizierungen zuständig ist.

In einer intensiveren Studie konzentrierte ich mich auf den Spezifizierungsprozess. Dabei führte ich Interviews mit sieben Mitarbeitern, die an den laufenden Projekten beteiligt waren. Die Länge dieser Interviews lag zwischen 15 und 32 Minuten und betrug im Mittel 24 Minuten.

Die Situation eines Interviews sollte möglichst vertraut und auch vertraulich sein [Lam05]. Deshalb wurden die Befragungen fast alle in einem Besprechungsraum der IAV vorgenommen, in dem sich außer mir und dem Befragten keine weiteren Personen befanden. In einzelnen Fällen wurden die Interviews auch am Arbeitsplatz des Befragten vorgenommen. Bis auf in einem Fall geschah dies jedoch nur, wenn sich zum Zeitpunkt der Befragung keine weiteren Personen mit im Raum befanden. Zu einer lockeren Gesprächsatmosphäre trug neben den von mir zumeist angebotenen Gummibärchen auch das - abteilungsweit unter allen Mitarbeitern aller Hierarchien gepflegte - Duzen bei.

Als Orientierungshilfe für die Interviews zur genaueren Analyse des Spezifizierungsprozesses entwickelte ich einen Interview-Leitfaden, den ich nach einem Pilotinterview noch optimierte. Auch für die Interviews mit den leitenden Mitarbeitern erstellte ich Leitfäden. Die Leitfäden sind im Anhang (s. Abschnitt 12.2) einzusehen.

Zu Beginn des Interviews stellte ich den Beteiligten kurz mein Forschungsvorhaben vor und sicherte ihnen eine Anonymisierung der von mir erhobenen Informationen zu. Die Anonymität bezog sich dabei sowohl auf persönliche als auch auf projektspezifische Daten, da die Beteiligten für letztere einer Geheimhaltungspflicht unterlagen. Bei Herrn Wegner entfiel die Anonymitätszusicherung bezüglich persönlicher Daten.

Die Interviews nahm ich, soweit mir der Interviewpartner dies gewährte, mit einem Diktiergerät auf. Bis auf in einem Fall war dies möglich. Dadurch musste ich mich wäh-

rend der Befragungen nicht auf das Festhalten der Daten konzentrieren, sondern konnte darauf achten, ob die Antworten die von mir gewünschten Informationen enthielten. Falls dem nicht der Fall war, hakte ich mit Zusatzfragen nach.

Um bei der Datenanalyse für den Spezifizierungsprozess besser abstrahieren zu können, habe ich jeweils Interviews mit mehreren Mitarbeitern, die ungefähr die gleiche Rolle im Prozess einnahmen, durchgeführt. Die Mitarbeiter waren alle bis auf einen sowohl am Projekt A als auch am Projekt B beteiligt, was es mir ermöglichte, die Interviewpartner nicht nur zu einem Projekt zu befragen.

4.6 Datenanalyse

Die Transkriptionen der Befragungen habe ich nur stichpunktartig vorgenommen, da eine vollständige Transkription für meine Zwecke keinen weiteren Informationsgewinn erzielt hätte. Aus demselben Grund verzichtete ich auch auf die Protokollierung nonverbaler Aspekte während der Interviews.

In der zweiten Phase der Analyse sortierte ich die Aussagen nach den in 4.3 aufgelisteten Bereichen. Das war erforderlich, da der Ablauf der Interviews teilweise stark von der im Leitfaden vorgegebenen Struktur abwich. Auch wurden mitunter durch den Gesprächscharakter der Interviews mehrere Bereiche in den Antworten vermischt, die ich während der Analyse wieder trennen musste. Um detailliertere Informationen über die einzelnen, während des Prozesses erzeugten, Dokumente und deren Formate zu erhalten, analysierte ich zusätzlich noch Exemplare dieser Dokumente. Sich widersprechende Aussagen habe ich durch Nachfragen bei mehreren IAV-Mitarbeitern validiert.

Aufgrund des Umfangs habe ich den Ergebnissen dieser Studie zwei gesonderte Kapitel gewidmet. Der Aufbau des Gesamtprozesses ist in Kapitel 5 erläutert, während die Lupenbetrachtung des Spezifizierungsprozesses in Kapitel 6 folgt.

4.7 Diagrammerstellung

Für die einzelnen Teilprozesse und deren Anordnung erstellte ich jeweils ein Diagramm mit dem Werkzeug UMLStudio. Die in den Diagrammen enthaltenen grafischen Elemente sind, wie in Abbildung 13 gezeigt, zu interpretieren.

Abbildung 13: Legende für Prozessdiagramme

Eine Unterscheidung zwischen in Teilprozessen erzeugten und verwendeten Produkten ergibt sich jeweils aus den in den Diagrammen eingezeichneten Pfeilen zwischen einem Teilprozess und einem Produkt. Ein Pfeil, der von einem Teilprozess ausgehend auf ein Produkt zeigt, beschreibt die Erzeugung dieses Produkts, während ein von einem Produkt auf einen Teilprozess zeigender Pfeil die Verwendung darstellt.

Pfeile zwischen zwei Teilprozessen beschreiben deren Abfolge. Ein gestrichelter Pfeil deutet dabei an, dass der Teilprozess, auf den der Pfeil gerichtet ist, nicht zwingend in der Abfolge erforderlich ist.

5 Der Entwicklungsprozess TeleDrive VISION

Dieses Kapitel beschreibt zunächst die Motivation für den TeleDrive VISION Prozess und dessen Grundidee. Danach erläutere ich den grobe Aufbau des momentanen Zustands des Prozesses. Den Sollzustand des Prozesses stelle ich anhand einer kurzen Vorstellung des von der IAV entwickelten Werkzeugs HMIStudio vor. Abschließend nehme ich eine Einordnung in bekannte Prozessmodelle vor.

Die Informationen stammen, soweit nicht anders gekennzeichnet, aus Interviews mit Herrn Wegner und mit leitenden Mitarbeitern der beiden untersuchten Projekte (s. Kapitel 1.3). Für die Abschnitte 5.3.7 und 5.3.8 führte ich Interviews mit leitenden Mitarbeitern der IAV-Abteilung „Systemintegration“. Die Interviews habe ich nach der im 4. Kapitel beschriebenen Methodik durchgeführt und ausgewertet.

5.1 Motivation

Immer mehr Fahrzeughersteller gehen dazu über, die von ihnen produzierten Fahrzeuge mit einem ins Cockpit integrierten Infotainmentsystem als Teil der Grund- oder Zusatzausstattung zu vermarkten. Da eine Entwicklung dieser Geräte jedoch nicht zu ihren Kernkompetenzen gehört, beauftragen sie hierfür zumeist eine Automobilzuliefererfirma (im Folgenden kurz Zulieferer genannt), so wie sie es auch bei anderen für die Fertigung der Fahrzeuge notwendigen Teilen, wie Scheinwerfern oder Reifen, handhaben.

Neben den von dem Infotainmentssystem zur Verfügung gestellten Funktionalitäten wollen die Fahrzeughersteller auch das Aussehen und Verhalten des HMI der Systeme nach ihren Wünschen spezifizieren können, da dies ein entscheidendes Kriterium für die Akzeptanz beim Käufer darstellt. Wie Herr Wegner aus seiner langjährigen Erfahrung berichten konnte, geht dabei der Trend immer mehr dahin, dass die Fahrzeughersteller die Spezifizierung des HMI der in ihren Fahrzeugen eingebauten Infotainmentsysteme bis ins letzte Detail vornehmen wollen. Dieser Wunsch ist auch berechtigt, da der Fahrzeughersteller seine Kunden und deren Wünsche im Zweifel besser kennt als ein Zulieferer. Schon im Jahr 1971 erklärte Hansen es als das erste Prinzip für das Design von interaktiven Systemen, den Benutzer zu kennen [Shn80, S. 249]. Nach Shneiderman sollte dies das Motto eines jeden Benutzerschnittstellen-Designers sein.

Dadurch, dass die Fahrzeughersteller das HMI ihrer Infotainmentsysteme selber spezifizieren, aber von Zulieferern entwickeln lassen wollen, ist ein spezielles Übergabeformat für die Spezifikationsdokumente erforderlich. In vielen Fällen wird dieses Problem heute noch dadurch gelöst, dass der Fahrzeughersteller seine Spezifikation in Papierformat (z.B. (ausgedruckte) PDF-, Word- oder Excel-Dokumente) an den Zulieferer weitergibt. So werden z.B. bei BMW die Anforderungen an ein Gerät und dessen HMI mit einem Werkzeug für das Anforderungsmanagement (namentlich Doors) verwaltet und später in Form eines per Export erzeugten Word-Dokuments an die Geräteentwickler gegeben. Diese Form der Spezifizierung bringt ein Problem mit sich: Die bereits maschinell festgehaltenen Anforderungen können nicht in den Entwicklungsprozess mit einfließen und müssen erneut, wenn auch in einer anderen Form, codiert werden.

Dieses Problem zieht sich mitunter durch den gesamten Entwicklungsprozess. Wie im Abschnitt 2.2 beschrieben, sind an der Entwicklung einer Mensch-Maschine-Schnittstelle oftmals verschiedenste Rollen beteiligt, unter anderem Ergonomen, Designer, Soft- und Hardwareentwickler. In den bisherigen Prozessansätzen verwenden die Rollen zumeist jeweils unterschiedliche Werkzeuge bei ihrer Arbeit. Damit geht einher, dass ihre Arbeitsergebnisse in unterschiedlichen Datenformaten erzeugt werden, die teilweise nur schwierig ineinander überführbar sind [TDV06]. Aufgrund dieser Medienbrüche wird eine Weiterverwendung der Prozesszwischenprodukte für nachgelager-

te Teilprozesse erschwert und mitunter nur durch manuelles Übertragen möglich. Daraus resultieren Mehraufwände durch doppelte Arbeiten. Außerdem kann es dadurch passieren, dass bei den Überführungen ungewollt Informationen verloren gehen oder andere Fehler auftreten. Als Folge kann die Qualität des zu entwickelnden Produkts darunter leiden.

Die für die Fahrzeughersteller gebauten Infotainmentsysteme werden zumeist in verschiedenen Varianten für verschiedene Fahrzeuge, Fahrzeugklassen und Märkte (z.B. Europa, China etc.) benötigt. Diese Varianten sind nur selten komplett unterschiedlich. So können sich Varianten für verschiedene Fahrzeugklassen oder Ausstattungsniveaus z.B. dadurch unterscheiden, dass es sich bei der einer Variante um ein Komplettsystem handelt, während bei einer anderen kein Navigationssystem verbaut ist. Bei den Varianten für die verschiedenen Märkte unterscheiden sich die Systeme häufig nicht nur in der Zusammensetzung aus Teilsystemen, sondern auch anhand der Bedienkonzepte. So wird z.B. für ein Navigationssystem für den chinesischen Markt ein anderes Bedienkonzept benötigt als für den nordamerikanischen Markt, was schon durch die unterschiedlichen Zeichensätze und Lese- / Schreibrichtungen gegeben ist. Das grafische Design der Benutzungsschnittstellen soll jedoch zumeist für eine Marke weltweit gleich bleiben.

Um eine bessere Verhandlungsgrundlage gegenüber den Zulieferern zu erhalten und um nicht von einem Zulieferer alleine abhängig zu sein, wollen die Fahrzeughersteller sich die Möglichkeit offen lassen, für jede der Varianten theoretisch einen anderen Zulieferer auswählen zu können. Da aber die einzelnen Varianten zumeist eine große Schnittmenge bei Funktionalität und Aussehen des HMI aufweisen, und somit eine Wiederverwendung von Spezifikationsdokumenten dringend erforderlich ist, bedarf es hier eines industrieweit standardisierten, maschinenlesbaren und plattformunabhängigen Austauschformats. Erst durch ein solches wäre es realisierbar, Teile der HMI-Spezifikation in die Software-Entwicklungsprozesse verschiedener Zulieferer einfließen lassen zu können, ohne diese immer wieder neu zu spezifizieren.

5.2 Grundidee

Die Grundidee der IAV war und ist es, mit dem TeleDrive VISION Prozess einen von der Idee des Fahrzeugherstellers bis zum Serienprodukt durchgängigen Prozess für die Entwicklung von Infotainmentsystemen zu konzipieren. Um dem erläuterten Problem der Medienbrüche gerecht zu werden, hat die Gesellschaft unter Federführung von Herrn Wegner beginnend im Jahr 2004 die Sprache IML entwickelt. Sie ermöglicht es einem Fahrzeughersteller, eine detaillierte maschinenlesbare Spezifikation für das HMI eines Infotainmentsystems zu erstellen, die ein Zulieferer für den Bau des Systems weiterverwenden kann.

Eine vereinfachte Darstellung des Zusammenspiels von Fahrzeughersteller - in der Branche oft auch als OEM (Initialwort für *Original Equipment Manufacturer*) bezeichnet - und Zulieferer in diesem Prozess ist in Abbildung 14 dargestellt. Der Fahrzeughersteller spezifiziert das HMI mittels IML. Die dabei entstehende IML-Datenbasis wird vom Zulieferer z.B. durch automatische Codegenerierung für die Geräteentwicklung wiederverwendet. Nach „Fertigstellung“ des Produkts wird es vom Fahrzeughersteller und dem Zulieferer unter Zuhilfenahme von aus der IML-Datenbasis generierten Testfällen geprüft. [1]

Die Spezifizierung des HMI mittels IML erlaubt es dem Fahrzeughersteller, schon während der Entwicklung einen Eindruck von dem späteren Produkt zu bekommen, indem aus der IML-Spezifikation ein Prototyp, z.B. in Form einer Simulation, erzeugt

[1] In den untersuchten Projekten wurde die IAV für die Spezifikation und die Gerätetests von einem Fahrzeughersteller beauftragt.

wird. Ein weiterer Vorteil ergibt sich daraus, dass die IML-Spezifikation direkt für die Gerätesoftwareentwicklung weiterverwendet wird: Der Fahrzeughersteller kann noch zu späten Zeitpunkten in einem Projekt Änderungen am Produkt vornehmen. Dies ist des Öfteren erforderlich, da dem Hersteller seine Wünsche oft erst während des Projekts, z.B. nach Begutachtung eines Prototypen, bewusst werden.

Kurz zusammengefasst sind die übergeordneten Ziele des Prozesses, eine Kostenreduktion und Produktqualitätsverbesserung bei der Entwicklung von Mensch-Maschine-Schnittstellen für Infotainmentsysteme zu erreichen, also letztlich eine Effizienzsteigerung.

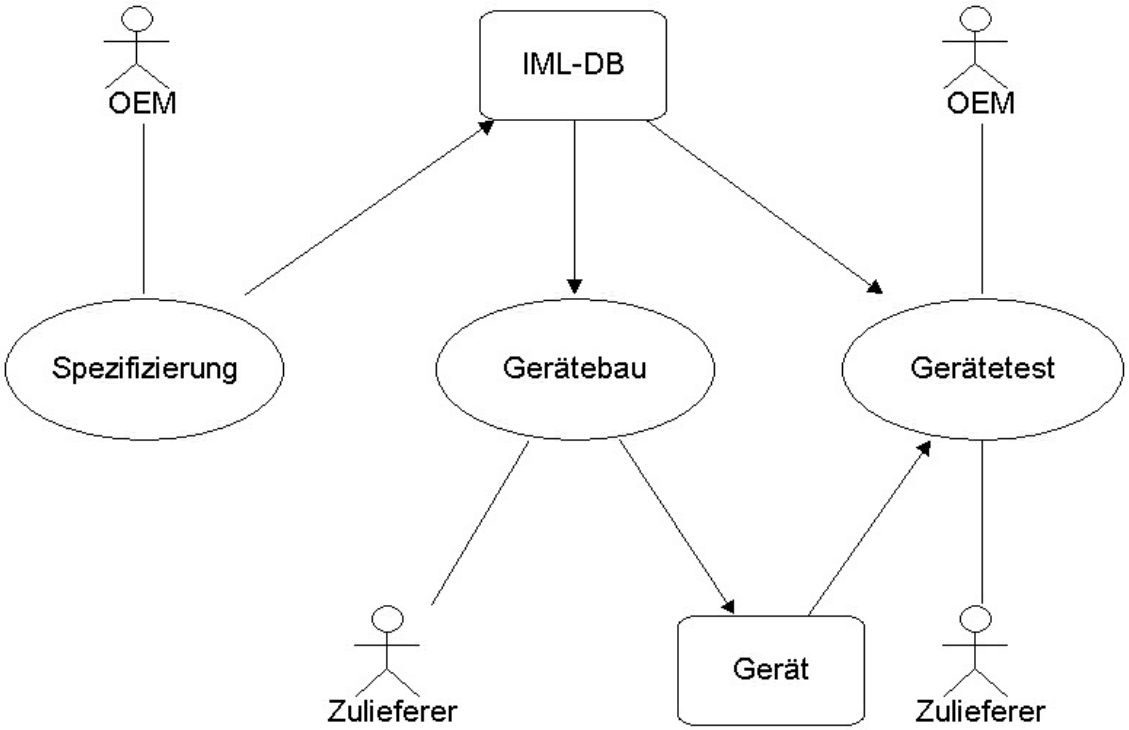

Abbildung 14: Vereinfachte Darstellung des TeleDrive VISION Prozesses

Es sei an dieser Stelle angemerkt, dass der Prozess nicht fixiert ist auf die hier beschriebene Rollenverteilung mit einem Fahrzeughersteller als Auftraggeber und einem Zulieferer als Auftragnehmer. Die Rollen Auftragnehmer und Auftraggeber können auch durch andere Parteien wahrgenommen werden. So kann ein Fahrzeughersteller den gesamten Prozess auch alleine ohne Zulieferer leben. Auch ist es denkbar, dass ein Zulieferer den Prozess alleine durchführt, um z.B. ein Gerät für den Ersatzteilmarkt (engl. *aftermarket*) zu entwickeln. Auftraggeber wäre in diesen Fällen das Produktmanagement der betreffenden Firma und Auftragnehmer eine Entwicklungsabteilung. Im weiteren Verlauf dieser Arbeit bleibe ich jedoch bei der häufiger auftretenden Konstellation mit einem Fahrzeughersteller als Auftraggeber und einem Zulieferer als Auftragnehmer.

5.3 Teilprozesse

Der Entwicklungsprozess TeleDrive VISION lässt sich grob in die in Abbildung 15 dargestellten Teilprozesse untergliedern. Die Abbildung stellt lediglich die Abfolge und die Abhängigkeiten der Teilprozesse untereinander dar. Außer der Abfolge sollen zeitliche Aspekte darin nicht enthalten sein, was durch den stark iterativen Charakter des Prozesses auch nur schwierig machbar wäre. Die in dem Graphen dargestellten Wege

werden nicht immer vollständig abgeschritten. Teilweise werden einzelne Teilprozesse übersprungen. So fließt z.B. das erstellte Pflichtenheft auch direkt in den Abnahmeprozess mit ein. Nachfolgend werden die Teilschritte, teilweise zusammengefasst, beschrieben.

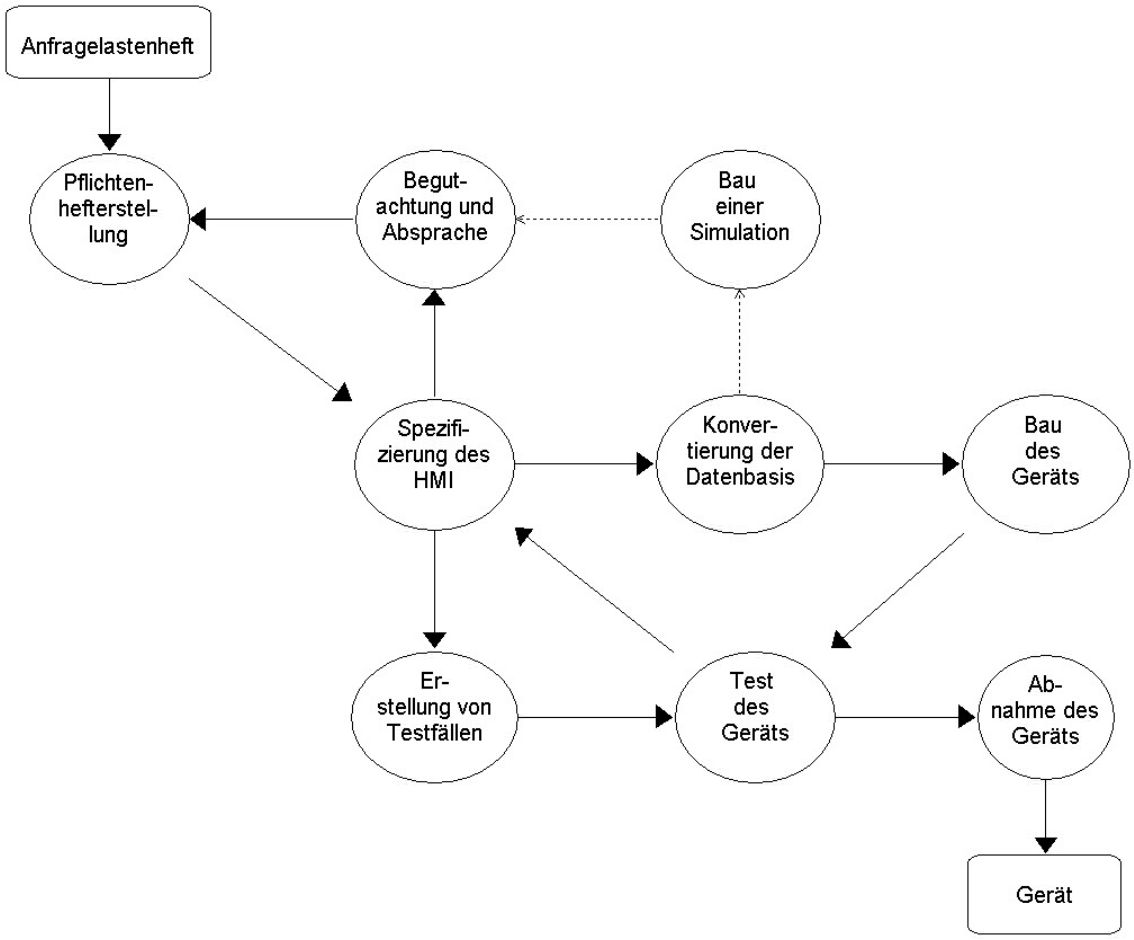

Abbildung 15: Prozessdiagramm des TeleDrive VISION Prozesses

5.3.1 Das Anfragelastenheft

Ausgangspunkt bei der Entwicklung eines neuen Infotainmentsystems ist ein Anfragelastenheft des Fahrzeugherstellers. Ein Lastenheft ist ein Dokument, das die vom Auftraggeber gewünschten fachlichen Basisanforderungen an das zu entwickelnde Produkt enthält [Bal96, S. 57 ff]. Im TeleDrive VISION Prozess beschreibt es die Anforderungen, die das von einem Zulieferer zu entwickelnde Infotainment-System aus der Sicht des Fahrzeugherstellers erfüllen muss.

Die Anforderungen sind zum einen funktionaler Art. Im Anfragelastenheft werden die Produktmerkmale (neudeutsch: Features), die das Infotainmentsystem anbieten soll, aufgelistet und auf einem abstrakten Niveau beschrieben. Dafür wird unter anderem dokumentiert, aus welchen Gerätekomponenten, wie z.B. einer Radio- oder einer Telefonkomponente, das Infotainmentsystem bestehen soll. Für jede der Komponenten wird angegeben, welche Funktionalitäten sie unterstützen soll.

Neben den funktionalen sind in dem Anfragelastenheft auch die nicht-funktionalen Anforderungen an das Infotainmentsystem vermerkt. Es werden unter anderem die technischen Spezifika beschrieben, die für das System gelten müssen. Diese Spezifika können z.B. beinhalten, wie das Unterspannungsverhalten des Systems sein muss oder

in welchem Temperaturbereich es arbeiten können muss. Es wird darin auch beschrieben, welche Schnittstellen das Gerät anbieten muss, damit es in das Fahrzeugnetz integriert werden kann. Das ist meistens erforderlich, da das System mit anderen im Fahrzeug verbauten Geräten kommunizieren können muss, wie z.B. der Klimaanlage oder dem Kombiinstrument[1].

Ein weiterer nicht-funktionaler Bestandteil des Anfragelastenhefts im TeleDrive VISION Prozess ist die Forderung, dass ein Zulieferer eine ihm in Form von XML-Dateien übergebene Spezifikation des HMI automatisiert in seinen Softwareentwicklungsprozess einfließen lassen muss. Das heißt ein Zulieferer muss einen Übersetzer schreiben, der dieses elektronische Format in ein für seinen Entwicklungsprozess passendes Format konvertiert, ohne dass dabei Informationen verloren gehen oder Defekte auftreten.

Mit diesem Bestandteil des Lastenhefts wird dem Problem der sich ständig ändernden Anforderungen in Softwareentwicklungsprojekten begegnet. Laut einer Studie der *Standish Group* aus dem Jahr 2001, bei der 500 größere Softwareentwicklungsprojekte untersucht wurden, entstehen durchschnittlich 43 Prozent der Softwarewartungskosten durch Änderungen an Anforderungen [Ver04]. Änderungen am HMI sind im TeleDrive VISION Prozess durch die Verwendung der Sprache IML nach Aussage von Herrn Wegner sehr kostengünstig geworden, da hierbei nicht mehr der Quellcode der Software an sich modifiziert werden muss. Dadurch, dass die Änderungen direkt vom Auftraggeber durchgeführt werden, hat er die dabei entstehenden Kosten selber unter Kontrolle. Er begibt sich jedoch auch in die Pflicht, die volle Verantwortung für das HMI zu übernehmen.

5.3.2 Erste Pflichtenhefterstellung

Nach der Planungsphase übergibt der Fahrzeughersteller sein Anfragelastenheft an potentielle Zulieferer, die daraufhin ein Angebot in Form eines Pflichtenhefts anfertigen. Ein Pflichtenheft ist ein vom Auftragnehmer, also dem Zulieferer, verfasstes Dokument, das die vertragliche Beschreibung des Lieferumfangs darstellt [Bal96, S. 105]. Es enthält eine Liste der im Anfragelastenheft aufgeführten Produktmerkmale, die ein vom Zulieferer entwickeltes Gerät im Falle eines Vertragsabschlusses unterstützen würde. Der Zulieferer verpflichtet sich zu der Umsetzung dieser Produktmerkmale. Er beschreibt, mit welcher Pflicht welche Last eingehalten werden würde.

Im Bereich der Entwicklung von Infotainmentsystemen können nicht immer alle Ideen des Fahrzeugherstellers vom Zulieferer in vollem Umfang technisch umgesetzt werden, zumal letzterer aus Kostengründen meistens auf bereits bestehende Plattformen zurückgreift. Deshalb wird für die Produktmerkmale jeweils mit angegeben, ob deren Umsetzung mit Restriktionen behaftet ist und, wenn das der Fall ist, wie diese im einzelnen aussehen.

Über die Restriktionen wird danach zwischen dem Fahrzeughersteller und dem potentiellen Zulieferer in einem feineren Abstimmungsprozess verhandelt, mit dem Ergebnis, dass entweder das Anfragelastenheft oder aber das Pflichtenheft diesbezüglich angepasst wird. Nach erfolgreichen Verhandlungen über Preise, Termine und anderen Vertragskonditionen bestimmt der Fahrzeughersteller den Zulieferer seiner Wahl.

[1] Instrumentenblock in Kraftfahrzeugen, der aus Komponenten, wie Tachometer, Drehzahlmesser, Tankanzeige etc. besteht

5.3.3 Spezifizierung des HMI

Auf Basis der durch das Pflichtenheft zugesicherten Anforderungen an das Infotainmentsystem spezifiziert der Fahrzeughersteller oder eine von ihm beauftragte Drittfirma, was im Fall der in dieser Arbeit untersuchten Projekte die IAV war, das HMI des Systems mit Hilfe der Sprache IML. In den untersuchten Projekten wurden als Spezifizierungszwischenschritt zuerst Zustandsdiagramme (engl. *statecharts*) erstellt, auf deren Basis dann die IML-Umsetzung geschah.

Ziel des Spezifizierungsprozesses ist es, eine weiterverwendbare Spezifikation des HMI für das zu entwickelnde Infotainmentsystem zu erstellen. Sie muss die Umsetzung der graphischen Bedienoberfläche für alle im Pflichtenheft beschriebenen Produktmerkmale enthalten. Sowohl die Menüstruktur als auch die darin vorkommenden Interaktionselemente müssen einheitlich und ganzheitlich beschrieben sein, so dass der Zulieferer die Spezifikation automatisch in seinen Softwareentwicklungsprozess einfließen lassen kann.

Die bei der Spezifizierung entstehende IML-Datenbasis hat drei wesentliche Einsatzzwecke:

1. Erstens dient sie als Teil des Lastenhefts, anhand dessen der Zulieferer das Infotainmentsystem bauen muss, und damit als Vertragsgrundlage. Da dieses Lastenheft in einem maschinenlesbaren Austauschformat verfasst ist, verwendet die IAV hierfür den Begriff des elektronischen Lastenhefts. Anpassungen am elektronischen Lastenheft werden durch die Verpflichtung des Zulieferers, dass er die Datenbasis in seinen Entwicklungsprozess mit einfließen lässt, praktisch direkt zu Anpassungen am Pflichtenheft. Dadurch bildet das Lastenheft gleichzeitig auch einen Teil des Pflichtenheftes. Der Prozess ist damit darauf ausgelegt, dass Änderungen und Erweiterungen am Pflichtenheft ohne weiteres während eines Projektes auch noch zu späten Zeitpunkten vorgenommen werden können.

   Damit der Zulieferer die IML-Datenbasis in seine Softwareentwicklung mit einfließen lassen kann, muss ein für seine Plattform passendes Format erzeugt werden. Dieses Format muss nicht zwangsläufig das IML-Format sein, weshalb eine Konvertierung der IML-Datenbasis in ein Zuliefererformat notwendig sein kann. Auf Basis der dabei entstehenden Datenbasis generiert der Zulieferer HMI-spezifische Softwarekomponenten, z.B. in Form von Java-Quelltextdateien für die Gerätesoftware. Eine Vision von TeleDrive ist es dabei, dass unter Bereitstellung eines Rahmenwerkes (engl. *framework*) und spezieller Gerätekomponenten, die z.B. Radio-Funktionalitäten anbieten, die komplette Gerätesoftware per Tastendruck erzeugt werden kann. Bisher ist dies nach Aussage von Herrn Wegner nur Theorie, da an verschiedenen Stellen immer noch manuelle Nacharbeit notwendig ist. Mit zunehmender Reife des Prozesses werde der Anteil daran aber nach und nach abnehmen.

2. Ein zweites Einsatzgebiet der IML-Datenbasis ist der Bau von Prototypen. Hierzu muss die Datenbasis zumeist ähnlich, wie bei der Generierung der Gerätesoftware in ein passendes Format konvertiert werden.

3. Das dritte Einsatzgebiet der IML-Datenbasis ist die automatische Erzeugung von Testfällen (engl. *test case*) und Anwendungsszenarien (engl. *use case*), auf deren Basis das vom Zulieferer entwickelte Gerät später getestet werden kann.

Außerdem lassen sich aus der IML-Datenbasis auch konventionelle Dokumentationen erzeugen, z.B. in Form von HTML- oder PDF-Dokumenten. Eine detaillierte Betrachtung des Spezifizierungsprozesses folgt im nächsten Kapitel.

5.3.4 Prototypenbau

Ein Prototyp ist "ein Vorab-Exemplar einer späteren Serienfertigung, das zur Erprobung von Eigenschaften [des zu entwickelnden Produkts oder dessen Umwelt] dient" [Wik05d]. In den von mir untersuchten Projekten wurden zwei verschiedene Arten von Prototypen gebaut und eingesetzt. Zum einen wurde ein virtueller Prototyp in Form einer auf einem PC ausführbaren Simulation entwickelt und zum anderen fertigten die Zulieferer zu vorab festgelegten Meilensteinen reale Prototypen an.

In dem hier betrachteten Kontext ist ein realer Prototyp im Gegensatz zu einem virtuellen ein Exemplar eines echten Gerätes, das in einem Fahrzeug eingebaut werden könnte. Er wird auch als Gerätemuster (kurz Muster) bezeichnet. Ein Muster kann vor allem dazu dienlich sein, die Umsetzung der Anforderungen, die das Einsatzgebiet des Systems, also das Fahrzeug, mit sich bringt, zu verifizieren. So kann mit einem Muster z.B. überprüft werden, ob das Gerät auch noch bei minus 20 Grad Celsius voll funktionsfähig ist. Mit ihm kann auch sichergestellt werden, dass die entwickelte Gerätesoftware auch auf der Zielhardware stabil und leistungsstark läuft. Dadurch, dass Muster aus Hardware bestehen und es sich bei deren Bau um Einzelanfertigungen handelt, sind sie im Vergleich zu den virtuellen Prototypen relativ teuer. Deshalb werden sie im Verlauf eines Projekts im Allgemeinen nur selten hergestellt. Des Öfteren wird daher, z.B. nach softwaretechnischen Defektbehebungen, lediglich die Software eines Musters aktualisiert.

Nach Aussage von Herrn Wegner ist der Bau eines virtuellen Prototypen, also einer Gerätesimulation, im TeleDrive VISION Prozess nicht zwingend erforderlich, weshalb auch in Abbildung 15 der Übergang zu diesem Prozessteilschritt als optional dargestellt ist. Theoretisch ist es möglich, ein Projekt auch ohne Prototypenbau durchzuführen, jedoch hat sich in der Praxis herausgestellt, dass die Verwendung eines Prototypen an vielen Stellen im Prozess hilfreich sein kann. Laut den IML-Entwicklern (s. Kapitel 6.1.3) war die Simulation für die Durchführung der Projekte unabkömmlich, da sie ihnen die Möglichkeit bot, ihre Arbeitsergebnisse zu verifizieren und Defekte in der IML-Datenbasis somit frühzeitig aufzudecken. Außerdem war dieser Prototyp hilfreich, um die später beschriebenen Paardurchsichten (s. Kapitel 8.5) durchzuführen, und es konnte mit ihm die Umsetzbarkeit der technischen Spezifikation gegenüber dem Zulieferer demonstriert werden.

Die Simulation wurde in den untersuchten Projekten unter Einsatz der TeleDrive Simulationsplattform, die aus einem Applikationsrahmenwerk in Verbund mit projektspezifischen Simulationskomponenten besteht, begleitend zur Spezifizierung entwickelt.

Ein weiteres Einsatzgebiet von Prototypen ist die Durchführung von Akzeptanztests, sogenannter Kundenkliniken. Noch bevor das Gerät gebaut wird, können mit deren Hilfe Benutzbarkeitsstudien durchgeführt werden und es kann schon vorab die Akzeptanz des Infotainmentsystems beim Kunden evaluiert werden. Je nach Ziel der Studie können dabei reale oder virtuelle Prototypen eingesetzt werden. Für Studien, bei denen es darum geht, sicherzustellen, dass die GUI übersichtlich und selbsterklärend gestaltet ist und die Bedienlogik auch ihren Namen verdient hat, kann ein virtueller Prototyp ausreichend sein, während für eine Studie über die Benutzbarkeit des Geräts im Fahrzeug ein realer Prototyp notwendig ist.

5.3.5 Begutachtungen und Absprachen

Zu bestimmten Zeitpunkten im Projektverlauf, zu sogenannten Meilensteinen, wird ein aktueller Stand der IML-Datenbasis an den Zulieferer geschickt. Dieser generiert daraus den für seine Plattform spezifischen Code. Dabei kann es vorkommen, dass er auf Defekte oder Unverträglichkeiten mit seinem Format oder seiner Plattform in der Datenba-

sis stößt. Ein Protokoll über diese Defekte schickt er daraufhin an die mit der Spezifizierung befassten Ingenieure des Fahrzeugherstellers. Zu Beginn meiner Untersuchungen gab es über das Format dieses Protokolls keinerlei Vorgaben.

In den untersuchten Projekten fertigte die IAV die Spezifikation im Auftrag des Fahrzeugherstellers an. Deshalb wurden die Datenbasis und die zugehörigen Zustandsdiagramme zu den Meilensteinen zusätzlich noch an eine weitere Firma, den Fahrzeughersteller, versendet. Dieser überprüfte anhand der Statecharts und dem virtuellen Prototypen, ob die von ihm verlangten Anforderungen vollständig und nach seinen Vorstellungen umgesetzt worden sind.

Die Entwicklung eines HMI hat im Allgemeinen einen stark iterativen Charakter, was daher rührt, dass der Umfang anfangs zumeist noch nicht überschaubar ist und dass einem Auftraggeber viele gewünschte Spezifika erst auffallen, wenn er eine (Vorab-) Version des HMI, z.B. in Form eines Prototypen, sieht. In den Projekten kam es deshalb des Öfteren vor, dass die Projektverantwortlichen des Fahrzeugherstellers, nachdem sie eine Version des HMI sahen, feststellten, dass sie sich an der einen oder anderen Stelle noch Änderungen oder Erweiterungen an Funktionalitäten wünschten. Dies ist laut den Projektleitern auch für spätere Projekte zu erwarten und spiegelt das bekannte Problem bzw. Charakteristikum von Softwareentwicklungsprojekten wider, dass nicht alle Anforderungen an die zu entwickelnde Software bereits zu Beginn eines Projekts feststehen.

Nicht nur nach den Meilensteinen sondern auch schon während der IML-Umsetzung des HMI kam es in den bisherigen Projekten regelmäßig zu Absprachen unter den drei Parteien. Zum einen fielen den Entwicklern bei der Umsetzung mitunter Spezifika auf, die sie, entgegen den Wünschen des Fahrzeugherstellers, aufgrund von fehlender oder zu aufwändiger technischer Machbarkeit, nicht umsetzen konnten. Und zum anderen sprachen die Entwickler sich mit den Zulieferern über technische Details ab, damit letztere nicht erst bei Erhalt einer Lieferung feststellen mussten, dass ihre Plattform die Spezifikation nicht umsetzen kann.

Die Sprache IML ist noch relativ jung. Sie wurde bisher nur in den beiden betrachteten Projekten eingesetzt. Dadurch kam es in den Projekten vor, dass der Wortschatz der Sprache den IML-Entwicklern nicht ausreichte, um ein von ihnen zu lösendes Problem zu spezifizieren. In diesen Fällen wurden noch während des Projekts Erweiterungen an der Sprache vorgenommen, die dann immer auch mit den Zulieferern abgesprochen werden mussten, damit sie ihre Plattformen dahingehend anpassen konnten.

Die aufgrund der erläuterten Begutachtungen und Absprachen gewünschten Anpassungen werden in der IML-Datenbasis und ggf. in vorgelagerten Spezifikationen vorgenommen und fließen somit in eine neue Version des Pflichtenhefts mit ein. In den untersuchten Projekten bedeutete dies, dass die Veränderungen sowohl in den vorgelagerten Zustandsdiagrammen als auch in der IML-Datenbasis vorgenommen werden mussten, wodurch also immer ein doppelter Aufwand entstand.

5.3.6 Bau des Geräts

Auf Basis des Pflichtenhefts entwickelt der Zulieferer unter Einsatz der IML-Datenbasis das Infotainmentsystem. Der TeleDrive VISION Prozess lässt es dabei offen, wie dieser Entwicklungsprozess zu gestalten ist. Der Prozess schreibt auch nicht vor, wie die Überführung der IML-Datenbasis in eine lauffähige Software zu geschehen hat.

Hier bieten sich mehrere Möglichkeiten an, weshalb Herr Wegner diesen Konvertierungsschritt auch “IML to anything“ nennt. Zum einen können die IML-Dateien oder andere daraus erzeugte XML-Dateien zur Laufzeit von einer entsprechend ausgelegten Plattform eingelesen und bearbeitet werden. So wurde dies z.B. bei der in den Projekten

verwendeten Simulationssoftware, dem virtuellen Prototypen, gelöst. Zum anderen ist es möglich, aus den IML-Dateien Quellcodedateien zu erzeugen, die dann wiederum in eine dafür geeignete Plattform eingebunden werden. Auch Mischformen der beiden Methoden sind vorstellbar. So könnten z.B. Verhalten und Aussehen in erzeugten Code und einzulesende XML-Dateien getrennt werden. Weniger effizient aber auch denkbar ist eine manuelle Überführung der Spezifikation, das heißt ein erneutes Programmieren[1] des HMI auf Basis der IML-Spezifikation.

Vorabversionen eines Geräts werden im Allgemeinen in Form von Mustern, also realen Prototypen, an den Fahrzeughersteller geliefert.

Eine genauere Betrachtung des Gerätebaus war im Rahmen meiner Diplomarbeit nicht möglich: Einerseits aus Zeitgründen und andererseits werden entsprechende Informationen in den meisten Firmen gerne geheimgehalten. Es hätten auch nur exemplarische Betrachtungen durchgeführt werden können, da die Gerätehersteller ihre eigenen, oft stark unterschiedlichen Prozesse leben.

5.3.7 Testfallerstellung

Nebenläufig zur Geräteentwicklung, noch bevor ein erstes Muster vom Zulieferer produziert worden ist, beginnen die zuständigen Tester mit der Erstellung von Testfällen für das Gerät. Die Fälle basieren auf dem Pflichtenheft und der Spezifikation des Geräts. Dadurch, dass die Tests alle Gerätekomponenten und alle dafür definierten funktionalen und nicht-funktionalen Anforderungen abdecken müssen, werden pro Gerät mehrere tausend Testfälle geschrieben. Auch verschiedene Anwendungsszenarien werden mit berücksichtigt, wie z.B. ein eingehender Telefonanruf, während der Benutzer Radio hört und das Navigationssystem eine Routenführung anzeigt.

Unter den Testfällen befindet sich eine Reihe von allgemeinen, für eine Vielzahl von Infotainmentsystemen geltenden Fällen. Dies sind unter anderem Testfälle zur Überprüfung von funktionalen Anforderungen an das Gerät, wie z.B., dass der CD-Spieler eine CD mit Kopierschutz abspielt oder auch dass eine CD-ROM, auf der sowohl MP3-Dateien als auch andere Daten gespeichert sind, im Zufallsmodus des Audiowiedergabegeräts korrekt abgespielt wird. Derartige Testfälle können von Projekt zu Projekt wiederverwendet werden. Die meisten Testfälle müssen jedoch speziell für das zu testende Gerät neu geschrieben werden.

Der TeleDrive VISION Prozess erfordert noch weitere nicht-produktspezifische sondern prozessspezifische Testfälle. Es muss überprüft werden, dass der Konvertierungsschritt, der für die Übersetzung der IML-Datenbasis in die HMI-Software ausgeführt wird, korrekt funktioniert. Bei einer automatischen Konvertierung entfallen dafür jedoch eine große Anzahl von HMI-spezifischen Testfällen: Wenn sich die am Prozess beteiligten Parteien auf die Korrektheit des automatischen Konvertierungsschritts durch entsprechende Tests verlassen können, ist es nicht mehr notwendig, diese Softwarekomponenten zu testen. Hierfür muss jedoch auch die Korrektheit der IML-Datenbasis gewährleistet sein.

Die Erstellung von Testfällen für Infotainmentsysteme geschieht heutzutage, zumindest in den von der IAV durchgeführten Projekten für Gerätetests, noch größtenteils manuell. In Projekt A wurden die Testfälle unter anderem durch Analyse der Spezifikation (in Form von Zustandsdiagrammen) erstellt. Die maschinenlesbare Spezifikation in IML hat den Vorteil, dass sich daraus Testfälle auch automatisch generieren lassen. Zur Zeit wird in einem Projekt erstmalig ein von der IAV entwickeltes Werkzeug erprobt, das Testfälle für die Menülogik eines HMI automatisch aus einer IML-Datenbasis er-

[1] Z.B. an einem sogenannten Low-Cost-Standort. Aber das ist vielleicht ja bald nicht mehr notwendig.

zeugt und in die Datenbank eines Testmanagement-Werkzeugs (hier: TestDirector von Mercury) einspeist.

Neben Testfällen für die Menülogik können per automatischer Generierung auch Testszenarien für Gerätefunktionen erzeugt werden. Dadurch, dass in IML auch Funktionsaufrufe spezifiziert werden, ist es möglich, Szenarien für das Zusammenspiel der einzelnen Funktionen untereinander zu erstellen. Ein Beispiel-Szenario: Das zu testende Gerät befindet sich im Radio-Kontext, und der Benutzer schaltet auf den CD-Kontext um. Hierbei werden mehrere Gerätefunktionen, meist asynchron, ausgeführt, z.B. die Audioquelle umstellen, den HMI-Kontext wechseln, den CD-Spieler ggf. initialisieren, das Abspielen des aktuellen Titels starten, Informationen über diesen zur Anzeige bringen lassen usw. Die Funktionen können in einzelnen Funktionstests schon vorab getestet werden. Das Zusammenspiel der Funktionen kann jedoch auch Defektquellen, die durch derartige Tests nicht berücksichtigt werden, mit sich bringen, wie Schmutzeffekte durch Nebenläufigkeiten. Deshalb sollte auch dieses getestet werden.

5.3.8 Test des Geräts

Wenn eine Geräteversion in Form eines Muster fertiggestellt ist, wird sie zunächst vom Zulieferer getestet. In den untersuchten Projekten führte die Abteilung "Systemintegration" der IAV diese Tests im Auftrag des Zulieferers durch, wodurch es mir möglich war, an diesbezügliche Informationen durch Interviews zu gelangen. Es werden nachfolgend nur die für den TeleDrive VISION Prozess relevanten Softwaretests beschrieben und keine Hardwaretests, wie z.B. die Gerätequalifizierungstests.

Ein Muster wird zuerst einem Akzeptanztest unterzogen. Ziel dieses Tests ist es, sicherzustellen, dass ein nachfolgendes intensives Testen des Musters überhaupt möglich und sinnvoll ist. Dafür werden grundlegende Testfälle, die zuvor aus der Gesamtmenge der Testfälle ausgewählt werden, am Gerät im Testlabor „durchgespielt". So wird z.B. für ein Navigationssystem überprüft, ob es möglich ist, ein gewünschtes Fahrziel einzugeben. Wenn das Muster die grundlegenden Testfälle erfolglos durchläuft, können viele andere Testfälle gar nicht erst geprüft werden, weshalb dann das Muster nicht weiter getestet wird und auf eine ausgereiftere Version der Software gewartet wird.

Nach einem erfolgreich verlaufenen Akzeptanztest kommt es zur Mustererprobung. Dabei wird das Muster im Testlabor für alle definierten Testfälle überprüft. Die Testfälle sind in Checklisten zusammengefasst, nach denen die Tester ihre Überprüfungen vornehmen. Sie überprüfen dabei auch das Layout per Augenmaß gegen im Spezifizierungsprozess erstellte Grafik-Spezifikationen.

Die Tests geschehen zumeist noch manuell. In zur Zeit noch in Bearbeitung befindlichen Arbeiten wird jedoch auch an automatischen Testwerkzeugen geforscht, so z.B. in einer Diplomarbeit an der TU Berlin im Zusammenarbeit mit der IAV über das automatische Testen eines Infotainmentsystems mit Hilfe eines Roboters. Ein per Software gesteuerter Roboter soll dabei das Gerät bedienen und eine Kamera die grafische Ausgabe des Geräts aufzeichnen. Die somit gewonnenen Daten werden dann unter anderem mit der IML-Spezifikation abgeglichen. Dadurch kann z.B. überprüft werden, ob die vom Gerät angezeigten Texte in der richtigen Größe und Farbe dargestellt werden.

Neben der Mustererprobung wird auch noch ein Integrationstest für die Muster durchgeführt. Dabei wird das Gerät in ein Testfahrzeug und/oder in eine entsprechende Testumgebung eingebaut. Ziel dieser Tests ist es, Defekte aufzudecken, die aufgrund der Umgebung des Einsatzortes, also der Fahrzeugumgebung, auftreten können. Es werden Testfälle und Anwendungsszenarien durchgespielt, deren Realisierung im Labor nicht oder nur aufwändig machbar wäre. Dabei kann es sich z.B. darum handeln, zu überprüfen, ob das Gerät auch noch korrekt funktioniert, wenn während der Fahrt tele-

foniert wird und das Mobilfunknetz, z.B. in einem Tunnel, verloren geht. Ein weiteres Beispiel ist die Erprobung einer Radiokomponente in realen und unterschiedlichen Senderlandschaften.

Mit den Integrationstests wird auch die Korrektheit spezieller Fehler- oder Warnhinweise getestet, die nur während einer Fahrt vom Gerät angezeigt werden. So ist es einem Fahrer z.B. nicht gestattet, bei einer Fahrgeschwindigkeit von mehr als sechs Km/h fernzusehen, weshalb Infotainmentsysteme bei einer schnelleren Fahrt zumeist einen entsprechenden Hinweis anzeigen, wenn der Benutzer den Wechsel in ein TV-Menü veranlasst. Ein entsprechender Test ist in einem Labor nicht ohne weiteres durchführbar.

Für die Integrationstests setzt die IAV jedoch auch speziell dafür konstruierte Testlaborplätze ein. Mit deren Hilfe kann einem Gerät z.B. vorgetäuscht werden, dass es sich in einem fahrenden Automobil befindet, indem simulierte oder bei Testfahrten aufgezeichnete Daten - über die momentane Fahrgeschwindigkeit und die Lenkradstellung (ESP-Signale) - über den CAN-Bus an das Gerät gesendet werden. Dadurch kann eine teure Testfahrt entfallen.

Die gefundenen Defekte werden je nach Zuständigkeit an den Zulieferer oder an den für die Spezifikation zuständigen Fahrzeughersteller in Form von Protokollen weitergeleitet. Das heißt der in Abbildung 15 dargestellte Rückwärts-Zyklus vom Test zur Spezifizierung kann auch kürzer sein. Der Teilprozess Spezifizierung kann übersprungen werden, wenn die entdeckten Defekte direkt dem Zulieferer zuzuordnen sind, wenn also bei der Geräteentwicklung Fehler gemacht wurden.

Durch eine Defektbehebung in einer Systemkomponente können wieder Defekte durch Seiteneffekte auf andere Komponenten auftreten, deshalb kann danach ein erneutes vollständiges Testen notwendig sein. Daraus folgt, dass Defekte, die erst während des Testens aufgedeckt werden, enorme Kosten verursachen können.

5.3.9 Abnahme des Geräts

Die Muster und das endgültige Gerät werden zusätzlich zu den vom Zulieferer durchgeführten Tests noch einem Abnahmetest vom Fahrzeughersteller unterzogen. „Der Abnahmetest dient zum Nachweis, dass die zwischen Auftraggeber und Auftragnehmer vereinbarten Leistungen erbracht wurden“ [SoK06c]. Die dafür erforderlichen Abnahmekriterien werden, wie üblich [Bal96, S. 962 f.], dem auf dem Lastenheft basierenden Pflichtenheft entnommen.

Außerdem wird das Gerät dahingehend getestet, ob es sich auch noch in der Fahrzeugumgebung in Verbindung mit den anderen Fahrzeugkomponenten korrekt verhält. Dies ist bei den Integrationstests des Zulieferers nicht unbedingt im vollem Umfang möglich, da der Fahrzeughersteller nicht immer alle Fahrzeugkomponenten eines neuen Modells vorab der Öffentlichkeit und damit auch nicht dem Zulieferer präsentieren will.

Genauere Informationen über den Abnahmeprozess hätten den Rahmen dieser Arbeit überschritten. Die Tests laufen aber nach Aussage von Herrn Wegner ähnlich ab wie die oben unter 5.3.8 beschriebenen vom Zulieferer durchgeführten Gerätetests.

Wenn die letzte Version des Geräts erfolgreich abgenommen wurde, werden Exemplare dieses Geräts in die neu produzierten Fahrzeuge eingebaut und somit an die Kunden weiterverkauft.

5.4 Das HMIStudio – Der Sollzustand des Prozesses

Um einen weiteren Schritt in Richtung des unter 2.4 bereits erwähnten Ziels der Anwendungsentwicklung ohne Programmierer zu gehen, entwickelt die IAV das HMIStudio. Mit diesem Programm soll ein Werkzeug entstehen, „mit dem alle Mitwirkenden des kompletten Entwicklungsprozesses ohne besondere Programmierkenntnisse ein HMI [...] erstellen können.“ [TDV06]

Das Werkzeug ist auf das IML-Format ausgelegt. Es bietet einem Entwickler die Möglichkeit, IML-Widgets auch ohne XML-Kenntnisse zu erstellen. Zu Beginn meiner Untersuchungen war es eine (TeleDrive) Vision, in einem Fenster des Werkzeugs die spezifizierten Widgets den Entwicklern in einer grafischen Vorschau anzuzeigen. Es sollte also ein IML-WYSIWYG-Editor entstehen (WYSIWYG steht für engl. *What-You-See-Is-What-You-Get*). Nach Aussage von Herrn Wegner war diese Vision zum Entwicklungstand im Februar 2006 bereits ansatzweise Realität geworden (s. Abbildung 16).

Das Werkzeug wird (mittlerweile) in der IAV bereits in Projekten eingesetzt. In den untersuchten Projekten kam es noch nicht voll zum Einsatz. Es war zum Zeitpunkt des Starts von Projekt B noch nicht einsatzbereit und eine diesbezügliche Anpassung des Prozesses mitten im Projekt wäre aufgrund der strengen Zeitvorgaben nicht praktikabel und zu riskant gewesen. Das Projektteam verwendete dennoch zumindest in Teilbereichen der IML-Entwicklung das HMIStudio, unter anderem mitunter für die Verifizierung der IML-Datenbasis.

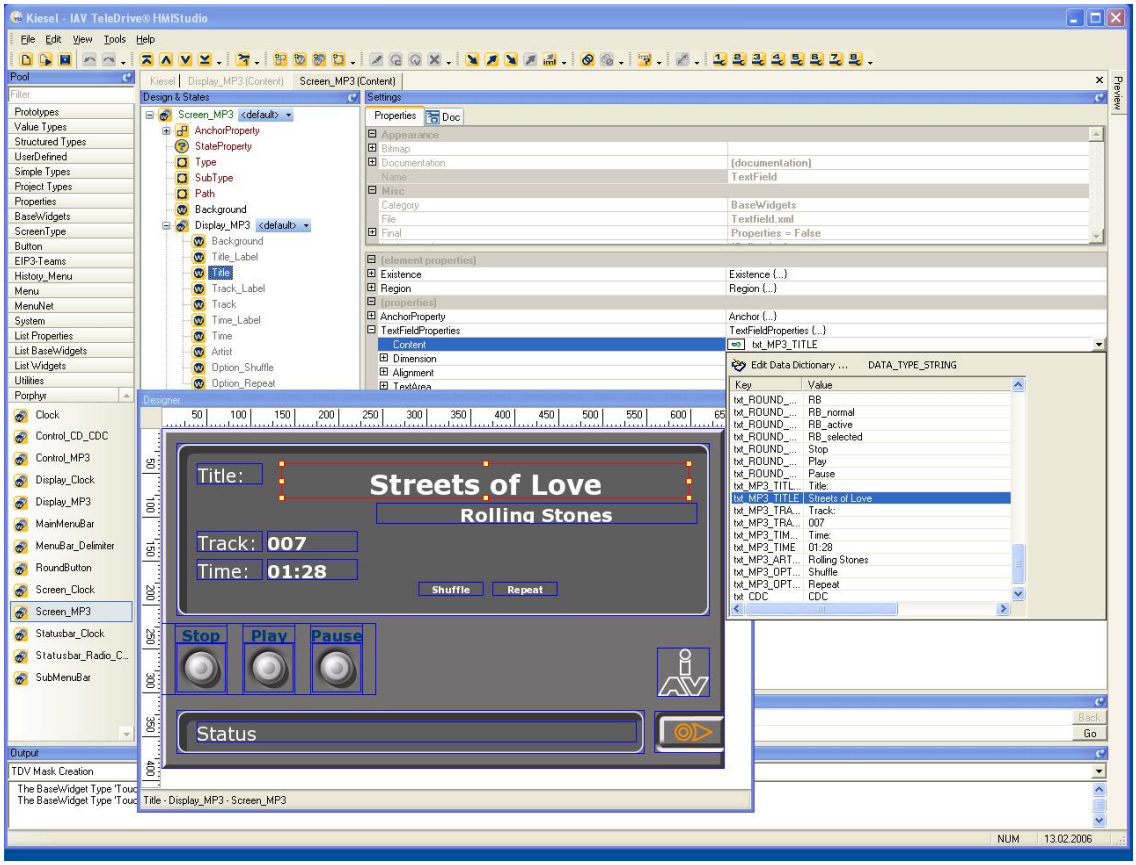

Abbildung 16: Bildschirmschnappschuss des HMIStudios (Februar 2006)

5.5 Einordnung in bekannte Prozessmodelle

In diesem Abschnitt wird eine Einordnung des in diesem Kapitel beschriebenen TeleDrive VISION Prozesses in Prozessmodelle, die aus dem Softwareingenieurwesen bekannt sind, vorgenommen. Er ließ sich nicht auf ein Modell festlegen.

5.5.1 Plangetriebenes, iteratives Modell

Die Entwicklung von Infotainmentsysteme unterliegt sehr strengen zeitlichen Vorgaben, da sie an den Fahrzeugentwicklungsprozess gekoppelt ist. Spätestens zum Zeitpunkt des Produktionsstarts (engl. *start of production* (SOP)) des betreffenden Fahrzeugs muss der oben beschriebene Entwicklungsprozess für die Infotainmentsysteme abgeschlossen sein. Die Verzögerung eines Entwicklungsprojekts hätte enorme finanzielle Auswirkungen, da sich dadurch im schlechtesten Fall auch die Markteinführung des Fahrzeugs verzögern könnte. Deshalb ist hier ein plangetriebenes Vorgehen zwingend. Ein agiler Ansatz, wie z.B. *eXtreme Programming* (abgekürzt XP), wäre hier nicht fruchtbar bzw. zu riskant.

In den untersuchten Projekten standen schon zu Beginn der Projekte die Zeitpunkte fest, zu denen bestimmte Musterstände erreicht sein mussten. Auch der SOP des Geräts stand zu Beginn fest. Da jedoch die Anforderungen anfangs nicht vollständig bekannt waren (und auch nicht sein sollten), konnte auch nicht exakt definiert werden, welche konkreten Anforderungen an die Muster bestehen. Diese wurden jeweils ein paar Wochen vor dem Mustertermin festgelegt. Es war meistens nur bekannt, welche Produktmerkmale mit welchem Umfang zu dem Termin umgesetzt sein mussten. Da die Termine in jeden Fall eingehalten werden sollten, einigten sich die drei Parteien (Fahrzeughersteller, IAV, Zulieferer) in kritischen Fällen darauf, die Fertigstellung bestimmter Produktmerkmale innerhalb der Spanne der Gesamtprojektlaufzeit zu verschieben. Es handelte sich bei den in den Projekten eingesetzten Prozess also um einen plangetriebenen.

Ein stark planendes Modell, ist aufgrund folgender Gegebenheiten in Projekten für die Entwicklung von Infotainmentsystemen notwendig:

1. Ein (relativ) wohldefiniertes Resultat musste in definierter Zeit erreicht werden [Pre06, F.29].
2. Große, verteilte Projektgruppen mussten koordiniert werden [Pre06, F. 29].
3. „Parallele Entwicklung von Hardware (z.B. Auto) und SW“ [Pre06, F. 30]
4. „Örtlich verteilte Entwicklung (auch: in mehreren Firmen)“ [Pre06, F. 30]

Diese Gegebenheiten trafen auch auf die untersuchten Projekte zu.

Wie schon des Öfteren weiter oben im- und explizit erwähnt, handelt es sich bei TeleDrive VISION um einen iterativen Prozess. Die Iterativität bezieht sich dabei nicht nur auf die oben beschriebene Meilensteinentwicklung. Der Austausch von Datenständen der HMI-Spezifikation zwischen den Parteien fand in den untersuchten Projekten auch regelmäßig (in etwa wöchentlich) zu zuvor nicht festgelegten Zeitpunkten statt. Wie oben beschrieben waren zum Start der Projekte die Anforderungen an die zu spezifizierenden Geräte nur grob festgelegt. Eine detaillierte Fassung der (elektronischen) Lastenhefte entstand jeweils im Laufe der Projekte.

5.5.2 Schnelle Anwendungsentwicklung durch Prototypenbau

Durch den hohen Grad an Interdisziplinarität in Projekten zur Entwicklung von HMI wird eine gemeinsame Arbeitsgrundlage für die beteiligten Rollen benötigt. Im TeleDrive VISION Prozess ist dies die Simulation (also der virtuelle Prototyp (s. Kapitel 5.3.4)). Dieser Prototypenbau entspricht dem Ansatz des *Rapid Prototyping* (s. [Tha93, S. 56 ff]). Der „Zweck von Rapid Prototyping muß es sein, dem Benutzer oder Kunden innerhalb kurzer Zeit einen benutzbaren Prototypen der Software zeigen zu können“ [Tha93, S. 57].

Vor allem die von Thaller beschriebene dritte Ausprägung des *Rapid Prototyping* trifft auf den TeleDrive VISION Prozess zu. Denn dort „werden für die Erstellung der Spezifikation formale Sprachen eingesetzt, die wiederum direkt in ausführbaren Code übersetzt werden können.“ [Tha93, S. 57]

Rapid Prototyping kann nach Thaller besonders geeignet sein für die Entwicklung des HMI eines Softwaresystems, da bei der Schnittstelle zum Benutzer die Korrektheit nur subjektiv bestimmbar ist [Tha93, S. 57]. „Bei allen anderen Schnittstellen der Software tut man sich da leichter.“ [Tha93, S. 57] Denn bei allen anderen Schnittstellen, wie z.B. einer Schnittstelle zur Hardware, ist die Korrektheit zumeist objektiv bestimmbar.

Die IAV geht bei der Einordnung ihres Prozesses nach Aussage von Herrn Wegner noch einen Schritt weiter und bezeichnet ihren Ansatz als „Real Prototyping“, da die iterativ erstellte Software auch auf den Hardwaremustern bereits laufen kann und somit Teile der Software bereits während der Entwicklung im Fahrzeug erlebt und getestet werden könnten.

Die Verbindung des Prototypenbaus mit Sprachen der vierten Generation, wie hier mit IML, wurde auch von Pressmann als Modell für die Entwicklung von Mensch-Maschine-Schnittstellen vorgeschlagen (s. Abbildung 17) [Joh92]. Da zumindest die detaillierte Anforderungserhebung (engl. *requirements gathering*) im TeleDrive VISION Prozess nicht wasserfallartig geschieht und dem Zulieferer für das Vorgehen bei der Produktentwicklung kaum Vorgaben gemacht werden (können), ist hier lediglich das „Prototyping“ (der mittlere Kasten) relevant.

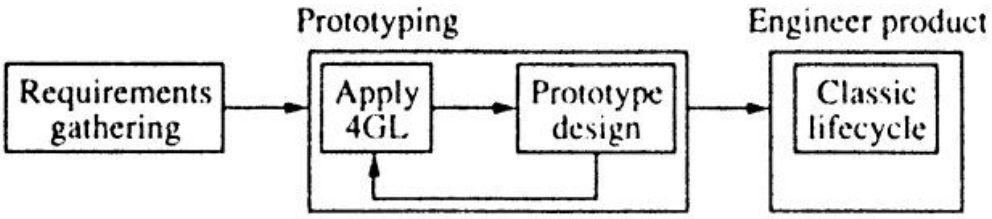

Abbildung 17: Prozessmodell von Pressman für die Entwicklung von HMI [Joh92]

5.5.3 V-förmiges Modell

Nach der grafischen Darstellung des Teledrive VISION Prozesses auf einer Webseite der IAV [TDV06] basiert der Prozess auf dem Vorgehensmodell, kurz V-Modell. „Das V-Modell ist ein Leitfaden zum Planen und Durchführen von [Softwareentwicklungs-] Projekten“ [KBSt05]. Der Name stammt vielleicht auch von seiner V-förmigen grafischen Darstellung, wie sie in Abbildung 18 zu sehen ist. „Diese Darstellung wird manchmal das V-Modell genannt.“ [Som01, S. 431]

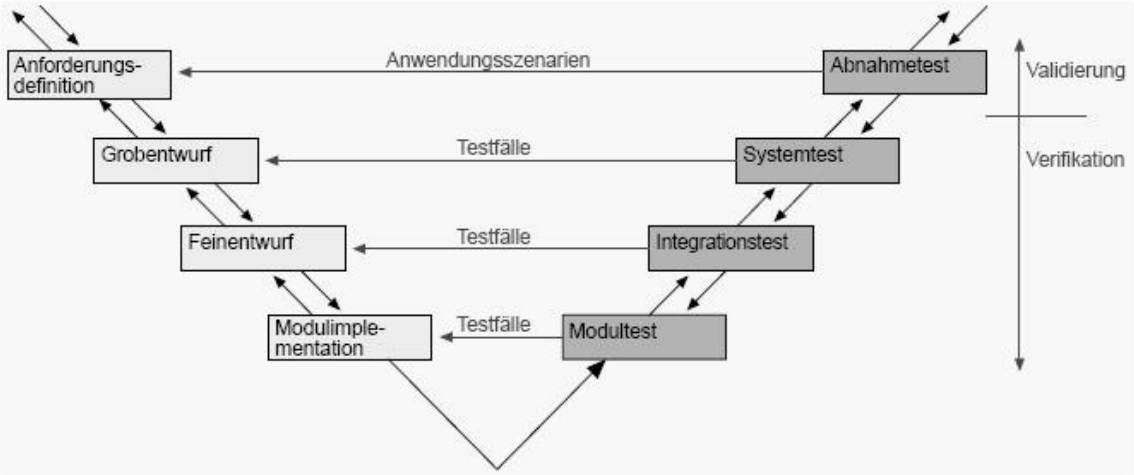

Abbildung 18: Prozessschema des V-Modells [Pre03]

Die erste Version dieses Prozessmodells entstand 1986 im Bundesministerium für Verteidigung als Erweiterung des Wasserfallmodells von Boehm um eine Qualitätssicherung. Seitdem wurde es mehrmals überarbeitet und erweitert, um neuere Methoden und Techniken der Softwareentwicklung mit zu berücksichtigen. Seit Februar 2005 liegt die Version XT (Abkürzung für *eXtreme Tailoring*) des Modells vor, die unter anderem mit der Zielsetzung entstand, das Modell „dem neuesten Stand des Systems Engineering" anzupassen [Hei06]. Es „erlaubt je nach Projektart ein maßgeschneidertes [engl. *tailored*] Vorgehen" [Hei06]. Eine recht detaillierte (533 Seiten) Beschreibung dieses Modells (bzw. dieses Richtlinienkatalogs) ist in [KBSt05] zu finden. Das V-Modell ist nicht nur bloße Theorie: Es wird in sämtlichen IT-Projekten, die von der Bundesregierung in Auftrag gegeben werden, verwendet, worunter sich auch Großprojekte wie die Einführung eines LKW-Mautsystems befinden [Hei06]. Auch in vielen deutschen Unternehmen, wie z.B. Siemens[1], die an der Weiterentwicklung des V-Modells beteiligt waren [Hei06], kommt es mitunter zum Einsatz. Eine Aussage darüber, inwieweit das V-Modell, wie es auf den 533 Seiten vorgeschrieben ist, in den IT-Projekten auf Bundesebene und in den Unternehmen umgesetzt wird, konnte ich in der Literatur nicht finden. Nach Prechelt haben, die „meisten Leute, die behaupten sich damit auszukennen, [...] nur Grundkenntnisse" [Pre06, F. 33].

Der TeleDrive VISION Prozess stellt nach Aussage eines IAV-Mitarbeiters keine vollständige Implementierung des V-Modells, wie es in [KBSt05] oder in älteren Versionen von Dokumentationen des V-Modells beschrieben ist, dar. Der Prozess ist jedoch in soweit an das V-Modell angelehnt, als das er die, für dieses Modell typischen Teilprozesse, wie sie in Abbildung 18 zu sehen sind, durchläuft. Um dieser Anlehnung gerecht zu werden, lässt sich das in Abbildung 15 dargestellte Prozessdiagramm auch wie in Abbildung 19 geschehen V-förmig darstellen.

[1] Zumindest nach den Stellenanzeigen des Unternehmens zu urteilen

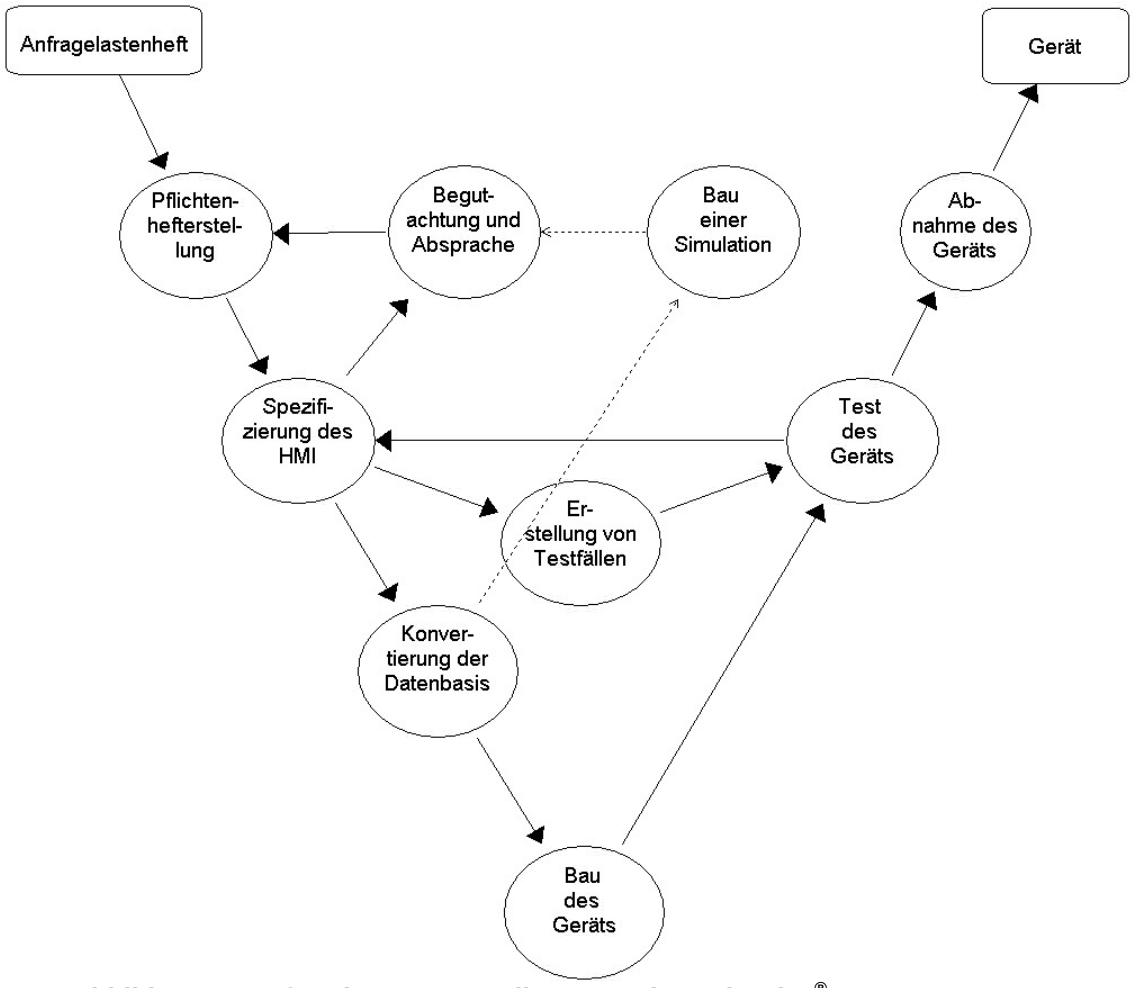

Abbildung 19: V-förmiges Prozessdiagramm des TeleDrive® VISION Prozesses

6 Der Spezifizierungsprozess von TeleDrive VISION

Dieses Kapitel ist Teil des Ergebnisses der unter Kapitel 4 beschriebenen Studie. Die hier zusammengetragenen Informationen stammen aus den Interviews, die ich mit den an den untersuchten Projekten beteiligten Mitarbeitern führte. Informationen über die Ausgangsbasis und Ziele dieses Teilprozesses sind im Abschnitt 5.3.3 nachzulesen. Nachfolgend beschreibe ich den detaillierten Ablauf des Spezifizierungsprozesses und die daran beteiligten Rollen.

6.1 Beteiligte Rollen

An der Spezifizierung des HMI sind mehrere Rollen beteiligt, die nachfolgend vorgestellt werden. Eine wichtige hier nicht näher betrachtete Rolle ist die der Projektleiter. Sie koordinieren die Arbeit der Rollen bzw. die „Exemplare“ der Rollen und stimmen die Arbeit mit anderen am jeweiligen Projekt beteiligten Parteien, wie z.B. einem Zulieferer, ab. Die dabei anfallenden Arbeiten sind, wie es der Name schon impliziert, projektspezifisch.

Es sei an dieser Stelle angemerkt, dass am TeleDrive VISION Prozess noch weitere Rollen beteiligt sind, wie z.B. die Tester oder die für den Simulationsbau zuständigen Softwareentwickler. Sie sind in dieser Arbeit aufgrund der Konzentration auf den Spezifizierungsprozess jedoch nicht näher beschrieben.

6.1.1 Der HMI-Experte

Die HMI-Experten befassen sich im TeleDrive VISION Prozess mit den Aufgabengebieten eines Softwareergonomen und denen eines technischen Experten. Letzterer bezeichnet hier einen Experten auf dem Gebiet der Technik von Infotainmentsystemen. Eine Trennung dieser beiden Rollen ist meiner Ansicht nach nicht praktikabel, da sie sehr eng miteinander verzahnt sind. In den untersuchten Projekten nahmen die HMI-Experten jeweils beide Rollen zusammen ein.[1]

Der Begriff Ergonomie setzt sich zusammen aus den griechischen Wörtern *ergon* (dt.: Arbeit, Mühe, Werk) und *nomos* (dt.: Gesetz, Lehre, Regel) – das Gesetz der Arbeit. Die Softwareergonomie ist die Wissenschaft von der Benutzbarkeit und Gebrauchstauglichkeit von Computerprogrammen [Wik06a]. In der englischsprachigen Literatur wird in diesem Zusammenhang von *usability* gesprochen.

Die HMI-Experten sind für die Gestaltung von benutzbaren Bedienoberflächen für Infotainmentsysteme zuständig. Sie entwerfen das Aussehen und Verhalten des HMI. Für die Benutzbarkeit ist es erforderlich, die vielfältigen Funktionalitäten des Systems auf eine intuitive und übersichtliche Weise dem Benutzer zur Verfügung zu stellen. Das gesamte Erscheinungsbild des HMI muss dafür in sich konsistent sein.

Die ergonomische Gestaltung wird dadurch beeinflusst, dass die Infotainmentsysteme im Fahrzeug zum Einsatz kommen, denn die Geräte müssen auch während der Fahrt von dem Autofahrer bedienbar sein. Es ist also eine andere Bedienphilosophie erforderlich als z.B. für eine Stereo-Anlage im Wohnzimmer. Zum einen müssen die HMI-Experten deshalb darauf achten, dass dem Benutzer kurze, prägnante Wege zu seinem Bedienziel angeboten werden. Zum anderen müssen sie dabei auch berücksichtigen, dass die Bedienung des Systems den Autofahrer nur sehr begrenzt vom Fahren ablenken darf.

[1] Es sei angemerkt, dass die Rolle des technischen Experten im Prozess auch in „Reinform“ existiert. Diese sind an der Anfragelastenheft- und Pflichtenhefterstellung beteiligt.

Bei ihrer Arbeit spielen aber nicht nur ergonomische Gesichtspunkte eine Rolle. Sie benötigen auch ein ausgeprägtes technisches Verständnis über die Geräte, die sie spezifizieren. So muss ein HMI-Experte, der das HMI eines Radios entwirft, über ein fundiertes Wissen über moderne Radiogeräte verfügen. Er muss den Leistungsumfang verschiedenster spezieller Funktionalitäten kennen, wie z.B. *Traffic Program* (TP) oder *Traffic Message Control* (TMC).

Die befragten HMI-Experten erfinden die für das HMI notwendigen Bedienkonzepte nicht unbedingt alle neu, sondern sie orientieren sich auch an Vorgängermodellen und an am Markt üblichen Geräten. Sie benötigen dafür einen guten Überblick über den Stand der Technik. Außerdem müssen sie sich mit dem Patentrecht auskennen, denn das Kopieren von bekannten Bedienkonzepten ist nicht immer ohne weiteres möglich, da viele Konzepte von den jeweiligen Herstellern patentrechtlich geschützt sind.

6.1.2 Der Grafikdesigner

Die Akzeptanz eines Infotainmentsystems beim Benutzer hängt nicht nur von den zur Verfügung gestellten Funktionalitäten und dem Bedienkonzept ab, sondern auch von dem Erscheinungsbild des Gerätes. Deshalb sind am Entwicklungsprozess oft auch Grafikdesigner beteiligt. Sie gestalten die grafische Repräsentation des gesamten HMI, was sowohl die GUI der Software als auch das äußere Erscheinungsbild der Gerätehardware beinhaltet. Zu letzterem gehört unter anderem die Festlegung von Formen und Farben für den Rahmen des Bildschirms und den am Gerät befindlichen Tasten.

Am Anfang eines Projekts entwickeln sie Ideen und Konzepte für Interaktionselemente, die in der GUI zum Einsatz kommen sollen. Dabei kooperieren sie mit den HMI-Experten. Im weiteren Projektverlauf konkretisieren die Grafikdesigner die Ideen. Sie spezifizieren die Parameter der grafischen Elemente, wie Farben, Positionen und Dimensionen und erstellen Grafiken für die Elemente und Hintergrundbilder.

Wie bei den HMI-Experten ist auch die Arbeit der Grafikdesigner davon beeinflusst, dass die zu entwickelnden Infotainmentsysteme später in einem Fahrzeug eingesetzt werden. Es ist einem Autofahrer beispielsweise nicht möglich, sich während der Fahrt für längere Zeit auf den Bildschirm des Infotainmentsystems zu konzentrieren. Deshalb muss der Grafikdesigner dafür Sorge tragen, dass die angezeigten Informationen auch mit einem kurzen Blick lesbar sind. Dementsprechend muss er die Schrifttypen, Größen, Farben und Positionen der Texte auswählen.

Sowohl das Erscheinungsbild der Gerätehardware als auch das der GUI muss mit dem Interieur des Fahrzeugs abgestimmt sein. Auch hierauf müssen die Grafikdesigner Acht geben und müssen sich dafür mit den übrigen Produktdesignern der Fahrzeugentwicklung absprechen.

6.1.3 Der IML-Entwickler

Der IML-Entwickler ist zuständig für die Überführung der von den Grafikdesignern und HMI-Experten erarbeiteten Entwürfe in ein formales, maschinenlesbares Format. Dafür beschreibt er Widgets in der Spezifikationssprache IML, weshalb die Beherrschung dieser Sprache für ihn unabdingbar ist. Dadurch, dass IML auf der Sprache XML basiert, benötigt er auch zumindest Grundkenntnisse über diese Sprache.

Ein IML-Entwickler sollte auch Programmiererfahrung besitzen. In den untersuchten Projekten waren die IML-Entwickler fast ausschließlich ausgebildete Informatiker. Das Spezifizieren in IML kommt dem Programmieren noch sehr nahe. Für die Beschreibung des Verhaltens von Widgets müssen beispielsweise auch Funktionen für die Interaktion mit Gerätekomponenten deklariert werden. Dafür muss ein IML-Entwickler unter ande-

rem die Bedeutung von einzelnen Parametertypen kennen und den Unterschied zwischen synchroner und asynchroner Funktionsausführung verstehen.

Für die Verifizierung der von ihnen erstellten IML-Datenbasis benutzen sie die Simulationsplattform, weshalb sie den Umgang damit erlernt haben müssen. Sie müssen unter anderem bei der Defektbehebung wissen, welche Fehlermeldungen der Plattform auf welche Art von Defekten zurückzuführen sind.

Die Arbeit des IML-Entwicklers im TeleDrive VISION Prozess erfordert viel Kommunikation mit anderen Prozessbeteiligten. Einerseits muss er sich mit den Softwareentwicklern, die für den Bau der Simulation verantwortlich sind, darüber absprechen, wie welches Verhalten von Widgets umgesetzt wird und wie welche Funktionen der Gerätekomponenten arbeiten sollen. Andererseits muss er mitunter auch Rücksprache mit den Grafikdesignern darüber halten, wie Spezialfälle bei z.B. der Vermassung gehandhabt werden sollen.

Der kooperationsaufwändigste Bereich im Gesamtprozess ist, nach allem was ich über den Prozess kennen gelernt habe, die Schnittstelle zwischen den IML-Entwicklern und den HMI-Experten. Hier sind ständige Absprachen darüber notwendig, wie sich die HMI-Experten die Umsetzung des HMI konkret vorgestellt haben. Dies ist, wie weiter unten noch genauer erläutert wird, meines Erachtens nach dem Austauschformat zwischen den beiden Rollen, den illustrierten Statecharts, geschuldet (s. Kapitel 7.1).

6.2 Teilprozesse

Der Spezifizierungsprozess lässt sich grob in zwei Teilprozesse unterteilen: HMI-Experten und Grafikdesigner entwerfen das HMI mit dem Ergebnis einer semiformalen Spezifikation. Diese wird von IML-Entwicklern in eine formale IML-Spezifikation überführt. Eine feinere Zerlegung ergibt sechs Teilprozesse, die nachfolgend vorgestellt werden. Einen Überblick über die Beziehungen zwischen den Teilprozessen verschafft die in Abbildung 36 dargestellte Grafik (s. Anhang 12.1, Seite 114). Da es sich hier um einen iterativen Prozess handelt, sollen in der Grafik keine zeitlichen Aspekte enthalten sein. Die Grafik ist im Zusammenhang mit der in Kapitel 5 abgebildeten Darstellung des Gesamtprozesses (s. Abbildung 15) zu sehen. Es handelt sich hier nicht um einen einmalig linear durchlaufenen Prozess.

6.2.1 Entwurf elementarer Widgets

Ausgehend vom Pflichtenheft entwickeln HMI-Experten in Zusammenarbeit mit Grafikdesigner ein Bedienkonzept für das HMI des Infotainmentsystems. Beschreibungen von den für dieses Konzept benötigten elementaren Widgets, wie z.B. Tasten oder Listen, stellen sie in einem Katalog zusammen: dem Widget-Katalog.

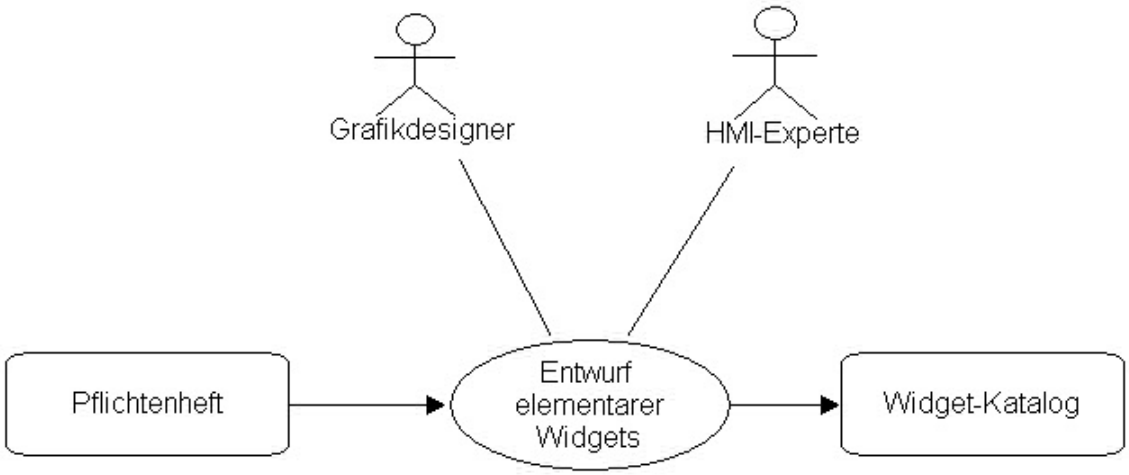

Abbildung 20: Entwurf elementarer Widgets

Eingaben

Grundlage für diesen Teilprozess ist das Pflichtenheft. Es enthält eine Beschreibung aller Produktmerkmale, die das zu entwickelnde Infotainmentsystem bereit stellen soll.

Aktivitäten

Die HMI-Experten überlegen sich zusammen mit den Grafikdesignern ein Bedienkonzept, mit dem es möglich ist, alle im Pflichtenheft aufgelisteten Produktmerkmale zu realisieren. Aus dem Pflichtenheft entnehmen sie außerdem die von ihnen benötigten Informationen über Gerätespezifika, wie z.B. die Vermassung des Bildschirms. Damit das Bedienkonzept in sich konsistent ist, definieren sie Interaktionselemente, also elementare Widgets, mit denen die Menüs des HMI aufgebaut werden sollen. Diese sollen systemweit ähnlich aussehen und sich auch ähnlich verhalten. Abweichungen kann es z.B. bei der Farbgebung oder der Beschriftung geben. Das Grundkonzept der elementaren Widgets sollte aber gleich sein. Ein Grund dafür ist, dass ein Benutzer des Infotainmentsystems dadurch einen geringeren Lernaufwand für die Bedienung aufbringen muss. Aber auch der Bau der Gerätesoftware inklusive Tests wird dadurch vereinfacht.

Elementare Widgets können z.B. Widgets für Tasten, Listenelemente oder Textfelder sein. Die Grafikdesigner und die HMI-Experten stellen in dem Widget-Katalog eine Auflistung der in den Menüs zu benutzenden Widgets zusammen. Sie beschreiben jeweils den Verwendungszweck des Widgets und geben an, welche Zustände es haben kann. Außerdem werden die Eigenschaften des Widgets, die später im IML-Format anpassbar sein sollen, aufgelistet und erklärt. Bei einem Widget, das ein Textfeld spezifizieren soll, kann es sich dabei z.B. um die Anzahl der Zeilen, den Schrifttyp oder auch den Inhalt handeln. Falls sich das Widget aus mehreren anderen Widgets zusammensetzt, wird diese Struktur verbal oder grafisch, wie z.B. in Abbildung 3 geschehen (s. nummeriertes Element 2.4.2), beschrieben. Zusätzlich wird, falls notwendig, das Verhalten des Widgets erklärt. Die Grafikdesigner zeichnen erste Skizzen für die grafische Repräsentation der Widgets und fügen sie dem Dokument hinzu.

Ergebnis

Das Ergebnis des Teilprozesses ist der Widget-Katalog. In dem Dokument werden die verschiedenartigen elementaren Widgets des zu spezifizierenden HMI beschrieben, die später für die Erstellung der Menü-Widgets benötigt werden. Informationen über die

Menü-Widgets selber sind in dem Katalog nicht enthalten. Konventionen für das Format des Widget-Katalogs sind nicht vorhanden. Die Form kann von Projekt zu Projekt verschieden sein. Ein Ziel bei der Erstellung dieses Kataloges sollte es jedoch sein, die Widgets so allgemein zu beschreiben, dass der jeweilige Fahrzeughersteller den Katalog für verschiedene von ihm in Auftrag gegebene Geräte wiederverwenden kann.

6.2.2 Entwurf von Menüs

Anhand des Pflichtenhefts und des Widget-Katalogs wird eine Beschreibung der Mensch-Maschine-Schnittstelle für das Infotainmentsystem entworfen. Die hierfür zuständige Rolle ist der HMI-Experte, wie es in Abbildung 21 zu sehen ist. Er entwirft das Menünetz des HMI und die grobe Struktur der einzelnen Menüs. In den untersuchten Projekten stellten die HMI-Experten diese Informationen in Form von *illustrierten Statecharts*, einer Form von Menüdiagrammen, dar.

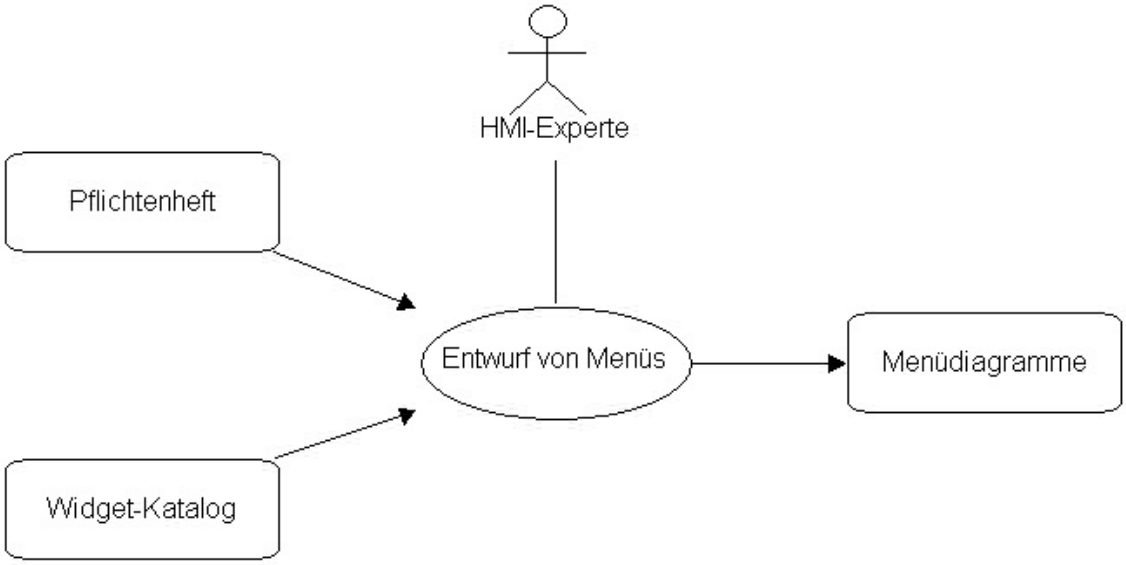

Abbildung 21: Entwurf von Menüs

Eingaben

In dem vom Zulieferer verfassten Pflichtenheft befindet sich eine Auflistung der Produktmerkmale, die das Infotainmentsystem später zur Verfügung stellen soll. Die Produktmerkmale sind üblicherweise nach den einzelnen Kontexten des Systems sortiert, welche die verbauten Gerätekomponenten, wie z.B. CD-Spieler, Radio oder Navigationskomponente, widerspiegeln. Ansonsten kann das Format dieser Auflistung von Projekt zu Projekt verschieden sein. Die Produktmerkmale sind informal beschrieben. Für den HMI-Experten stellt die Auflistung den Rahmen dessen dar, was er in den HMI-Entwurf mit aufnehmen muss.

Auch andere technische Details über das zu entwickelnde Infotainmentsystem sind für den Entwurf des HMI notwendig. Die HMI-Experten müssen die Dimension des Bildschirms kennen und auch wissen, mit wie vielen und mit welcher Art von Tasten das Gerät von außen bedient werden kann. Auch diese Vorgaben entnimmt er dem Pflichtenheft.

Zusätzlich verwenden die HMI-Experten den im vorangegangenen Teilprozess zusammengestellten Widget-Katalog. Die darin aufgeführten Widgets stellen die Bausteine für den Entwurf des HMI dar.

Aktivitäten

Um eine Übersichtlichkeit der Bedienoberfläche zu gewährleisten, teilt der HMI-Experte die Produktmerkmale eines Kontextes zunächst auf mehrere Menüs auf. Danach entwirft er für jedes dieser Menüs ein grobes Design. Er entscheidet, welche Interaktionselemente, wie z.B. Textfelder, Tasten oder auch Grafiken für das Menü benötigt werden und wie diese in etwa angeordnet sein sollen. Dabei kommen vor allem ergonomische Prinzipien zum Einsatz. Die Menüs müssen so gestaltet sein, dass sie für den Benutzer einfach und selbsterklärend bedienbar sind. Trotzdem darf aber keines der geforderten Produktmerkmale fehlen.

Die Menüdiagramme werden mit einem geeigneten Visualisierungs-Werkzeug erstellt. Für die grafische Repräsentation der Menüs würde es ausreichen, ein Zeichenprogramm zu verwenden. In den Menüdiagrammen modelliert der HMI-Experte jedoch zusätzlich noch eine Art Zustandsmaschine für das Menü. Er spezifiziert, unter welchen Bedingungen es verlassen wird und in welches Menü daraufhin gesprungen wird, wodurch ein Entwurf eines Menünetzes entsteht. Auch hierfür muss das verwendete Werkzeug eine Unterstützung bieten.

Für die im Menü vorkommenden Interaktionselemente, die vom Benutzer aktivierbar sind, spezifiziert der HMI-Experte in verbaler Form, welche Funktionalität bei dessen Betätigung ausgeführt werden soll. Das kann die Ausführung einer geräteinternen Funktion sein, wie z.B. das Starten des Sendersuchlaufs des Radios. Es kann aber auch eine vom HMI angebotene Funktion sein, wie z.B. das Starten einer Animation.

Eine weitere Aufgabe bei der Erstellung der Menüs besteht darin, dem Benutzer die Funktion der Interaktionselemente mit Hilfe von Texten oder Symbolen zu erklären. Die Texte und Symbole müssen dabei so gewählt werden, dass sie in die Dimension des Interaktionselements passen und trotzdem eine ausreichende Erklärung bieten.

Die HMI-Experten müssen während ihrer Arbeit auch Entscheidungen darüber treffen, welche technischen Details vor dem Benutzer verborgen bleiben sollen und welche nicht. Zum einen müssen sie entscheiden, welche vom Gerät gelieferten Daten dem Benutzer angezeigt werden sollen. So kann z.B. ein digitales Radio eine Vielzahl von Senderinformationen liefern, die ein Benutzer nicht unbedingt alle wissen will. Zum anderen muss auch die Konfiguration der einzelnen Gerätekomponenten benutzerfreundlich konzipiert werden. Es muss dem Benutzer möglich sein, alle ihm wichtigen Einstellungen vornehmen zu können. Er darf dabei aber auch nicht überfordert werden. Deshalb werden technisch zu komplizierte Einstellungsmöglichkeiten dem Benutzer gar nicht erst zur Verfügung gestellt, sondern es wird ein Standardwert festgelegt.

Ergebnisse

Das Resultat dieses Teilprozesses ist die Darstellung des HMI-Entwurfs in Form von Menüdiagrammen für alle Menüs, die für das Infotainmentsystem benötigt werden. In Abbildung 22 ist beispielhaft ein Menüdiagramm in Form einer illustrierten Statechart, wie sie in den untersuchten Projekten verwendet wurden, zu sehen. Es stellt einen Teil der Menüstruktur des Radiokontexts eines Infotainmentsystems dar. Eine genauere Betrachtung des Spezifikationsformats der illustrierten Statecharts folgt in Kapitel 7.1.

In den Menüdiagrammen sind verschiedenartigste Informationen enthalten. Das Aussehen der Menüs wird grob dargestellt, indem die einzelnen Interaktionselemente eingezeichnet werden. Dabei werden auch die Texte, die auf den Elementen angezeigt werden sollen, mit eingetragen. Sie können entweder statisch sein, wie z.B. der Text „Speicher“ einer Taste in dem Menü *TUN_SELECT*, das in Abbildung 22 dargestellt ist. Oder die Texte sind dynamisch, das heißt sie werden von einer Gerätekomponente geliefert, wie z.B. der Sendername „Bayern 3“ in dem Menü *TUN_SELECT*. Diese Unterschei-

dung wird mitunter in den Diagrammen für die das Dokument weiterverarbeitenden IML-Entwickler kenntlich gemacht.

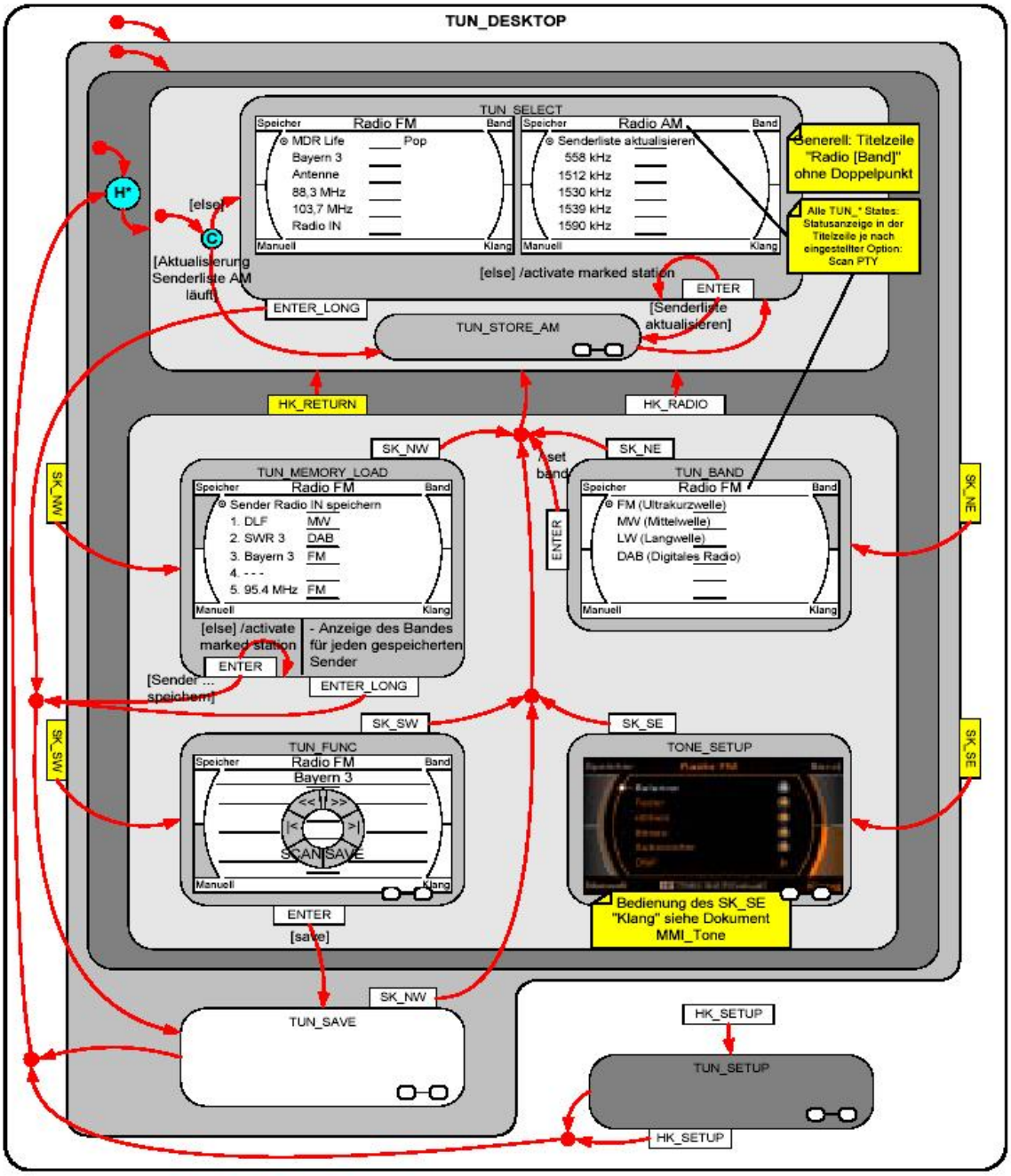

Abbildung 22: Illustrierte Statechart für ein Infotainmentsystem [HaM03]

Falls ein eingezeichnetes Interaktionselement aktivierbar ist, wie z.B. eine Taste, wird zumeist natürlichsprachig beschrieben, was bei einer Aktivierung geschehen soll. Dabei ist Genauigkeit und Eindeutigkeit wichtig, da die Beschreibungen die Grundlage für spätere Funktionsspezifikationen bilden.

Die Menüdiagramme beinhalten außerdem noch Informationen über Zustände und Zustandsübergänge des Menüs. Es muss angegeben werden, auf welches Ereignis hin ein Menü seinen Zustand wechseln soll und wie diese Zustände aussehen. Ein Zustandsübergang kann auch ein Menüwechsel sein. Es muss mit Zustandsübergangspfeilen dargestellt werden, welche anderen Menüs unter welchen Bedingungen zu erreichen sind. Von dem in Abbildung 22 dargestellten Menü *TUN_SELECT* gelangt man z. B. in das

Menü *TUN_MEMORY_LOAD*, wenn die Taste „Speicher“ („SK_NW“) gedrückt wurde. Auch die Zustandsübergänge von den im Menü enthaltenen Interaktionselementen müssen kenntlich gemacht werden.

6.2.3 Erstellung der Grafikspezifikation

Das in den Menüdiagrammen von den HMI-Experten suggerierte Layout der Menüs wird von den Grafikdesignern konkretisiert. Dabei erstellen sie eine sogenannte Grafikspezifikation, ein Dokument, in der die genauen Vermassungen und andere für die grafische Repräsentation der Menüs entscheidenden Werte angegeben sind. Zusätzlich fertigen sie die für das HMI benötigten Grafiken an.

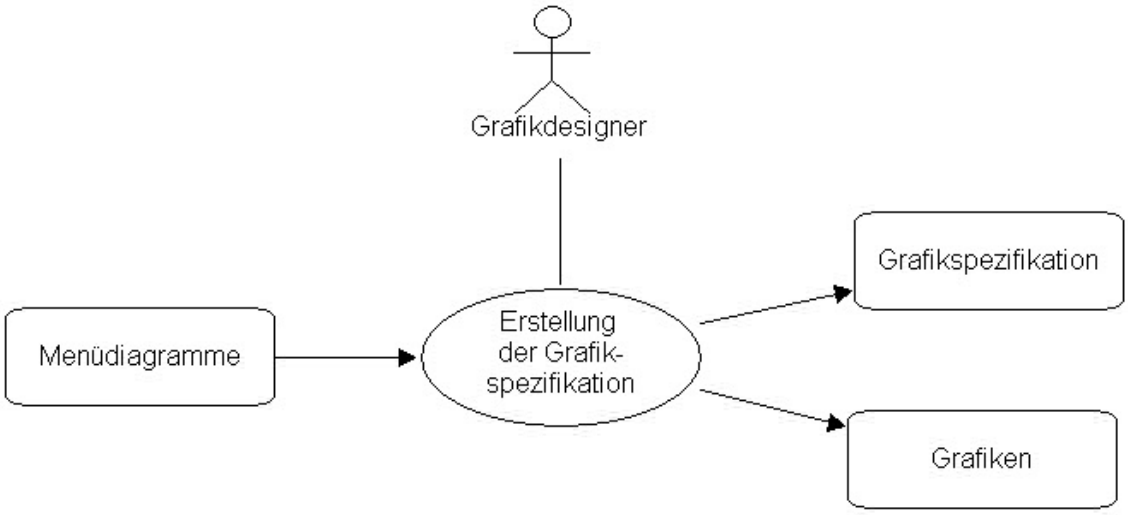

Abbildung 23: Erstellung der Grafikspezifikation

Eingaben

Die von den HMI-Experten erstellten Menüdiagramme bilden die Arbeitsgrundlage der Grafikdesigner in diesem Teilprozess.

Aktivitäten

Für alle in den Menüdiagrammen dargestellten Menüs und die darin enthaltenen Elemente legen die Grafikdesigner Werte für die Farben, Positionen und Dimensionen fest. Falls ein Element beschriftet ist, spezifizieren sie den Typ, die Größe und die Farbe der Schrift. Auch der Zeilenabstand und die Laufweite des Textes werden vermerkt. Zusammengefasst werden alle Informationen in der Grafikspezifikation. Dieses Dokument enthält außerdem noch Skizzen und Beschreibungen für kompliziertere grafische Darstellungen, wie z.B. Animationen.

Falls für die Darstellung eines Elements eine Grafik erforderlich ist, so erstellen die Grafikdesigner sie mit einem geeigneten Zeichenprogramm. Der Name der Datei, in der die Grafik gespeichert wurde, wird für das betreffende Element in der Grafikspezifikation mit aufgenommen.

Dieser Teilprozess ist eng gekoppelt mit dem Entwurf elementarer Widgets und dem von Menüs. Das heißt die Grafikdesigner kooperieren vor allem mit den HMI-Experten. Durch die Kopplung und die Kooperation fließen auch Informationen aus dem Pflichtenheft in die Ergebnisse mit ein.

Ergebnisse

Die Ergebnisse des Teilprozesses sind die erstellten Grafikdateien und das Grafikspezifikationsdokument, das die oben beschriebenen Informationen enthält. Der TeleDrive VISION Prozess schreibt nach Aussage von Herrn Wegner nichts bezüglich der Form dieses Dokuments vor.

6.2.4 Spezifizierung elementarer Widgets

Auf Basis des Widget-Katalogs spezifizieren IML-Entwickler die im HMI vorkommenden elementaren Widgets in IML (s. Abbildung 24).

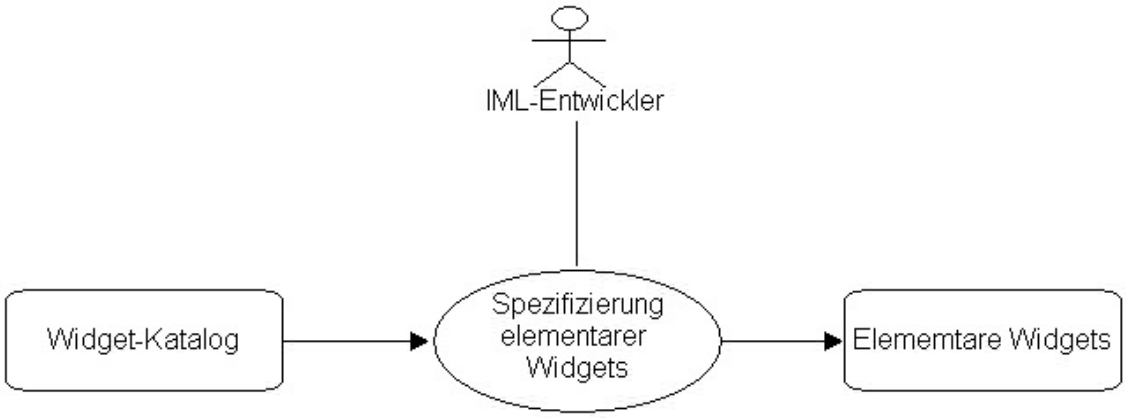

Abbildung 24: Spezifizierung elementarer Widgets

Eingaben

Die in diesem Teilprozess benötigten Informationen erhalten die IML-Entwickler in Form des Widget-Katalogs. Darin sind die elementaren Widgets, die für die Menüerstellung benötigt werden, verbal beschrieben.

Aktivitäten

Die verbale Beschreibung der im Widget-Katalog aufgeführten Widgets setzen die IML-Entwickler mittels IML technisch um. In den untersuchten Projekten verwendeten sie hierfür einen XML-Editor.

Zuerst bauen sie dafür eine passende Struktur auf. Sie besteht aus Basis-Widgets und anderen bereits vorhandenen elementaren (zusammengesetzten) Widgets. Dafür kann es unter Umständen notwendig sein, weitere elementare Widgets zu spezifizieren. Gegebenenfalls nehmen sie Anpassungen an den Eigenschaften der in der Struktur verwendeten Widgets vor, indem sie Werte auf der Ebene des zu spezifizierenden Widgets überschreiben. Danach werden die im Widget-Katalog beschriebenen Eigenschaften und Zustände in der IML-Datei spezifiziert. Außerdem werden darin mit Hilfe einer Zustandstabelle die Zustandsübergänge des Widgets umgesetzt. Auch die für das Verhalten des Widgets notwendigen Eigenschaften werden in der IML-Struktur mit aufgenommen. Teilweise müssen die IML-Entwickler auch Erweiterungen an den im Widget-Katalog beschriebenen Eigenschaften oder Strukturen der Widgets vornehmen, da die aus dem Katalog stammenden Vorgaben nicht immer ausreichen, um das Widget in der Praxis umzusetzen.

Die elementaren Widgets werden so generisch wie möglich spezifiziert, damit sie später an so vielen Stellen wie möglich wiederverwendet werden können. Die Wieder-

verwendung ist dabei nicht nur auf ein Projekt beschränkt. Sondern es sollte darauf geachtet werden, dass die elementaren Widgets auch in nachfolgenden Projekten des Fahrzeugherstellers eingesetzt werden können.

Die in diesem Teilprozess erstellten IML-Dateien stellen keine endgültige Version dar, da sich viele technische Probleme erst beim Einsatz der Widgets in den Menüs herausstellen. Wenn das der Fall ist, werden nach einer Absprache mit den für die Spezifierung der Menüs zuständigen IML-Entwicklern Anpassungen nachgezogen.

Auch kann es vorkommen, dass der Zulieferer Probleme mit der Umsetzung eines Widgets feststellt. Auch in diesem Fall werden nach einer Absprache Anpassungen an der Spezifikation des Widgets vorgenommen.

Ergebnisse

Für jedes der in dem Widget-Katalog aufgeführten elementaren Widgets schreiben die IML-Entwickler eine oder mehrere IML-Dateien, welche die formale Spezifikation des Widgets enthalten. Die Dateien bilden das Ergebnis dieses Teilprozesses.

6.2.5 Spezifizierung der HMI-API

Aus den Menüdiagrammen ermitteln die IML-Entwickler, welche Funktionen die Gerätekomponenten anbieten müssen. Diese spezifizieren sie in der sogenannten HMI-API. Um Verwirrungen zu vermeiden, sei an dieser Stelle erwähnt, dass es sich bei diesem Dokument nicht um eine Spezifikation der Funktionen handelt, die das HMI anbietet, wie die Bezeichnung HMI-API vielleicht vermuten lässt. Vielmehr werden in diesem Dokument die Funktionen spezifiziert, die das HMI importiert. Es handelt sich um die von den Gerätekomponenten bereitzustellenden Funktionen, die in der IML-Spezifikation aufgerufen werden.

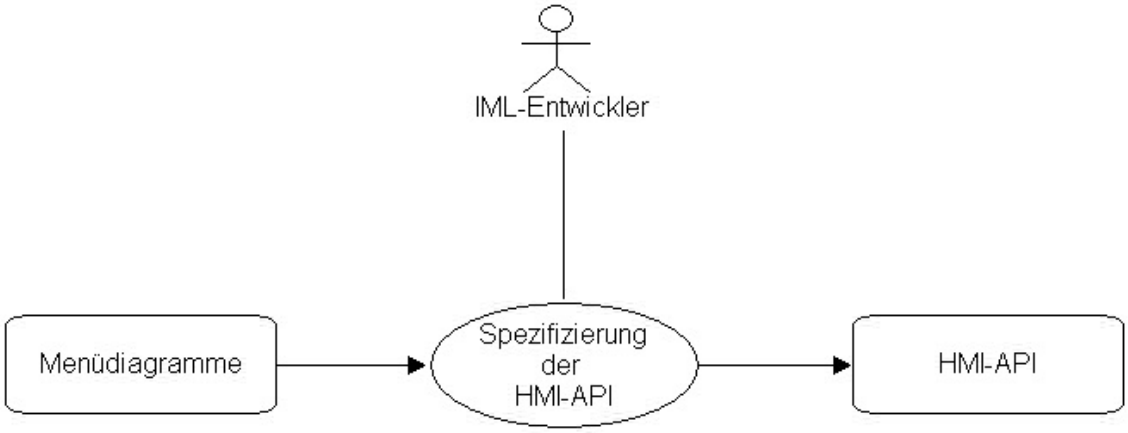

Abbildung 25: Spezifizierung der HMI-API

Eingaben

Als Eingabe für die Spezifizierung der HMI-API erhalten die IML-Entwickler die von den HMI-Experten erstellten Menüdiagramme, wobei nicht alle darin enthaltenen Informationen, wie z.B. das Layout, für diesen Teilprozess gebraucht werden. Wichtig sind darin vor allem die Beschreibungen der Funktionalitäten, die das jeweilige Menü anbieten soll.

Aktivitäten

In den Menüdiagrammen geben die HMI-Experten für die in den Menüs enthaltenen Interaktionselemente jeweils mit an, welche Funktionalität bei einer Betätigung ausgeführt werden soll. Die für die Umsetzung der Funktionalitäten benötigten Funktionen spezifizieren die IML-Entwickler in einer speziellen IML-Datei, der HMI-API. Außerdem sind in den Menüdiagrammen darzustellende Texte eingetragen, die von den Gerätekomponenten zu liefernde (dynamische) Informationen enthalten. Auch hierfür spezifizieren die IML-Entwickler Funktionen in der HMI-API.

Die Spezifikation einer geräteinternen Funktion besteht immer aus dem Funktionsnamen und dem Namen der Gerätekomponente, aus der sie stammt. Zusätzlich wird verbal beschrieben, was die Funktion leisten soll, und es werden die Ein- und Ausgabeparameter spezifiziert. Für letztere werden jeweils der Name, der Typ und der Wertebereich angegeben. Beispielhaft ist in Abbildung 26 eine in einer HMI-API spezifizierte Funktion dargestellt.

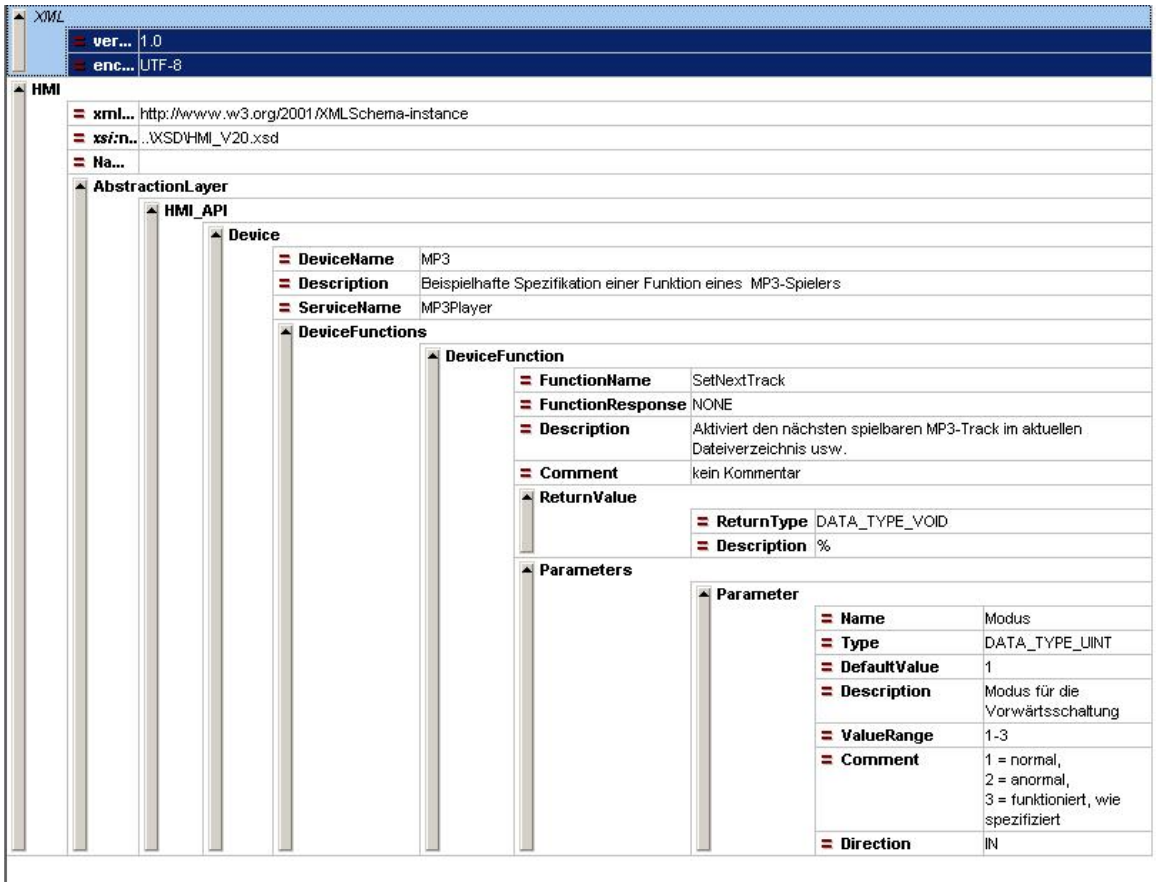

Abbildung 26: Beispielhafte Spezifikation einer HMI-API

Da dieser Teilprozess ein Bindeglied zwischen dem Entwurf des HMI und der Menüspezifizierung ist, und das Ergebnis, die HMI-API, auch für den Bau von Gerätesoftware- und Simulationskomponenten verwendet wird, ist für die Aufgabenerfüllung viel Kommunikation mit anderen Prozessbeteiligten erforderlich. Zum einen müssen die IML-Entwickler sich mit den HMI-Experten über die konkret benötigten Funktionsweisen absprechen, da nicht immer alle Informationen den Menüdiagrammen im Detail zu entnehmen sind (s. Kapitel 7.1). Zum anderen sind Absprachen mit den für die Menüerstellung zuständigen IML-Entwicklern notwendig. Es muss entschieden werden, wie die Parameter der Funktionen spezifiziert und in welcher Form Daten geliefert werden sol-

len. So kann es z.B. sinnvoll sein, mehrere Daten in einer Feldstruktur zusammenzufassen.

In Projekt B entstand mitunter starke Iterativität bei der Erstellung der HMI-API. Es war z.B. des Öfteren der Fall, dass den IML-Entwicklern erst bei der Spezifizierung der Menüs auffiel, dass eine weitere oder modifizierte Funktion benötigt wird. Daraufhin wurde sie nachträglich in die HMI-API eingetragen.

Auch mit den Softwareentwicklern, die für den Bau der Gerätesimulationskomponenten zuständig sind, müssen sich die IML-Entwickler bei der Erstellung der HMI-API abstimmen. Es können Probleme bei der Implementierung der Funktionen auftreten, wenn die technische Umsetzung der Spezifikation nicht oder nur umständlich machbar ist. Daraufhin müssen Anpassungen vorgenommen werden.

Die HMI-API wird auch von dem Zulieferer verwendet bzw. implementiert. Dabei kann es vorkommen, dass er eine Funktion mit der von ihm verwendeten Plattform nicht so, wie es in der HMI-API beschrieben ist, umsetzen kann. Z.B. kann es sein, dass für die Umsetzung noch ein zusätzlicher Eingabeparameter benötigt wird. Hier bedarf es einer Abstimmung zwischen den IML-Entwicklern und den Entwicklern des Zulieferers. In der Grafik über den Gesamtprozess (Abbildung 15) sind diese Abstimmungen Bestandteil des Teilprozesses „Begutachtung und Absprache“.

Ergebnisse

Das Ergebnis des Teilprozesses ist die HMI-API, die in einem speziellen Format der Sprache IML verfasst ist. Dieses Format ist in dem IML-Widget-Schema hinterlegt. In der HMI-API sind die im HMI des Infotainmentsystems verwendeten Funktionen der Gerätekomponenten spezifiziert. Der Übersichtlichkeit halber sind die Funktionen nach den Gerätekomponenten sortiert.

6.2.6 Spezifizierung von Menüs

Den von den HMI-Experten in den Menüdiagrammen angefertigten HMI-Entwurf übersetzen IML-Entwickler in das IML-Format. Unter Verwendung von in vorausgegangenen Teilprozessen erstellten Dokumenten spezifizieren sie die Menü-Widgets und was damit zusammenhängt.

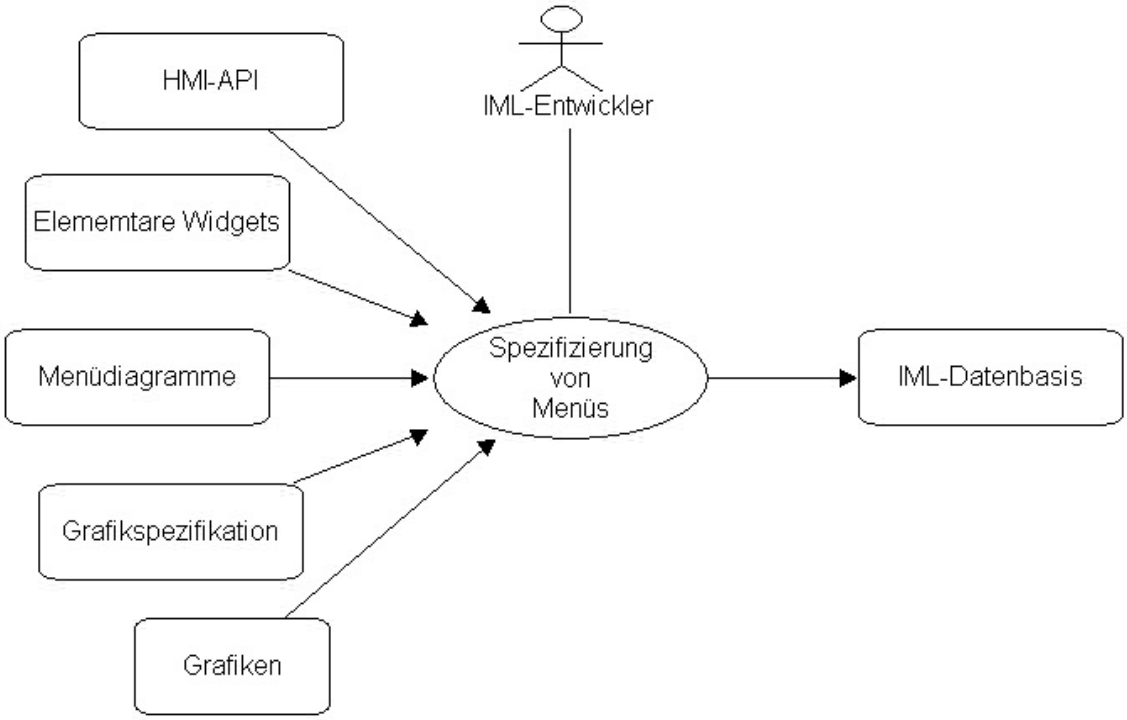

Abbildung 27: Spezifizierung von Menüs

Eingaben

Wie in Abbildung 27 zuerkennen ist, werden in diesem Teilprozess viele Dokumente verwendet. Die IML-Entwickler erhalten von den HMI-Experten die Menüdiagramme und von den Grafikdesignern die Grafikspezifikation und die Grafiken. Außerdem verwenden sie bei ihrer Arbeit die bereits in vorausgehenden Teilprozessen spezifizierten elementaren Widgets und die HMI-API.

Aktivitäten

Die zuvor spezifizierten elementaren Widgets dienen den IML-Entwicklern als Bausteine für die Menüerstellung. Sie bauen damit die Struktur der Menüs auf. Für komplexere Menüs werden zusätzlich extra IML-Dateien angelegt, die einen Ausschnitt des Menüs, wie z.B. eine Leiste mit mehreren Tasten, beschreiben. Auch diese werden in die Struktur der Menüs mit aufgenommen.

Anschließend werden die Eigenschaften der elementaren Widgets angepasst bzw. überschrieben. Dies kann z.B. für die Anzahl der Einträge in einer statischen Liste oder aber auch für den Inhalt eines Textfeldes notwendig sein. Die dabei zu verwendenden Werte entnehmen die IML-Entwickler den Menüdiagrammen. Auch die grafischen Eigenschaften der Widgets, wie z.B. Farben und Positionen, müssen angepasst werden. Informationen hierzu finden sie in der Grafikspezifikation. Darin sind auch die zu verwendenden Dateinamen der Grafiken angegeben, die z.B. als Hintergrundbild eines Menüs erscheinen sollen.

Die Beschreibung des Verhaltens der Menü-Widgets besteht zu einem Teil aus der Spezifizierung der im Menü verwendeten Funktionen. Die IML-Entwickler geben dabei unter anderem den Namen der Funktion, den Namen der Gerätekomponente, aus der sie stammt, und die Werte für die Eingabeparameter an. Die benötigten Informationen entnehmen sie der HMI-API. Falls die Funktion Daten zurückliefert, wird in einem Attribut vermerkt, ob die Daten bei einem Funktionsaufruf permanent oder nur einmalig von dem Gerät gesendet werden sollen.

Eine so spezifizierte Funktion können sie nun unter Angabe des Funktionsnamens verwenden, um die Herkunft von dynamischen Inhalten, also z.B. einem Radiosendernamen, zu definieren.

Eine weitere Aufgabe bei der Spezifizierung des Verhaltens eines Menü-Widgets ist die Modellierung der Zustandsmaschine. Zunächst werden hierfür die eingehenden Nachrichten, auf die das Menü reagieren soll, spezifiziert. Sie lassen sich in zwei Arten unterscheiden. Zum einen kann es sich um ein geräteinternes Ereignis handeln, wie z.B. der Eingang eines Anrufs in der Telefonkomponente. Zum anderen kann es eine HMI-interne Nachricht sein. Diese werden von Widgets des betreffenden Menüs oder von anderen Menü-Widgets versendet. Beispielsweise kann dies eine Nachricht sein, die dem Menü signalisiert, dass ein im Menü enthaltenes Interaktionselement betätigt wurde.

In einer Zustandstabelle spezifizieren die IML-Entwickler, wie auf den Empfang einer Nachricht reagiert werden soll. Sie setzen dabei die in den Menüdiagrammen eingezeichnete Zustandsmaschine um. Auf eine eingehende Nachricht wird entweder eine Funktion ausgeführt, eine andere Nachricht versendet oder der Zustand des Menüs gewechselt.

Um die in den Menüdiagrammen dargestellte Logik für Menüwechsel in IML umzusetzen, schreiben sie außerdem IML-Dateien für die Menü-Netze. Die Elemente der Menünetz-Widgets sind Menü-Widgets. Die Menüübergänge werden in der Zustandstabelle der Menünetze spezifiziert.

Um die Korrektheit der IML-Dateien, die sie mit einem XML-Editor erstellen, zu testen, benutzen sie die TeleDrive VISION Simulationsplattform. Eine genauere Betrachtung der in den Projekten eingesetzten Prüfmethoden folgt weiter unten.

Ergebnisse

Zusammen mit der HMI-API, den elementaren Widgets, den Grafik- und anderen Ressourcen-Dateien bilden die in diesem Teilprozess erstellten Widgets die IML-Datenbasis: die Spezifikation des HMI eines Infotainmentsystems.

7 Identifizierte Probleme und Lösungsvorschläge

Ein Ziel der empirischen Prozessanalyse war es, Problembereiche im Prozess zu identifizieren. Da ich den Fokus der Analyse auf den Spezifizierungsprozess setzte, lag die Konzentration der Problemidentifizierung vor allem auf der Schnittstelle zwischen den HMI-Experten und den IML-Entwicklern. Dadurch musste ich mich vor allem mit dem in den betrachteten Projekten verwendeten Austauschformat zwischen den beiden Rollen, den illustrierten Statecharts, auseinandersetzen. Im weiteren Verlauf der Arbeit deckte ich durch die Konzipierung der unten beschriebenen Durchsicht und einer dazu durchgeführten Studie weitere Probleme auf. Auch diese werden in diesem Kapitel behandelt.

7.1 Illustrierte Statecharts

In den Projekten wurde für die Menüdiagrammerstellung das Format der illustrierten Statecharts verwendet. Grundlagen über dieses Spezifikationsformat und die durch dieses in den Projekten aufgetretenen Probleme werden nachfolgend erläutert.

7.1.1 Grundlagen

Bei dem Modell der illustrierten Statecharts (abgekürzt ill. Statecharts) handelt es sich um einen von Audi [HaM03, S. 230 f] entwickelten Zustandsdiagramm-Dialekt (engl. statechart dialect). Zustandsdiagramme sind eine Erweiterung von Zustandsautomaten, die Harel 1987 in [Har87] beschrieb [Bal96, S. 277].

Nach Ryser [Rys03, S. 119] sind „Statecharts [...] weniger geeignet, um Kausalabhängigkeiten und bestimmte Zeitabhängigkeiten [...] darzustellen. Zusammenhänge und Beziehungen, die von Objektzuständen oder von Dateninhalten oder Zustandsvariablen abhängig sind [...] werden von der Statechart-Notation nur unzureichend unterstützt.“

Zustandsdiagramme sind außerdem ein Diagrammtyp der Modellierungssprache UML (Initialwort für *Unified Modeling Language*) [Vog05]. Diesen Diagrammtypen haben Hamberger und Mauter [HaM03] zu einer Modellierungssprache für HMI von Infotainmentsystemen erweitert: den illustrierten Statecharts. Wie in Abbildung 22 auf Seite 56 zu erkennen ist, werden darin - zusätzlich zur Beschreibung der Menüzustände und der Zustandswechsel (Transitionen) - skizzierte Darstellungen der jeweiligen Menügrafik in die Zustandsdiagramme eingefügt. Außerdem werden verschiedenste Merkmale des jeweiligen Menüs mit Hilfe von Kommentaren beschrieben.

Nach [HaM03] haben die illustrierten Statecharts „aufgrund ihrer Genauigkeit, Vollständigkeit und allgemeinen Verständlichkeit einen sehr effektiven Austausch zwischen Design, Ergonomie und Technik ermöglicht ...“. „Auf diese Weise kann keine wichtige Information z.B. zwischen Design- und Softwareentwicklung verloren gehen bzw. missverstanden werden.“ [HaM03]

Durch die natürlichsprachige und damit informale Beschreibungsform in den Kommentaren haben die illustrierten Statecharts höchstens semiformalen Charakter. Keines falls sind sie formal, da eine Notation als formal charakterisiert ist, falls Syntax und Semantik der Notation präzise und eindeutig definiert sind [Rys03, S. 28].

Durch den höchstens semiformalen Charakter der illustrierten Statecharts können theoretisch Missinterpretationen durch inkonsistente Informationen entstehen. Z.B. können sich zwei Aussagen in den Kommentaren widersprechen. Auch kann sich eine gezeichnete Information, wie z.B. ein Menüsprung, mit einem Kommentar widersprechen. Die Überprüfung der Korrektheit von illustrierten Statecharts kann - zumindest bisher - nur manuell geschehen. Es können bei deren Erstellung also ähnlich viele In-

konsistenzen entstehen, wie wenn ein Softwareentwickler ohne Übersetzer programmieren würde.

Auch ist es beim Einsatz dieses Spezifikationsformats nicht garantiert, dass Informationen verloren gehen bzw. nur unzureichend festgehalten werden, wie in 7.1.4 beschrieben. Weder eine Widerspruchsfreiheit noch eine Vollständigkeit der Spezifikation sind also durch dieses Format garantiert.

Illustrierte Statecharts und ähnliche Statechart-Dialekte werden nach Aussage von Herrn Wegner bei zahlreichen Fahrzeugherstellern in Projekten zur Entwicklung von Mensch-Maschine-Schnittstellen für Infotainmentsysteme eingesetzt.

7.1.2 Einsatz in den untersuchten Projekten

In den von mir untersuchten Projekten erstellten die HMI-Experten die illustrierten Statecharts mit VISIO, einem universellen Werkzeug aus den Häusern Microsoft für die Erstellung von Diagrammen. Für die Weitergabe der Spezifikation an die IML-Entwickler exportierten sie die Statecharts in das *Portable Document Format* (PDF). Ein derartiges Statechart-Dokument enthielt jeweils die Statecharts für alle Menüs eines Kontexts (z.B. Radio, Navigation, CD).

Aufgrund der starken Iterativität des Prozesses wurden die Statechart-Dokumente von den HMI-Experten in unvollständigen Versionen an die IML-Entwickler weitergegeben und danach inkrementell erweitert und angepasst. Wenn neue Menüs hinzukamen oder Veränderungen vorgenommen worden sind, wurde eine neue Version des Dokuments verfasst. Das meist geänderte Dokument in Projekt B hatte in einem Zeitraum von ca.16 Monaten 30 Versionen. Im Mittel gab es pro Dokument gerundet 14 Versionen.[1]

7.1.3 Unzureichende Konventionen für die Diagrammerstellung

Das Modell der illustrierten Statecharts sieht lediglich Richtlinien für die Darstellung von Menüzuständen, Transitionen und Funktionsaufrufen vor. Diese orientieren sich an UML-Notationen [HaM03]. Richtlinien zur Gestaltung der skizzierten Darstellungen der Menüs kann das Modell nicht vorgeben, da es für die Spezifikation von verschiedensten HMI anwendbar sein soll. Auch die Erstellung von Diagrammen mit dem Werkzeug VISIO unterliegt keinen Vorgaben. Einem Benutzer ist die Gestaltung hier völlig frei gestellt.

Die Menüs eines HMI sind immer aus elementaren Widgets zusammengesetzt, die über die verschiedenen Menüs und Kontexte hinweg gleich aussehen und sich gleich verhalten sollen. Damit die IML-Entwickler wissen, bei welchen Widgets, die in den illustrierten Statecharts eingezeichnet sind, es sich um die gleichen handelt, müssen alle HMI-Experten eines Projekts die gleichen Notationsformen für die Widgets verwenden. Um dies garantieren zu können, muss deshalb bei Verwendung des Spezifikationsformats der illustrierten Statecharts, ein Richtlinien-Dokument erstellt werden, an dem sich alle orientieren. In den Projekten gab es ein derartiges Dokument, was für Projekt A geschrieben und in Projekt B wiederverwendet wurde. Das Dokument enthielt jedoch keine Konventionen zur Gestaltung der Menüskizzen und der Widgets, sondern lediglich Konventionen zur Darstellung der Zustände, Transitionen und Kommentare.

Die grafischen Darstellungen der Widgets in den illustrierten Statecharts ließen laut den IML-Entwicklern in den Projekten nicht immer eindeutig erkennen, um welche Art von Widget es sich handelte und welche Eigenschaften sie haben sollten. Dadurch war mitunter auch nicht zu erkennen, ob es sich bei einem gezeichneten Widget um ein bereits in der IML-Datenbasis spezifiziertes Widget handelte. Zu Anfang des Projekts B

[1] Gemessen am 26. Januar 2006

sprachen sich die HMI-Experten laut eines HMI-Experten zwar ab, wie die Skizzen gezeichnet werden sollten, indem sie einen von allen zu benutzenden Satz von VISIO-Figuren (engl. *shapes*) festlegten, also Grafiken für z.B. eine Taste oder einen Schieberegler. Jedoch wurde sich nicht durchgängig an die Verwendung dieses Satzes gehalten. So wurden z.B. Tasten nie einheitlich gezeichnet. In einem Statechart-Dokument wurden sie abgerundet in einem anderen rechteckig gezeichnet, obwohl es später als das gleiche IML-Widget umgesetzt wurde und auch werden sollte.

Auch die Benennung von Transitionen auslösenden Ereignissen war nicht einheitlich. So wurde z.B. in einem Dokument auf einem Transitionspfeil die Beschriftung der Taste, die das Ereignis auslöst (z.B. „Ok“), als Bedingung für das Eintreten der Transition angegeben. In einem anderen Dokument hingegen wurde jeweils noch ein Präfix, der den Ereignistyp spezifizieren soll, hinzugefügt, wie z.B. bei „Bt_ok“ für eine Taste namens „Ok“. Teilweise kam es laut eines IML-Entwicklers auch vor, dass sich die Beschriftung einer Taste von einer Version des Statechart-Dokuments zu der nächsten geändert hatte, das zugehörige Ereignis jedoch nicht umbenannt wurde, so dass es für einen Fremden, also jemanden, der weder an der Erstellung des betreffenden illustrierten Statecharts noch an dessen IML-Umsetzung beteiligt war, nicht mehr nachvollziehbar war, wodurch dieses Ereignis ausgelöst werden soll.

Auch die unten beschriebene Versionsgeschichte der Statechart-Dokumente (s. Kapitel 7.1.5) wurde nicht einheitlich verfasst. So wurde in Projekt B z.B. zumeist am Anfang des Dokuments kurz verbal beschrieben, was sich zur vorherigen Version im Dokument verändert hatte. In Dokumenten eines anderen HMI-Experten wurde dies wiederum komplett anders gelöst. Für jede Version wurde eine Farbe vergeben, mit der Veränderungen durch diese Version markiert wurden, so dass Veränderungen z.B. rot für Version X und grün für Version Y umrahmt waren. Bei mehr als acht Versionen wäre dies irgendwann wahrscheinlich zu bunt geworden, weshalb zusätzlich noch verschiedene Muster gewählt wurden: z.B. Wellenlinien oder grüne Linien mit roten Punkten. Der HMI-Experte versuchte dadurch das Problem 7.1.5 zu umgehen.

Insgesamt hatten die fehlenden Konventionen zur Folge, dass die von unterschiedlichen HMI-Experten erstellten Statecharts in ihrer Form mitunter stark voneinander abwichen. Die unterschiedlichen Darstellungsweisen führten in Projekt B dazu, dass IML-Entwickler sich in das Format des jeweiligen HMI-Experten erst einmal einarbeiten mussten, wenn sie zuvor nur illustrierte Statecharts von anderen HMI-Experten in IML umgesetzt hatten. Daraus resultierte also ein Entwicklungsmehraufwand.

Eine mögliche Lösung kann es hier sein, zu Beginn eines Projekts verschärftere Richtlinien für die Diagramm-Erstellung festzulegen. Wie die unterschiedlichen Widgets in den Diagrammen dargestellt werden müssen, könnte z.B. Bestandteil des Widget-Katalogs des jeweiligen Projekts werden.

7.1.4 Interpretationsspielraum

Da die Methode der Erstellung der illustrierten Statecharts, wie oben bereits erläutert wurde, nur eine semiformale Form der Spezifikation ist, sind die Diagramme nicht unbedingt vollständig und eindeutig. Es bleiben Interpretationsspielräume, die hauptsächlich durch die Unschärfe der natürlichsprachigen Texte entstehen, welche unter anderem für die Beschreibung von Funktionalitäten verwendet werden: den Kommentaren. Diese Unschärfe wurde in den Projekten außerdem noch dadurch verstärkt, dass die Beschreibungen auf Englisch waren, geschrieben und interpretiert von zumeist Nichtmuttersprachlern. Dies ist jedoch durch die zunehmende Globalisierung nicht zu vermeiden.

Nicht alle Interpretationsspielräume sind auf die oben beschriebenen, fehlenden Konventionen zurückzuführen. In vielen Fällen wurde von den IML-Entwicklern erwartet,

dass sie Informationen über bestimmte Funktionalitäten z.B. aus einer Tastenbeschriftung implizieren. Es wurde nicht explizit notiert, welche Funktionalität ausgeführt oder was für ein Zustandswechsel ausgelöst werden soll. Mit einem solchen Problem konfrontiert meinte ein HMI-Experten, dass er nicht für eine vollständige Spezifikation zuständig sei. An vielen Stellen ließen die HMI-Experten wichtige Details weg, weil sie dachten, dass es klar wäre, was dort passieren müsse. So wurde z.B. in der unten beschriebenen Durchsichtsstudie (s. Kapitel 10) ein Defekt entdeckt, der auf folgendes zurückzuführen war: Es wurde vergessen, bei dem Verlassen eines Menüs eine Gerätekomponentenfunktion ausführen zu lassen. Diese Funktionsausführung war jedoch auch nicht in den Statecharts angegeben. In einem Gespräch zwischen dem HMI-Experten und dem IML-Entwickler - während der gemeinsamen Sitzung einer Paardurchsicht - stellte sich heraus, dass diese Auslassung nicht dadurch entstand, dass der HMI-Experte vergessen hatte, sie anzugeben, sondern er war der Meinung, dass es selbstverständlich sei, dass die Ausführung der Funktion bei dem Verlassen des Menüs stattfinden müsse. Deshalb habe er sie gar nicht erst notiert.

Noch ein weiterer Punkt führte zu Interpretationsspielräumen in den illustrierten Statecharts. Transitionen eines Menüs (vor allem Menüsprünge), die durch ein Tastendruck-Ereignis ausgelöst werden sollten, bekamen zumeist nur eine Beschriftung wie z.B. „Bt_Ok“. Die Tasten bekamen (und bekommen) in illustrierten Statecharts jedoch keine Namen, sondern die IML-Entwickler mussten aus den Beschriftungen der Tasten ablesen, welches Tastendruck-Ereignis gemeint war. Bei einem Ereignis „Bt_Ok“ auf eine Taste zu schließen, die mit „Ok“ beschriftet ist, ist dabei sicherlich noch machbar. Wenn jedoch das Textfeld einer Taste mit einer dynamischen Beschriftung belegt wird, kann es mitunter schwieriger werden, wie z.B. bei einem in Projekt B notwendigen Schluss vom Ereignis „Band“ auf die Taste mit der Beschriftung „FM“. Ein Experte ist gefragt, wenn die Taste, wie des Öfteren in Projekt B, gar keine Beschriftung sondern ein Symbol in den illustrierten Statecharts erhält. Dann ist z.B. von einer Taste bei der fünf waagerechte Linien eingezeichnet sind auf ein Ereignis „BT_text“ zu schliessen.

Die IML-Entwickler mussten durch den Interpretationsspielraum „immer wieder [bei den betreffenden HMI-Experten] nachfragen, wie was gemeint ist“. Das heißt der Kommunikationsaufwand zwischen den beiden Rollen war dadurch relativ groß.

Ein weiteres Problem, was hier auftreten kann, ist, dass den IML-Entwicklern nicht unbedingt auffällt, dass ein Interpretationsspielraum gegeben ist. Sie lösen also die Aufgabe aus ihrer Sicht. Erst wenn die HMI-Experten, welche die Statecharts entworfen haben, das Endergebnis in Form eines Prototypen sehen, wird bemerkt, dass etwas missinterpretiert wurde. Dieses Problem ist um einiges gravierender, da es zum einen zu unnötigen Mehraufwand bei der IML-Spezifikation führt und zum anderen die Gefahr birgt, dass derlei Defekte erst sehr spät oder erst vom Benutzer bemerkt werden, was eine Defektbehebung bekanntlich [Boe86, S. 34f] teurer macht.

Wie sich bei Befragungen zu späteren Defektkosten- und Defektklassenanalysen herausstellte, betrifft das Problem der Interpretationsspielräume nicht nur die Umsetzung nach IML. Auch die Tester sind davon betroffen. Nach Aussage eines IAV-Mitarbeiters trat in Projekt A des Öfteren folgendes Problem auf. Die Tester berichteten nach einem Abgleich von Menüs eines Gerätemusters mit den zugehörigen illustrierten Statecharts über Defekte, bei denen sich letztlich herausstellte, dass sie die illustrierten Statecharts missinterpretiert hatten. Das heißt nicht nur bei der IML-Umsetzung verursachen die Interpretationsspielräume einen Mehraufwand, sondern zusätzlich auch in der Testphase. Hier ist der Aufwand sogar noch höher, da der Kommunikationsweg länger ist.

7.1.5 Unzureichende Versionsgeschichte

Den Statechart-Dokumenten wurde eine Versionsgeschichte beigefügt, in der zumeist stichpunktartig verbal festgehalten wurde, was sich von Version zu Version des Dokuments geändert hatte. Das Format war dabei nicht einheitlich, worauf weiter oben (7.1.3) bereits eingegangen wurde. Die Auflistung der Veränderungen war auch nicht immer vollständig. Wenn z.B. in einem Menü lediglich eine Beschriftung angepasst wurde, dann wurde die Veränderung laut den HMI-Experten nicht immer festgehalten.

Das dadurch entstandene Problem war, dass die IML-Entwickler in einer unnötigen Phase die neueren Versionen der Statechart-Dokumente noch einmal von Anfang bis Ende durchsehen mussten, um die Veränderungen zur vorherigen Version festzustellen. Diese Arbeit war laut den IML-Entwicklern zeitaufwändig und unmotivierend. Der Grund dafür ist, dass die Dokumente zum einen mitunter sehr umfangreich waren - teilweise über hundert Seiten lang - und zum anderen PDF-Dokumente nicht mit Werkzeugen, die verschiedene Versionen von Dokumenten miteinander vergleichen und daraufhin Unterschiede aufzeigen, bearbeitet werden können. Zumindest kannte das Projektteam kein derartiges Werkzeug. Die Vergleiche mussten also manuell von den IML-Entwicklern vorgenommen werden. Eine Frage, die hier aufkommt, ist, ob PDF ein geeignetes Austauschformat für die illustrierten Statecharts darstellt.

Durch die hier beschriebenen Probleme mit der Versionsgeschichte der Statechart-Dokumente kam es vor allem in Projekt B mitunter zu Unstimmigkeiten zwischen HMI-Experten und IML-Entwicklern.

7.1.6 Lösungsvorschläge

Verstärkte Kooperation: Die hier aufgeführten Probleme resultieren meiner Meinung nach zu einem großen Teil daraus, dass mit den HMI-Experten und den IML-Entwicklern zwei anders geartete Welten aufeinander treffen: die technische und die ergonomische. Beide Rollen können sich nicht vollständig in die Arbeit der jeweils anderen Rolle hineinversetzen. Um dem zu begegnen, ist es vorstellbar, die Kooperation zwischen den beiden Rollen gezielt zu verstärken. Schon jetzt ist es natürlich so, dass die IML-Entwickler die auftretenden Probleme nicht alleine lösen. Sobald sie ein Problem bemerken, fragen sie bei dem Verfasser der illustrierten Statecharts nach, wie dieser sich die Umsetzung vorgestellt hat. Diese Form der Kooperation ist unkoordiniert, recht zeitaufwändig für beide Seiten und in der Praxis nicht immer machbar, da sich in manchen Projekten die HMI-Experten und die IML-Entwickler nicht am selben Ort befinden.

Eine bessere Form der Kooperation bieten die von mir im nächsten Kapitel vorgeschlagenen rollenkooperativen Durchsichten der IML-Menüs. Diese verursachen zwar kurzfristig einen Mehraufwand, jedoch konnte gezeigt werden, dass Durchsichten sich langfristig auszahlen [Gra97, S. 66]. Es wäre außerdem vorstellbar, dass die illustrierten Statecharts zusammen von den HMI-Experten und den IML-Entwicklern vor der betreffenden IML-Entwicklung durchgesehen werden. Dabei müsste der HMI-Experte dem IML-Entwickler erklären, wie er sich die Umsetzung vorstellt. Dadurch könnte das „ständige Nachfragen“ während der Entwicklung reduziert werden.

Strengere Konventionen: Wie sich oben gezeigt hat, führten die unzureichenden Konventionen für die Erstellung der illustrierten Statecharts zu einigen Problemen in den untersuchten Projekten. Auch hat sich gezeigt, dass selbst, wenn Richtlinien, z.B. in Form eines Word-Dokuments, aufgestellt werden, diese nicht unbedingt befolgt werden, zumeist sicherlich unbewusst. Schließlich kann auch ein HMI-Experte Fehler machen.

Um dem entgegen zu wirken, muss bereits bei der Entwicklung der Menüdiagramme darauf geachtet werden, dass Defekte nicht auftreten können. Es ist vorstellbar, eine Art automatisches Testwerkzeug für Menüdiagramme zu entwickeln. Dazu muss jedoch das Modell, welches den Menüdiagrammen zugrunde liegt, strengere Richtlinien befolgen als das Modell der illustrierten Statecharts.

In einer folgenden Diplomarbeit, die in Kooperation mit der IAV entstehen wird, wird mit Hilfe der Metasprache MOF (Initialwort für *Model Object Facility*) versucht, ein entsprechendes Modell aufzustellen. MOF ist ein offener Standard der *Object Management Group* (OMG) und stellt ein Metamodell für die UML dar. Es „beschreibt eine Semantik zur Formulierung von Daten, Modellen und Metamodellen“ [GiS05].

Mit einem Menüdiagrammeditor, der nur die Erstellung von Diagrammen erlaubt, welche die Regeln eines derart formalen Menüdiagrammmodells befolgt, kann auch ein weiterer Schritt in Richtung Automatisierung gegangen werden. Es könnten zumindest Teile des IML-Codes, wie z.B. die Menüübergänge, automatisch erzeugt werden. Die IAV plant einen entsprechenden Editor im HMIStudio zu integrieren.

7.2 Fehlende Konventionen für die IML-Entwicklung

Das oben (s. Kapitel 5.4) vorgestellte HMIStudio wurde in den untersuchten Projekten nicht eingesetzt, da es laut IAV-Mitarbeitern zu spät einsatzbereit war und eine diesbezügliche Anpassung des Prozesses mitten im Projekt aufgrund der strengen Zeitvorgaben nicht praktikabel gewesen wäre. Deshalb verwendeten die IML-Entwickler für die Spezifizierung der HMI den XMLSPY. Dieses Werkzeug von Altova hat gegenüber einem Texteditor den Vorteil, dass der XML-Code nicht zeichenweise eingegeben werden muss, sondern Sprachkonstrukte, in diesem Fall XML-Elemente mit zugehörigen Attributen, per Maus hinzugefügt und verschoben werden können und dass der XML-Code in einer für den Menschen genießbareren Form grafisch dargestellt wird.

Die Darstellung des XML-Codes auf dieser grafischen Metaebene hat außerdem den Vorteil, dass der Code, egal von wem er erstellt wurde, immer den selben Kodierrichtlinien entspricht, da das Programm die eigentliche Codeerstellung übernimmt. Vorraussetzung dafür ist lediglich, dass alle Entwickler den XMLSPY gleich konfiguriert haben, was vor allem Konventionen zur Einrückung des Codes betrifft. Auf entsprechende Code-Konventionen wurde sich unter den Entwicklern in den Projekten geeinigt. Der Prozess schreibt diesbezüglich nichts vor.

Außerdem gab es bei der IML-Entwicklung Konventionen, die der Zulieferer vorgegeben hatte. So durften im Projekt B Aufrufe von Gerätekomponentenfunktionen z.B. immer nur in Menü-Widgets stattfinden, obwohl die Sprache IML hierzu keine Vorgaben macht.

Was jedoch meiner Meinung nach fehlte, waren Konventionen für das Widget-Design in IML. Es gab keine Vorgaben darüber, aus welcher Art von Widgets sich ein Menü zusammensetzen muss. Das führte dazu, dass Menüs, die vom Aufbau her relativ ähnlich waren, auf völlig unterschiedliche Weise von verschiedenen IML-Entwicklern spezifiziert wurden.

Auch für die Überschreibung von Eigenschaften gab es keine Richtlinien. Angenommen ein IML-Entwickler muss in einem Menü den Inhalt eines Textfelds spezifizieren. Außerdem sei dieses Menü in einer siebenstufigen Widget-Hierarchie aufgebaut, wobei sich das Textfeld als Basis-Widget auf letzter bzw. Blatt-Ebene befindet. In diesem Fall ist es einem IML-Entwickler theoretisch freigestellt, auf welcher der sieben Ebenen er die Eigenschaft Textfeld-Inhalt spezifiziert. Von dieser Freiheit machten die Entwickler in den Projekten mitunter auch Gebrauch.

Aus den fehlenden Richtlinien für die IML-Entwicklung resultierten mindestens zwei Probleme:

1. Zum einen war es einem codefremden IML-Entwickler, also einem Entwickler der an der Erstellung des betreffenden IML-Codes nicht beteiligt war, selbst mit Erfahrung auf Anhieb nicht möglich herauszufinden, in welchem Widget welche Eigenschaft gesetzt wurde. Er musste eine in Projekt B bis zu 14-stufige Hierarchie[1] (vom Menü-Widget ausgehend) nach dieser Eigenschaft absuchen - das heißt bis zu 14 Dateien, die bis zu 2,15 MB groß[1] waren, durchsuchen. Hingegen hätte sich der Entwickler, der den Code erstellt hat, vielleicht noch an die Stelle erinnern können. Dieses Problem machte sich vor allem bei der Defektbehebung bemerkbar, da die Entwickler dabei auch den nichteigenen Code bearbeiten mussten. Auch wenn ein Entwickler einmal wegen Krankheit oder Urlaub ausfiel, trat dieses Problem auf.
2. Zum anderen wird durch das Fehlen entsprechender Richtlinien die Wiederverwendung von Widgets erschwert. Es wurde in den Projekten nicht darauf geachtet, die Widgets unterhalb der Menüebene so generisch wie möglich zu halten. Wenn ein neues Menü erstellt werden sollte, das einem anderen ähnelte, wurde dies zumeist so gelöst, dass ein Großteil der Hierarchie kopiert und umbenannt wurde. Dies geschah laut den IML-Entwicklern, weil es (zunächst einmal) schneller geht. Dadurch wurde jedoch z.B. die Datenbasis in Projekt B unnötig aufgebläht. Nur 173 von insgesamt 1595[1] in der Datenbasis enthaltenen Widgets, also ca. 11% wurden mehr als einmal verwendet und folglich wiederverwendet. Langfristig gesehen sollte die Zeitersparnis, die aus einer erhöhten Generalisierung aufgrund von geringeren Wartungsarbeiten resultiert, höher als die kurzfristige Zeitersparnis sein.

Die Probleme würden auch beim Einsatz des HMIStudios erhalten bleiben, da es auch dort möglich ist, die Widget-Strukturen beliebig anzulegen. Meine These ist außerdem: Je mehr Widgets in einer IML-Datenbasis wiederverwendet werden, desto einfacher gestaltet sich der Bau einer anderen Variante aus dieser Datenbasis – z.B. für einen anderen Markt. Wenn z.B. Spezifika des Aussehens von fünf sich ähnelnden Menüs in fünf verschiedenen Dateien spezifiziert sind, so ist der Aufwand für das Anpassen dieser Spezifika für eine andere Variante in jedem Fall höher, als wenn diese generisch in einem Widget unterhalb der Menüebene definiert sind.

Eine Lösung für dieses Problem könnte es sein, Richtlinien für die IML-Entwicklung aufzustellen, die sich an Regeln zum Entwurf von wiederverwendbaren Klassen ausrichten, wie sie für die objektorientierte Programmierung aufgestellt wurden (s. z.B. in [JF88]). Dies impliziert jedoch auch, dass die Benutzer des HMI-Studios später Kenntnisse in Objektorientierung besitzen müssten. Es sei denn, das Werkzeug bringt die Fähigkeit mit, die Daten von selbst generisch zu strukturieren.

Bei Hewlett-Packard konnte Grady zeigen, dass durch erhöhte Wiederverwendung die Zeit bis zur Markteinführung eines Produkts (engl. *time to market*) erheblich verkürzt werden konnte [Gra97, S. 179]. Die Einführung eines GQM-Programms mit dem Ziel der erhöhten Wiederverwendung von Widgets besäße also auch ein übergeordnetes Geschäftsziel.

[1] Gemessen am 23.01.06 (Hauptvariante)

7.3 Zu spätes und unzureichendes Verifizieren

Die IML-Datenbasis wurde in den untersuchten Projekten immer zu bestimmten Meilensteinen an den Zulieferer und an den Fahrzeughersteller geschickt. Da der Zeitplan sehr eng gesteckt war, mussten kurz vor den Abgaben meistens „schnell noch mehrere Menüs“ erstellt werden. Für größere Tests und Durchsichten war dadurch oftmals keine Zeit vorhanden. Defekte wurden dadurch mitunter erst beim Zulieferer entdeckt. Im Projekt A wurden erst in einer sehr späten Phase des Projekts, als schon eine erste Betaversion der Spezifikation fertiggestellt war, intensivere Einzeldurchsichten (s. Kapitel 8.3) vorgenommen.

Das Problem dabei ist, dass bekanntermaßen die Defektbehebung umso teurer wird, je später im Prozess ein Defekt entdeckt wird (vgl. [Boe86, S. 34 f]). Wenn ein Defekt erst mehrere Wochen nach der Erstellung des betreffenden Menüs gefunden wird, so muss sich der IML-Entwickler, der für das Menü verantwortlich ist, erst einmal wieder in den IML-Code einlesen, um ihn erneut zu verstehen und den Defekt dann beheben zu können. Würde der Defekt früher aufgedeckt werden, entfiele dieser Aufwand.

Vor den Abgaben wurde zwar die Spezifikation zumeist getestet, jedoch nie vollständig, da dies laut eines IAV-Mitarbeiters zu aufwändig gewesen wäre.

Die Tests wurden von den IML-Autoren selber getätigt. Eine Prüfung hinsichtlich der oben beschriebenen Interpretationsspielräume (s. Kapitel 7.1.4) wurde nicht vorgenommen. Meines Erachtens nach ist es aber zwingend erforderlich, dass der HMI-Experte, der die Menüs entworfen hat, auch später überprüft, ob die IML-Spezifikation seinen Vorstellungen entspricht. Aufgrund des semiformalen Formats der illustrierten Statecharts und der damit verbundenen Interpretationsspielräume kann nur er dies feststellen.

Das TeleDrive VISION Prozessmodell sieht eine entwicklungsbegleitende Prüfung der Prozesszwischenprodukte bislang noch nicht vor. Aber: „Zu jeder Phase des Softwareprozesses gehören auch Aktivitäten der Verifikation und Validierung." [Som01 , S. 427]

Zur Lösung der in diesem Unterkapitel beschriebenen Probleme habe ich mich eingehender mit manuellen Prüfmethoden befasst und eine auf den Prozess zugeschnittene rollenkooperative Durchsichtsform konzipiert. Diese wird im nachfolgenden Kapitel erläutert.

7.4 Benutzbarkeit der Simulation

Die in diesem Unterkapitel beschriebenen Probleme identifizierte ich bei der Konzipierung der im nächsten Kapitel beschriebenen Paardurchsicht.

7.4.1 Für Nichtinformatiker

Die Motivation für den Bau einer Simulation des HMI im TeleDrive VISION Prozess liegt nicht alleine darin, dass die IML-Entwickler mit Hilfe der Simulation die IML-Datenbasis besser verifizieren können. Vielmehr sollte die Simulation auch für Validierungen durch Prozessbeteiligte geeignet sein, deren Informatikkenntnisse begrenzt sind, wie im Zweifel z.B. dem Management eines auftraggebenden Fahrzeugherstellers. Auch für die HMI-Experten kann eine Simulation ein geeignetes Werkzeug zur Unterstützung ihrer Arbeit sein.

In den untersuchten Projekten war eine Verwendung der Simulation im vollen Umfang jedoch nur den IML-Entwicklern und den am Bau der Simulation beteiligten Softwareentwicklern möglich, da spezielle Kenntnisse für die Konfigurierung der Simulation benötigt wurden.

Wenn jemand mit der Simulation des Projekts B z.B. ein Menü begutachten wollte, dass nur angezeigt wurde, falls eine DVD im internen DVD-Laufwerk eingelegt war, so musste, da es sich nur um eine Simulation handelte und das Programm dies deshalb nicht wirklich überprüfte, eine XML-Konfigurationsdatei angepasst werden, um einen entsprechenden Wert zu verändern. Wo sich diese Datei befand und wie diese verändert werden musste, wussten im Zweifel nur die Entwickler. Bei der Konzipierung der Durchsichtsstudie (s. Kapitel 10) stellte sich zumindest heraus, dass die HMI-Experten dies nicht wussten.

Durch diese Konfigurationsmethoden war die Benutzbarkeit der Simulation beschränkt, zumal auch nur vereinzelt Dokumentationen hierüber existierten. Aber selbst wenn ausreichend Dokumentationen vorhanden gewesen wären, wäre eine eingeschränkte Benutzbarkeit geblieben, da ein Benutzer sich im Zweifel nicht mit der Konfigurierung von XML-Dateien beschäftigen will.

Eine Abhilfe könnte es hier sein, dem Benutzer während der laufenden Simulation ein geeignetes Einstellungsmenü zur Verfügung zu stellen. Falls es technisch zu aufwändig wäre, die Einstellungen zur Laufzeit der Simulation zu ermöglichen, wäre ein Konfigurationswerkzeug vorstellbar, dass der Benutzer vor dem Starten der Simulation ausführen kann. Entsprechende Ansätze sind mit den Werkzeugen *TestContainer* und *RepWatcher* schon in der IAV vorhanden.

Eine weitere Einschränkung der Simulation in Bezug auf ihre Benutzbarkeit rührte daher, dass in den untersuchten Projekten nicht immer alles simuliert wurde. In dem Projekt A wurde bis auf kleine Details sowohl das Verhalten als auch das Aussehen des HMI vollständig simuliert. Das Projekt B war jedoch um einiges umfangreicher (s. Kapitel 1.3).[1] Deshalb beschränkte man sich hier darauf, das Aussehen komplett zu simulieren und von dem Verhalten nur so viel zu simulieren, wie es der enge zeitliche Rahmen des Projektes zu ließ. An einigen Stellen wäre der Aufwand für eine vollständige Simulation nach Aussage eines Projektleiters so hoch gewesen, dass man auch gleich das ganze Gerät hätte bauen können. So beschränkte sich z.B. die Simulation der Darstellung einer Landkarte für den Navigationskontext darauf, dass immer das selbe Bild angezeigt wurde. Bei Auswahl eines POI (Initialwort für *Points Of Interest*, s. 2.1) in der Karte wurde immer das gleiche Ereignis ausgelöst, so dass verschiedene mögliche Fälle, wie z.B. der Druck auf eine in der Karte symbolisierte Tankstelle nicht simulierbar waren.

Es wurden auch viele Listen mit dynamischen Daten, wie z.B. eine Liste von favorisierten Radiosendern, in den Simulationskomponenten mit statischen Einträgen gefüllt, was zur Folge hatte, dass nach dem Löschen eines Listeneintrags in einem Menü des HMI der Eintrag noch immer in der Liste vorhanden war. Wenn also z.B. ein HMI-Experte ein solches Menü verifizierte, wusste er nicht, ob es sich dabei um einen Defekt in der Spezifikation handelte oder lediglich um eine unvollständige Simulation.

7.4.2 Für IML-Entwickler

Die Benutzbarkeit der Simulation war auch für die Arbeit der IML-Entwickler eingeschränkt. Sie verwendeten die Simulation hauptsächlich zur Verifizierung der von ihnen erstellten IML-Dateien. Um dabei Defekte aufzudecken und um deren Ursache auf den Grund zu gehen, benutzen sie einen auf die Simulationssoftware zugeschnitten Ablaufverfolger (engl. *tracer*), das sogenannte *TraceTool*. Dieses Werkzeug listet nebenläufig

[1] Projekt A: 579 verschiedene Widgets (319 Menüs), Projekt B: 1323 verschiedene Widgets (476 Menüs) (Stand: 29.9.2005), wobei die einzelnen Widgets in Projekt B im Schnitt umfangreicher waren und Verhalten von Widgets in Projekt A nicht in IML spezifiziert wurde.

zur Simulation Meldungen auf, die von der Simulationssoftware zur Laufzeit generiert werden.

Darunter befinden sich neben Meldungen, die aufgrund einer defektbehafteten IML-Spezifikation erzeugt werden, auch Meldungen, die Defekte in der Simulationssoftware signalisieren, wie z.B. eine Ausnahme (engl. *exception*) in einer Funktion einer Simulationskomponente. Auch in der IML-Datenbasis spezifizierte Ereignisse, wie z.B. ein Zustandswechsel eines Widgets oder ein Funktionsaufruf in einem Widget erzeugen Meldungen im *TraceTool*.

Zum Zeitpunkt der Konzipierung der Paardurchsichtsstudie waren diese Meldungen nicht in jedem Fall von einander trennbar. Es war den IML-Entwicklern nicht möglich, sich nur Meldungen über IML-Defekte anzeigen zu lassen. Sie mussten bei der Begutachtung der Ausgabe des *TraceTools* immer die Spreu vom Weizen trennen. Da ich bei der Erstellung der Durchsichtsanleitung nicht dieses selektive Lesen erklären wollte, setzte ich mich mit einem Entwickler der Simulationssoftware für ca. eine halbe Stunde zusammen und wir behoben dieses Problem. Diese Maßnahme erwies sich als sehr effizient. Wir legten dabei eine Klassifizierung für die verschiedenen Meldungstypen an. Da das *TraceTool* Einstellungsmöglichkeiten anbietet, mit denen das Aufzeichnen und die Anzeige einzelner Meldungsklassen an- und abschaltbar sind, war es nun möglich, sich nur „echte“ Defekte der IML-Spezifikation anzeigen zu lassen.

8 Manuelle Prüfmethoden

Um dem identifizierten Problem 7.3 des unzureichenden Verifizierens zu begegnen, entschied ich mich dazu, eine Form von manueller Prüfmethode zu konzipieren. Mit den nachfolgenden Untersuchung werden die Themen Kooperationsaspekte und Qualitätskriterien in Verbund mit Messungen vereinigt.

In diesem Kapitel werden zunächst Grundlagen zu manuellen Prüfmethoden und deren Aufwand und Nutzen erläutert. Darauf folgt eine Diskussion über die Handhabung der Durchsichten, wie sie in den untersuchten Projekten vorgenommen wurden. Das Kapitel schließt mit einer detaillierten Erläuterung der von mir konzipierten rollenkooperativen Paardurchsicht ab.

8.1 Grundlagen

Um eine hohe Qualität des zu erzeugenden Produktes sicherzustellen, sollten in einem Softwareentwicklungsprozess die Ergebnisse der einzelnen Prozessaktivitäten durch Verifizierung und Validierung überprüft werden. Eine Validierung geht dabei der Frage nach, ob ein Produkt die Erwartungen der Kunden erfüllt, also ob das richtige Produkt erstellt wird. Eine Verifizierung hingegen beantwortet die Frage, ob ein Produkt richtig erstellt wird, das heißt ob ein Produkt seine Spezifikation erfüllt. [Som01, S. 427 ff.]

Nicht alle Produkt- und Prozesseigenschaften können automatisch überprüft werden. Vor allem eine Validierung oder eine Verifizierung semantischer Aspekte ist oftmals nicht automatisch möglich. So ist es z.B. nicht realisierbar, mit einem Werkzeug zu verifizieren, ob eine Software ihre in einem natürlichsprachigen Anforderungsdokument verfasste Spezifikation erfüllt. In diesen Fällen bietet es sich an, die Qualität durch manuelle Prüfmethoden zu untersuchen.

Es existieren unterschiedliche manuelle Prüfmethoden, die in den verschiedenen Bereichen und Phasen des Lebenszyklus eines Softwareentwicklungsprozesses einsetzbar sind. Balzert unterscheidet in [Bal98, S. 302 ff.] in folgende wesentliche Methoden: das Durchgehen (engl. *walkthrough*), die technische Durchsicht[1] (engl. *techincal review*) und die Inspektion. Der Formalisierungsgrad der Methoden nimmt in der Reihenfolge dieser Aufzählung zu. Da eine Abgrenzung zwischen den verschiedenen Methoden in der Literatur nicht einheitlich ist, verwende ich den Begriff der Durchsicht im Folgenden als Oberbegriff.

Bei einer Durchsicht wird das zu untersuchende Dokument von einer oder mehreren Personen mit dem Ziel analysiert, Mängel aufzudecken. Sie ist theoretisch für alle Arten von Zwischen- und Endprodukten eines Prozesses einsetzbar. Für die Organisation und den Ablauf von Durchsichten sind verschiedenste Formen und Formalisierungsgrade bekannt. Eine Durchsicht kann von rein informell und ungeplant bis hinzu durchstrukturiert und detailliert geplant ablaufen. Auch die Anzahl der Beteiligten und ob diese verschiedene Rollen bei der Durchsicht einnehmen sollen, ist nicht fest vorgegeben, sondern ist auf den jeweiligen Kontext, in dem die Durchsicht durchgeführt werden soll, und den Einsatzzweck abzustimmen. [Lai02]

Bekanntermaßen werden die Kosten, die für das Beheben eines Defekts entstehen, umso höher, je später der Defekt während des Entwicklungsprozesses gefunden wird. Nach- und Umbauarbeiten werden billiger, je früher sie getätigt werden. [Gra97, S. 66] Deshalb sollten Durchsichten vor allem auch für Zwischenprodukte, die in frühen Prozessschritten entstehen, durchgeführt werden, und nicht nur für Quelltexte. In einer Me-

[1] Balzert verwendet hier nur den Begriff „R*eview*“, erwähnt aber auch, dass er ihn im Sinne der „technischen *Reviews*“ gebraucht.

tastudie von Briand et al. [Bri98] konnte nachgewiesen werden, dass Durchsichten in den frühen Phasen besonders kosteneffektiv sind.

Um eine größtmögliche Effektivität von Durchsichten zu erzielen, sollten diese in den Softwareentwicklungsprozess durchgehend integriert sein [Lai02]. Eine vereinfachte Darstellung eines an das V-Modell angelehnten Prozessmodells, in dem eine solche Integration angedeutet wird, zeigt die Abbildung 28. Für den Spezifizierungsprozess des TeleDrive VISION Prozesses bedeutet dies unter anderem, dass die illustrierten Statecharts gegen die im Pflichtenheft formulierten Anforderungen und die IML-Datenbasis gegen die illustrierten Statecharts inspiziert werden müssten.

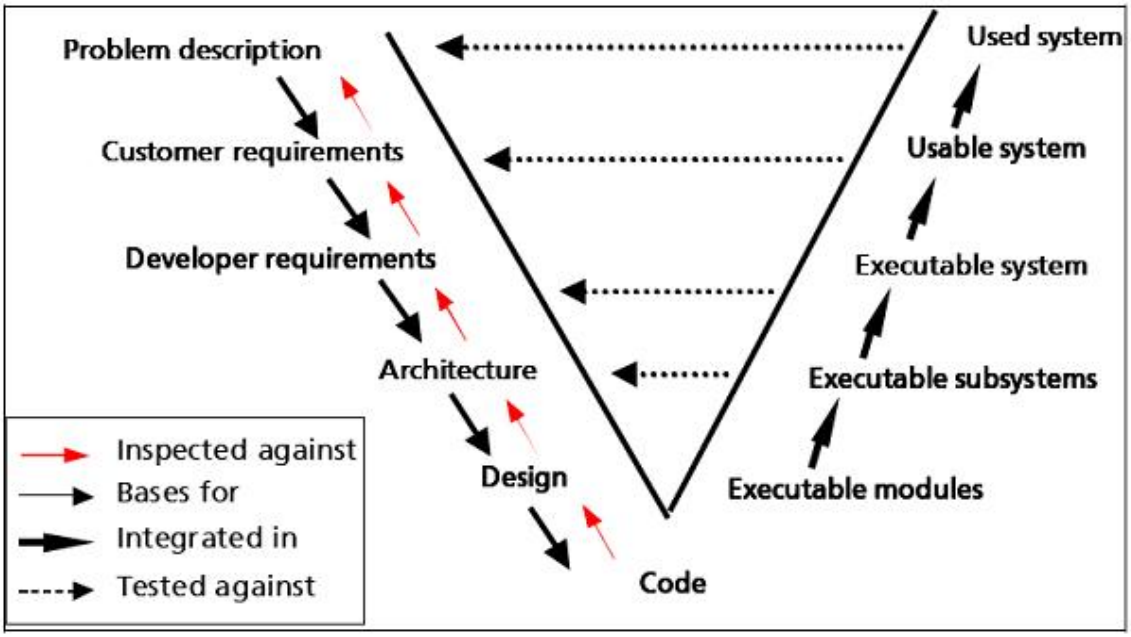

Abbildung 28: Durchgehende Integration von Durchsichten in das Schema des V-Modells [Lai02]

Eine Integrierung von kontinuierlichen Durchsichten in den TeleDrive VISION Prozess bietet mehrere Vorteile:

1. Es ist möglich, auf entdeckte Defektquellen frühzeitig zu reagieren und den Entwicklungsprozess daraufhin so anzupassen, dass derartige Defekte vermieden werden. Das heißt die Prozessbeteiligten können aus ihren Fehlern lernen und der Prozess kann kontinuierlich verbessert werden.
2. Eine Durchsicht eines kompletten Standes eines Prozesszwischenprodukts, wie z.B. der IML-Datenbasis, ist wahrscheinlich weniger effektiv, da Ermüdungserscheinungen und geringere Motivation bei den Beteiligten zu erwarten sind, als wenn die Durchsichten kontinuierlich z.B. einmal im Monat oder am Ende jeder Iteration vorgenommen werden würden.
3. Das Ansehen der für den betreffenden Prozessteilschritt zuständigen Firma bei anderen am Prozess beteiligten Firmen wird dadurch gestärkt, dass etwaige Defekte frühzeitig im eigenen Betrieb aufgedeckt werden und nicht z.B. erst beim Zulieferer.

8.2 Aufwand und Nutzen von Durchsichten

Da es sich bei Durchsichten um eine von Personen ausgeführte Tätigkeit handelt, lässt sich der dafür eingesetzte Aufwand in Personenstunden angeben. Die bei Durchsichten entstehenden materiellen Aufwänden sind im Allgemeinen vernachlässigbar gering.

Durch eine Einführung von Durchsichten entsteht ohne Zweifel kurzfristig zunächst einmal ein Mehraufwand. Dieser zahlt sich jedoch nach Grady langfristig aus [Gra97, S. 66] (engl.: *return on investment (ROI)*).

Studien haben gezeigt, dass der Aufwand für das Finden von Defekten mit Hilfe von Durchsichten und dem nachfolgenden Beheben dieser Defekte um einiges geringer ist, als der Aufwand für das Finden und Beheben der gleichen Defekte in einer späteren Phase des Entwicklungsprozesses, wie z.B. dem formalen Testen. In Studien von Kaner, Remus und Kan wurden für dieses Verhältnis Werte zwischen 1:10 und 1:34 gemessen [Lai02]. In von Grady bei Hewlett-Packard untersuchten Studien [Gra97, S. 250] wurde gezeigt, dass durch die Einführung von Design-Inspektionen aus diesem Grund auch die Zeit bis zur Markteinführung (engl.: *time-to-market*) eines Produktes erheblich verringert werden kann.

Der Gesamtaufwand für die Durchführung von Durchsichten in einem Projekt wurde in der Literatur bisher nur wenig betrachtet [Lai02], obwohl gerade diese Information wichtig für projektleitende Mitarbeiter sein kann, um zu entscheiden ob und wie Durchsichten in einem Projekt durchgeführt werden sollen.

Der Hauptnutzen von Durchsichten ist die oben erwähnte Effizienzsteigerung des Findens und Behebens von Defekten. Da in der Fachliteratur zum Thema Durchsichten die Semantik der Begriffe Effizienz und Effektivität nicht einheitlich ist, definiere ich kurz, was ich im Folgenden darunter verstehe, wobei ich den Aufwand für die Behebung der entdeckten Defekte nicht berücksichtige.

Nach der Norm ISO 9000 ist Effizienz definiert als das „Verhältnis zwischen dem erreichten Ergebnis und den eingesetzten Ressourcen“ [DIN00, S. 22]. Die Effizienz des Findens von Defekten einer Durchsicht D lässt sich also aus dem Verhältnis zwischen der Anzahl, der durch D gefundenen Defekte und dem dafür eingesetzten Aufwand berechnen:

$$\text{Effizienz}(D) = \#\text{Defekte}(D) / \text{Aufwand}(D)$$

Effektivität gibt den Grad der Wirksamkeit einer Tätigkeit an. In der Norm ISO 9000 wird Wirksamkeit als das „Ausmaß, in dem geplante Tätigkeiten verwirklicht und geplante Ergebnisse erreicht werden" [DIN00, S. 22] definiert. Die Effektivität einer Durchsicht eines Prüfgegenstandes mit dem Ziel des Aufdeckens von Defekten lässt sich also aus dem Verhältnis von der Anzahl der Defekte, die während der Durchsicht gefunden werden, zu der Anzahl der in dem Prüfgegenstand insgesamt vorhandenen Defekte bestimmen. Da die Bestimmung der Gesamtanzahl von Defekten in einem Prüfgegenstand ein nicht berechenbares und nicht messbares Problem darstellt, wird dieses Verhältnis in der Literatur zumeist abgeschwächt formuliert, als das Verhältnis von der Anzahl der während einer Durchsicht D gefundenen Defekte zu der Anzahl der Defekte, die während der Durchsicht und dem Testen T zusammen gefunden werden:

$$\text{Effektivität}(D) = \#\text{Defekte}(D) / (\#\text{Defekte}(D) + \#\text{Defekte}(T))$$

Diese Art der Effektivitätsberechnung entspricht auch der von Humphrey vorgeschlagenen Berechnung des Durchsichtsertrags (engl. *review yield*) [Hum95, S. 248]. Grady stellte bei den von ihm untersuchten Studien laut [Lai02] eine durchschnittliche Effekti-

vität von 55 Prozent bei Entwurfsinspektionen und 60 Prozent bei Quelltextinspektionen fest.

Der jeweilige Aufwand der beiden Prüfmethoden findet bei der Effektivitätsberechnung keine Berücksichtigung. Da davon ausgegangen wird, dass das frühzeitige Beheben von Defekten in jedem Fall billiger ist (vgl. [Boe86, S. 34f]), impliziert hier eine durch Veränderung der Durchsicht erzielte höhere Effektivität folglich auch eine höhere Effizienz. Sowohl bei der Effizienzberechnung als auch bei der Effektivitätsberechnung finden hier noch keine Gewichtungen der Defekte bezüglich ihres Schweregrades statt. Die Formeln könnten aber nachträglich für diese Berücksichtigung erweitert werden.

Neben einer Qualitätsverbesserung des erstellten Produkts und einer Kostensenkung für die Erstellung können mit Durchsichten auch andere zumeist nicht quantitativ messbare Nutzen erzielt werden. So verbreitert sich z.B. die Wissensbasis, der an einer Durchsicht teilnehmenden Mitarbeiter [Bal98]. Sie lernen Arbeitsmethoden von Kollegen kennen und erfahren, an welchen Stellen häufiger Defekte auftreten, also an welchen Stellen sie bei der Erstellung besonders Acht geben müssen. Nützlich kann es auch sein, dass durch die Durchsichten die Kommunikation der Mitarbeiter untereinander steigt.

8.3 Momentane Handhabung der Durchsichten

In den Projekten war zum Zeitpunkt meiner Untersuchungen eine Verifizierung der erstellten IML-Datenbasis bezüglich ihrer Spezifikation, also den illustrierten Statecharts, wie oben unter 7.3 beschrieben, nicht explizit in den laufenden Entwicklungsprozess integriert. Erst nachdem alle Menüs für das zu spezifierende Infotainmentsystem fertiggestellt waren, begannen die IML-Entwickler mit systematischen Einzeldurchsichten der IML-Datenbasis bezüglich der illustrierten Statecharts. Dabei glichen sie die Statechart-Dokumente Seite für Seite unter Zuhilfenahme der Simulation mit der IML-Datenbasis ab.

Bei dem zum Zeitpunkt der Untersuchung noch laufenden Projekt B sollten die Durchsichten wieder wie im Projekt A gehandhabt werden. Das Projekt A war zum Zeitpunkt meiner Untersuchung soweit fortgeschritten, dass der Spezifizierungsprozess bis auf kleinere Nachbesserungen bereits abgeschlossen war. Zu Durchsichten kam es in diesem Projekt, nachdem die IML-Datenbasis einen ersten vollständigen Stand erreicht hatte.

Für die Durchführung der Einzeldurchsichten gab es keine feste Vorgehensweise, weshalb sie von verschiedenen IML-Entwicklern teilweise unterschiedlich gehandhabt wurden. So wurden z.B. entdeckte Defekte von einigen erst einmal notiert, während andere diese sofort korrigierten. In den meisten Fällen sah der Autor, die von ihm erstellten IML-Dateien durch und verglich seine Arbeit dabei mit den zugehörigen illustrierten Statecharts. Teilweise wurden die Dateien auch von jemanden, der nicht am Erstellungsprozess beteiligt gewesen war, also einem Fremden, inspiziert. In einigen Fällen war der Autor nicht eindeutig bestimmbar, da im Verlauf des Projekts verschiedene IML-Entwickler an der Erstellung der Dateien beteiligt waren. Die Durchsichten wurden ausschließlich alleine durchgeführt.

Ziel dieser Durchsichten war es zum einen zu überprüfen, ob in der IML-Datenbasis alles umgesetzt wurde, was in den Statecharts illustrativ spezifiziert wurde. Die IML-Datenbasis wurde also auf Vollständigkeit geprüft. Zum anderen wurde ihre Korrektheit begutachtet, indem überprüft wurde, ob alle Menüsprünge, alle textlichen und grafischen Anzeigeelemente, alle Zustandswechsel usw. eines Menüs, so umgesetzt worden waren, wie es in den illustrierten Statecharts semiformal spezifiziert wurde.

Die Durchsichten liefen laut den IML-Entwicklern zumeist so ab, dass der IML-Autor den aktuellen Stand der IML-Datenbasis (bzw. Teile von dieser) gegen die aktuelle Version eines Statechart-Dokuments (eine PDF-Datei, die alle illustrierten Statecharts eines Kontexts enthält) auf Korrektheit und Vollständigkeit überprüfte. Hierfür benutzte er zum einen die Simulation. Zum anderen war es für ihn in vielen Fällen jedoch auch notwendig die IML-Dateien genauer zu inspizieren, und zwar z.B. dann, wenn es um das Überprüfen von bedingten Menüsprüngen oder Funktionsausführungen ging. Es ließ sich mit der Simulation dabei zumeist nur ein Fall der jeweiligen Bedingung überprüfen (s. Kapitel 7.4). Deshalb wurde vereinzelt auch die Ausgabe des auf die Simulationssoftware zugeschnittenen Ablaufverfolgers, des *TraceTools* (s. Kapitel 7.4.2), überprüft.

Bei der Durchsicht gingen die IML-Entwickler das Statechart-Dokument meistens Menü für Menü durch. Sie prüften, ob das betreffende Menü überhaupt in der IML-Datenbasis vorhanden war und ob es auch wie in den illustrierten Statecharts spezifiziert, von den anderen Menüs aus erreichbar war. Auch überprüften sie die Menüs dahingehend, ob sie alle gewünschten Elemente enthielt und dass diese auch wie spezifiziert funktionierten.

Zu einen Abgleich der Datenbasis mit den illustrierten Statecharts durch den HMI-Experten, der das Statechart-Dokument verfasst hat, kam es nur teilweise. Im Prozessmodell vorgesehen war dies nicht. Die HMI-Experten überprüften für sich zwar hin und wieder anhand der Simulation, ob die IML-Umsetzung ihren Vorstellungen entspricht, jedoch taten sie dies nicht systematisch.

8.4 Probleme bei der momentanen Handhabung

Eine Durchsicht durch den HMI-Experten ist wie unter 7.1.4 und 7.3 schon erläutert unabdingbar, da nur er überprüfen kann, ob die Umsetzung auch seiner Spezifikation entspricht. Durch den semiformalen Charakter der Spezifikation in Form von illustrierten Statecharts ist es schließlich möglich, dass ein IML-Entwickler ein illustriertes Statechart anders interpretiert, als es vom Verfasser beabsichtigt wurde.

Da die HMI-Experten die Sprache IML zumeist nicht oder nur unzureichend beherrschen, können sie eine Durchsicht nur mit Hilfe der Simulation ausüben. Für eine vollständige Durchsicht reicht dies aber nicht unbedingt aus, da nicht immer alles vollständig simuliert wird (s. Kapitel 7.4). Außerdem ist die Simulation durch die unter 7.4 beschriebenen Probleme für die HMI-Experten nur beschränkt einsatzfähig, so dass sie spezielle Konfigurationen des HMI zumeist nicht selber testen könnten. Sie benötigen hierfür die Unterstützung eines IML-Entwicklers.

Die bis zum Zeitpunkt meiner Untersuchungen durchgeführten Durchsichten waren, wie oben beschieben, ausschließlich Einzeldurchsichten, die zumeist vom Autor der IML-Dateien selber getätigt wurden. Oft ist aber eine Durchsicht besonders dann effektiv, wenn sie nicht nur von dem Autor ausgeführt wird, sondern auch von jemandem, der nicht am Erstellungsprozess beteiligt war, einem Fremden. Bei den in der Literatur (vgl. [Bal98]) vorgestellten Durchsichtsmethoden wird deshalb in der Regel davon ausgegangen, dass es sich bei dem Gutachter nicht um den Autor handelt. Eine Ausnahme bildet die persönliche Durchsicht (engl. *personal review*), wie sie z.B. von Humphrey in [Hum95] beschrieben wurde. Dabei überprüft der Autor seine Arbeitsergebnisse selber.

Ein weiteres Manko der durchgeführten Einzeldurchsichten war, dass keine Messungen stattfanden. Dadurch war es nicht möglich diesen Teilprozess zu analysieren und zu bewerten.

8.5 Einführung von Paardurchsichten

Um den erläuterten Problemen zu begegnen, ist es vorstellbar, spezielle Paardurchsichten durchführen zu lassen. Das heißt, dass ein IML-Entwickler die von ihm in der IML spezifizierten Menüs eines Kontexts zusammen mit dem HMI-Experten, der die illustrierten Statecharts für diesen Kontext verfasst hat, nach einer individuellen Vorbereitung unter Zuhilfenahme der Simulation durchsieht.

Nach einigen Diskussionen und Absprachen mit den IML-Entwicklern, den HMI-Experten und den Projektleitern habe ich eine entsprechende Vorgehensweise erarbeitet. Die Zielsetzung der Paardurchsichten, deren Form und ein möglicher Ablauf mit Planungs- und Vorbereitungsphase, gemeinsamer Sitzung, Nachbereitungs- und Korrekturphase wird im Folgenden erläutert.

8.5.1 Zielsetzung

Das primäre Ziel der Paardurchsichten ist eine Verifizierung der IML-Datenbasis bezüglich ihrer Spezifikation in Form von illustrierten Statecharts. Es sollen dabei auch etwaige Defekte, die durch Missinterpretation bei der Überführung der illustrierten Statecharts in das Format der IML entstanden sind, identifiziert werden. Die Ursache kann z.B. darin liegen, dass bei der Erstellung der IML-Dateien ein in den Statecharts illustriertes Detail übersehen wurde. Dafür ist eine Paardurchsicht besonders geeignet, da hier der Effekt des Vier-Augen-Prinzips eintreten kann, dass zwei Personen zusammen mehr Defekte finden, als eine alleine. Dies wurde unter anderem in einem Experiment von Porter [Por97] bestätigt.

Es sollen vor allem Defekte, die das Verhalten und die Struktur der Menüs betreffen, aufgedeckt werde. Nach designspezifischen Defekten, wie z.B. einer falsch gewählten Schriftgröße, wird bei diesen Durchsichten nicht explizit gesucht. Eine diesbezügliche Erweiterung unter Berücksichtigung der Grafikspezifikation (s. Kapitel 6.2.3) ist jedoch vorstellbar.

Die kooperative Durchsicht dient auch dazu, die Qualität der illustrierten Statecharts und dessen Umsetzung zu verbessern. Der HMI-Experte erfährt bei der gemeinsamen Sitzung, an welchen Stellen Verständnisproblem auftraten und kann dann beim späteren Erstellen von illustrierten Statecharts versuchen, ähnliche Probleme zu vermeiden. Analog gilt dies für den IML-Entwickler. Auch er erfährt, an welchen Stellen er bei seiner Arbeit vermehrt Defekte produziert und kann daraus lernen.

Für beide Rollen bietet eine Paardurchsicht die Möglichkeit die Spezifikationen einmal aus der Sicht der jeweils anderen Rolle erläutert zu bekommen, wodurch sie sich später vielleicht besser in die andere Rolle hineinversetzen können und Verständnisprobleme somit verringert werden können. Falls dies zutrifft und die Paardurchsichten regelmäßig stattfinden würden, wäre dies ein Prozess zur kontinuierlichen Prozessverbesserung.

8.5.2 Form der Durchsicht

Wie die meisten Softwareentwicklungsprojekte, unterliegen auch Projekte, die mit dem TeleDrive VISION Prozess durchgeführt werden, zeitlichen und finanziellen Restriktionen. Deshalb ist es eine Rahmenbedingung für die Form der Durchsicht, den zeitlichen und damit auch finanziellen Aufwand so gering wie möglich zu halten, ohne jedoch deren Effektivität erheblich zu schmälern. Das heißt eine Rahmenbedingung war es, eine effiziente Durchsichtsform zu konzipieren.

Aus diesem Grund sah ich von einer weitergehenden Vergrößerung des an der Durchführung beteiligten Teams ab. In den in der Literatur [Bal98, Lai02] vorgestellten

Durchsichtsmethoden werden Teamgrößen von bis zu sieben Personen vorgeschlagen. In einem kontrollierten Experiment über Quelltextinspektionen von Porter et al. [Por97] wurde jedoch herausgefunden, dass bei Inspektionen, an denen vier Inspektoren beteiligt waren, nicht signifikant mehr Defekte gefunden wurden als bei Inspektionen mit zwei Inspektoren, das heißt eine Vergrößerung der Teamgröße auf vier Inspektoren führte zu keiner Effektivitätssteigerung und war folglich ineffizienter. Wie auch in der von Bisant und Lyle vorgestellten Zwei-Personen-Inspektion [Bis89], verzichtete ich deshalb auf die ansonsten bei Durchsichten übliche Rolle des Moderators.

Der Vorschlag dieser Teamgröße ist jedoch nicht als generelle Aussage anzusehen. Die Teamgröße ist abhängig vom Kontext der Durchsicht. Wenn hoch komplexe Algorithmen mit Hilfe von Quelltextinspektionen durchgesehen werden, und der Projekterfolg von der Korrektheit der Software stark abhängig ist, wie z.B. bei der Entwicklung einer Steuerungssoftware einer ESA-Rakete, kann es durchaus sinnvoll sein, eine Teamgröße von mehr als zwei Personen zu wählen.

Es hat sich nach mehreren Diskussionen mit den IML-Entwicklern herausgestellt, dass der Autor für eine effiziente Durchsicht unabdingbar ist, weshalb es sich bei dem an der Paardurchsicht beteiligten IML-Entwickler auch um den Autor handeln sollte. Auch in den in der Literatur vorgestellten Durchsichtsmethoden (vgl. [Bal98, Lai02]) ist der Autor immer einer der Beteiligten.

Der Grund, warum die Einzeldurchsichten zumeist vom Autor durchgeführt wurden, war laut eines IML-Entwicklers, dass der Autor die Struktur des betreffenden Menüs bereits kennt. Für einen Fremden bedürfe es hier einer Einarbeitungszeit.

Dadurch, dass es in IML möglich ist, Eigenschaften und Verhaltensweisen eines Menüs auf mehreren Stufen einer Widget-Hierarchie zu überschreiben, benötigt ein Fremder bei einer Prüfung von IML-Code zunächst einmal einen zeitlichen Aufwand zum Suchen der Stelle, an der die zu prüfende Eigenschaft definiert wurde. Bei einer Widget-Hierarchietiefe von im Mittel sieben und bis zu 14 in den Menüs einer Variante des Projekts B[1] ist es im schlechtesten Fall theoretisch möglich, dass ein Fremder für die Prüfung einer Eigenschaft 14 Dateien inspizieren muss. Da er die Struktur des Menüs selber festgelegt hat, erinnert sich der Autor zumeist, an welcher Stelle, in welcher Datei er nachsehen muss. Das Problem entsteht unter anderem dadurch, dass es in den untersuchten Projekten keine Konventionen für die Erstellung der Widgets gab, wie unter 7.2 erläutert. Wenn fest vorgegeben wäre, an welcher Stelle in einer Widget-Hierarchie welche Art von Eigenschaften zu definieren sind, würde ein fremden IML-Entwickler eine gesuchte Information wahrscheinlich schneller finden können

Ein wichtiger Grund, warum der Autor bei einer Durchsicht beteiligt sein sollte, ist außerdem, dass zumeist er es ist, der die gefundenen Defekte später korrigieren muss. Dafür ist es sinnvoll, wenn der Autor während der Durchsicht die entdeckten Defekte notiert, so dass er für die spätere Korrektur nicht zunächst einmal die Aufzeichnungen eines Fremden nachvollziehen muss.

Im Gegensatz zu den bisher durchgeführten Einzeldurchsichten sollten während der Paardurchsicht Messungen vollzogen werden. Dafür werden die Namen der durchgesehenen Menüs, die benötigte Zeit und die dabei gefundenen Defekte von den Beteiligten in einem Durchsichtsprotokoll notiert. Das Protokoll bildet die Grundlage für die nachfolgende Korrektur und die Bewertung des Prozesses. Damit die Daten dieser Messungen für spätere Analysen vergleichbar sind, habe ich eine Anleitung (s. Anhang 12.3) ausgearbeitet, nach der die Durchsichten vorzunehmen sind. Ich habe dabei einen hohen Grad der Detaillierung gewählt, um eine maximale Vergleichbarkeit für die nachfolgend beschriebene Paardurchsichtsstudie zu erhalten. Bei einer „echten“ Einführung der

[1] gemessen am 11.11.2005

Paardurchsicht in einem Projekt, sollten sich die Projektbeteiligten zuvor über Sinn und Unsinn der einzelnen Schritte absprechen. Die Vorgehensweise sollte dabei an das jeweilige Projekt angepasst werden.

Die Perspektive, welche die Beteiligten bei der hier beschriebenen Paardurchsicht einnehmen sollen, orientiert sich jeweils an ihrer Rolle, das heißt der IML-Entwickler überprüft die technischen Aspekte der IML-Umsetzung, während der HMI-Experte verifiziert, ob seine Spezifikation erfüllt wird.

Für spätere Projekte wäre es meiner Meinung nach aus den oben unter 8.1 genannten Gründen sinnvoll, wenn die Durchsichten immer gleich dann durchgeführt werden würden, wenn die Menüs erstellt worden sind, das heißt, wenn die Paardurchsichten in den Entwicklungsprozess integriert werden würden. Als Zeitpunkt bietet es sich hier an, diese jeweils ca. eine Woche vor einer Abgabe durchführen zu lassen, so dass die gefundenen Defekte noch behoben werden können.

8.5.3 Planungsphase

Wie gerade erläutert, sollten die Durchsichten ein fester Bestandteil des Entwicklungsprozesses sein. Es sollte bereits mit den Durchsichten begonnen werden, sobald eine unbestimmte jedoch nicht zu große Anzahl von Menüs von den IML-Entwicklern fertiggestellt wurde. Die Entscheidung darüber, wann der Zeitpunkt der Fertigstellung erreicht ist, sollte beim Autor liegen [Hum89]. Er muss daraufhin den zuständigen HMI-Experten darüber informieren, welche Menüs durchzusehen sind.

Außerdem ist ein Termin für die gemeinsame Sitzung zu vereinbaren. Um Ermüdungserscheinungen der Beteiligten vorzubeugen, sollte die Dauer der Sitzung begrenzt werden und maximal zwei Stunden betragen [Bal98, S. 318]. Da nicht absehbar ist, wie lange für die Durchsicht der betreffenden Menüs benötigt wird und wegen der begrenzten Sitzungsdauer, kann es sein, dass mehrere Sitzungen notwendig sind, falls nicht alles in einer Sitzung durchgesehen werden konnte.

8.5.4 Individuelle Vorbereitungsphase

Die Vorbereitung ist, wie bei Durchsichten üblich (vgl. [Bal98, S. 320]), individuell von den Beteiligten zu erledigen. Der an der Paardurchsicht beteiligte IML-Entwickler sollte sich zur Vorbereitung vor der Sitzung mit den zu begutachtenden Menüs anhand des Statechart-Dokuments, der IML-Datenbasis und der Simulation (wieder) vertraut machen. Er überprüft dabei, ob alle in dem Statechart-Dokument spezifizierten Menüs, die zu inspizieren sind, in der IML-Datenbasis vorhanden sind. Für die Menüs prüft er jeweils, dass alle in den illustrierten Statecharts eingezeichneten Elemente in der IML spezifiziert worden sind. Entsprechende Abweichungen notiert er sich dabei im Durchsichtsprotokoll. Außerdem überprüft er die Ausgabe des *TraceTools* auf etwaige zur Laufzeit generierte Meldungen über IML-Defekte.

Die auf den illustrierten Statecharts von dem Verfasser notierten Kommentare und die Menüübergänge sollen von ihm bei der Vorbereitung noch nicht verifiziert werden. Dies geschieht durch den HMI-Experten bzw. in späteren Phasen der Durchsicht. Jedoch sollte er sie sich durchlesen und nachvollziehen, damit er dies nicht in der gemeinsamen Sitzung nachholen muss. Außerdem präpariert er das Protokoll für die gemeinsame Sitzung.

Auch der HMI-Experte bereitet sich auf die Sitzung vor. Mit Hilfe einer aktuellen Simulation überprüft er die Menüsprünge und Kommentare der zu inspizierenden Menüs. Auf einem ausgedruckten Statechart-Dokument hakt er mit einem Stift alle korrekt umgesetzten Punkte ab. Wenn er aus den oben erläuterten Gründen etwas nicht nachprüfen kann oder wenn er sich unsicher ist, bleibt der Punkt erst einmal offen und wird

später während der gemeinsamen Sitzung verifiziert. Von ihm aufgedeckte Defekte notiert er sich in einer vorgefertigten Tabelle.

8.5.5 Gemeinsame Sitzung

Die gemeinsame Sitzung soll dazu dienen die während der Vorbereitung gefundenen Defekte zusammenzutragen und weitere Defekte, wie z.B. solche, die auf Missinterpretationen zurückzuführen sind, aufzudecken. Da der IML-Entwickler die Simulation während der Sitzung bedienen und konfigurieren muss, sollte die Sitzung in seiner vertrauten Arbeitsumgebung, also an seinem Arbeitsplatz, stattfinden

Ein zu inspizierendes Menü wird dabei zuerst einmal in der Simulation angezeigt. Es werden nun die in der Vorbereitungsphase entdeckten Defekte, die das Menü betreffen, diskutiert. Wenn das Paar darin übereinstimmt, dass ein gefundener potentieller Defekt wirklich einer ist, übernimmt ihn der IML-Entwickler in das Durchsichtsprotokoll, falls er dort nicht schon vorhanden ist. Dafür ist es auch notwendig zu erörtern, in welchem Produkt sich der Defekt befindet. Dabei kann es sich unter anderem um die IML-Datenbasis, ein illustriertes Statechart oder die Simulationssoftware handeln.

Danach werden die Kommentare überprüft, die bei der vorbereitenden Durchsicht durch den HMI-Experten offen geblieben sind. Das Paar verifiziert, ob die IML-Umsetzung dem entspricht, was mit dem Kommentar spezifiziert wurde. Der IML-Entwickler leitet dabei die Prüfung, indem er die für die Überprüfung notwendigen Konfigurationen an der Simulation vornimmt. Wenn ein Punkt nicht mit der Simulation überprüft werden kann und eine Inspektion der IML-Dateien dafür notwendig wäre, wird er nur soweit durchgesprochen, dass ihn der IML-Entwickler in der folgenden Nachbereitungsphase alleine überprüfen kann.

Auch die in der Vorbereitungsphase noch nicht überprüften Menüsprünge sind in der Sitzung nicht unbedingt vollständig zu verifizieren, jedoch sollten die Bedingungen für die Sprünge, wieder soweit durchgesprochen werden, dass der IML-Entwickler nachvollziehen kann, wann es zu dem Sprung kommen muss und keine diesbezüglichen Missverständnisse bei einer späteren Verifizierung durch ihn mehr auftreten können.

Der IML-Entwickler fungiert während der Sitzung als Protokollant. Die in der Durchsicht gefundenen Defekte werden vom ihm in dem Durchsichtsprotokoll notiert. Auch hakt er auf dem in der Vorbereitung vom HMI-Experten bearbeiteten Statechart-Dokument, die Punkte ab, die während der Sitzung vollständig überprüft werden konnten. Dadurch behält er den Überblick darüber, was von ihm später noch zu verifizieren ist.

8.5.6 Nachbereitungsphase

Der IML-Entwickler verifiziert nach der Sitzung die Punkte, die bis dahin noch nicht vollständig überprüft worden sind. Dabei hakt er wieder die geprüften Punkte auf dem bereits in der Sitzung verwendeten Statechart-Dokument ab, bis kein Punkt mehr offen ist.

Er inspiziert dabei die IML-Dateien an den betreffenden Stellen. Für die Überprüfung von Funktionsaufrufen, die z.B. nach einem Tastendruck veranlasst werden sollen, kontrolliert er mittels der zugehörige Ausgabe des *TraceTools*, ob der Aufruf stattfand. Etwaige Defekte, die er entdeckt, werden zuerst wieder schriftlich im Durchsichtsprotokoll festgehalten.

8.5.7 Korrekturphase

Erst nach Abschluss der Prüfungen werden die entdeckten Defekte von dem an der Durchsicht beteiligten IML-Entwickler korrigiert. Defekte, deren Behebung er nicht auf Anhieb alleine bewerkstelligen kann, sollte er entweder an den verantwortlichen IML-Entwickler direkt herantragen oder aber detailliert und für alle Entwickler zugänglich notieren bzw. mit einem Werkzeug verwalten. Als Werkzeug würde sich hier das in dem Projekt B vereinzelt schon eingesetzte Testmanagement-Werkzeug *TestDirector* der Firma Mercury eignen. Der Einsatz eines anderen, ähnlichen Produkts ist auch vorstellbar

Falls bei der Durchsicht Defekte in dem Statechart-Dokument gefunden worden sind, so sind sie von dem HMI-Experten zu korrigieren. Dabei und auch bei anderen nachfolgenden Veränderungen des Statechart-Dokuments sollten entweder alle Veränderungen als noch nicht geprüft in den Statechart-Dateien kenntlich gemacht werden oder aber in einem Werkzeug zur Verwaltung von Anforderungen - auch hierfür wäre der *TestDirector* geeignet - erfasst werden. Somit ist es möglich, falls später eine erneute Durchsicht eines Statechart-Dokuments vorgenommen wird, lediglich die neu hinzugekommenen Punkte zu überprüfen. Dabei wird jedoch das Risiko eingegangen, dass sich durch die Korrektur ein Defekt an einer anderen Stelle eingeschlichen hat.

Auch das unter 7.1.5 beschriebene Problem der unzureichenden Versionsgeschichte würde durch die Verwendung eines Anforderungsmanagementsystems entfallen, da die IML-Entwickler sofort erkennen können, welche neu hinzugekommenen Anforderungen noch zu implementieren sind.

9 Ein Umfangsmaß für IML

Ein Ziel der nachfolgend beschriebenen Studie (s. Kapitel 10) war es per Hochrechnung abzuschätzen, wie aufwändig die von mir konzipierte Paardurchsicht (s. Kapitel 8.5) wäre, wenn sie für alle Menüs des Projekts B vorgenommen werden würde. Deshalb benötigte ich ein Maß, welches den Umfang eines Menü-Widget bestimmt. Zum einen wurde damit der Umfang der in der Studie (s. Kapitel 10) durchgesehenen Menüs bestimmt. Zum anderen ließ sich damit bestimmen, wie umfangreich alle Menüs einer IML-Datenbasis zusammen sind. Die Definition des nachfolgend beschriebene Maßes ist allgemein gehalten. Da ein Menü-Widgets ein zusammengesetztes Widget ist, lässt sich dessen Umfang mit dem in Kapitel 9.3 (Umfang eines zusammengesetzten Widgets) vorgestellten Maß bestimmen. Das Maß stelle ich nach einer Erläuterung zu Grundlagen von Komplexitätsmaßen vor.

9.1 Grundlagen

Normalerweise wird bei Durchsichten von Quelltexten als Maß die Anzahl der Quelltextzeilen gezählt (engl.: *lines of code* (LOC)), wobei zumeist Kommentarzeilen nicht mitgezählt werden, um die Aussagekraft zu erhöhen (engl.: *non-commented lines of code* (NLOC)). Beide Maße zählen zu den Komplexitätsmaßen, deren Zweck es ist, die Größe eines Softwareprodukts zu beschreiben.

Komplexität ist ein „hinreichend verwaschener Begriff“ [BeR03]. Das Wort ist ein gutes Beispiel für die Mehrdeutigkeit informaler Sprachen. Im Bereich der Informatik existieren verschiedenste Komplexitätsarten. In [BeR03] wird in folgende Arten unterschieden: Problemkomplexität, algorithmische, strukturelle und kognitive Komplexität. Die Vielfältigkeit von Komplexität zeigt auch die Abbildung 29.

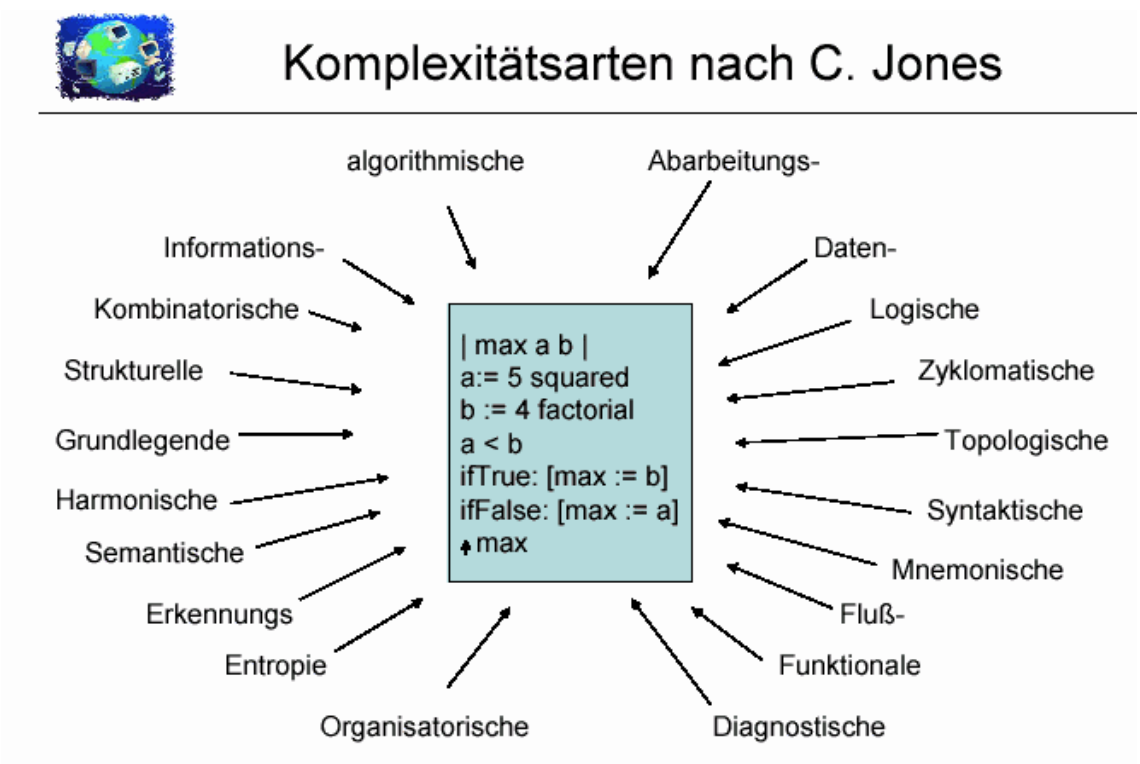

Abbildung 29: Komplexitätsarten nach Casper Jones [Dum06]

Um Missverständnisse, die aus dieser Mehrdeutigkeit resultieren könnten, zu vermeiden, bezeichne ich die in diesem Kapitel definierten Maße genauer als Umfangsmaße.

9.2 Umfang einer Widget-Hierarchie

Ein Beispiel für eine Widget-Hierarchie ist (noch einmal) in Abbildung 30 zu sehen. Um in Erfahrung zu bringen, wie umfangreich eine Widget-Hierarchie ist, reicht es aus zu ermitteln, wie umfangreich das System-Widget (s. Abbildung 30 oder auch Kapitel 2.4.2) ist, da es (induktiv) alle Widgets der Hierarchie enthält. Weil das System-Widget (zumindest in der Praxis) ein zusammengesetztes ist, muss nur betrachtet werden, wie sich der Umfang eines zusammengesetzten Widgets bestimmen lässt.

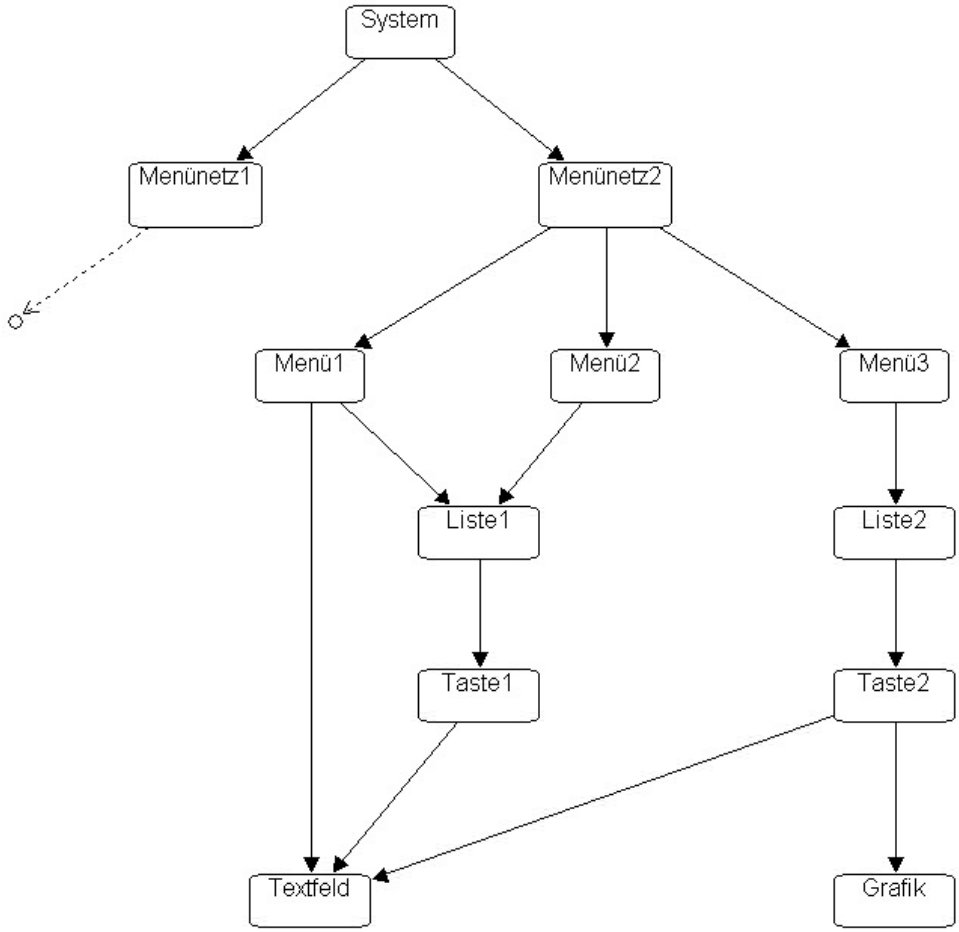

Abbildung 30: Beispielhafte IML-Widget-Hierarchie

9.3 Umfang eines zusammengesetzten Widgets

Der Umfang eines zusammengesetzten Widgets ergibt sich aus dem Umfang des Widgets selbst plus der Summe der Umfänge der in dem betreffenden Widget eingebundenen Widgets. Dabei wird jedoch jeder Widget-Typ nur einmal mitgezählt, auch wenn er in der Hierarchie in mehreren Exemplaren eingebunden ist. In Abbildung 31 ist die Widget-Exemplar-Hierarchie für die in Abbildung 3 auf Seite 17 zu sehende Hierarchie beispielhaft dargestellt.

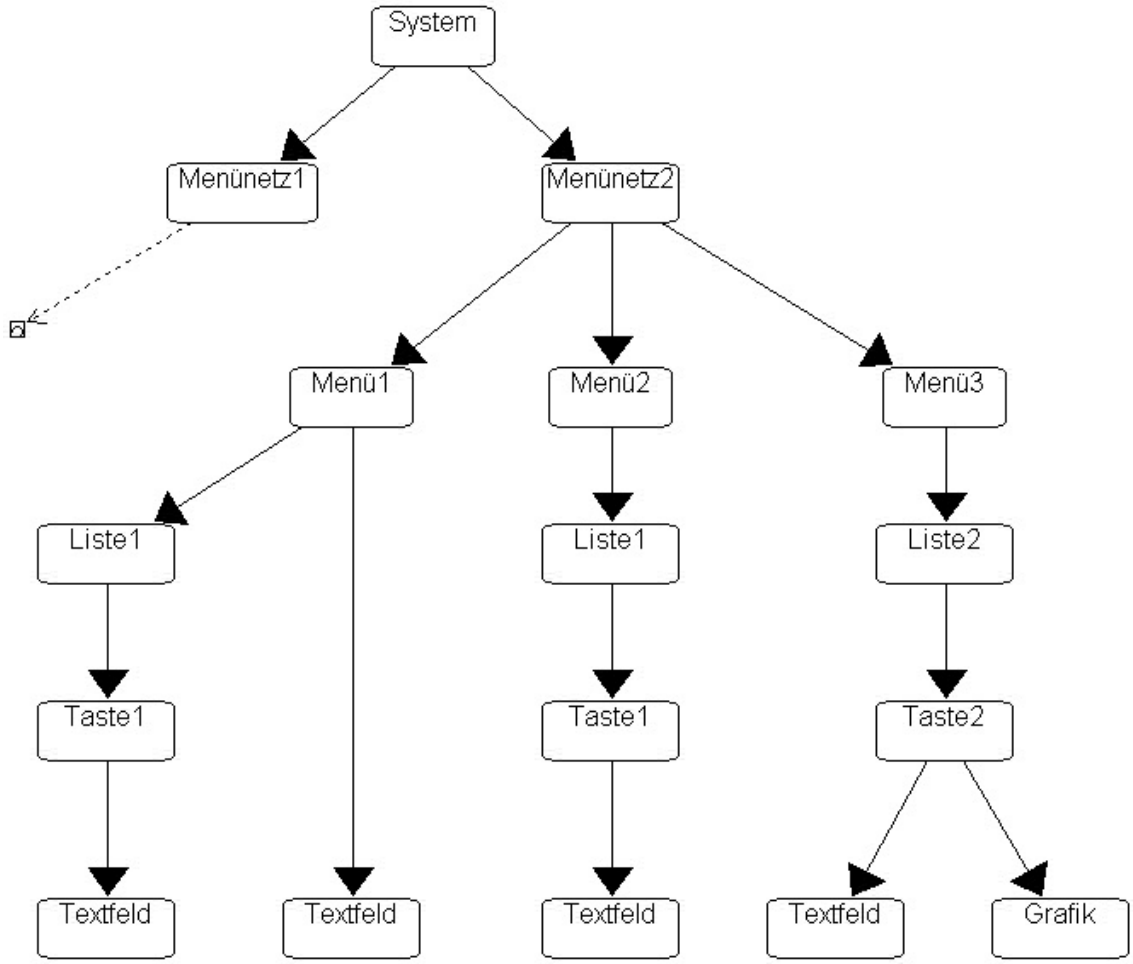

Abbildung 31: Beispielhafte Widget-Exemplar-Hierarchie

Für eine Umfangsberechnung des Menünetz2 würde hier also der Umfang des Widgets Textfeld nur einmal anstatt viermal gezählt werden. Auch das Widget Liste1 würde nur einfach berücksichtigt werden, obwohl es zweimal in der Hierarchie auftritt.
Dies geschieht aus folgendem Grund: Die Durchsicht eines Menüs, wie sie in Kapitel 8.5 beschrieben ist, erfordert keinen zehnmal so großen Aufwand, wenn ein Menü aus zehn gleichen Textfeldern besteht anstatt nur aus einem. Erst wenn die Eigenschaften der Textfelder, wie z.B. die Textfarbe, (z.B. auf Menüebene) überschrieben werden, muss jedes Textfeld einzeln betrachtet werden. Die Überschreibungen werden jedoch nicht im Textfeld-Widget vorgenommen, sondern in darüber liegenden Widgets. Durch die nachfolgend beschriebene Messung nimmt der Umfang eines Widgets zu, wenn in ihm eine Überschreibung eines eingebundenen Widgets stattfindet.

9.4 Umfang eines einzelnen Widgets

Der Umfang eines einzelnen Widgets lässt sich anhand des dafür geschriebenen XML- / IML-Codes berechnen. In XML wie auch in anderen formalen Sprachen ist es theoretisch möglich, den gesamten Code in eine Zeile zu schreiben. Deshalb habe ich mich dazu entschieden, die Anzahl der Befehle zu zählen. Dies entspricht in XML in etwa der Anzahl der Elemente.

Ein Element kann in XML respektive IML zwei Ausprägungen haben. Ein leeres Element ist durch ein Leeres-Element-Tag (<ZumBeispiel/>) begrenzt. „Ein vollständiges *Element*, [...] besteht aus dem *Start-Tag*, dem Inhalt sowie dem *End-Tag*“ [W3C05] (<So>ZumBeispiel</So>).

Die im XML-Quelltext enthaltenen Kommentare werden nicht mitgezählt. Da aber Kommentare in IML-Dateien zumindest in den von mir untersuchten Projekten kaum vorkamen (in der IML-Datenbasis des Projekts B maß ich am 17.11.2005 eine Kommentarzeilenanzahl von insgesamt 200 in den 1534 IML-Widget-Dateien), würde auch aus dem Mitzählen von Kommentarzeilen keine große Abweichung resultieren.

Durch das Zählen der Elemente entfällt die für das Zählen von LOC übliche Fragestellung, ob Leerzeilen mitgezählt werden sollen und wie mit Zeilen, die aus mehr als einer Anweisung bestehen, verfahren werden soll. Auch das von Prechelt angesprochene Problem [Pre99] der unzureichenden Interpretierbarkeit erhobener Daten beim Mitzählen von Leerzeilen besteht hier deshalb nicht.

Im weiteren Verlauf der Arbeit bezeichne ich mit dem Begriff Umfangseinheit (UE) ein XML-Element, welches mittels der in diesem Kapitel beschriebenen Methode gemessen wurde.

9.5 Validierung des Umfangsmaßes

Im Rahmen einer Diplomarbeit war es mir leider nicht mehr möglich, das hier vorgeschlagene Umfangsmaß bezüglich seiner Tauglichkeit als Schätzmaß für den Durchsichtsaufwand zu validieren. Ich beschäftigte mich hiermit zwar einige Wochen lang, fand jedoch keine brauchbare Methode, um zu validieren, ob das Maß mit dem Durchsichtsaufwand korreliert.

Da jedoch das Maß LOC für Aufwandsschätzungen (z.B. bei COCOMO) herangezogen wird (vgl. [SoK06e]) und das hier vorgestellte Maß diesem ähnelt, ist eine weitere Validierung möglicherweise überflüssig.

9.6 Verwendetes Messwerkzeug

Für die Messungen passte ich ein von einem IAV-Mitarbeiter in C# (nicht Cis-Dur) geschriebenes IML-Komplexitätsmesswerkzeug an. Dieses Werkzeug maß unter anderem die Anzahl von Widgets in einer Hierarchie und die darin enthaltene Anzahl von Widget-Eigenschaften und wurde für Optimierungsarbeiten an der Simulationsplattform eingesetzt.

Ich erweiterte es während meiner Arbeit um die Messung mit dem hier vorgeschlagenen Maß. Außerdem baute ich eine Exportfunktion für die erhobenen Daten ein, um sie mit Excel statistisch auswerten zu können.

Die von dem Werkzeug ausgegebenen Daten validierte ich (bzw. ich verifizierte die Algorithmen) durch manuelles Nachzählen für ausgewählte, kleinere Widgets, die alle aus den oben erläuterten Eigenschaften des Maßes ableitbaren Spezialfälle abdeckten.

10 Empirische Studie zur Einführung von Durchsichten

Um die oben beschriebene Form der Paardurchsicht (s. Kapitel 8.5) „auszuprobieren“, führte ich eine Studie durch, die ich nachfolgend beschreibe.

10.1 Ziel der Studie

Das primäre Ziel der Studie war es zu untersuchen, ob eine wie in Kapitel 8.5 beschriebene Paardurchsicht einer IML-Spezifikation effizient in Projekten, die den TeleDrive VISION Prozess leben, einsetzbar ist. Dazu wurde eine Fallstudie in dem Projekt B nach der GQM-Methode (s. Kapitel 3.3) durchgeführt. Die Zieldefinition für die in der Studie durchgeführten Messungen lautete wie folgt:

Objekt: Paardurchsichten
Zweck: Bewertung und Beurteilung
Qualitätsfokus:
1. Anzahl gefundener Defekte
2. Defektklassen
3. Aufwand für die Durchsicht

Blickwinkel: IML-Entwickler und HMI-Experten
Kontext: Projekt B

Es sollte untersucht werden, ob die Einführung von Paardurchsichten ein effizientes Mittel sind, um Mängel in einer IML-Datenbasis aufzudecken und damit dessen Qualität zu steigern. Dabei lag ein Fokus auf Defekten, die auf Missinterpretationen bei der Überführung der illustrierten Statecharts in das Format der IML zurückzuführen sind, da diese durch die bisher angewandte Methodik der Durchsichten (s. Kapitel 8.3) nicht entdeckt werden können. Wenn durch die Studie gezeigt werden kann, dass derartige Defekte in dem Überführungsprozess entstehen und sie durch die hier vorgeschlagenen Paardurchsichten aufgedeckt werden können, wären die Paardurchsichten ein effizientes, wenn nicht sogar notwendiges, Mittel für eine Qualitätssteigerung des Prozesses.

Ziel der Studie war es auch herauszufinden, wie groß der zeitliche Aufwand der Paardurchsichten ist und wie groß dieser wäre, wenn die Durchsichten systematisch über die gesamte IML-Datenbasis eines Projekts durchgeführt werden würden. Da eine diesbezügliche Beurteilung der Ergebnisse nur von leitenden Mitarbeitern, wie z.B. einem Projektleiter, erfolgen kann, ging es hier lediglich um eine Datenerhebung und die Analyse der Daten nicht aber um deren Interpretation.

An Durchsichten beteiligte Gutachter finden es im Allgemeinen hilfreich, eine Checkliste zu verwenden [Hum89], die ihnen einen Rahmen dafür vorgibt, auf welche Aspekte hin der Prüfling untersucht werden soll. Darin sollten unter anderem die häufigst auftretenden Defekttypen aufgelistet sein. Da eine Checkliste für die Durchsicht der IML-Datenbasis noch nicht existierte, war es ein weiteres Ziel der Untersuchung, eine erste Auflistung der möglichen Defekttypen zu erstellen.

Außerdem sollte eine Analyse der Defekte dazu dienen, Schwachstellen im Prozess aufzuzeigen und daraufhin Lösungsansätze zu entwickeln. Bei seiner Arbeit als Leiter der Software-Vermessung-Programme der Firma Hewlett-Packard fand Grady heraus [Gra94], dass Defektanalysen in einem Softwareentwicklungsprozess, das größte Potential für eine kurzfristige Prozessverbesserung bieten. Durch die Analyse der Ergebnisse der hier durchgeführten Studie sollten Rückschlüsse gezogen werden, an welchen Stellen die illustrierten Statecharts formal werden müssen, damit ein fehlerfreies und nachfragefreies oder zumindest nachfragearmes Überführen der illustrierten Statecharts er-

möglicht werden kann. Es sollten Schwächen dieses Spezifikationsformats aufgedeckt werden, um Ansätze für eine Prozessverbesserung zu erlangen.

10.2 Abgeleitete Fragestellungen

Nachfolgend sind die aus den Zielen der Studie abgeleiteten Fragen aufgeführt.

F.1 Wie groß ist der Aufwand für die Paardurchsichten?

F.2 Wie groß wäre der Aufwand für Paardurchsichten von allen Menüs der IML-Datenbasis in dem untersuchten Projekt?

F.3 Wie viele Defekte werden bei den Paardurchsichten entdeckt?

F.4 Von welchem Typ sind die Defekte, die bei den Paardurchsichten gefunden werden?

F.5 Wie viele der entdeckten Defekte sind auf Missinterpretationen zurückzuführen?

10.3 Maße zur Fragenbeantwortung

Nachfolgend sind die aus den Fragen abgeleiteten Metriken aufgelistet.

M.1 Dauer der Paardurchsicht

M.1.1 Verwendete Zeit für die Vorbereitung durch den IML-Entwickler

M.1.2 Verwendete Zeit für die Vorbereitung durch den HMI-Experten

M.1.3 Verwendete Zeit für die gemeinsame Sitzung

M.1.4 Verwendete Zeit für die Nachbereitung durch den IML-Entwickler

M.2 Version des durchgesehenen Statechart-Dokuments

M.3 Namen der Menüs, die durchgesehenen wurden

M.4 Umfang der durchgesehenen Menüs

M.5 Summe der Umfänge von allen Menüs

M.6 Anzahl der bei der Paardurchsicht gefundenen Defekte

M.6.1 Anzahl der Defekte, die während der Vorbereitung durch den IML-Entwickler gefundenen wurden

M.6.2 Anzahl der Defekte, die während der Vorbereitung durch den HMI-Experten gefundenen wurden

M.6.3 Anzahl der Defekte, die während der gemeinsamen Sitzung gefundenen wurden

M.6.4 Anzahl der Defekte, die während der Nachbereitung gefundenen wurden

M.7 Klassifikation der gefundenen Defekte

M.8 Anzahl der bei der Durchsicht gefundenen Defekte pro Klasse aus M.7

M.9 Durchschnittliche Anzahl der bei der Durchsicht gefundenen Defekte

10.4 Methodik

Nachfolgend wird die bei der Studie verwendete Methodik beschrieben.

10.4.1 Wahl der Studiensubjekte

Nach einer Absprache mit einem Projektleiter des Projekts B bestimmte ich fünf Paarungen für die Studie. Dabei war es eine Vorgabe möglichst immer verschiedene Personen auszuwählen, um (noch) keine Lerneffekte der Beteiligten in die Ergebnisse mit einfließen zu lassen, also eine Vergleichbarkeit der Daten vor allem bezüglich der benötigten Zeit und aufgedeckten Defekte zu garantieren. Da sich zum Zeitpunkt der Untersuchung nur drei HMI-Experten in Berlin befanden, war diese Bedingung nur für die IML-Entwickler einzuhalten.

Die Beteiligten sollten wegen 8.5.2 jeweils die IML-Autoren und die Autoren der illustrierten Statecharts sein. Da die IML-Entwickler zumeist jeweils für einen Kontext (z.B. Radio oder Navigation) zuständig waren, entschied ich mich dazu, für jede Durchsicht einen anderen Kontext zu wählen. Dadurch konnte ich auch ausschließen, dass die spätere Hochrechnung auf zu speziellen Daten basieren würde.

10.4.2 Wahl der Studienobjekte

Für die Auswahl der zu inspizierenden Menüs, durchsuchte ich die Statechart-Dokumente. Eine durchzusehende Seite musste dabei folgende Kriterien erfüllen:

1. Es wurden nur Seiten gewählt, welche die Spezifikation von einem oder mehreren Menüs *vollständig* abdeckten. In Projekt B kam es mitunter vor, dass ein Menü und die zugehörige Menülogik (Transitionen) auf mehreren Seiten verteilt spezifiziert war. Diese Menüs wurden von mir nicht berücksichtigt, sofern es sich dabei um mehr als zwei Seiten handelte.
2. Auch die berechneten Umfangswerte der Menüs habe ich betrachtet und in meine Wahl mit einfließen lassen. Die Messungen wurden nach der Methode, wie sie in Kapitel 9 nachzulesen ist, vorgenommen. Bei den fünf Paarungen sollte ca. der gleiche Umfang durchgesehen werden. Ich wollte damit erreichen, dass die Dauer der einzelnen Durchsichten in etwa gleich ist.
3. Menüs, welche die ersten beiden Kriterien erfüllten, legte ich dem betreffenden IML-Autor vor und fragte ihn, ob die Spezifizierung der Menüs beendet ist. Es sollte schließlich immer der betreffende Autor entscheiden können, ob eine Durchsicht bereits vorgenommen werden kann (vgl. 8.5.3).

Da mir das erste und das dritte Kriterium wichtiger waren, wurde das zweite Kriterium nicht immer eingehalten. Dadurch schwankten die Umfangswerte der durchzusehenden Menüs mitunter um mehr als das doppelte.

10.4.3 Durchsichtsanleitung und -Protokoll

Für die Paardurchsichten konzipierte ich eine Anleitung zur Vorgehensweise, nach der die Durchsichten in der Studie vorgenommen werden sollten. Dadurch sollten die erhobenen Daten später besser vergleichbar sein. Wenn jeder die Durchsicht nach seiner Facon durchgeführt hätte, wären die Daten bezüglich der benötigten Zeit und der aufgedeckten Defekte nicht mehr vergleichbar gewesen, da einer intensiver als ein anderer inspiziert hätte. Passend zur Anleitung erstellte ich außerdem ein Durchsichtsprotokoll. Die Anleitung und das Protokoll orientieren sich an der in Kapitel 8.5 beschriebenen

Durchsichtsmethodik und sind im Anhang einzusehen. Bei der Entwicklung der Durchsichtsmethodik fragte ich mehrere Entwickler nach ihrer momentanen Vorgehensweise. Die Anleitung ließ ich von zwei IML-Entwicklern und einem HMI-Experten vor der Studie durchsehen. Danach und nach einer ca. einstündigen Pilotstudie mit einem Paar überarbeitete ich die Dokumente.

In der ersten Version der Anleitung war es vorgesehen, dass ein durchzusehendes Menü immer als Startmenü der Simulation definiert werden sollte, dann erst sollte die Simulation gestartet werden. Nach der Durchsicht des Menüs sollte die Simulation wieder beendet werden. Das wurde nach der Pilotstudie von den Beteiligten kritisiert, da es bei vielen Menüs schneller gehe die Simulation standardmäßig zu starten und zu dem entsprechenden Menü zu navigieren. In der nächsten Version der Anleitung war die Festlegung des Startmenüs vor dem Simulationsstart als optional gekennzeichnet.

Nach einer Absprache mit meinen Betreuern entschied ich mich dazu Anpassungen an der Durchsichtsanleitung auch noch während der Studie vorzunehmen. Dadurch konnte ich aus meinen Fehlern lernen und den Durchsichtsprozess kontinuierlich verbessern.

Folgende Veränderungen an der Anleitung nahm ich während der Studie vor:

4. Menürücksprünge: In der zweiten Durchsicht wurde ein Defekt gefunden, der noch nicht zwangsläufig durch die von mir verfasste Vorgehensweise entdeckt wurde. Ich hatte Menürücksprünge, sogenannte Back-Sprünge, noch nicht berücksichtigt. Ab der dritten Durchsicht war dann ein entsprechender Hinweis, dass für jedes Menü, bei der Überprüfung eines Menüsprungs auch der Zurücksprung überprüft werden soll, in der Durchsichtsanleitung vorhanden.
5. Stil: Während einer Durchsicht kam es bei dem anschließenden Interview (s. Kapitel 10.5) zu einer Kritik an der Formulierung der Anleitung. Die beteiligte IML-Entwicklerin beschwerte sich, dass immer vom *dem* IML-Entwickler die Rede sei. Deshalb verfasste ich anschließend eine geschlechtsneutrale Version der Durchsichtsanleitung für IML-EntwicklerInnen. Die an der Studie beteiligten HMI-Experten waren ausschließlich männlich.
6. Formulierungen in der Anleitung passte ich auch schon während der Studie an, wenn sich herausstellte, dass etwas nicht verständlich formuliert war.

10.5 Durchführung

Nach einer Aufklärung der Beteiligten über den Zweck und die Vorgehensweise der Studie, wurden Termine für die Durchsichten vereinbart. Die Durchführung der Studie erstreckte sich über ca. eine Woche.

Die Vorbereitungsphase wurde von den Beteiligten jeweils an ihrem eigenen Arbeitsplatz vorgenommen. Meistens geschah dies direkt vor dem Termin der gemeinsamen Sitzung. In einem Fall wurde die Vorbereitung des IML-Entwicklers bereits am Vorabend durchgeführt.

Die gemeinsame Sitzung fand immer am Arbeitsplatz des IML-Entwicklers satt. Ich war zwar bei den Sitzungen mit dabei, wirkte jedoch nicht aktiv mit. In Einzelfällen beantwortete ich Fragen zur Durchführung. Dies war notwendig, da die Anleitung aufgrund des informalen Formats nicht immer eindeutig geschrieben war. Außerdem beobachtete ich die Durchsicht. Dabei lag ein Fokus darauf, herauszufinden, ob entdeckte Defekte auf Missverständnisse zurückzuführen waren.

Die Gespräche der Beteiligten nahm ich mit einem Audioaufnahmegerät mit ihrer (freundlichen) Genehmigung auf.

Die Messungen für die Maße M.1 – M.3 und M.6 (s. 10.3) wurden von den Beteiligten vorgenommen und in dem Protokoll notiert. Die restlichen Messungen wurden von mir nachträglich vorgenommen.

Um nach der Studie Verbesserungsvorschläge an der Vorgehensweise in Erfahrung bringen zu können, verfasste ich einen zur Studie passenden Interviewleitfaden (s. Anhang 12.2.4). Die Interviews fanden zumeist im Anschluss an die gemeinsame Sitzung statt und wurden mit beiden Beteiligten (dem Paar) gleichzeitig geführt.

10.6 Datenanalyse

10.6.1 Defektklassifizierung

Anhand der IML-Dateien, der illustrierten Statecharts und der Audioaufnahmen der gemeinsamen Sitzung überprüfte ich die Defekte hinsichtlich ihrer Ursache. Zunächst ging es dabei darum, herauszufinden, ob es sich bei dem Defekt um einen Defekt in der IML-Datenbasis, den illustrierten Statecharts oder aber in der Simulationssoftware handelte. Die Konzentration lag danach auf den IML-Defekten, da die Studie vor allem auf eine Analyse dieser Defekte abzielte. Wie es unter 10.1 beschrieben ist, sollte eine Defektklassifizierung für die IML-Defekte vorgenommen werden.

Nach Freimut [Fre01] gibt es zwei Strategien zur Entwicklung einer Defektklassifizierung. Entweder man schneidert eine bereits existierende Klassifizierung auf den betreffenden Kontext zu oder aber man entwickelt eine neue. Da eine Klassifizierung für den hier untersuchten Kontext noch nicht existierte, fiel die Entscheidung auf letztere Strategie. Die Klassifizierung orientiert sich an der Sprache IML. Die mit den Durchsichten aufgedeckten Defekte waren externe Defekte, also äußerlich sichtbare Mängel im IML-Code; im Gegensatz zu Defekten, die nicht oder nur unter Umständen von außen sichtbar sind, wie z.B. von einem C-Programm erzeugte Speicherlöcher.

Neben der Defektklassifizierung nahm ich eine eingeschränkte Fehleranalyse vor, um Defekte, die auf die in Kapitel 7 beschriebenen Probleme zurückzuführen sind, herauszufiltern. Sowohl für die Defektklassifizierung als auch für die Fehleranalyse verwendete ich die Audioaufzeichnungen der gemeinsamen Sitzung. In Zweifelsfällen befragte ich die Studienbeteiligten im Nachhinein nach der Ursache der Defekte und Fehler oder ließ mir eine vermutete Ursache bestätigen.

Einige der aufgedeckten Defekte wurden aufgrund der Vorgehensweise doppelt in den Protokollen notiert. So kam es z.B. vor, dass ein IML-Entwickler während der Vorbereitung einen Defekt aufgrund der Anzeige des *TraceTools* entdeckte und dieser Defekt auch in der Vorbereitung des HMI-Experten anhand der Simulation aufgedeckt wurde. Der Defekt wurde dadurch zweimal im Protokoll notiert. Defekte dieser Art versuchte ich während der Datenanalyse herauszufiltern, um die korrekte Gesamtanzahl der aufgedeckten Defekte zu ermitteln.

Auch wurden mitunter Defekte vom IML-Entwickler während der Durchsicht als IML-Defekte gekennzeichnet, die sich später als Defekte der Simulationssoftware herausstellten. Diese probierte ich auszusortieren.

Bei einer Durchsicht, bei der drei ähnliche Menüs inspiziert wurden, ist es außerdem vorgekommen, dass in allen drei Menüs der gleiche Defekt gefunden wurde. Dies war wahrscheinlich auf die Kopiertechnik (s. Kapitel 7.2), die bei der IML-Entwicklung eingesetzt wurde, zurückzuführen. Defekte dieser Art wurden bei der Datenanalyse nur einfach berücksichtigt, da sie durch den gleichen Fehler entstanden waren.

10.6.2 Abschätzung des Gesamtaufwands

Um den Aufwand für eine Durchsicht der gesamten IML-Datenbasis hochzurechnen, ging ich wie folgt vor. Zunächst berechnete ich den Gesamtaufwand der einzelnen Durchsichten, der sich aus der Summe der für die einzelnen Phasen benötigten Zeiten ergibt. Die Dauer der gemeinsamen Sitzung wurde dabei doppelt gezählt, da daran zwei Personen beteiligt gewesen waren.

Den so errechneten Gesamtzeitaufwand stellte ich dem Gesamtumfang der durchgesehenen Menüs gegenüber. Dadurch erhielt ich pro Durchsicht einen Wert, der angibt, wie viele Umfangseinheiten (im Folgenden kurz UE) jeweils pro Minuten durchgesehen wurden: die Durchsichtsrate.

Humphrey schlägt für Schätzungen von Durchsichtsaufwänden in einem Buch [Hum00] über den *Personal Software Process* (kurz PSP) folgende Methode vor: Die aus vorherigen Durchsichten gewonnenen Daten bzw. Erfahrungswerte werden für eine lineare Regression benutzt. Dadurch erhält man eine Gradengleichung, die für eine Abschätzung verwendet werden kann. Die in der Studie gesammelten Daten trug ich in Excel-Tabellen zusammen. Auch eine lineare Regression nahm ich in Excel vor, um den Gesamtaufwand abzuschätzen.

10.7 Resultate

Nachfolgend präsentiere ich die Resultate dieser Studie, wobei zunächst die Rohdaten betrachtet werden und danach die aufbereiteten Daten. Bei letzteren handelt es sich zum einen um Ergebnisse, die nach der soeben beschriebenen Methodik entstanden. Zum anderen orientierte ich mich außerdem an den in [Hum89, S. 319 ff] vorgeschlagenen Darstellungsformen für Durchsichtsraten.

Im nachfolgenden Text werden die an den Durchsichten beteiligten Rollen teilweise wie folgt abgekürzt: IML-Entwickler (IML), HMI-Experte (HMI).

10.7.1 Rohdaten

Folgende Daten wurden bei den Paardurchsichten erhoben.
„Vor“ steht jeweils für die Vorbereitungsphase, „Nach“ für die Nachbereitungsphase.
Die Dauer ist immer eine Minutenangabe. Die Gesamtdauer ergibt sich durch doppelte Zählung der Sitzungsdauer. Bei den Defekten handelt es sich ausschließlich um IML-Defekte.

1. Durchsicht (D1)

Anzahl durchgesehener Menüs: 5
Summe der Umfänge der durchgesehenen Menüs: 12251 UE

	Vor (IML)	Vor (HMI)	Sitzung	Nach	Gesamt
Dauer	22	24	18	41	123
Defekte	1	3	0	0	3

Durchsichtsrate: 99,6 UE/min

2. Durchsicht (D2)

Anzahl durchgesehener Menüs: 10
Summe der Umfänge der durchgesehenen Menüs: 17248 UE

	Vor (IML)	Vor (HMI)	Sitzung	Nach	Gesamt
Dauer	28	32	25	0	110
Defekte	3	5	0	0	7

Durchsichtsrate: 156,8 UE/min

3. Durchsicht (D3)

Anzahl durchgesehener Menüs: 6
Summe der Umfänge der durchgesehenen Menüs: 17192 UE

	Vor (IML)	Vor (HMI)	Sitzung	Nach	Gesamt
Dauer	30	13	35	30	143
Defekte	1	0	3	0	4

Durchsichtsrate: 120,22 UE/min

4. Durchsicht (D4)

Anzahl durchgesehener Menüs: 2
Summe der Umfänge der durchgesehenen Menüs: 8766 UE

	Vor (IML)	Vor (HMI)	Sitzung	Nach	Gesamt
Dauer	25	30	42	70	209
Defekte	0	10	0	2	12

Durchsichtsrate: 41,94 UE/min

5. Durchsicht (D5)

Anzahl durchgesehener Menüs: 4
Summe der Umfänge der durchgesehenen Menüs: 7734 UE

	Vor (IML)	Vor (HMI)	Sitzung	Nach	Gesamt
Dauer	30	50	25	0	130
Defekte	1	1	0	0	1

Durchsichtsrate: 59,49 UE/min

Summe über die Durchsichten

Insgesamt wurden in gerundet 12 Personenstunden 27 Menüs mit einem Umfang von 63191 UE durchgesehen. Dabei wurden 27 Defekte in der IML-Datenbasis aufgedeckt.

10.7.2 Defektklassifizierung

Aus der Analyse der aufgedeckten IML-Defekte ergaben sich folgende Defektklassen:

Zustandswechsel: Hierunter fallen alle Defekte, die auf falsche Menüsprünge, Schaltuhren (engl. *timer*) oder andere Widget-Zustandswechsel zurückzuführen sind.

Language-ID: Texte, z.B. für Tastenbeschriftungen, werden in IML zumeist in gesonderten Sprachdateien gespeichert (s. Kapitel 2.4). In den Menü-Widgets werden dafür nur Platzhalter, sogenannte Language-IDs, als Inhalt für Textfelder angegeben. Die Durchsichten zeigten: Es wird mitunter vergessen die Platzhalter, in den Sprachdateien anzugeben. Das heißt es wird in den Menü-Widgets ein nicht definierter Platzhalter verwendet. Auch kam es vor, dass falsche Platzhalter verwendet wurden, so dass in dem betreffenden Menü nicht der gewünschte Text angezeigt wurde.

Funktion: Funktionen werden in IML verwendet um Daten mit den Gerätekomponenten auszutauschen (so könnte z.B. ein Funktionsaufruf „Hörfunk.GetCurrentStationName“ den Namen des aktuell eingestellten Radiosenders liefern. Diesbezügliche Defekte können durch folgende Fehler auftreten: Funktion vergessen, falsche Funktion verwendet, Funktion falsch verwendet (z.B. falscher Parameter).

Bedingung: Es gibt in IML sogenannte "Conditions" (s. Kapitel 2.4), mit denen es möglich ist, Fallunterscheidungen für z.B. Funktionsaufrufe oder Layoutspezifika vorzunehmen. Defekte können hier z.B. dadurch entstehen, dass ein Fall vergessen oder falsch angegeben wurde.

Layout: Viele der in den Widgets spezifizierten Informationen sind aussehensspezifische. Defekte können dabei z.B. durch folgende Fehler auftreten: Falsche Schriftfarbe, -Größe oder –Position gewählt, falscher Dateiname oder Pfad für ein anzuzeigendes Bild spezifiziert.

In der nachfolgenden Tabelle ist die Anzahl der aufgedeckten Defekte nach den beschriebenen Klassen aufgeschlüsselt zusammengestellt.

Defektklassifizierung	**Anzahl aufgedeckter Defekte**
Zustandswechsel	10
Language-Id	3
Funktion	4
Bedingung	2
Layout	6
Widget fehlt/zuviel	2
Summe	27

Tabelle 3: Defektklassifizierung

Die zehn Defekte in den Zustandswechseln waren in vier Fällen auf falsche oder fehlende Schaltuhren zurückzuführen und in sechs Fällen auf fehlerhafte oder fehlende Menüsprünge. Die defekten Schaltuhren bedingten alle gleichzeitig auch defekte Menüsprünge, da sie für sich automatisch schließende Menüs eingesetzt wurden. Die zehn defekten Zustandswechsel waren also ausschließlich defekte Menüsprünge.

Erwähnenswerte Fehler

Mindestens zwei der aufgedeckten Defekte hätten nicht von einem IML-Entwickler alleine aufgedeckt werden können. Einer war auf die Interpretationsspielräume in den illustrierten Statecharts zurückzuführen. Er wurde bereits in Kapitel 7.1.4 erwähnt. Bei der IML-Spezifizierung wurde vergessen, beim Verlassen eines Menüs eine Gerätekomponentenfunktion ausführen zu lassen. Diese Funktionsausführung war jedoch auch nicht in den Statecharts angegeben. In einem Gespräch zwischen dem HMI-Experten und dem IML-Entwickler - während der gemeinsamen Sitzung einer Paardurchsicht - stellte sich heraus, dass diese Auslassung nicht dadurch entstand, dass der HMI-Experte vergessen hatte, sie anzugeben, sondern er war der Meinung, dass es selbstverständlich sei, dass die Ausführung der Funktion bei dem Verlassen des Menüs stattfinden müsse. Deshalb habe er sie gar nicht erst notiert.

Der zweite Defekt entstand aus der unzureichende Versionsverwaltung. Eine Veränderung der Spezifikation bezüglich eines Menüsprungs teilte der HMI-Experte dem IML-Entwickler (vorerst) nur mündlich mit. Er nahm zwar auch eine Modifizierung der Statecharts vor, gab jedoch noch keine neue Version des Statechart-Dokuments an die IML-Entwickler weiter. Dies lag daran, dass sich die IML-Entwickler zuvor über die sich ständig ändernden Statechart-Dokumente beschwerten. Die mündliche Überlieferung wurde vom IML-Entwickler anders gedeutet als erwünscht. Zitat: „Dann haben wir uns da wohl missverstanden.“ [HMI-Experte während der gemeinsamen Sitzung]

10.7.3 Abschätzung des Gesamtaufwands

In der Statistik ist es bei der linearen Regression üblich, zu überprüfen, ob eine hinreichende Korrelation der gegenübergestellten Werte (hier durchgesehener Umfang und Dauer der Durchsicht) gegeben ist. Dazu wird der Korrelationskoeffizient r, wie es z.B. unter [Ger06] beschrieben ist, berechnet. Nach Humphrey ist eine hinreichende Korrelation bei Ressourcenaufwandsschätzungen gegeben, falls r^2 größer oder gleich 0,5 ist [Hum00, S. 12]. Eine diesbezügliche Betrachtung der in den fünf Durchsichten erhobenen Wertepaare ergab jedoch einen Wert von gerundet 0,23.

Die Werte der vierten Durchsicht unterschieden sich - vor allem bezüglich der aufgedeckten Defekte - stark von den Werten der anderen Durchsichten (s. auch Kapitel 10.8.3). Deshalb entschied ich mich dazu, dieses Wertepaar aus der Abschätzung des Gesamtaufwands herauszulassen. Eine Berechnung des Korrelationskoeffizienten für die Wertepaare der restlichen vier Durchsichtsergebnisse ergab jedoch auch noch keinen gewünschten Wert. Ich vermutete, dass vier Wertepaare zu wenig für eine sinnvolle Regression sind. Deshalb entschied ich mich dazu, die Daten einer Nulldurchsicht anstelle der Daten der vierten Durchsicht zu verwenden. Unter einer Nulldurchsicht verstehe ich, eine Durchsicht bei der Menüs mit einem Umfang von Null UE innerhalb von Null Minuten durchgesehen werden. Diese Durchsicht wurde von einem IML-Entwickler alleine nachträglich aufgrund meiner Bitte durchgeführt. Der Korrelationskoeffizient für die so gewählten Wertepaare ist gerundet 0,66, also hinreichend groß.

Die nachfolgende Grafik zeigt die Durchsichtsraten der fünf Durchsichten. Die erwähnte vierte Durchsicht liegt über der 200-Minuten-Marke. Neben den Datenpunkten

der Durchsichtsraten, ist zusätzlich noch die zugehörige magentafarbene Regressionsgerade eingezeichnet.

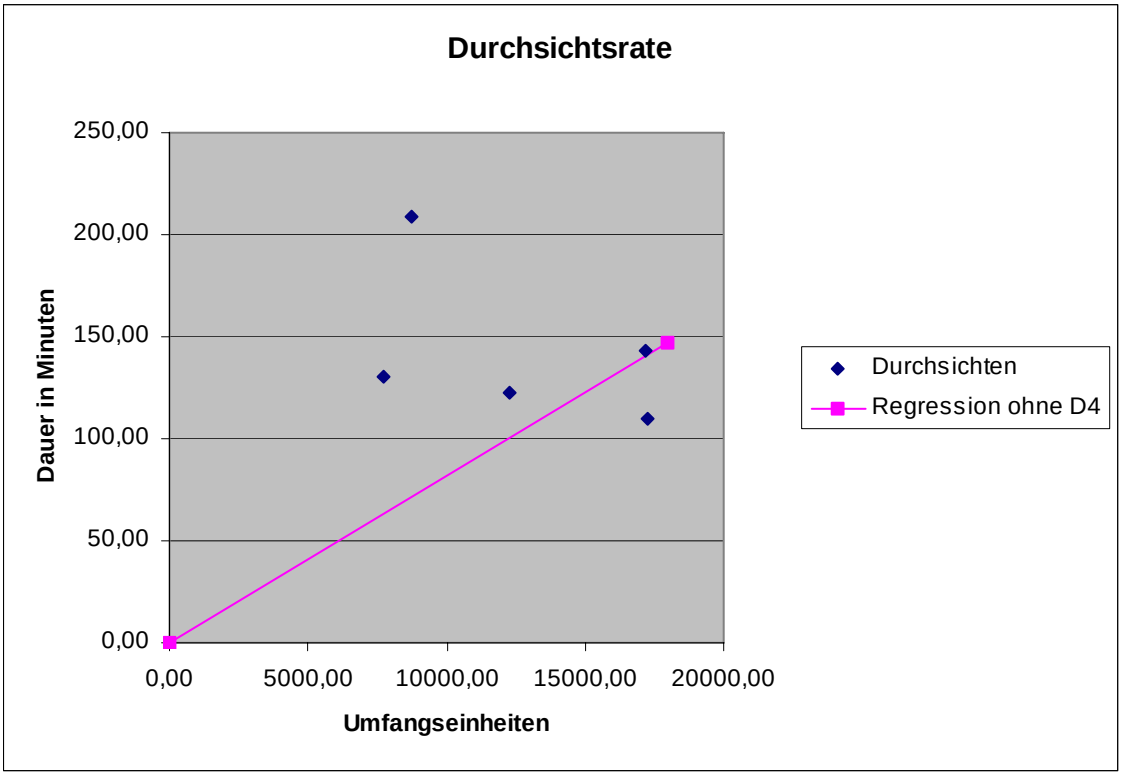

Abbildung 32: Resultat der Durchsichtsstudie (Durchsichtsrate)

Am Tage der letzten Durchsicht maß ich einen Gesamtumfang von 1211970 UE, wobei ich diesen aus der Summe der Umfänge aller in der IML-Datenbasis des Projekts B vorhandenen Menüs berechnete. Die lineare Regression ergab, dass für eine Durchsicht aller Menüs 7860,81 Personenminuten, also 131,01 Personenstunden benötigt werden würden.

10.7.4 Diverse Grafiken

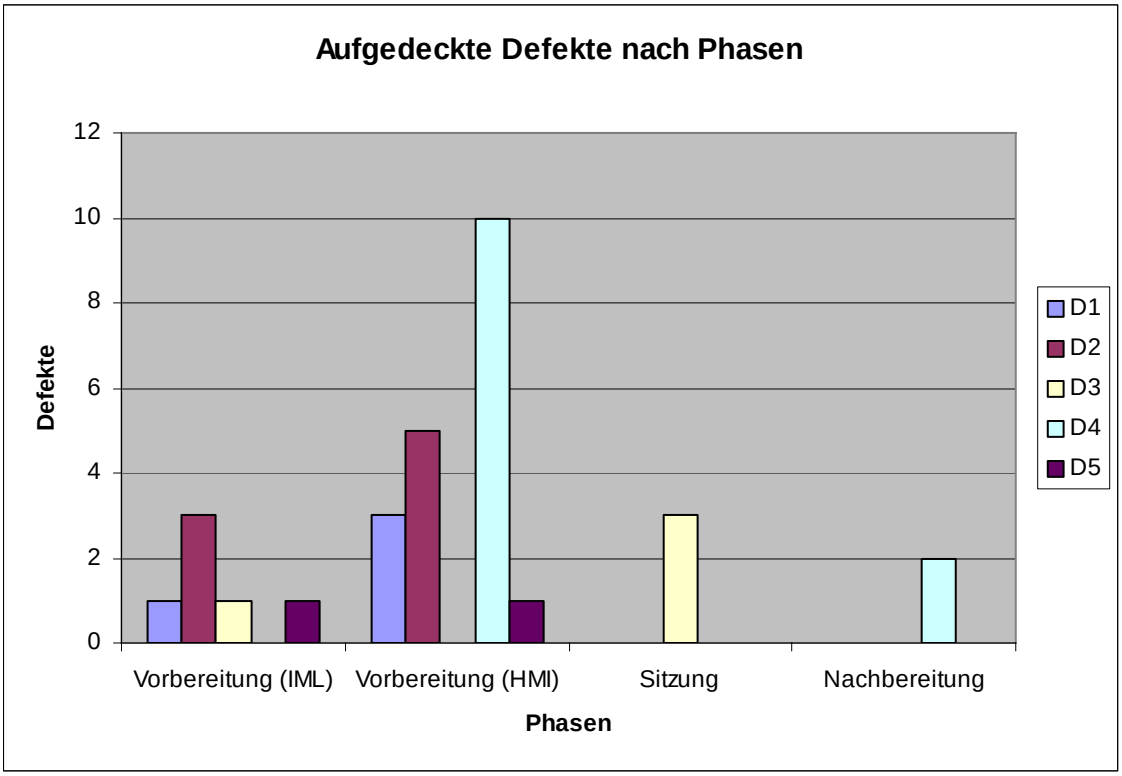

Abbildung 33: Resultat der Durchsichtsstudie (Defekte nach Phasen)

Bei den nach Phasen sortierten Defekten ist die Summe aller Defekte größer, als die Gesamtanzahl. Dies ergibt sich aufgrund der Methodik der Datenanalyse (s. Kapitel 10.6). Defekte, die während der Vorbereitungen gefunden wurden, wurden mitunter von beiden „Partnern“ entdeckt.

Wie in der Grafik zu sehen ist, wurden die meisten der Defekte während der Vorbereitungsphase aufgedeckt. Der HMI-Experte identifizierte dabei im Schnitt mehr Defekte als der IML-Entwickler. Während der Sitzung und der Nachbereitungsphase wurden nur noch kaum Defekte entdeckt.

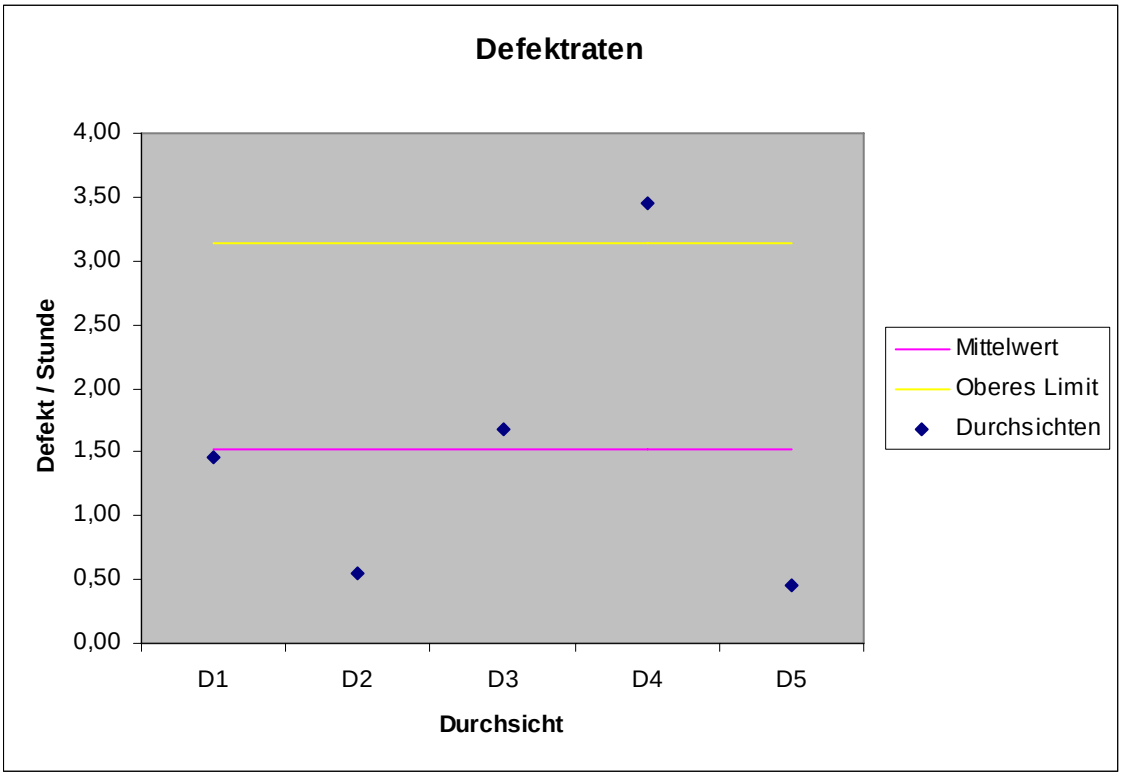

Abbildung 34: Resultat der Durchsichtsstudie (Defektraten)

10.8 Diskussion

Nachfolgend werden die Resultate hinsichtlich der Defektklassifizierung, der Gesamtaufwandsschätzung und weiterer Aspekte diskutiert.

10.8.1 Defektklassifizierung

Die Analyse der Durchsichten hat gezeigt, dass mehr als ein Drittel der in der IML-Datenbasis aufgedeckten Defekte auf fehlerhafte Spezifizierungen von Menüsprüngen zurückzuführen waren. Daraus ziehe ich den Schluss, dass hier ein großes (Prozess-)Verbesserungspotential liegt. Die momentane Handhabung der Spezifizierung von Menüübergangen ist nicht nur zeitaufwändig (doppelte Arbeit), sondern auch fehleranfällig. Hier bedarf es eines besseren Spezifikationsformats als es die illustrierten Statecharts bieten. Optimal wäre eine direkte Überführung der von den HMI-Experten verfassten Spezifikation in das IML-Format. Ein entsprechender Ansatz ist in Kapitel 7.1.6 beschrieben.

Die zweitgrößte Anzahl von Defekten wurde in der Klasse der Layout-Spezifikation festgestellt. Dies ist jedoch nicht verwunderlich, da ein großer Anteil der Informationen in einer HMI-Spezifikation aussehensspezifisch sind.[1] Bemerkenswert ist jedoch, dass diese Defekte, obwohl die Form der Paardurchsicht nicht darauf ausgelegt war, überhaupt aufgedeckt wurden. Das liegt wahrscheinlich daran, dass aussehensspezifische Defekte einem Gutachter sofort „ins Auge springen“.

Eine kritische Defektklasse sind die fehlenden Widgets. Bei einem der beiden Defekte handelte es sich um ein nicht in der IML-Datenbasis vorhandenes Menü. Dadurch,

[1] Dieser Anteil ist (noch) nicht messbar, da im IML-Format das Aussehen nicht vom Verhalten getrennt ist (s. 2.4.3).

dass der Prozess nur unzureichende Unterstützung des Arbeitsflusses (engl. *workflow*) bietet, fiel vor der Durchsicht keinem das Fehlen des Menüs auf. Es existierte in dem Projekt zwar eine Liste über alle Menüs, die in IML bis zu einem gewissen Meilenstein spezifiziert sein mussten. Es handelte sich hier jedoch wieder (wie auch bei den Statechart-Dokumenten) nur um ein schwierig von Maschinen lesbares PDF-Dokument. Ein entsprechendes maschinenlesbares (XML-)Dokument könnte Defekte dieser Klasse vermeiden. Denn damit wäre es realisierbar, ein (kleines) Werkzeug zu bauen, das überprüft, ob alle zu spezifizierenden Menüs vorhanden sind.

10.8.2 Abschätzung des Gesamtaufwands

Wie sich herausstellte wurden von mir zu wenige Datenpunkte gesammelt, um eine vernünftige Abschätzung geben zu können. Erst mit mehr Erfahrungswerten kann eine Berechnung des Gesamtaufwands vorgenommen werden. Da die Durchsichtsraten der verschiedenen Paare enorm schwankten (41,94 UE/min bis 156,8 UE/min) müsste für spätere Hochrechnungen auch eine gesonderte Betrachtung der Paare erfolgen. Da ein Paar hier immer für einen Kontext zuständig war, ist eine Hochrechnung für die einzelnen Kontexte angebracht. Aus der Summe dieser Hochrechnungen würde sich dann wahrscheinlich ein genauerer Schätzwert für den Gesamtaufwand ergeben.

Ein weiterer Kritikpunkt an der verwendeten Methodik ergibt sich aus der unter 10.4.2 beschriebenen Wahl der durchzusehenden Menüs. Nach Aussage eines IML-Entwicklers ist die Spezifizierung und die Durchsicht von Menüs, deren Spezifikation über mehrere Seiten des Statechart-Dokuments verteilt ist, um einiges aufwändiger. Das liegt daran, dass viele benötigte Informationen zunächst einmal im gesamten Dokument gesucht werden müssen. Dieser Faktor fließt in die hier vorgenommen Aufwandsschätzung nicht ein.

Auch die Berechnung des Menüumfangs ist nicht exakt genug. Die Menülogik wurde zu einem großen Anteil in Widgets (Menünetze, „Menünetz-Handler“), die den Menüs übergeordneten sind, spezifiziert. Dieser Anteil wird bei der Berechnung des Menüumfangs nicht berücksichtigt.

Für eine aussagekräftige Aufwandsschätzung müsste also noch einiges an der Methodik verbessert werden. Es ist jedoch zu beachten, dass, wie in Kapitel 3.2 bereits erwähnt wurde, die Suche nach einem perfekten Schätzmaß ein Widerspruch in sich.

Wie oben schon des Öfteren erwähnt (s. Kapitel 8), sollte die hier vorgeschlagene Form der Durchsicht in den Entwicklungsprozess integriert werden, um eine größtmögliche Effizienz zu erzielen. Bei einer Integrierung würde das Schätzen des Gesamtaufwands entfallen. Der geschätzte Wert von gerundet 131 Personenstunden zeigt jedoch, dass der Mehraufwand einer Integrierung nicht besonders hoch wäre im Vergleich zu den Personenjahren, sie für die Spezifizierung und die Tests in Projekt B aufgewendet wurden.

10.8.3 Weitere Aspekte

Um eine effiziente Durchsicht zu garantieren, ist es zwingend notwendig, dass die Resultate, die an das Management weitergegeben werden, anonymisiert werden. Ansonsten könnte es das Ziel der Beteiligten sein, so wenig Defekte wie möglich aufzudecken. Das Ziel einer Durchsicht ist aber genau das Gegenteil. Da die Endresultate für die Bewertung des Prozesses und des Projektsstands gerade für das Management interessant sind, bedarf es also eines Durchsichtsbeauftragten. Dieser muss unabhängig vom Management und der Projektleitung sein und die Daten so aufbereiten, dass ein Rückschluss auf an der Durchsicht beteiligte Personen nicht möglich ist [Hum89]. In dieser Studie war z.B. eine Aufschlüsselung der Daten (Defekte, Dauer) nach den einzelnen Kontex-

ten nicht möglich, da daraus ersichtlich gewesen wäre, welche Personen an welcher Durchsicht beteiligt waren. Es muss ausgeschlossen werden, dass mit den mittels Durchsichten erhobenen Daten eine Projekt-Ressourcen-Vermessung vorgenommen werden kann. Theoretisch könnte es ansonsten sein, dass das Management die Daten für Personalentscheidungen verwendet.

Nach der Studie stellte sich heraus, dass die Menüs, die in Durchsicht vier inspiziert wurden, aufgrund der hohen Defektrate komplett überarbeitet werden mussten. In so einem Fall wird in der Literatur vorgeschlagen, eine erneute Durchsicht (der Menüs) durchführen zu lassen. Normalerweise hat dies der Moderator am Ende einer Durchsicht zu entscheiden. Die Form der Paardurchsicht sieht jedoch, wie unter beschrieben, diese Rolle aus Effizienzgründen nicht vor (s. Kapitel 8.5.2). Die Frage, die sich hier deshalb aufdrängte war, ob es möglich ist, aus den erhobenen Daten abzulesen, wann eine erneute Durchsicht erforderlich ist. Humphrey schlägt für die Lösung dieses Problems [Hum89, S. 319 ff.] vor, ein oberes Kontrolllimit (engl. *upper contol limit* (UCL)) der Defektrate (aufgedeckte Defekte pro Stunde) mit Hilfe folgender Formel zu ermitteln.

$$UCL = \text{Mittelwert(Defektraten)} + N * \text{Standardabweichung(Defektraten)}$$

Der Faktor N ist dem Untersuchungskontext anzupassen und sollte zwischen Null und drei liegen [Hum89, S. 330]. Durchsichten, bei denen die Defektrate über diesem Limit liegt, sollten wiederholt werden.

Die Defektraten der Durchsichten und ein oberes Kontrolllimit ist in Abbildung 34 dargestellt. Aus mangelnder Erfahrung mit diesem Maß wählte ich für den Faktor N den Wert 1,5 , was der Mitte des von Humphrey vorgeschlagenen Bereiches entspricht. Da die Anzahl der Datenpunkt nicht ausreichend ist, handelt es sich hier nur um einen ersten Versuch, dem beschriebenen Problem zu begegnen. Das heißt, dass Kontrolllimit müsste bei weiteren vermessenen Durchsichten kontinuierlich angepasst werden. Erst mit mehr Erfahrung ließe sich dieses Maß als Indikator verwenden. Einem Durchsichtsbeauftragtem könnte dessen Betrachtung bei der Entscheidung bezüglich einer Wiederholung der Durchsicht behilflich sein.

Das Erstellen der Durchsichtsanleitung war gleichzeitig auch ein Zusammenstellen der besten Verfahrensweise (engl. *best practice*) für von den Entwicklern eingesetzte Werkzeuge. Es stellte sich auch heraus, dass die Entwickler nicht immer den vollen Funktionsumfang der Werkzeuge kannte. Ein regelmäßiges Zusammenstellen der besten Verfahrensweise in Verbindung mit den Paardurchsichten, kann deshalb auch als Schulungsmaßnahme dienen und so zu einer Effizienzsteigerung des Entwicklerteams beitragen.

Ein weiteres Resultat der Erforschung der besten Verfahrensweise war das Aufdecken von Schwächen der verwendeten Werkzeuge, wie es unter anderem in Kapitel 7.4 beschrieben ist. Ein Indiz für eine Schwäche bezüglich der Benutzbarkeit eines Werkzeugs war jeweils, dass eine zu verwendende Verfahrensweise nur kompliziert zu beschreiben war (vgl. 7.4.2).

11 Abschließende Beurteilungen

In diesem Kapitel beurteile ich abschließend die Vorgehensweise, die ich in dieser Arbeit angewendet habe. Dies betrifft die Untersuchungsmethodiken, die dieser Arbeit zugrunde lagen und die in der Arbeit konzipierten Paardurchsichten. Auch Ausblicke zu den Themen werden erläutert.

11.1 Beurteilung der Untersuchungsmethodik

In diesem Abschnitt beurteile ich zunächst die in der Arbeit verwendete Untersuchungsmethodik. Dabei geht es zum einen um die allgemeine Methodik und zum anderen um die von mir gewählte Form der Interviews.

11.1.1 Allgemeines Vorgehen

Nach ISO 9000:2000 [DIN00, S. 14] ist es (verallgemeinert) eine der Grundfragen zur Beurteilung eines Prozesses, ob der Prozess festgelegt und in geeigneter Weise beschrieben ist. Für den TeleDrive VISION Prozess war zumindest der zweite Teil der Frage zu Beginn der Arbeit mit nein zu beantworten. Deshalb war es notwendig, zuerst einmal das Prozessmodell aufzunehmen. Erst nachdem ich das Prozessmodell analysiert hatte, war es mir möglich, Stärken und Schwächen des Prozesses aufzuzeigen.

Neben der Stärken-Schwächen-Analyse war es ein Ziel dieser Arbeit, eine Quantifizierung der Prozessqualität zu ermöglichen. Auch hierfür bedurfte es des schriftlichen Prozessmodells, denn „Metriken sind nur dann sinnvoll, wenn ein Prozeß [Prozess] zur Software-Entwicklung bereits definiert und festgeschrieben wurde.“ [Tha93, S. 262]

Wie sich im Laufe der Untersuchung gezeigt hat, ist der TeleDrive VISION Prozess an vielen Stellen noch nicht reif genug für ein effizientes Messprogramm. Humphrey führt in seinem Prozessreifegradmodell *Capability Maturity Model* (CMM) [Hum89] nicht ohne Grund Messungen erst auf der vierten Stufe („Managed Process“) ein [Tha93, S. 262]. Diese Stufe ist erst erreicht, wenn die im Prozess verwendeten Verfahren sowohl bei der Softwareentwicklung als auch im Management verfeinert und standardisiert sind. [Tha93, S. 217] „Erst auf dieser Ebene sind nämlich die Voraussetzungen dafür gegeben, um mit den Ergebnissen von Messungen sinnvolle Verbesserungen einführen zu können.“ [Tha93, S. 262]

Die Verwendung der GQM-Methode (s. Kapitel 3.3) für die Messungen stellte sich mitunter als schwierig heraus. Es wäre sicherlich einfacher gewesen, ziellos zu messen. Der Aufwand hat sich jedoch gelohnt, denn viele der Ergebnisse und Lösungsvorschläge dieser Arbeit sind nur entstanden, weil ich zielgerichtet gemessen habe. Als Beispiel sei hier die Defektklassifizierung erwähnt, die ich in der Paardurchsichtsstudie (s. Kapitel 10) vorgenommen habe.

11.1.2 Interviews

Die durch die Interviews gewonnenen Daten bildeten für die Aufnahme des Prozessmodells und die Identifizierung der Schwachstellen im Prozess lediglich einen Ausgangspunkt, der in weitere Untersuchungen mündete. Die Interviews waren sehr hilfreich, weil sie mir erste Einblicke boten, wo sich z.B. welche Schwachstellen befinden. Sie hätten jedoch alleine niemals ausgereicht, da mir viele Problembereiche erst bei der späteren Analyse der Aufzeichnungen auffielen. Daraufhin kam es des Öfteren zu formlosen Zusatzbefragungen.

Bei diesen Befragungen ging es zum einen um den Erhalt von Informationen über Prozessdetails, wie z.B. dem Aufbau von bestimmten Dokumenten, die ich für die Erstellung des Prozessmodells benötigte. Zum anderen kam häufig zu Zusatzbefragungen bezüglich identifizierter potentieller Probleme und dazu passenden Lösungsvorschlägen. Mitunter führte dies auch zu längeren Diskussionen. Mit mehr Erfahrung hätte ich aber teilweise schon während der Interviews verstärkt auf bestimmte Bemerkungen – vor allem in Bezug auf Schwachstellen im Prozess - reagieren können. Da mir diese Erfahrung aber zumindest zum Zeitpunkt der Studiendurchführung noch fehlte, war ich dazu nicht immer in der Lage.

Eine Alternative wäre es gewesen, die Interviews in zwei Etappen durchzuführen. Das hätte den Vorteil gehabt, dass ich mich in einem gesonderten Interview auf die Probleme allein hätte konzentrieren können. Andererseits vermute ich, dass es bei meinen Interviews gerade von Vorteil war, dass es nicht nur um Probleme ging. Dadurch konnten die Befragten sich erst einmal mit dem Berichten über unbestrittene Fakten ihres Expertenwissens „warm reden“. Und die Interviews hatten somit keinen ausschließlich kritikübendenden Charakter.

Bei der Analyse der Daten aus den Interviews stellte es sich mitunter als schwierig heraus, eine Trennung zwischen dem der Vorstellung der Erfinder entsprechenden Zustand des Prozesses und dem in den Projekten gelebten herzustellen. Auch die Trennung von projektspezifischen und prozessimmanenten Charakteristika war nicht immer einfach. Vor allem die Beschreibung des Gesamtprozesses sollte eher präskriptiven als deskriptiven Charakter haben. Auch die Detailbetrachtung des Spezifizierungsprozesses sollte nur immanent notwendige und keine projektspezifischen Merkmale enthalten. Darunter fallen in erster Linie Verschiebungen des Prozessablaufs durch unideale Zeitplanungen. Es kam in Projekt B z.B. mitunter vor, dass die Grafikspezifikation noch nicht die Vermassung eines bereits zu spezifizierenden Menüs enthielt. Zur Validierung befragte ich bei einzelnen Merkmalen mehrere IAV-Mitarbeiter, ob die Merkmale projektspezifisch oder prozessimmanent sind.

Bei der Untersuchung des Spezifizierungsprozesses stellte es sich mitunter als schwierig heraus, die einzelnen Teilprozesse und Rollen von einander zu trennen, da die an den Projekten beteiligten Mitarbeiter zumeist nicht nur eine Rolle einnahmen und auch nicht nur für einen Teilprozess zuständig waren. So waren manche der IML-Entwickler z.B. mitunter auch für den Bau der Simulationssoftware zuständig. Nach weiteren Befragungen ließen sich diese Probleme jedoch lösen.

Ein Schwachpunkt meiner Untersuchung bezüglich des Prozessmodells ist, dass sie durch die Betrachtung von lediglich zwei Projekten nicht repräsentativ genug ist, um ausreichend Verallgemeinerungen für ein Prozessmodell treffen zu können. Eine Betrachtung weiterer Projekte, die den Prozess mit IML-Entwicklung leben, war jedoch nicht möglich, weil es zum Zeitpunkt der Untersuchung keine weiteren gab.

11.2 Beurteilung der Paardurchsicht

Eine Einführung von Paardurchsichten wäre nicht kostenlos. Ihr Einsatz würde sich jedoch im Laufe eines Projektes sicherlich auszahlen. Um dies zu erklären, werden in diesem Abschnitt zunächst ausgewählte Gründe dargelegt, die für eine Einführung der Paardurchsicht sprechen. Danach werde ich einen Ausblick geben, wie die Paardurchsichten zukünftig noch effizienter eingesetzt werden könnten. Außerdem werde ich betrachten, inwiefern die Einführung des HMIStudios eine Bedrohung (engl. *threat*, vgl. SWOT-Analyse) darstellen könnte.

11.2.1 Gründe für eine Einführung

Ich plädiere in erster Linie für eine Integration der Paardurchsicht in den TeleDrive VISION Prozess, weil eine Verifizierung der Mensch-Maschine-Schnittstelle bisher noch nicht durch deren Erfinder erfolgt. Mit der Anfertigung der illustrierten Statecharts spezifizieren die HMI-Experten das Aussehen und Verhalten der Schnittstelle. Die IML-Entwickler setzen diese Spezifikation „lediglich“ in Form von Software um. Nur die HMI-Experten können überprüfen, ob die Umsetzung ihren Vorstellungen entspricht.

Neben diesem Hauptaspekt existieren noch weitere Gründe für die Einführung der Paardurchsicht. Wie sich gezeigt hat, ließen sich eine Reihe von Prozessverbesserungsvorschlägen aus den im Zuge der Durchsichten erhobenen Daten ableiten und dies, obwohl hier nur rund zwölf Personenstunden aufgewendet wurden.[1] Aus einer Integration der Paardurchsicht in den Prozessablauf könnten derartige Vorschläge kontinuierlich resultieren. Dadurch ließe sich der Prozess, wenn auf die Vorschläge eingegangen wird, auch kontinuierlich verbessern. „Das Ziel der ständigen Verbesserung [...] besteht darin, die Wahrscheinlichkeit zu steigern, die Zufriedenheit der Kunden und anderer interessierter Parteien zu erhöhen.“ [DIN00, S. 16] Im Falle der IAV wären dies die Fahrzeughersteller und die Zulieferer.

Auch ließe sich mit Hilfe der Paardurchsicht die Produktqualität höchstwahrscheinlich effizienter steigern als mit den bisherigen Prüfmethoden. In zwölf Personenstunden wurden 27 Defekte direkt an die jeweiligen Entwickler herangetragen. Wie sich in den gemeinsamen Sitzungen herausstellte, war ihnen zumeist sofort bewusst, an welcher Stelle in der IML-Datenbasis sich der Defekt befindet.

Wenn Defekte dieser Art von einem Zulieferer oder einem Tester berichtet werden, ist der Kommunikationsaufwand hierfür um einiges größer. In Projekt B wurden z.B. Defekte, die ein Zulieferer berichtete, zumindest bis Januar 2006 per Email an die IAV verschickt. Hier fiel zumindest der Aufwand für die Identifizierung des Defekts und für das Schreiben einer Email auf Seiten des Zulieferers an. Danach musste ein IAV-Mitarbeiter die formatlose, natürlichsprachige Email analysieren und entscheiden, ob es sich wirklich um einen Defekt handelte und wenn dies der Fall war, welcher IML-Entwickler für die Behebung zuständig war. Sodann musste der betreffende IML-Entwickler suchen, wo sich der Defekt in der IML-Datenbasis befindet. Der Kommunikationsweg für einen spät entdeckten Defekt war also enorm lang.

Da diesbezügliche Zeiten nicht (mehr) von mir gemessen werden konnten, war ein Effektivitätsvergleich leider nicht mehr durchführbar. Jedoch sollte man sich ein Bild aus der vorrangegangenen Diskussion machen können.

Nicht nur eine (vermutliche) Effizienzsteigerung beim Aufdecken und Beheben von Defekten würde aus einer Einführung der Paardurchsichten resultieren. Bei den Interviews und auch bei späteren Befragungen und Beobachtungen zeigte sich nicht nur ein Kooperationsdefizit zwischen den IML-Entwicklern und den HMI-Experten. Mitunter entstand der Eindruck, die HMI-Experten und die IML-Entwickler würden gegeneinander und nicht miteinander arbeiten (s. z.B. Kapitel 7.1.5). Um dem entgegenzuwirken und das Teamgefühl zu stärken, sind meiner Ansicht nach die Paardurchsichten ein geeignetes Mittel.

11.2.2 Zukünftige Datenerhebung

Wie jeder andere Prozess bedarf auch der hier vorgeschlagene Prozess der Paardurchsicht einer kontinuierlichen Verbesserung. Es folgen hier deshalb einige Verbesserungsvorschläge bezüglich der dabei stattfindenden Datenerhebung.

[1] Es fließt hier jedoch nicht die Zeit ein, die ich für die Studie aufgewendet habe.

Für die Analyse der aus Paardurchsichten stammenden Daten wird, wie oben bereits erläutert (s. Kapitel 10.8.3), ein Durchsichtsbeauftragter benötigt. Um dessen Arbeitsaufwand jedoch so gering wie möglich zu halten, sollte versucht werden, die Daten bei der Durchführung der Durchsicht so detailliert wie möglich zu erfassen. In der Studie wurden z.B. die IML-Defekte immer ohne Klassifizierung in den Protokollen notiert. Das lag daran, dass eine Defektklassifizierung für die Sprache IML zuvor nicht vorhanden war. Für mich als „Durchsichtsbeauftragter“ war der Aufwand für das Klassifizieren der aufgedeckten Defekte einer der größten. Bei späteren Durchsichten könnten die Defekte nun mit Hilfe der (kontinuierlich zu erweiternden) Defektklassifizierung bereits von den Paaren klassifiziert werden. Somit müsste ein Durchsichtsbeauftragter nur noch die Defekte der einzelnen Klassen addieren. Das Sammeln der Daten sollten die Paare also alleine vollziehen, ähnlich wie beim *Personal Software Process* von Humphrey (vgl. [Hum00]), nur dass es hier dann ein *Personal Pair Software Process* wäre.

Der Durchsichtsbeauftragte sollte also nur noch für die Datenanalyse zuständig sein. Diese Tätigkeit könnte auch von einem (noch zu entwickelnden) Werkzeug übernommen werden, dem die Paare die erhobenen Daten „mitteilen“. Dieses Werkzeug könnte dann die Daten (anonym!) aufbereiten. Das würde ein in Kapitel 10.8.3 bereits erörtertes Problem lösen. Das dort identifizierte Problem war, dass nur eine gegenüber dem Management anonymisierte Datensammlung eine effiziente Durchsicht garantiert. Die von einem Werkzeug anonym aufbereiteten Daten könnten von IAV-Mitarbeitern aller Ebenen verwendet werden, um notwendige Prozessverbesserungen abzuleiten.

Zusätzlich zu den unter Kapitel 10.3 beschriebenen Messungen sollte außerdem noch die Zeit, die zur Behebung von Defekten notwendig war, vermerkt werden, und zwar auch für Defekte, die durch andere Prüfmethoden aufgedeckt werden. Damit wäre ein Effektivitätsvergleich der Prüfmethoden zu bewerkstelligen.

11.2.3 Bedrohlicher Ausblick

Während der letzten Monate stellte ich mir des Öfteren die Frage, ob die Paardurchsichten überhaupt eine Zukunft haben. Ich fragte mich, ob dieser Verifizierungsschritt bei Einsatz des HMIStudios nicht hinfällig werden würde. Die IAV hat sich vorgenommen, mit der Einführung dieses Werkzeugs die Rollen IML-Entwickler und HMI-Experten später in einer Rolle zu vereinen. Dadurch gäbe es später möglicherweise keinen Bedarf mehr nach einer Durchsichtsform, wie ich sie mittels dieser Arbeit vorschlage. Nach Aussage von Herrn Wegner steht der Vereinigung der beiden Rollen aber noch ein langer Weg bevor. Auch werde es immer Stellen im Prozess geben, an denen ein „reiner“ IML-Entwickler benötigt wird, wie z.B. bei der Spezifizierung der HMI-API (s. Kapitel 6.2.5). Das impliziert, dass es immer Stellen im Prozess geben wird, an denen ein HMI-Designer und ein Entwickler kooperieren müssen. Dadurch werden manuelle Prüfmethoden auch bei einem späteren Einsatz des HMIStudios noch relevant für den Prozess sein.

Bis es soweit ist, dass eine Rolle alleine die Arbeiten beider Rollen zusammen verrichten kann, können die Paardurchsichten außerdem dazu dienen, den Rollen Einblicke in die Arbeit der jeweils anderen Rolle zu verschaffen. Wenn die Paardurchsicht dann noch auf eine Paarentwicklung erweitert werden würde, wären die derzeitigen Mitarbeiter beider Rollen nach einer gewissen Zeit vielleicht in der Lage, die gesamte Arbeit alleine zu verrichten und die Rollen IML-Entwickler und HMI-Experte würden zu einer Rolle „HMI-IML-Experte“ zusammenwachsen. Eine Paarentwicklung mit HMIStudio ist sogar in jedem Fall kostengünstiger als die bisherige Verfahrensweise, da zur Zeit auch zwei Personen ein Menü spezifizieren. Dies tun sie nur nicht zusammen sondern

nacheinander. Diese Vorgehensweise dauert zum einen länger (im Sinne von Zeit bis zur Markeinführung (engl. *time-to-market*)) und zum anderen entstehen, wie gezeigt wurde, Defekte auf dem Übertragungsweg. Bei einer Paarentwicklung würden diese Defekte höchstwahrscheinlich nicht entstehen. Dadurch, dass die Entwickler bei einem Einsatz des Werkzeugs später ihre Arbeit mit dem WYSIWYG-Editor überprüfen können, testen sie die IML-Datenbasis praktisch schon während der Spezifizierung.

Literaturverzeichnis

Wenn es nicht anders angegeben ist, handelt es sich bei den Büchern um die erste Auflage. Bei den Zitaten, die aus dem Wikipedia stammen, handelt es sich zumeist lediglich um eine Wiederverwendung schöner Formulierungen.

[Bal96] Helmut Balzert: "Lehrbuch der Software-Technik: Software-Entwicklung", Spektrum, Akademischer Verlag, Heidelberg, 1996

[Bal98] Helmut Balzert: "Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung", Spektrum, Akademischer Verlag, Heidelberg 1998

[BeR03] Marcel Bennicke, Heinrich Rust: „Software-Produktanalyse", TU Cottbus, 2003

[BeR04] Marcel Bennicke, Heinrich Rust: „Messen im Software-Engineering und metrikbasierte Qualitätsanalyse", Virtuelles Software Engineering Kompetenzzentrum, Januar 2004

[Bis89] D.B. Bisant, J.R. Lyle: "A Two-Person Inspection Method to Improve Programming Productivity", IEEE Transactions on Software Engineering, vol. 15, no. 10, pp. 1294–1304, Oktober 1989

[BMW05] WWW-Auftritt der BMW AG: "iDrive", http://www.bmw.com/generic/de/de/fascination /technology/lexicon/content_reload.html?articleUrlStartidrivearticleUrlEnd, letzter Abruf am 15. November 2005

[Boe86] Barry W. Boehm, „Wirtschaftliche Software-Produktion", Forkel-Verlag, Wiesbaden, 1986

[BR88] Victor R. Basili and H. Dieter Rombach: "The TAME Project: Towards improvement–oriented software environments", IEEE Transactions on Software Engineering, SE-14(6):S. 758–773, Juni 1988

[Bri98] L. Briand, K. E -Emam, T. Fussbroich, O. Laitenberger: "Using Simulation to Build Inspection Efficiency Benchmarks for Development Projects", Proceedings of the 20th International Conference on Software Engineering, pp. 340-349, 1998

[BW84] Victor R. Basili and David M.Weiss: "A methodology for collecting valid software engineering data", IEEE Transactions on Software Engineering, SE-10(6): S. 728–738, November 1984

[CSE06] Centre for Software Engineering: "ISO 9126: The Standard of Reference", Dublin City University, http://www.cse.dcu.ie/essiscope/sm2/9126ref.html, letzter Abruf am 16. Januar 2006

[DIN95] Europäische Norm DIN-EN 614-1: „Sicherheit von Maschinen – Ergonomische Gestaltungsgrundsätze", Beginn der Gültigkeit 10. Februar 1995

[DIN00] Europäische Norm DIN-EN ISO 9000:2000-12: „Qualitätsmanagementsysteme – Grundlagen und Begriffe", Dezember 2000

[Dum06] Reiner Dumke: „Softwarequalitätsmanagement – 2. Lehrhilfe zur Vorlesung: Produktqualität", http://ivs.cs.uni-magdeburg.de/~dumke/ST2/ST2Produkt.html, letzter Abruf am 17. Januar 2006

[Fäh02] Klaus-Peter Fähnrich: "Software-Management Vorlesung", 8. Vorlesung: Kontrolle, http://ais.informatik.uni-leipzig.de/studium/vorlesungen/vorlesungen_2002_ss.html , Sommersemester 2002, letzter Abruf am 17.11.2005

[Fen91] N.E. Fenton: "Software Metrics: A Rigorous Approach", Chapman and Hall, New York, 1991

[Fre01] Bernd Freimut: "Developing and Using Defect Classification Schemes", Fraunhofer IESE-Report Nr. 072.01/E, September 2001

[Ger06] Jutta Gerhard, “Lineare Regression“, Statistik2, http://members.chello.at/gut. jutta.gerhard/kurs/statistik2.htm, letzter Abruf 7. Februar 2006

[GHJV96] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (aka. “Gang of Four”): “Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software”, Addison-Wesley, Reading, Mass., 1996

[GiS05] Cord Giese, Rüdiger Schilling: „Modellgetriebene Generatorentwicklung“, bit-fabrik, März 2005

[Gra94] Robert B. Grady: “Successfully Applying Software Metrics”, IEEE Computer, vol. 27, no. 9, pp. 18-25, September 1994

[Gra97] Robert B. Grady: “Successful Software Process Improvement”, Hewlett-Packard Company, Prentice Hall, 1997

[HaM03] W. Hamberger, G. Mauter, “Audi Multi Media Interface (MMI) – Neue Spezifikationsmethoden zur interdisziplinären Bedienkonzeptentwicklung”, VDI-Berichte 1768, VDI Verlag GmbH, Düsseldorf, 2003

[Har87] D. Harel: „Statecharts: A Visual Formalism for Complex Systems“, Science of Computer Programming, vol. 8, #3, S. 231-274, 1987 (nicht von mir gelesen, aber vom Balzert)

[Hei06] heise online (Richard Sietmann), „Neuer Entwicklungsstandard für IT-Systeme des Bundes“, http://www.heise.de/newsticker/meldung/56028, letzter Abruf am 6. Februar 2006

[Hol97] Patricia Holland: „The television handbook“, Routledge, London, 1997, gelesen von Julius Stucke (vielen Dank für das Zitat)

[Hum89] Watts S. Humphrey: “Managing the Software Process”, Addison-Wesley, Reading, MA, 1989

[Hum95] Watts S. Humphrey: “A Discipline for Software Engineering”, Addison-Wesley, Januar 1995

[Hum00] Watts S. Humphrey: “The Personal Software Process (PSP)”, technischer Bericht, CMU/SEI-2000-TR-022, ESC-TR-2000-022, November 2000

[IAV05] IAV GmbH: „IAV – Die Ingenieursgesellschaft mit der Kompetenz für das ganze Fahrzeug“, Informationsbroschüre über das Firmenprofil, August 2005

[IAV06] WWW-Auftritt der IAV GmbH, http://www.iav.de, letzter Abruf am 8. Februar 2006

[IML05] IAV GmbH, „Infotainment Markup Language – Allgemeiner Aufbau und Struktur”, Version vom 16. September 2005

[Jai05] A. Jain: “Vorlesung Statistik 1”, Vorlesung und Übung des Wintersemesters 2005 /2006, http://www.uni-koeln.de/phil-fak/psych/methoden/statistik_WS0506/uebung/uebung_2.pdf, letzter Abruf am 18.11.2005

[JF88] Ralph E. Johnson, Brian Foote: „Designing reusable classes“, Journal of Object-Oriented Programming, 1(2):22–35, 1988

[KBSt05] Andreas Rausch et al. für die Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung: “V-Modell XT“, Version 1.1.0, Kaiserslautern, August 2005

[Lai02] O. Laitenberger: "A Survey of Software Inspection Technologies", Handbook on Software Engineering and Knowledge Engineering, vol. 2, World Scientific Publishing, pp. 517-555, 2002

[Lam05] Lamnek, S.: „Qualitative Sozialforschung. Ein Lehrbuch“, Seiten 329 – 407, 2005, 4. Auflage., Weinheim: Beltz Vlg..

[Joh92] Peter Johnson: “Human Computer Interaction: psychology, task analysis and software engineering”, McGraw-Hill, 1992

[John01] Phillip Johnson: “You can’t even ask them to push a button: Toward ubiquitous, developer-centric, empirical software engineering”, The NSF Workshop for New Visions for Software Design and Productivity: Research and Applications, Nashville (TN), Dezember 2001

[Jud04] Andreas Judt: “Konfigurierbare Benutzerschnittstellen zur Vereinfachung formularbasierter Datenerfassung“, Dissertation an der Universität Karlsruhe, 2004

[KoS05] Koschke, Simon: „Software-Metriken“, Vorlesung „Software-Reengineering“ an der Universität Stuttgart im Wintersemester 2004 / 2005, http://ftp.informatik.uni-stuttgart.de/iste/ps/Lehre/reengineering/neu/metriken.pdf, letzter Abruf am 18.11.2005

[Kit96] Barbara Kitchenham: “Software Metrics – Measurement for Software Process Improvement”, Blackwell Publishers Inc., Cambridge, 1996

[Kru06] Andreas Krutscher: “Modell-Ansicht-Bediener“, http://www.fh-wedel.de/~si/seminare/ws97/Ausarbeitung/3.Krutscher/archmu3.htm , letzter Abruf am 20. Januar 2006

[Mil88] Everald E.Mills: “Software Metrics“, SEI Curriculum Module SEI-CM-12-1.1, Carnegie Mellon University, Dezember 1988

[MS05] WWW-Auftritt der Microsoft Corporation: “Microsoft Technology Hits the Road in BMW 7 Series“, http://www.microsoft.com/presspass/press/2002/mar02/03-04BMWpr.mspx, letzter Abruf am 15. November 2005

[MS06] WWW-Auftritt der Microsoft Corporation: “Code Name Avalon - Create Real Apps Using New Code and Markup Model“, http://msdn.microsoft.com/msdnmag/issues/04/01/Avalon/ , letzter Abruf am 20. Januar 2006

[Pfl97] S. L. Pfleeger, R. Jeffery, B. Curtis, B. Kitchenham: “Status Report on Software Measurement”, IEEE Software, vol. 14, no. 2, Seiten 33-43, März / April 1997

[Por97] Adam A. Porter , Harvey P. Siy , Carol A. Toman , Lawrence G. Votta: “An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development”, IEEE Transactions on Software Engineering, v.23 n.6, pp.329-346, Juni 1997

[Pre99] Lutz Prechelt: “Ausgewählte Kapitel der Softwaretechnik“, http://page.mi.fu-berlin.de/~prechelt/swt2/skript.html , Vorlesung im Sommersemester 1999, letzter Abruf am 18. November 2005

[Pre03] Lutz Prechelt: „Prozeßmodelle“, Vorlesung „Softwaretechnik“, Wintersemester 2002 / 2003, letzter Abruf am 5. Januar 2006

[Pre06] Lutz Prechelt, „Software-Prozessmodelle“, Vorlesung „Softwaretechnik“, Kapitel 16, Wintersemester 2005 / 2006, letzter Abruf am 5. Januar 2006

[Pres97] Roger S. Pressman: “Software Engineering: A Practitioner’s Approach”, 4. Auflage, McGraw Hill, New York, 1997

[Rei02] Ralf Reißing: “Bewertung der Qualität objektorientierter Entwürfe”, Dissertation an der Universität Stuttgart, August 2002

[Rys03] Johannes Ryser: “Szenarienbasiertes Validieren und Testen von Softwaresystemen”, Dissertation der wirtschaftswissenschaftlichen Fakultät der Universität Zürich, Februar 2003

[Shn80] Ben Shneiderman: “Software Psychology: Human Factors in Computer and Information Systems”, Winthrop Publishers, Cambridge (USA, MS), 1980

[SoB99] Rini van Solingen, Egon Berghout: “The Goal/Question/Metric Method: a practical guide for quality improvement of software development", McGraw-Hill, 1999

[SoK06a] Virtuelles Software Engineering Kompetenzzentrum: “Erläuterung: Qualität”, http://www.software-kompetenz.de/?14884, letzter Abruf am 2. Januar 2006

[SoK06b] Virtuelles Software Engineering Kompetenzzentrum: “Prozess zur Definition eines Maßes – Zielbestimmung”, http://www.software-kompetenz.de/?10954, letzter Abruf am 10. Februar 2006

[SoK06c] Virtuelles Software Engineering Kompetenzzentrum: “Abnahmetest”, http://www.software-kompetenz.de/?17364, letzter Abruf am 10. Februar 2006

[SoK06d] Virtuelles Software Engineering Kompetenzzentrum: “SPICE”, http://www.software-kompetenz.de/?17149, letzter Abruf am 10. Februar 2006

[SoK06e] Virtuelles Software Engineering Kompetenzzentrum:“LOC“, http://www.software-kompetenz.de/?4822&highlight=LOC, letzter Abruf am 2. Februar 2006

[Som01] Ian Sommerville: “Software Engineering“, 6. Auflage, Addison-Wesley, 2001.

[TDV06] Internetauftritt der IAV mbH zum Thema TeleDrive® VISION, http://www.teledrive.de, letzter Abruf am 6. Februar 2006

[Tha93] Georg Erwin Thaller: “Qualitätsoptimierung der Software-Entwicklung“, Vieweg, Braunschweig, 1993

[Thi05] Caja Thimm: „Vorlesung Medientheorie – Neil Postman”, Universität Bonn, Wintersemester 2004/2005, http://www.ikp.uni-bonn.de/ZfKM/newzfkm/archiv/altekvvs/ws0405/downloads_ws0405/2_th3952/theoriepostman.pdf , letzter Abruf am 17. Januar 2006

[UFS03] Unternehmensberatung Frost & Sullivan: “Der Europamarkt für In-car-Infotainment”, http://www.kfz-elektronik.de/b_030130.htm, letzter Abruf am 4. Januar 2006

[Ver04] Gerhard Versteegen und Rupert Wiebel: “Klare Anforderungen schaffen bessere Programme – Unerfüllte Träume“, iX, Ausgabe 11 / 2004, S. 102ff

[Vog05] Birgit Vogel-Heuser: „Modellierung OO: UML in der Systementwicklung“, Skript zur Vorlesung „Systems und Software Engineering“, Bergische Universität Wuppertal, Wintersemester 2005

[W3C02] W3C: “Extensible Markup Language (XML) 1.0“, deutsche Übersetzung, http://edition-w3c.de/TR/2000/REC-xml-20001006/, letzter Abruf am 18. Januar 2006

[W3C05] World Wide Web Consortium: „Extensible Markup Language (XML) 1.1 – Deutsche Übersetzung“, http://edition-w3c.de/TR/2004/REC-xml11-20040204/ , letzter Abruf am 16. November.2005

[Wat06] William C. Waterhouse: “widget”, http://alt-usage-english.org/excerpts/fxwidget.html, letzter Abruf am 17. Januar 2006

[Wik05a] Wikipedia: “Infotainment”, http://de.wikipedia.org/wiki/Infotainment, letzter Abruf am 14. November 2005

[Wik05b] Wikipedia: „Mensch-Computer-Interaktion“, http://de.wikipedia.org/wiki/Mensch-Computer-Interaktion, letzter Abruf am 15. November 2005

[Wik05c] Wikipedia: „Metrik“, http://de.wikipedia.org/wiki/Metrik, letzter Abruf am 18. November 2005

[Wik05d] Wikipedia: „4GL“, http://de.wikipedia.org/wiki/4GL, letzter Abruf am 19. Dezember 2005

[Wik05e] Wikipedia: „V-Modell“, http://de.wikipedia.org/wiki/V-Modell, letzter Abruf am 9. Dezember 2005

[Wik06a] Wikipedia: „Software-Ergonomie“, http://de.wikipedia.org/wiki/Software-Ergonomie, letzter Abruf am 9. Januar 2006

[Wik06e] Wikipedia: „Widget“, http://de.wikipedia.org/wiki/Widget, letzter Abruf am 3. Januar 2006

[Wik06f] Wikipedia: „MVC“, http://de.wikipedia.org/wiki/MVC, letzter Abruf am 3. Januar 2006

[Wit00] Andreas Witzel (2000, Januar). Das problemzentrierte Interview [26 Absätze]. Forum Qualitative Sozialforschung / Forum: Qualitative Social Research [On-line Journal], 1(1). Verfügbar über: http://www.qualitative-research.net/fqs-texte/1-00/1-00witzel-d.htm

[Zus91] Horst Zuse: “Software Complexity”, Walter de Gruyter, Berlin, 1991

12 Anhang

12.1 Ganzseitige Abbildungen

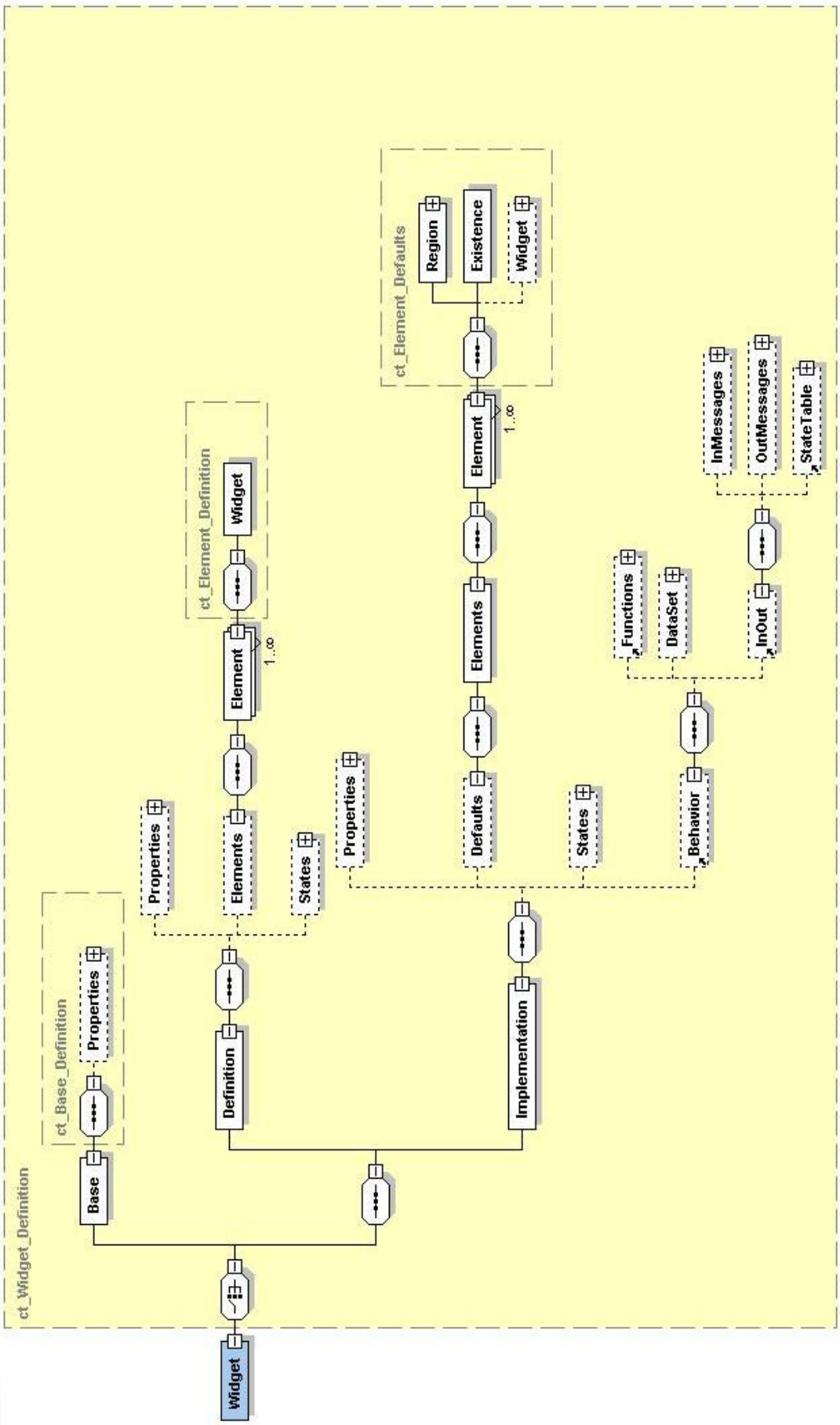

Abbildung 35: Grafische Darstellung des XML-Schemas für IML-Widgets

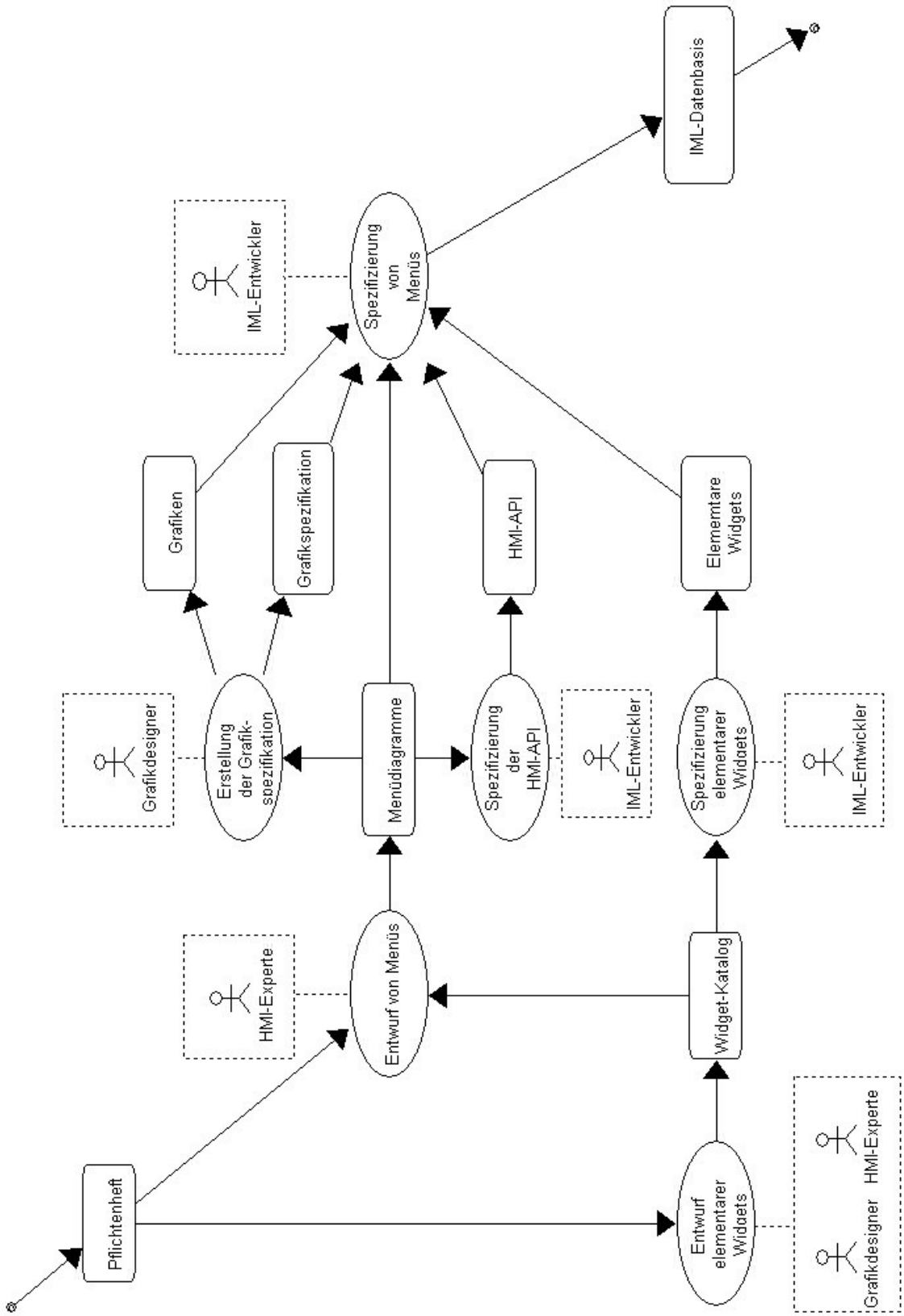

Abbildung 36: Grafische Darstellung des Spezifizierungsprozesses

12.2 Interviewleitfäden

12.2.1 Prozessanalyse

Begrüßung und Einleitung

1. Vorstellung des Diplomarbeitsthemas
2. Begründung für dieses Interview (Prozessanalyse)
3. Zusage von Anonymisierung der Daten geben (sowohl des Befragten als auch projektspezifischer Informationen)
4. Einverständnis zur Aufzeichnung einholen

Fragen

1. An welchem Projekt arbeitest Du zur Zeit?
2. Was ist das Ziel dieses Projekts?
3. Was ist Deine Aufgabe dabei?

4. Eingaben
    a. Welche Informationen und Dokumente brauchst Du, um Deine Arbeit zu erledigen?
    b. Von wem erhältst Du diese?
    c. Besteht dabei ein direkter Kontakt mit dem Informanten?
    d. Kommen die Informationen alle auf einmal?
5. Vorbedingungen
    a. In welchem Format müssen die Dokumente verfasst sein?
    b. Nach welchen Richtlinien oder Konventionen müssen diese verfasst sein?
6. Aktivitäten
    a. Wenn Du diese Dokumente erhältst, wie sieht dann Deine Arbeit im einzelnen aus?
    b. Aus welchen Arbeitsschritten besteht sie?
7. Werkzeuge
    a. Welche Werkzeuge verwendest Du für diesen Arbeitsschritt?
    b. Was muss das Werkzeug leisten können?
8. Ergebnisse
    a. Welche Informationen oder Dokumente erstellst Du bei Deiner Arbeit?
    b. Für wen?
    c. Wie reichst Du diese Ergebnisse weiter?
9. Nachbedingungen
    a. In welchem Format müssen Sie die Ergebnisse liefern?
    b. An welche Richtlinien oder Konventionen müssen Sie sich dabei halten?
10. Probleme
    a. Treten bei Deiner Arbeit irgendwelche generellen Probleme auf, die Dir auf anhieb einfallen?
11. Kommunikation
    a. Mit welchen Personen sprichst Du Dich bei der Arbeit ab?
    b. Worüber?
12. Rolle
    a. Welche Methoden und Sprachen musst Du für die Arbeit beherrschen?
    b. Welche Art von speziellen Wissen musst Du mitbringen?

12.2.2 Testabteilung

1. Welche Arten von Tests werden vorgenommen?

2. Wie lässt sich dies in Phasen und Teilprozesse einteilen?

3. Wie laufen diese im Einzelnen genau ab?
    a. Was wird gegen was getestet?
    b. Was wird manuell, was automatisch getestet?
    c. Welche Arten von Testfällen werden aus der IML-DB erzeugt?
    d. Welche Werkzeuge verwendest Du dabei?
    e. Mit wem wird dabei kommuniziert?

4. Was für Arten von Defekten werden gefunden?
    a. Grobe Einteilung auf Produktebene (Statecharts, IML, Gerätesoftware, ...)
    b. Feine Einteilung für IML

5. Welche IML-Defektklassen von den hier aufgelisteten sind dabei vorhanden?
    a. Zustandswechsel (Sprünge, Timer etc.)
    b. Lang-ID
    c. Funktionen
    d. Bedingungen
    e. Layout
    f. Struktur (Element zuviel oder fehlt)

6. Wie häufig werden welche der genannten Defektarten gemeldet? Welche treten besonders häufig auf?

7. Was schätzen Sie wie viele Defekte für das gesamte Projekt A/B gemeldet werden?

8. An wen werden die aufgedeckten Defekte herangetragen?

9. Welche Defektarten sind denn besonders teuer oder billig in ihrer Behebung?

10. Welche aufgedeckten Defekte können nur noch schwierig oder gar nicht behoben werden, weil es zu spät ist?

11. Welche Defektarten, werden gehäuft erst beim Testen gemeldet?

12. Werden alle Defekte im TestDirector eingetragen? Welche nicht? Warum nicht?

12.2.3 Defekte

1. Wie werden Defekte in der Spezifikation vom Zulieferer und vom Auftraggeber an Dich herangetragen? [Kommunikationsweg]

2. Wie sieht das Format der Protokolle aus?

3. An wen werden die Defekte herangetragen? Immer alle direkt an Sie?

4. Von wem kommen die Defekte?

5. Kommen die Defekte einer nach dem anderen oder in einer größeren „Lieferung“?

6. Um was für Arten von Defekten handelt es sich dabei?

7. Welche IML-Defektklassen von den hier aufgelisteten sind dabei vorhanden?
    a. Zustandswechsel (Sprünge, Timer etc.)
    b. Lang-ID
    c. Funktionen
    d. Bedingungen
    e. Layout
    f. Struktur (Element zuviel oder fehlt)

8. Falls eine nicht dabei ist: Wann / In welcher Phase wird diese entdeckt?

9. Wie häufig werden welche der genannten Defektarten gemeldet? Welche treten besonders häufig auf?

10. Um wie viele Defekte handelt es sich denn insgesamt pro Woche/Zeitraum (grobe Abschätzung)?

11. Was schätzen Sie wie viele Defekte für das gesamte Projekt gemeldet werden?

12. Wie wird bei der Behebung vorgegangen?

13. Welche Personen sind dabei involviert?

14. Welche Defektarten sind denn besonders teuer oder billig in ihrer Behebung?

15. Welche Defektarten, werden erst sehr spät gemeldet?

16. Werden alle Defekte im TestDirector eingetragen? Welche nicht? Warum nicht?

17. Ich werde im Rahmen meiner Diplomarbeit vorschlagen, für die Defekte im TestDirector jeweils noch mit eintragen zu lassen, wie lange deren Behebung gedauert hat. Welche Vor- und Nachteile würden daraus resultieren?

12.2.4 Paardurchsicht

1. Denkst Ihr, dass Ihr durch die Durchsicht die Qualität der IML-Datenbasis verbessert habt?

2. Glaubt Ihr, dass ihr mit der Durchsicht den Großteil der Fehler gefunden habt?

3. Gab es Fehler, die Du nicht hättest alleine aufdecken können?

4. Gibt es Fehlerarten, die durch die jetzige Form der Durchsicht nicht aufgedeckt werden können?

5. Waren die richtigen Personen an der Durchsicht beteiligt? (War jemand überflüssig?) Hat jemand gefehlt?

6. War die Anzahl der durchzusehenden Menüs zu viel / zu wenig?

7. Könnte die Durchsicht durch Veränderungen an der Durchführung effektiver ablaufen?

8. Hättest Du an irgendeiner Stelle Unterstützung durch Werkzeuge gebrauchen können?

9. Habt Ihr noch weitere Anregungen für die Verbesserung des Durchsichtsprozesses?

12.3 Durchsichtsanleitungen und -Protokolle

12.3.1 Durchsichtsanleitung für den IML-Entwickler

Anleitung für die Durchsicht (IML-EntwicklerIn)

Ziel dieser Durchsicht ist es, Fehler aufzudecken, die bei der Überführung der illustrierten Statecharts in die IML entstanden sind. Dabei geht es primär um Fehler, die das Verhalten oder die Struktur eines Widgets betreffen. Die entdeckten Fehler werden während der Durchsicht in einem Durchsichtsprotokoll notiert und erst nach der Nachbereitungsphase korrigiert. Es ist nicht notwendig einem entdeckten Fehler auf den Grund zu gehen. Das Notieren, dass ein Fehler vorliegt reicht aus. Falls jemandem bei der Durchsicht Design-Fehler, wie z.B. eine falsche Schriftgröße oder Tastenfarbe, auffallen, sollten diese natürlich auch notiert werden.

Jeder macht mal Fehler: Das Aufdecken der Fehler dient nicht der Beurteilung der Arbeit eines Mitarbeiters. Vielmehr sollte es das gemeinsame Ziel sein, zusammen ein qualitativ hochwertiges, also fehlerarmes Produkt zu erstellen.

Damit die Ergebnisse, die ihr erzielt, für meine Studie vergleichbar sind, habe ich eine relativ genaue Vorgehensvorschrift verfasst. Bitte probiert, Euch an diese zu halten.

Vorbereitungsphase

1. Notiere die Uhrzeit auf dem Deckblatt des Durchsichtsprotokolls.
2. Fülle das Deckblatt, so weit es möglich ist, aus
3. Bringe die Simulation auf den zu prüfenden Stand.
4. Öffne die durchzusehenden Seiten des Statechart-Dokuments mit dem Adobe Reader oder drucke sie aus.
5. Öffne das Trace-Tool.
6. Setze das Trace-Level und das Display-Level aller Komponenten auf “Error“. (mit der rechten Maustaste auf die Knoten “Widget Library“ und “IAV TeleDrive...“ klicken -> “Set trace level“)
7. Für die Knoten “WL::BaseBehavior (Events)“ und “WL::BaseBehavior (SM)“ setze das Display-Level auf “none“.
8. Starte die Simulation.
9. Für jedes der zu inspizierenden Menüs tue nun folgendes:
    a. Navigiere zu dem zu prüfenden Menü.
    Hinweis: Falls das Menü nicht oder nur umständlich in der Simulation erreichbar ist, kannst Du es auch wie folgt als Startmenü definieren: Öffne die Datei WidgetCont.xml, die sich im Pfad 18_Listings\Simulation\WidgetLibrary\platform\win32\bin befindet. Trage den Menünamen plus den Präfix „Menu_“ bzw. für Popup-Menüs „Overlay_“ als Wert des Attributs “Name“ des Knotens “UI / FirstMenue“ ein, speichere die Datei und starte die Simulation erneut.
    b. Überprüfe die Ausgabe des Trace-Tools auf Fehlermeldungen.
    c. Überprüfe, ob alle in der illustrierten Statechart eingezeichneten Elemente, wie Listen, (Close-) Buttons, Textfelder etc. in der Simulation zu sehen sind. Falls nicht, kann es sein, dass dieses Element nur zustandsabhängig erscheint. Wenn dies auf der illustrierten Statechart so einge-

zeichnet ist, dann überprüfe in der entsprechenden IML-Datei, ob das betreffende Widget für das Menü definiert wurde.

d. Notiere entdeckte Fehler in der Fehlertabelle des Durchsichtsprotokolls. Kreuze dabei das Kästchen „Nicht umgesetzt“ an, falls der Punkt bewusst noch nicht umgesetzt wurde.

e. Für Fehler in der IML-Umsetzung notiere jeweils noch die Klasse des Fehlers. Zur Auswahl stehen hier: **Z** für Zustandswechsel (Menüsprung, Timer etc.), **I** für Language-ID, **F** für Funktionen, **B** für Bedingungen (Conditions); **L** für Layout, **E** für fehlende Elemente bzw. Menüs und **X** für anderes. Notiere z.B. für einen fehlerhaften Menüsprung IML-Z.

f. Lösche die Ausgaben des TraceTools durch Drücken der Delete-Taste in dem Ausgabefenster des Tools.

10. Lese Dir die auf den illustrierten Statecharts notierten Kommentare und Menüübergänge durch.
11. Trage die für die Vorbereitung benötigte Zeit auf dem Deckblatt des Durchsichtsprotokolls ein.

Vielen Dank!

Gemeinsame Sitzung

Hinweis: Die Sitzung sollte nicht länger als zwei Stunden dauern. Da der/die IML-EntwicklerIn die Simulation während der Sitzung bedienen und konfigurieren muss, sollte die Sitzung in seiner vertrauten Arbeitsumgebung, also an seinem Arbeitsplatz, stattfinden.

1. Der/Die IML-EntwicklerIn notiert zu Beginn die Uhrzeit auf dem Deckblatt des Durchsichtsprotokolls
2. Er Trägt auch die vom HMI-Experten (derjenige der die Statecharts zeichnete) für die Vorbereitung benötigte Zeit ein.
3. Er startet nun die Simulation.
4. Für jedes der zu inspizierenden Menüs tut nun folgendes:
   a. Der/Die IML-EntwicklerIn navigiert in der Simulation zu dem zu prüfenden Menü.
   b. Geht nun die in der Vorbereitung vom HMI-Experten gefundenen Fehler durch. Der/Die IML-EntwicklerIn übernimmt diese dabei in das Durchsichtsprotokoll
   c. Überprüft nun die noch nicht vom HMI-Experten abgehakten Kommentare und Menüsprünge. Der/Die IML-EntwicklerIn bedient dabei die Simulation. Falls für eine vollständige Überprüfung des Punktes eine Inspektion oder Modifizierung einer IML-Datei notwendig ist, um z.B. eine Bedingung zu prüfen, wird er nur durchgesprochen, so dass ihn der/die IML-EntwicklerIn später in der Nachbereitung alleine überprüfen kann.
   d. Der/Die IML-EntwicklerIn notiert alle neu entdeckten Fehler in der Fehlertabelle des Durchsichtsprotokolls.
   e. Tragt die für die Sitzung benötigte Zeit auf dem Deckblatt des Durchsichtsprotokolls ein.
5. Der/Die IML-EntwicklerIn behält das ausgedruckte Statechart-Dokument und das Durchsichtsprotokoll.

Vielen Dank!

Nachbereitungsphase

1. Notiere zu Beginn wieder die Uhrzeit auf dem Deckblatt des Durchsichtsprotokolls.
2. Öffne das Trace-Tool.
3. Setze das Display-Level und das Trace-Level aller Komponenten bis auf das der Komponente “WL::BaseBehavior“ auf “Error“. Für letztere setze beide Level auf “Log“.
4. Für die Knoten “WL::BaseBehavior (Events)“ und “WL::BaseBehavior (SM)“ setze das Display-Level auf “none“.
5. Starte die Simulation.
6. Für jedes der zu inspizierenden Menüs tue nun folgendes, falls noch Punkte ungeprüft sind:
    a. Navigiere zu dem zu prüfenden Menü oder lasse es Dir als Startmenü anzeigen.
    b. Überprüfe nun die während der Sitzung offen gebliebenen Punkte durch Inspektion der betreffenden IML-Dateien, der Ausgabe des Trace-Tools und der Simulation.
    Wenn eine Funktionsausführung, verifiziert werden muss, dann überprüfe in der Ausgabe des Trace-Tools, ob die Funktion (mit den richtigen Parametern) ausgeführt wurde. (Für die Play-Funktion im Menu_AUDIOPLAYER_MAIN sollte z.B. folgende Zeile erscheinen: “`CIavFunction: Function AVDC.Audio.Play() is invoked.`“). Dazu ist es ggf. notwendig, in der IML-Datei des Menüs nachzuschauen, wie die Funktion benannt wurde.
    c. Trage etwaige Fehler wieder in die Fehlertabelle ein.
7. Trage die für die Nachbereitung benötigte Zeit auf dem Deckblatt des Durchsichtsprotokolls ein

Vielen Dank!

12.3.2 Durchsichtsanleitung für den HMI-Experten

Anleitung für die Durchsicht (HMI-Experte)

Ziel dieser Durchsicht ist es, Fehler aufzudecken, die bei der Überführung der illustrierten Statecharts in die IML entstanden sind. Dabei geht es primär um Fehler, die das Verhalten oder die Struktur eines Widgets betreffen. Die entdeckten Fehler werden während der Durchsicht in einem Durchsichtsprotokoll notiert und erst nach der Nachbereitungsphase korrigiert. Es ist nicht notwendig einem entdeckten Fehler auf den Grund zu gehen. Das Notieren, dass ein Fehler vorliegt reicht aus. Falls jemandem bei der Durchsicht Design-Fehler, wie z.B. eine falsche Schriftgröße oder Tastenfarbe, auffallen, sollten diese natürlich auch notiert werden.

Jeder macht mal Fehler: Das Aufdecken der Fehler dient nicht der Beurteilung der Arbeit eines Mitarbeiters. Vielmehr sollte es das gemeinsame Ziel sein, zusammen ein qualitativ hochwertiges, also fehlerarmes Produkt zu erstellen.
Damit die Ergebnisse, die ihr erzielt, für meine Studie vergleichbar sind, habe ich eine relativ genaue Vorgehensvorschrift verfasst. Bitte probiert, Euch an diese zu halten.

Vorbereitungsphase

2. Notiere zu Beginn die Uhrzeit.
3. Drucke Dir eine Fehlertabelle des Durchsichtsprotokolls aus.
4. Drucke Dir die durchzusehenden Seiten des Statechart-Dokuments aus.
5. Installiere eine Simulation für den zu prüfenden Stand auf Deinem Rechner.
6. Starte die Simulation.
7. Für jedes der zu inspizierenden Menüs tue nun folgendes:
    a. Navigiere zu dem zu prüfenden Menü.
    Hinweis: Falls das Menü nicht oder nur umständlich in der Simulation erreichbar ist, kannst Du es auch wie folgt als Startmenü definieren: Öffne die Datei WidgetCont.xml, die sich im Pfad 18_Listings\Simulation\WidgetLibrary\platform\win32\bin befindet. Trage den Menünamen plus den Präfix „Menu_“ bzw. für Popup-Menüs „Overlay_“ als Wert des Attributs “Name“ des Knotens “UI / FirstMenue“ ein, speichere die Datei und starte die Simulation erneut. Lasse Dir das ggf. beim ersten Mal von einem IML-Entwickler erklären.
    b. Überprüfe, sofern es mit der Simulation machbar ist, ob alle in der illustrierten Statechart notierten Kommentare und Menüsprünge korrekt umgesetzt worden sind.
    c. Verifiziere auch bei der Überprüfung eines Menüsprungs, ob ein eventuell im nachfolgenden Menü vorhandener Rücksprung („Backsprung“) korrekt funktioniert.
    d. Falls Du mit der Simulation den Punkt vollständig überprüfen konntest und er korrekt umgesetzt worden ist, dann hake den Punkt mit einem Stift auf dem ausgedruckten Statechart ab. Bei Unsicherheit lasse den Punkt besser offen.
    e. Notiere entdeckte Fehler in der Fehlertabelle des Durchsichtsprotokolls.
8. Notiere die für die Vorbereitung benötigte Zeit.

Vielen Dank!

12.3.3 Defektprotokoll

Durchsichtsprotokoll

Name des Statechart-Dokuments	
Version des Statechart-Dokuments	
Version der IML-DB (Variante / Datum)	
1: Dauer der Vorbereitung (IML-EntwicklerIn)	Beginn: Ende: Dauer:
2: Dauer der Vorbereitung (HMI-Experte)	Beginn: Ende: Dauer:
3: Dauer der gemeinsamen Sitzung	Beginn: Ende: Dauer:
4: Dauer der Nachbereitung (IML-EntwicklerIn)	Beginn: Ende: Dauer:

Kürzel	Durchgesehene Menüs (Menüname in der IML-Datenbasis)
A	
B	
C	
D	
E	
F	
G	
H	
I	
J	

Erstes Version

FEHLER

Nr.	Menü (Kürzel)	Fehler in **IML**-Datenbasis, **SC**: Statechart, **Sim**ulation	Nicht umge-setzt	Beschreibung	Wann entdeckt? 1, 2, 3, 4 (siehe Deckblatt)
BSP	B	IML		Die Funktion StartReview wird beim verlassen des Menüs nicht aufgerufen.	3
1					
2					
3					
4					
5					
6					
7					

Letzte Version

FEHLER

* **Z**: Zustandswechsel (Menüsprung, Timer.); **I**: Lang-ID; **F**: Funktion; **B**: Bedingung; **L**: Layout; **E**: Element/Menü fehlt oder zuviel; **X**: anderes

Nr.	Menü (Kürzel)	Fehler in **IML**-Datenbasis (*), **SC**: Statechart, **Sim**ulation	Nicht umge-setzt	Beschreibung	Wann entdeckt? 1, 2, 3, 4 (siehe Deckblatt)
BSP	B	IML-F		Die Funktion StartReview wird beim verlassen des Menüs nicht aufgerufen.	3
1					
2					
3					
4					
5					
6					
7					

12.3.4 Defektprotokoll für den HMI-Experte (Vorbereitungsphase)

FEHLER

Nr.	Menüname	Beschreibung
1		
2		
3		
4		
5		
6		
7		
8		