CARL VON OSSIETZKY universität OLDENBURG

Abschlussarbeit

zur Erlangung des Grades B.Sc.
im Studiengang Informatik

Bewertung der Usability eines Fahrkartenautomaten mit Hilfe von Engineering-Techniken zur Quantifizierung von Benutzungsoberflächen

Jutta Fortmann

29. September 2008

Gutachter:
Dr. Andreas Lüdtke
Prof. Dr. Susanne Boll

Zusammenfassung

In dieser Arbeit wurde eine Usability-Evaluation eines Fahrkartenautomaten durchgeführt. Dazu wurde die Interaktion mit zwei Methoden der GOMS-Familie (Goals, Operators, Methods, Selection Rules), namentlich KLM (Keystroke Level Model) und NGOMSL (Natural GOMS Language) modelliert und anschließend analysiert. Durch eine zusätzliche Nutzerbeobachtung wurde die formale Analyse ergänzt. Durch die Analyse wurden konkrete Optimierungspotentiale des Fahrkartenautomaten aufgedeckt. Usability-Schwächen lagen in den Bereichen Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität und Lernförderlichkeit. Abschließend wurden die eingesetzten Techniken kritisch hinterfragt. Eine wichtige Erkenntnis war, dass die Zielhierarchie des erstellten NGOMSL-Modells es ermöglichte, konkrete Problemstellen in der Interaktion zu identifizieren. Damit ging die Aussagekraft der NGOMSL-Analyse weit über die Berechnung von Bedien- und Lernzeiten hinaus. Darüber hinaus erwies sich die vorhergehende KLM-Analyse als sinnvolle Grundlage für die Erstellung des NGOMSL-Modells. Schließlich wird in dieser Arbeit ein Erweiterungskonzept von GOMS vorgestellt, welches es ermöglicht, Bedienfehler in die Berechnung der Bedienzeiten einzubeziehen.

Inhaltsverzeichnis

1 Einleitung

Dieses Kapitel führt den Leser in das Themenfeld dieser Arbeit ein. Außerdem werden die Aufgabenstellung und Motivation der Arbeit formuliert. Abschließend folgt eine Beschreibung des Aufbaus dieser Arbeit.

1.1 Einordnung des Themas

Das Design einer Benutzungsoberfläche ist entscheidend für den Erfolg und die effiziente Arbeit mit dieser. Auch die leistungsstärkste Soft- und Hardware ist nutzlos, wenn der Nutzer auf Grund schlechten Designs oder Ablaufs der Software langsamer arbeiten muss, als er es könnte. Um dauerhafte und effiziente Nutzung von Software zu gewährleisten, ist es deshalb wichtig, ihre Usability zu beachten. Usability wird laut DIN EN ISO 9241-11 mit Gebrauchstauglichkeit übersetzt und beschreibt „Das Ausmaß in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele **effektiv**, **effizient** und **zufriedenstellend** zu erreichen“. Diese drei Begriffe sind also als Maßstab für die Gebrauchstauglichkeit zu sehen.
Nicht selten wird Usability auch mit „Benutzer- oder Benutzungsfreundlichkeit“ übersetzt. Diese Wortwahl ist allerdings irreführend, da sie nicht alle Aspekte der Usability beschreibt und das Augenmerk ausschließlich auf den Nutzer selbst richtet. Die Aussage „Ein Gerät ist benutzerfreundlich.“ sagt nichts über den Kontext aus, in dem das Gerät verwendet wird, sondern nur darüber, wie der Benutzer dazu eingestellt ist. Der Begriff scheint also stark subjektiv, denn er lässt vermuten, dass einige Nutzer ein Gerät nur dann (gut) bedienen können, wenn es besonders „nett“ ist. Das Problem dabei ist allerdings, dass „nett“ von jeder Person anders definiert wird. Auch über Effizienz macht der Begriff keine Aussage. Tatsächlich bezieht sich Usability gerade auf den Kontext, in dem eine Interaktion mit einem Gerät statt findet. Nur in einem konkreten Kontext kann auch die Effizienz des Geräts sinnvoll betrachtet werden. Der Begriff Tauglichkeit suggeriert bereits diesen Bezug, nämlich, dass es Geräte gibt, die für eine bestimmte Aufgabe taugen und andere Geräte, die dagegen für diese Aufgabe untauglich sind. Gebrauchstauglichkeit bezieht sich folglich sowohl auf die Einschätzung des Benutzers, als auch auf den Kontext, also

die Aufgabe, die gelöst werden soll.
Es gibt Forschungsgebiete im Bereich Usability, die sich mit der Entwicklung von Geräten beschäftigen, die individuell an die Bedürfnisse des Nutzers angepasst werden können. Sogenannte „Adaptable User Interfaces“ (AUI) helfen dabei, Benutzungsoberflächen passend zu machen. Hier haben sich die Möglichkeiten mit den Fortschritten in der Hard- und Sofware-Technik weiterentwickelt. Kantorowitz und Sudarsky (KS89) beschreiben 1989 ein AUI als eine Schnittstelle zwischen Nutzer und Software, die es ermöglicht, problemlos und spontan zwischen der Programmsteuerung mittels Konsole und grafischer Benutzungsoberfläche zu wechseln. Während 1989 grafische Benutzungsoberflächen noch wenig verbreitet waren und futuristisch anmuteten, kann sich heute kaum ein Nutzer vorstellen, ohne ein Graphical User Interface (GUI) zu arbeiten. So fokussiert die sogenannte „User Interface Façades“-Technik Methoden, die es dem Nutzer ermöglichen, z.B. durch drag&drop[1], die grafische Benutzungsoberfläche an eigene Bedürfnisse anzupassen (Dei07). Der Eindruck des Benutzers spielt hier folglich eine zentrale Rolle.
Wichtige Kriterien, die beachtet werden, um ein Gerät gebrauchstauglich zu machen, sind zum Beipiel Belastungsminderung und die leichte Erlernbarkeit der Bedienung des Geräts. Um festzustellen, wie gebrauchstauglich ein System ist, werden verschiedene Kriterien untersucht. Das können z.B. die Erfolgsrate, die Zeit, die erforderlich ist, um ein Ziel (erstmals und wiederholt) zu erreichen, die Anzahl von Fehlern, die die Software erzeugt oder die Zeit, die der Nutzer durch Korrigieren von Fehlern benötigt, sein. Weitere mögliche Kriterien wären z.B. das Verhältnis zwischen Erfolg und Scheitern bis zum Ziel, das Ausmaß der Zufriedenheit des Nutzers oder auch die Anordnung von Tasten. Soll also ein Gerät auf Tauglichkeit getestet werden, so können neben der Bedienerführung demnach z.B. auch Hardwareelemente analysiert werden. Letztere seien hier aber nur der Vollständigkeit halber erwähnt, da diese Arbeit ihren Schwerpunkt auf die Analyse der Gebrauchstauglichkeit bezüglich der Bedienerführung legen wird. Bei solch einer Analyse helfen verschiedene Techniken, wie beispielsweise die GOMS-Technik (G.O.M.S. = Goals, Operators, Methods, Selection Rules) und die Varianten der GOMS-Familie, auf die in dieser Arbeit näher eingegangen wird.

Die Wissenschaft, die sich mit der Sicherstellung der Gebrauchstauglichkeit von Geräten beschäftigt, aber auch der Prozess selbst, der zu dieser Sicherstellung führt, werden Usability Engineering genannt. Dieser Prozess findet i.d.R. parallel zur Entwicklungsphase eines Geräts statt, soll aber auch darüber hinaus in der Nutzungsphase laufend zur Optimierung der Tauglichkeit beitragen. Idealerweise endet der Prozess „Usability Engineering“ bezüglich eines bestimmten Geräts folglich noch nicht nach der Entwicklung, sondern hilft auch dabei, optimierte Nachfolgeprodukte auf den Markt zu bringen.

[1] Unter drag&drop (engl. für „Ziehen und Fallenlassen“) versteht man eine Bedienungsmethode in GUIs, bei der grafische Elemente mittels eines Zeigegerätes verschoben werden.

Der Begriff Usability wird häufig im Zusammenhang mit Webseiten, Unterhaltungs- und Haushaltselektronik, wie z.B. Mobiltelefonen, Fernsehgeräten oder Video-Abspielgeräten benutzt. Gerade in diesen Bereichen spielt Usability eine große Rolle, da die Geräte nicht nur von (technischen) Laien, sondern auch von allen Altersgruppen bedienbar sein sollen und weitesgehend freiwillig und in der Freizeit benutzt werden. Das bedeutet, die Geräte müssen besonders leicht bedienbar sein und sollen in Design und Bedienerführung, aber auch Effizienz und Effektivität, positiv auf sich aufmerksam machen, um so ihre große Konkurrenz vom Markt zu verdrängen. Ein aktuelles Beispiel dafür ist das Apple iPhone, das vor allem durch seine intuitive und stark vereinfachte Touch-Pad-Bedienung Schlagzeilen machte.
Aber auch die Usability von anderen Alltagsgeräten, wie z.B. aus dem Transportwesen ist äußerst relevant. Denn hier kommen häufig sicherheitskritische Systeme, wie z.B. Fahrassistenzsysteme, zum Einsatz, deren leichte Bedienbarkeit besonders wichtig ist. Aber auch Service-Geräte, wie beispielsweise Fahrkartenautomaten, sollten über eine hohe Gebrauchstauglichkeit verfügen, da sie ohne diese ungenutzt blieben und so keinen Service-Dienst mehr bereitstellen würden. Ihr eigentlicher Sinn und Zweck wäre dann also verfehlt. Diese Arbeit widmet sich solch einem Gerät. Hier soll die Usability der Bedienerführung von Fahrkartenautomaten bewertet werden.

1.2 Aufgabenstellung

Ziel dieser Bachelorarbeit ist es zunächst, Optimierungspotentiale bezüglich der Bedienerführung der Fahrkartenautomaten der Deutschen Bahn mit Hilfe der Familie der GOMS-Analyse-Techniken aufzudecken. Des Weiteren sollen die verwendeten Analyse-Techniken kritisch betrachtet werden und der Ansatz eines Erweiterungskonzeptes im Rahmen dieser Techniken vorgestellt werden. Das Hauptaugenmerk liegt hier auf der Verwendung quantitativer Analyseverfahren im Bereich der Bewertung von Gebrauchstauglichkeit.

1.3 Motivation

In meinem Bekanntenkreis habe ich oft gehört, dass sich jemand über die Bedienung der Fahrkartenautomaten der Deutschen Bahn beklagt hat. Es würde zu lange dauern, bis man an dem Ziel, eine Fahrkarte zu kaufen, angekommen sei und die Bedienung sei oft nicht intuitiv sondern eher überraschend. Auch ich selbst habe solche Erfahrungen bereits gemacht. Eigentlich ist es überraschend und gerade deshalb so interessant, dass die Bedienerführung der Automaten vielen Personen erhebliche Schwierigkeiten bereitet, denn

oberflächlich betrachtet scheint die Funktionalität dieser Automaten nicht sonderlich groß zu sein. Auch ist die grafische Darstellung der Benutzungsoberfläche solcher Automaten sehr einfach gehalten. Es werden keine besonderen Multimedia-Techniken verwendet, sondern lediglich grafisch sehr nüchterne Schaltflächen, Textfelder, Symbole und Texte. Man sollte also annehmen, dass auf Grund des doch eher geringen Aufgabenspektrums, das mit diesen Automaten erfüllt werden kann, auch die Bedienbarkeit recht einfach intuitiv gestaltet werden kann.
Die Tatsache, dass an großen deutschen Bahnhöfen seit 2006 zur Unterstützung „Automatenguides“ eingestellt sind, die den Fahrgästen bei der „als schwierig empfundenen“ Bedienung des Automaten helfen sollen, verdeutlicht die Aktualität der hier behandelten Thematik, zeigt akuten Handlungsbedarf bei der Software der Automaten auf und unterstützt außerdem die Motivation dieser Arbeit. Auch der starke Anwendungsbezug an dieser Arbeit ist reizvoll.

1.4 Aufbau der Arbeit

In Kapitel zwei werden zunächst die Grundlagen dieser Arbeit erklärt. Es werden die internationale Normung der Usability und das untersuchte Gerät, der Fahrkartenautomat, vorgestellt. Das dritte Kapitel beschreibt den Stand der Wissenschaft. Dabei wird auf unterschiedliche Methoden zur Evaluation von Gebrauchstauglichkeit eingegangen. Den Schwerpunkt dieses Kapitels bildet die Einführung in die GOMS-Familie, die in dieser Arbeit eine zentrale Rolle einnimmt. Mit der Vorstellung einiger Werkzeuge für GOMS-Analysen schließt das Kapitel ab. Mit Kapitel vier folgt der analytische Teil dieser Arbeit. Nachdem auf die Auswahl geeigneter Analysemethoden und das Vorgehen bei den Analysen eingegangen wurde, wird die Durchführung und Auswertung dieser Analysen präsentiert. Die Bewertung der Analysen wird in Kapitel fünf vorgenommen. Anhand dieser werden konkrete Optimierungspotentiale des Fahrkartenautomaten formuliert und Verbesserungsvorschläge geäußert. Anschließend werden in Kapitel sechs die eingesetzten Techniken bewertet. Das Kapitel endet mit dem Aufzeigen der Grenzen dieser Techniken und bildet damit den Übergang zum siebten Kapitel, in welchem ein neues Erweiterungskonzept der GOMS-Techniken vorgestellt wird. Abgerundet wird diese Arbeit durch eine Zusammenfassung und einen Ausblick im achten Kapitel.

2 Grundlagen

Dieses Kapitel befasst sich mit der internationalen Normung der Usability und stellt das in dieser Arbeit untersuchte Gerät, den Fahrkartenautomaten, vor.

2.1 Die Usability-Normung

Der Begriff Usability wurde in der Normenreihe DIN EN ISO 9241: „Ergonomie der Mensch-System-Interaktion“ international definiert und spezifiziert. Diese Reihe bestand bis März 2006 aus insgesamt 17 Teilen, wurde seitdem aber um einige Normen erweitert. Wie in Kapitel 1.1 bereits zitiert, sind, laut DIN EN ISO 9241-11, Effektivität, Effizienz und Zufriedenstellung zentrale Begriffe bei der Definition von Gebrauchstauglichkeit.
Der Titel der Normenreihe lautet erst seit dem 1.4.2006, mit der Überarbeitung der Norm 9241-10, so. Zuvor hieß sie „Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten“. Da das Beachten von Usability mittlerweile längst nicht mehr nur bei Büroarbeit wichtig ist, wurde diese Einschränkung aufgehoben. Des Weiteren werden nun alle interaktiven Systeme, d.h. sowohl Software als auch Hardware, angesprochen. Dies wird in der Neudefinition der Benutzungsschnittstelle[1] deutlich, wie sie in der DIN EN ISO 9241-110 formuliert ist. Danach zählen zu einer Benutzungsschnittstelle (engl.: „User Interface“) „Alle Bestandteile eines interaktiven Systems (Software oder Hardware), die Informationen und Steuerelemente zur Verfügung stellen, die für den Benutzer notwendig sind, um eine bestimmte Arbeitsaufgabe mit dem interaktiven System zu erledigen“(Gei07).
Diese grundlegenden Überarbeitungen der Normenreihe zeigen, dass die Einhaltung von Usability als zunehmend wichtig erachtet wird. Zudem ist die Relevanz von Usability, durch die rasante Weiterentwicklung in der Informationstechnologie, längst nicht mehr nur im anfänglichen Bürobereich präsent, sondern bereits in die unterschiedlichsten Anwendungsfelder vorgedrungen, nämlich in solche, bei denen eine Interaktion zwischen Mensch und Maschine eine zentrale Rolle spielt, was heutzutage in nahezu jedem Anwendungsfeld der Fall ist. Außerdem wird Usability nicht mehr nur auf Software bezogen. Auch Hardware kann entscheidende Auswirkungen auf die Usability eines Geräts haben. Mit

[1] Dieser Begriff wird äquivalent zu dem Begriff „Benutzungsoberfläche“ verwendet

den in dieser Arbeit eingesetzten Analysetechniken lassen sich indirekt auch Rückschlüsse auf die Auswirkungen der Hardware auf die Usability des Geräts ziehen. Die für die Analysen ermittelten Antwortzeiten des Systems können auf Optimierungsmöglichkeiten der Hardware deuten. Verursacht der Fahrkartenautomat beispielsweise unerwartet lange Ladezeiten nach einem Tastendruck, so könnte es sich für die Usability des Geräts lohnen, die Hardware, z.B. den Prozessor oder Arbeitsspeicher, zu erneuern. Im Vordergrund steht bei den hier verwendeten Analysetechniken allerdings die Bewertung der Bedienerführung, also eher Soft- als Hardware.
Die DIN EN ISO 9241-10 legt in sieben Grundsätzen allgemeingültige Verhaltensregeln für interaktive Systeme fest. Dazu gehören die Aufgabenangemessenheit, die Selbstbeschreibungsfähigkeit, die Erwartungskonformität, die Fehlertoleranz, die Steuerbarkeit, die Individualisierbarkeit sowie die Lernförderlichkeit eines interaktiven Systems. Aus der Überarbeitung dieser Norm ist die DIN EN ISO 9241-110 „Grundsätze der Dialoggestaltung“ als Nachfolgerin entstanden, die die sieben Grundsätze grundsätzlich beibehält, aber weiter präzisiert. Die sieben Grundsätze beschreiben im Einzelnen Folgendes:

- **Aufgabenangemessenheit:** Ein interaktives System ist dann aufgabenangemessen, wenn es seinen Benutzer dabei unterstützt, seine Aufgabenziele vollständig und korrekt und mit einem annehmbaren Aufwand zu erledigen. Nicht aufgabenangemessen wäre z.B. ein Formular, bei dem mehrere Eingabefelder in einer für den Menschen untypischen Reihenfolge angeordnet sind. Sollte z.B. eine Adresse mit Namen, Straße, Hausnummer, Postleitzahl und Wohnort eingegeben werden und die Reihenfolge wäre nicht standardmäßig wie oben angegeben, sondern der Ort käme beispielsweise vor der Straße, dann würde sich das System hier unangemessen verhalten und eine uneffiziente Erledigung der Aufgabe verursachen.

- **Selbstbeschreibungsfähigkeit:** Ein interaktives System ist dann selbstbeschreibungsfähig, wenn für den Benutzer jederzeit offensichtlich ist, in welchem Dialog und an welcher Stelle im Dialog er sich befindet. Außerdem muss stets deutlich sein, welche Handlungen der Nutzer unternehmen kann und wie diese auszuführen sind. Wird beispielsweise in einem Formular keine Anmerkung zu einem Eingabefeld gegeben, dessen Formatierung unklar ist (Beispiel: Datums-Eingabe), so wird dieser Grundsatz hier missachtet.

- **Erwartungskonformität:** Ein interaktives System ist dann erwartungskonform, wenn es konsistent ist und den Merkmalen des Benutzers entspricht. Diese Merkmale sind beispielsweise seine Kenntnisse aus dem Arbeitsgebiet, aus seinen Erfahrungen und aus allgemein anerkannten Koventionen, wie z.B. der Norm 9241. Das Programm „WinZip“ zum Komprimieren von Dateien beispielsweise weist keine Funktion „Komprimieren“ auf, sondern versteckt diese hinter der Funktion „Neu“.

Der Nutzer hat die Erwartungshaltung, bei einem Komprimierprogramm auch eine „Komprimieren-“Funktion zu finden, wird aber enttäuscht und ist gezwungen, Vermutungen nachzugehen und auszuprobieren. Das Dialogprinzip der Erwartungskonformität wird hier also verletzt.

- **Fehlertoleranz:** Ein interaktives System ist fehlertolerant, wenn es den Benutzer vor Fehleingaben bewahrt und im Fehlerfall unmittelbar auf die Fehleingabe hinweist und den Nutzer dann konstruktiv bei der Behebung des Fehlers unterstützt. Ein Beispiel hierfür ist die Reaktion des Betriebssystemes Microsoft Windows Vista, wenn man versucht, einer Datei im Explorer oder auf dem Desktop einen Namen mit bestimmten Sonderzeichen, wie z.B. dem Slash („/“), zu geben. Es erscheint eine Meldung, die darauf hinweist, dass solche Zeichen unzulässig sind und die Eingabe des Zeichens wird gleichzeitig nicht angenommen bzw. verarbeitet.

- **Steuerbarkeit:** Ein interaktives System ist steuerbar, wenn der Nutzer in den Dialogablauf eingreifen kann, d.h. über Start, Richtung und Rückschritte bestimmen kann. Der Nutzer muss also jederzeit alle Aktivitäten des Systems steuern und auch vorangegangene Schritte rückgängig machen können. Ein Beispiel wäre die Rückgängig-Funktion in Grafik- oder Textverarbeitungsprogrammen, die z.B. einen gerade gelöschten Absatz wieder herstellen kann.

- **Individualisierbarkeit:** Ein interaktives System sollte sich individuell auf die Bedürfnisse und Vorlieben seines Benutzers einstellen können. Beispiele wären hier die Möglichkeit, die Schrift zu vergrößern, oder zusätzliche Menüs und Funktionen an- und abschalten zu können. Vor zu starker Anpassung ist allerdings auch Vorsicht geboten. Menüelemente oder Icons, die sich ständig, je nach Häufigkeit der Verwendung durch den Nutzer, unterschiedlich anordnen, können zu mehr Irritation als Nutzen führen. Ein Beispiel für eine Anwendung, die dieses Kriterium umsetzt, ist Microsoft Word. Die Anordnung der Icons in der Werkzeugleiste von Microsoft Word passt sich ständig an die Zugriffe des Nutzers an.

- **Lernförderlichkeit:** Ein interaktives System sollte die Erlernung von Bedienung und Funktionalität fördern. D.h. der Benutzer sollte durch intuitive Darstellung und Hilfestellungen möglichst schnell und einfach den Umgang mit dem System erlernen können. Dabei helfen z.B. aussagekräftige Icons und Bezeichner für bestimmte Funktionen, oder die Anordnung von Standardfunktionen in oberster Menüebene. Ein weiteres Beispiel ist die Verwendung von Unterstrichen unter Funktionsbezeichnern in Menüs, um Kommandokürzel anzugeben, wie es bei vielen Microsoft-Programmen angewendet wird. Lernförderlichkeit bezieht sich also auch auf das Erlernen alternativer Methoden zur Systemsteuerung.

Neben der oben erläuterten Normenreihe spielt hier ebenfalls die Norm ISO 13407 eine wichtige Rolle. ISO 13407 „Benutzer-orientierte Gestaltung interaktiver Systeme“ stellt ein grundsätzliches Vorgehensmodell vor, wie der Benutzer durchgängig in den Entwicklungsprozess mit einzubinden ist und fordert dieses gleichzeitig. Durch die frühzeitige und dauerhafte Einbeziehung der Nutzerbedürfnisse sollen Usability-Mängel und daraus resultierender zusätzlicher Optimierungsaufwand, verbunden mit hohen Kosten, vermieden werden. Laut der Norm soll ein iterativer Prozess, der parallel zur Entwicklung der Software abläuft, die Gebrauchstauglichkeit dieser gewährleisten. Dabei soll zunächst der Nutzungskontext erfasst werden, um darauffolgend die Anforderungen an die Software und auch organisatorische Anforderungen zu ermitteln. Damit können dann Gestaltungsvorschläge für die Software gemacht und diese letztlich bewertet werden. Allerdings handelt es sich hierbei lediglich um ein allgemeines Vorgehensmodell, nicht jedoch hält die ISO 13407 konkrete Gestaltungsvorschläge für den Entwicklungsprozess vor. Hierfür eignet sich der DATech[2] Prüfbaustein Usability-Engineering-Prozess, ein Verfahren zur Überprüfung der Normkonformität mit DIN EN ISO 9241-10/-11 und DIN EN ISO 13407. Solch ein definiertes Prüfverfahren erleichtert Prüfern den Vergleich von Prüfergebnissen und deren Reproduktion. Verbindlich angewendet wird dieses Prüfverfahren von akkreditierten Prüfstellen. Es gilt allerdings nicht als allgemeingültiges Prüfverfahren, sondern vielmehr als eine geeignete Richtung, die sich in der Konformitätsprüfung bewährt hat. Das Prüfverfahren ist im Einzelnen in dem „Leitfaden Usability“ (DAT08) veröffentlicht.

2.2 Der Fahrkartenautomat

Am Standort Oldenburg Hauptbahnhof befinden sich insgesamt elf Fahrkartenautomaten, davon fünf für den Nahverkehr und sechs für den Fernverkehr. Insgesamt sind fünf am Ausgang Hauptbahnhof Süd, zwei im Service-Center der Bahn und vier weitere Automaten am Ausgang ZOB installiert. Beide Automatentypen sind von der Firma Höft & Wessel AG entwickelt worden. Diese Hersteller liefern allerdings nur die Hardware, die Software wird von der Bahn selbst, genauer gesagt der DB Systel GmbH, entwickelt. Softwareupdates finden etwa einmal pro Quartal statt. Diese Releases erreichen dann über Netzwerk die einzelnen Automaten. Die Software ist für alle Automaten eines Typs identisch, die unterschiedlichen Verbundverbindungen werden rein datengesteuert entsprechend eingefügt. Die Hardware, also die Automaten an sich, kosten, nach Angaben des Herstellers, pro Stück ca. 20.000 Euro. Die Abbildung 2.1 zeigt einen Fahrkartenautomaten für den Fernverkehr am Standort Oldenburg in Oldenburg.

[2]Deutsche Akkreditierungsstelle Technik e.V.

Die letzte große Änderung der Software fand im Jahr 2002 statt, als die Bahn ein neues Preissystem einführte. Zu diesem Anlass wurden neben den Eingabemasken auch das Design und die Abläufe der Software komplett überarbeitet.

Abb. 2.1: Automat für den Fernverkehr am Standort Oldenburg (Oldb)

Um die Usability der Software sicherzustellen, beauftragt die Deutsche Bahn AG gelegentlich Firmen, die dann diesbezüglich Studien durchführen. Zuletzt geschah dies im Rahmen eines Projektes des Fraunhofer Institut für Arbeitswirtschaft und Organisation (IAO) in Stuttgart über den Zeitraum eines Jahres zwischen 2006 und 2007 (Fra08). Dabei wurden die Bedürfnisse und Wünsche der Kunden anhand von Tests und Befragungen analysiert und daraus mentale Modelle der verschiedenen Kundentypen (Pendler, Gelegenheitsfahrer, computer-unerfahrene Nutzer) erstellt. Neben diesen Aufträgen fließen aber auch die Erfahrungen und Vorschläge der eigenen Mitarbeiter in Ideen für eine bessere Gestaltung der Software mit ein. Als wichtigste Messgröße für Usability gab die Bahn die Kundenzufriedenheit an. Aus diesem Grund werden zur Bewertung der Usability stets Kundenbefragungen durchgeführt, deren Ergebnisse dann in Veränderungen der Software mit einfließen [3].

[3] Quelle: mündliches Interview mit Markus Grafe, DB Vertriebs GmbH

Diese Arbeit wird sich mit den Automaten für den Fernverkehr beschäftigen. Diese ermöglichen neben dem Verkauf von Fahrkarten auch die Reservierung von Sitzplätzen und die Fahrplanauskunft. Bedient wird solch ein Automat über einen Touch-Screen und ein PIN-Eingabefeld. Die Bezahlung ist ausschließlich über eine Geldkarte möglich, um, laut Angaben der Bahn, u.a. Vandalismus an den Automaten vorzubeugen.

3 State-of-the-Art

In diesem Kapitel werden unterschiedliche Methoden zur Evaluation von Gebrauchstauglichkeit vorgestellt. Ein besonderer Schwerpunkt wird dabei auf die Vorstellung der GOMS-Famile gelegt, der ein eigenes Unterkapitel gewidmet ist.

3.1 Ansätze zur Evaluation von Gebrauchstauglichkeit

Es gibt unterschiedliche Ansätze, um die Usability eines Geräts zu bewerten. Man unterscheidet u.a. zwischen formaler, heuristischer und empirischer Evaluation. Nachfolgend werden diese Evaluationsmethoden genauer beschrieben.

3.1.1 Formale Evaluation

Quantitative Analysetechniken, wie es auch die Varianten der GOMS-Familie sind, gehören zu den formalen Evaluationsverfahren. Diese eignen sich gut für die Erstellung von Leistungsaussagen bzgl. trainierter Nutzer. Sie beschreiben quantitativ Handlungsabläufe, um Schwachstellen zwischen der Mensch-System-Interaktion aufzudecken. Ihr Ziel ist es u.a., die Benutzungsleistung, die ein Nutzer bei der Arbeit mit einer Benutzungsoberfläche erbringt, zu beschreiben und vorherzusagen. Dabei gibt es z.B. Verfahren, die vorhersagen, wie lange ein geübter Nutzer zur Erledigung einer Aufgabe benötigt oder wieviel Lernzeit eine Benutzungsoberfläche beansprucht. Es gibt viele verschiedene formale Evaluationsverfahren für die Untersuchung unterschiedlicher Parameter, auf die im Folgenden näher eingegangen wird.

3.1.1.1 GOMS

Die Analyse-Technik GOMS (G.O.M.S. = Goals, Operators, Methods, Selection Rules) und die Varianten der GOMS-Familie helfen dabei, Benutzungsoberflächen quantitativ zu

untersuchen und zu bewerten. Sie wurden Anfang der 80er Jahre von Card, Moran und Newell entwickelt und sind in der Lage, a-priori-Abschätzungen zu Bearbeitungs- und Lernzeiten einer bestimmten Aufgabe zu liefern. Die Grundidee von GOMS ist es, die Interaktion zwischen Mensch und Maschine auf elementare Aktionen zu reduzieren und den Ablauf dieser in einem Modell darzustellen um dann die Effizienz dieser Schritte zu ermitteln. Das Akronym GOMS steht dabei für

- **G**oals: Goals steht für die Ziele des Benutzers. Sie klären somit die Frage: Was will der Benutzer erreichen? Oft werden Ziele in Unterziele aufgeteilt. Dabei gilt, dass alle Unterziele durchgeführt werden müssen, um das Gesamtziel zu erreichen. Ausgenommen davon sind mehrere Unterziele, die zum selben Ziel führen. Von diesen muss nur eins ausgeführt werden (s. Selektionsregeln).
- **O**perators: Operatoren sind die kleinsten Aktionen des Benutzers. Sie können wahrnehmender, kognitiver, motorischer Art oder auch eine Zusammensetzung dieser sein. Operatoren können den mentalen Zustand des Nutzers oder, durch körperliche Aktionen, den der Umgebung verändern. Die Parameter der Operatoren, wie z.B. die Ausführungszeit, sollten unabhängig von dem Operatorverlauf, also frei von einem bestimmten Kontext, sein.
- **M**ethods: Methoden sind Sequenzen von Operatoren und Unterzielaufrufen, die ein Ziel vollenden. Sind die Ziele hierarchisch geordnet, dann existiert auch eine dazu passende Hierachie von Methoden.
- **S**election Rules: Selektionsregeln werden benötigt, wenn mehrere Methoden bzw. Unterziele geeignet sind, um ein bestimmtes Ziel zu erreichen. Die Regeln klären, welche Methode ausgeführt bzw. welches Ziel verfolgt wird.

Die Abbildung 3.1 verdeutlicht diese Einteilung anhand eines Beispiels aus dem Alltag. Zu sehen ist ein unvollständiger Ausschnitt der Zielhierarchie zum Oberziel „Frühstück zubereiten“. Hier sei erwähnt, dass dieses Beispiel nur der Anschaulichkeit halber verwendet wurde und solche Aufgaben in der Realität nicht mit GOMS modelliert werden.

Diese Zielhierarchie kann in eine Hierarchie von Methoden abgebildet werden. D.h., zu jedem der hier aufgeführten Ziele existiert auch eine Methode, die dieses Ziel erreicht. Die Selektionsregel klärt, ob die Methode für das Ziel „Mit Kaffeemaschine“ oder die für das Ziel „Ohne Kaffeemaschine“ ausgeführt wird. Einige Operatoren, die die Methode „Filter positionieren“ formen, wären beispielsweise:
Denkpause, Zum Filter gehen, Filter nehmen, Zur Maschine gehen, Vorrichtung ausschwenken, Filter einsetzen.

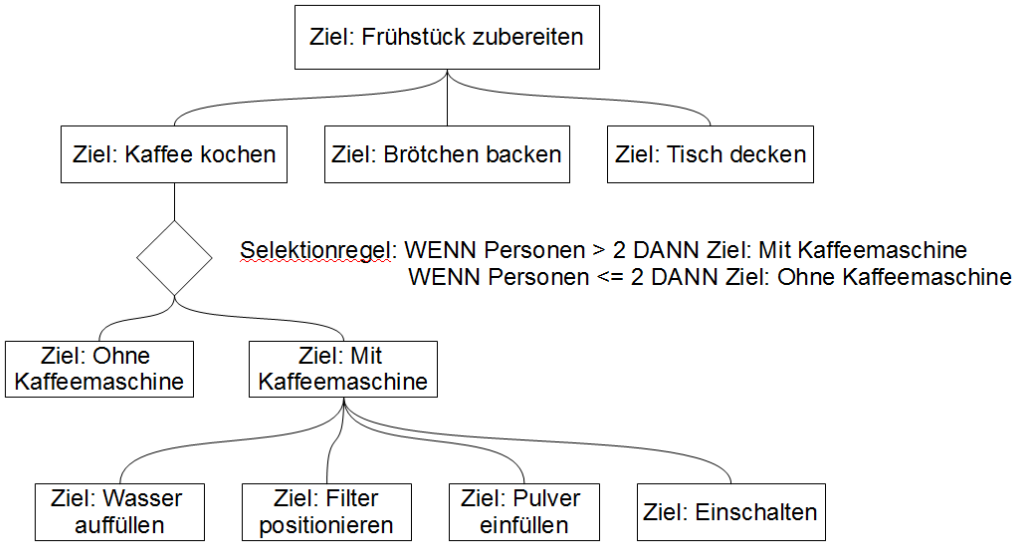

Abb. 3.1: Zielhierarchie zu Top-Level-Ziel „Frühstück zubereiten“

Den Weiterentwicklungen der GOMS-Methode zur GOMS-Familie wird ein eigenes Kapitel 3.2 gewidmet, da diese eine zentrale Stellung in dieser Arbeit einnehmen.

3.1.1.2 TAG

Neben der GOMS-Familie ist TAG (Task-Action Grammars) ein weiteres formales Evaluationsverfahren. TAG ist in der Lage, vorherzusagen, wie die Lern- oder Bearbeitungszeiten einer bestimmten Aufgabe von zwei unterschiedlichen Benutzungsoberflächen ausfallen werden (Jun02). TAG wird seit 1986 von Steve Payne und Thos Green entwickelt. Es ist eine Task-Action-Grammatik, die die Konsistenz einer Interaktionssprache beschreiben kann. Dafür nutzt diese eine feature-basierte Darstellung der Aufgaben und eine Form von Attribut-Grammatik, mit der die Aktionen generiert werden, die für die Durchführung dieser Aufgaben nötig sind. Mit feature-basierter Darstellung ist hier gemeint, dass die Darstellung der Aufgabe so gewählt ist, dass sie die Eigenschaften der Aufgabe verdeutlicht. Die TAG-Notation ist wesentlich kompakter als GOMS und nutzt eben eine andere Repräsentation von Zielen, nämlich TAG Attribute mit bestimmten Werten statt, wie bei GOMS, natürliche Sätze (Gre98). Die Besonderheit von TAG ist, dass Vorwissen und Erlernbarkeit berücksichtigt werden (BN02).

3.1.1.3 Kognitive Architekturen

Neben GOMS und TAG existieren auch kognitive Architekturen in Form von Produktionensystemen, wie ACT-R (Adaptive Control of Thought-Rational), SOAR (State, Operator

and Result - eine Weiterentwicklung von GPS (General Problem Solver)) und EPIC (Executive Process-Interactive Control). Kognitive Architekturen beschreiben, wie Wissen verarbeitet wird. Sie bestehen meistens aus einer Wissensbasis (Langzeitgedächtnis), einem Arbeitsgedächtnis (Kurzzeitgedächtnis), einer Wissensverarbeitung, einer Lernkomponente, einer Perzept- und einer Motorkomponente. Während die Wissensverarbeitung aktuell benötigtes Wissen aus der Wissensbasis in das Arbeitsgedächtnis lädt, manipuliert die Lernkomponente die Wissensbasis (Lü05). Ein Produktionensystem beinhaltet neben deklarativem Wissen, also reinem Faktenwissen, prozedurales Wissen. Dieses Wissen entsteht durch Erfahrungen und beschreibt Zusammenhänge, genauer gesagt „Handlungswissen“. Es liegt hier in Form von Produktionsregeln vor. Dies sind Wenn-Dann-Regeln, die Anweisungen enthalten, welche Aktion in welcher Situation oder für das Erreichen welches Ziels auszuführen ist. Diese Produktionsregeln steuern folglich die Handlungen eines Menschen. Deklaratives Wissen kann man demnach eher als „Wissen“ bezeichnen und prozedurales Wissen als „Können“.
Insbesondere ACT-R und EPIC besitzen eine besondere perzeptive Schnittstelle für visuelle und auditive Eingangssignale und eine handlungsmotorische Schnittstelle für die aktive Handlungsausführung. Diese drei kognitiven Architekturen können, dank ihrer Form als Produktionensysteme, neben einfacheren kognitiven Fähigkeiten auch Problemlöseverhalten erklären. ACT-R macht Aussagen über die grundsätzliche kognitive Architektur von Individuen, d.h. es beschränkt sich dabei nicht nur auf Teilaspekte menschlicher Kognition. Sie wurde deshalb von Newell als „erste einheitliche Theorie menschlicher Kognition“ bezeichnet und gilt vermutlich als ausgereifteste Theorie von Produktionensystemen (Lin07). SOAR basiert auf dem Problemraumprinzip nach Newell und Simon. D.h., dass die Verarbeitung von Wissen stets die Suche in Problemräumen bedeutet, die aus Anfangszustand, Zielzuständen und Operatoren bestehen. EPIC betrachtet im Gegensatz zu den anderen beiden erwähnten kognitiven Architekturen vor allem das Multitasking, d.h. in der EPIC-Architektur wird berücksichtigt, wie sich Menschen verhalten, wenn sie mehrere Aufgaben zur selben Zeit erledigen. Daher baut EPIC teilweise auch auf der CPM-GOMS-Variante auf. Allerdings befindet sich EPIC noch in der Entwicklung, sodass es noch nicht ausgereift genug ist, um für Evaluationszwecke eingesetzt zu werden.

3.1.2 Empirische Evaluation

Als Standardtechnik für die Entwicklung eines gebrauchstauglichen Systems gilt noch immer die empirische Evaluation (Kie96). Dabei werden experimentelle Vorgehensweisen zur Bewertung von Usability herangezogen. D.h., diese Bewertungen beruhen nicht auf theoretischen Grundlagen, sondern auf nachvollziehbaren Erfahrungen und beschreib- bzw. messbaren Beobachtungen. Dazu gehören z.B. Tests mit potentiellen Nutzern der definier-

ten Zielgruppe und die Beobachtung von Testpersonen. Auch das sogenannte „Thinking Aloud“, bei dem Testpersonen während der Testphase laut ihre (subjektiven) Gedanken zu dem Testgerät formulieren, ist empirischer Natur. Das „Usability Testing“ greift auch auf solche Experimente zurück, die Ziele dieser Methode sind jedoch andere. Die experimentelle Forschung verfolgt das Ziel, Zusammenhänge und ggf. Gesetzmäßigkeiten zu entdecken und zu verstehen um damit Aussagen über die Wahrscheinlichkeit des zukünftigen Auftretens bestimmter Nutzervorgänge machen zu können. Hingegen nutzt das Usability Testing diese Erkenntnisse bereits als Grundlage um darüber hinaus zu versuchen, durch entsprechende Produktgestaltung zukünftiges Nutzerverhalten zu kontrollieren (Jun02). Ein weiteres, bekanntes, empirisches Evaluationsverfahren ist der „Cognitive Walkthough“, der im Folgenden beschrieben wird.

Im Gegensatz zu GOMS und TAG wird bei dem Verfahren des Cognitive Walkthrough von Polson und Lewis nicht von einem geübten Nutzer ausgegangen, sondern es wird das explorative Verhalten von Laien erforscht. Dazu durchlaufen einige Usability-Experten einen simulierten Weg eines Nutzers durch die Bedienerführung einer Benutzungsoberfläche. Sie versetzen sich in die Nutzer und analysieren vorgegebene Handlungsabläufe unter der Annahme, dass ein Nutzer stets den Weg des geringsten kognitiven Aufwands gehen wird. Dies setzt voraus, dass die Experten Kenntnis über vermutliches Vorwissen und Verhalten der Nutzer haben, was wiederum eine aufwendige Vorbereitungsphase vor dem eigentlichen Walkthrough voraussetzt. Geeignet ist ein Cognitive Walkthrough inbesondere, um herauszufinden, wie leicht die Bedienung eines Geräts erlernt werden kann. Anhand der Ergebnisse eines Cognitive Walkthrough können Rückschlüsse auf Schwachstellen des Designs gezogen werden. Genauer gesagt ist festzustellen, ob, wo und warum das verwendete Design die Interaktion zwischen Nutzer und System stört (BGJK04).

3.1.3 Heuristische Evaluation

Der heuristische Ansatz greift auf Heuristiken zurück, um Gestaltungsmängel aufzudecken. Nielsen zählt in (Nie93) die, seiner Ansicht nach, sieben wichtigsten Heuristiken auf. Diese sind

1. Wahrnehmbarkeit des Systemstatus
2. Zusammenhang zwischen System und Realität
3. Benutzerkontrolle und -freiheit
4. Konsistenz und Normen
5. Fehlerverhütung

6. Wiedererkennen und Erinnerung
7. Flexibilität und Benutzereffizienz

Anhand dieser Heuristiken untersuchen drei bis fünf Experten unabhängig voneinander die Bedienerführung eines Geräts. Nachdem diese Untersuchungen durchgeführt wurden, werden alle Beobachtungen zusammengefasst und den einzelnen Heuristiken zugeordnet. Anschließend werden die Beobachtungen hinsichtlich ihres Schweregrades bewertet. Dabei wird berücksichtigt, wie oft ein Problem auftritt und wieviel Einfluss es auf die Bedienung des Geräts hat. So kann eine Prioritätenliste erstellt werden, nach der die einzelnen Probleme behoben werden sollen. Abschließend formulieren die Experten Optimierungsvorschläge zu jedem Problem.
Die heuristische Evaluation gilt als kostengünstige und wenig Zeit benötigende Evaluationsmethode. Ein Kritikpunkt ist allerdings, dass die Experten, die diese Untersuchungen durchführen, keine potentiellen Nutzer des Geräts sind und somit i.d.R. nicht über nötiges Domänenwissen oder Erfahrungen bzgl. der Benutzungsoberfläche verfügen, von denen jedoch ausgegangen wird, dass reale Nutzer sie hätten. Darüber hinaus wird bei der heuristischen Evaluation u.a. die Berücksichtigung von Benutzer und Aufgabe vermisst, da Nielsen lediglich Konstruktionshinweise für Aufgaben gibt, aber keine konkreten Vorgehensmodelle formuliert (BH00).

3.2 GOMS-Familie

Das Original CMN-GOMS, benannt nach seinen Erfindern Card, Moran und Newell, wurde weiterentwickelt zu einer Familie von GOMS-Verfahren, der GOMS-Familie. Dabei wurden zwei konzeptionelle Richtungen eingeschlagen:

- Sequenzform: Diese Form zeichnet sich durch eine feste, sichtbare Operatorsequenz ohne allgemeine Parameter aus. Sie ist vergleichbar mit einem einfachen, nicht objektorientierten, also völlig sequentiellen, Computerprogramm.
- Programmform: Diese Form ist allgemein. Es gibt Prozeduren und Parameter. Der Ausführungspfad ist variabel. Vergleichbar ist diese Form deshalb mit einem objektorientierten Computerprogramm.

Neben dem ursprünglichen GOMS unterscheidet man zwischen drei weiteren Varianten: dem Keystroke-Level-Model (KLM), der Natural GOMS Language (NGOMSL) und dem Cognitive-Perceptual-Motor oder auch Critical-Path-Method GOMS.

3.2.1 Keystroke-Level-Model (KLM)

3.2.1.1 Konzept

Das Keystroke-Level-Model wurde von Card, Moran und Newell als stark vereinfachte Variante des CMN-GOMS entwickelt. Sie ist damit die einfachste GOMS-Technik und verwendet die Sequenzform. Mit dieser Variante kann die Bearbeitungszeit einer bestimmten Aufgabe berechnet werden. Dazu listet der Analyst die einzelnen Operatoren der Aufgabe sequentiell auf und summiert dann deren Ausführungszeiten zur Gesamt-Bearbeitungszeit der Aufgabe auf. KLM verwendet keine Ziele, Methoden oder Selektionsregeln. Lediglich sechs primitive Keystroke-Level Operatoren stehen zur Verfügung, um die einzelnen Interaktionen darzustellen. Jeder dieser Operatoren benötigt eine bestimmte Zeit, um ausgeführt zu werden. Die Operatoren lauten wie folgt:

- K: K steht für einen Tastendruck (Keystroke). Die Dauer eines K-Operators ist u.a. von der Tippgeschwindigkeit des Nutzers und der Art des Druckes, d.h. Tastaturdruck oder Mausdruck, abhängig. Daher variiert dessen geschätzte Dauer zwischen 0.1 und 1.2 Sekunden.
- P: Mit P wird das Richten (point) der Maus auf ein Ziel beschrieben. Die geschätzte Dauer hierfür beträgt 1.10 Sekunden.
- H: H drückt eine Handbewegung (homing), wie z.B. das Bewegen der Hand zur Tastatur, aus; Dauer: 0,4 Sekunden.
- D: D bezeichnet das Zeichnen (draw) eines ca. 10cm langen Linienabschnitts mit der Maus. Die Dauer hierfür variiert je nach Anwendung.
- M: M steht für die mentale Vorbereitung auf eine Aktion, also eine Denkpause. Sie wird mit dem geschätzten Durchschnittwert von 1.35 Sekunden Dauer berechnet.
- R: R ist die Antwortszeit des Systems (Response time) und dessen Dauer somit systemabhängig.

Die hier erwähnten Ausführungszeiten wurden u.a. von Card abgeschätzt und in (CMN83) veröffentlicht.
Die M-Operatoren werden nach fünf Regeln angeordnet (CMN83, Seite 265). Dabei beschreibt die erste Regel zunächst, wo M-Operatoren gesetzt werden, während nach den folgenden Regeln dann schrittweise wieder M-Operatoren entfernt werden. Diese Regeln lauten im Einzelnen:

1. Platziere M-Operatoren vor alle K-Operatoren, die nicht Teil von Text- oder Nummer-Folgen sind (z.B. Text oder Zahlen). Platziere M-Operatoren vor alle P-Operatoren, die Kommandos auswählen (keine Argumente).
2. Ist ein Operator, der einem M-Operator direkt folgt, völlig zu erwarten, dann lösche den M-Operator (z.B. wird PMK zu PK).
3. Gehört eine Folge von MK-Operatoren zu einer kognitiven Einheit (z.B. Kommandoname), dann lösche alle M-Operatoren außer dem ersten.
4. Ist ein K-Operator ein redundanter Abschluss (Dem Abschluss eines Kommandos folgt der Abschluss dessen Arguments), dann lösche den M-Operator davor.
5. Wenn ein K-Operator einen konstanten String (z.B. Kommandoname) beendet, dann lösche den M-Operator davor; wenn ein K-Operator einen variablen String (z.B. Argument) beendet, dann lasse den M-Operator davor.

3.2.1.2 Bewertung

Das Keystroke-Level-Model bietet einige Vorteile:
Der wohl entscheidenste Vorteil bei KLM ist, dass sich Modelle besonders schnell und einfach erstellen lassen. Außerdem braucht die Person, die das Modell erstellt, keine besonderen Vorkenntnisse über GOMS zu haben.
KLM eignet sich besonders gut für Vergleichsschätzungen um z.B. zu klären, ob für eine bestimmte Aufgabe eine Benutzungsoberfläche X oder Y effizienter ist. Auch lässt sich mit KLM feststellen, wie externe Parameter die Bedienzeit beeinflussen. Dafür kann beispielsweise beim Operator K die unterschiedliche Tippgeschwindigkeit von Nutzern berücksichtigt werden.
Aber die Einfachheit der Technik bringt auch einige Nachteile mit sich. So werden weder Parallelität, noch Unterbrechungen oder die Verschachtelung von Zielen berücksichtigt. Es wird also u.a. davon ausgegangen, dass ein Nutzer sämtliche Interaktionen sequentiell ausführt und nicht in der Lage ist, mehrere Aktionen gleichzeitig zu verrichten. Dies ist natürlich unrealistisch, da ein Mensch z.B. durchaus in der Lage ist, zu denken, während er sich bewegt. Des Weiteren ist KLM nur für kleine Aufgaben bis maximal fünf Minuten Dauer geeignet, da ein, mit KLM erstelltes, Modell auf Grund fehlender Struktur sonst zu komplex und unübersichtlich werden würde. Außerdem ist eine Zielhierarchie nur implizit vorhanden, da die Operatoren bei dieser Variante lediglich in einer bestimmten, festen Reihenfolge aufgelistet, aber keinen Zielen zugeordnet werden. Die Nutzbarkeit dieser Variante bzgl. dieser Arbeit wird in Kapitel 4.1 erläutert.

3.2.1.3 Beispiel

Im Folgenden wird diese Technik an einem Alltagsbeispiel angewendet: In ein Navigationsgerät soll zwecks einer Zielführung das Ziel „Bremen“ eingegeben werden. Wie für ein KLM-Modell üblich, werden Beschreibungen, Operatoren und Ausführungszeiten sequentiell aufgelistet. Dafür werden die in der Fachliteratur üblichen englischen Bezeichner verwendet.

Beschreibung	**Operator**	**Dauer in Sekunden**
Mental Preparation	M	1.35
Move cursor to button „Ziel eingeben“	P	1.10
Click mouse button	K	0.20
Mental Preparation	M	1.35
Move cursor to button „B“	P	1.10
Click mouse button	K	0.20
Move cursor to button „R“	P	1.10
Click mouse button	K	0.20
Move cursor to button „E“	P	1.10
Click mouse button	K	0.20
Move cursor to button „M“	P	1.10
Click mouse button	K	0.20
Move cursor to button „E“	P	1.10
Click mouse button	K	0.20
Move cursor to button „N“	P	1.10
Click mouse button	K	0.20
Mental Preparation	M	1.35
Move cursor to button „Enter“	P	1.10
Click mouse button	K	0.20
Mental Preparation	M	1.35
Move cursor to button „Bevorzugte Straßenart wählen“	P	1.10
Click mouse button	K	0.20
Mental Preparation	M	1.35
Move cursor to button „Autobahn“	P	1.10
Click mouse button	K	0.20
Mental Preparation	M	1.35
Move cursor to button „Route berechnen“	P	1.10
Click mouse button	K	0.20
System response time	R	r
Geschätzte Gesamt-Bearbeitungszeit der Aufgabe		**22,4+r Sekunden**

3.2.2 Card, Moran & Newell GOMS (CMN-GOMS)

3.2.2.1 Konzept

CMN-GOMS gilt als das Original-GOMS. Es wurde, wie der Name bereits andeutet, von Card, Moran und Newell in den 80er Jahren entwickelt. Diese Variante verwendet die Programmform. Mit ihr kann die Bearbeitungszeit einer bestimmten Aufgabe berechnet, sowie die Operatorsequenz dieser vorhergesagt werden. Die Programmform unterstützt die strikte Zielhierarchie eines CMN-GOMS-Modells. Neben den Zielen werden Methoden, Selektionsregeln und „Verify"-Operatoren verwendet, um Interaktionen zwischen Mensch und Maschine auszudrücken. Die „Verify"-Operatoren ersetzen die in KLM kennengelernten M-Operatoren.
Um eine Analyse durchzuführen, muss zunächst eine Zielhierarchie aufgestellt werden. Dazu müssen Ziele und Teilziele der Aufgabe ermittelt werden. Erreicht werden die Ziele mit einzelnen Operatoren, die für jedes Ziel sequentiell aufgelistet werden. Die Gesamt-Bearbeitungszeit der Aufgabe wird, wie bei KLM, durch die Aufsummierung der Operatoren-Ausführungszeiten, unter Berücksichtigung der Zielhierarchie, berechnet.
CMN-GOMS basiert auf zwei der Operationsprinzipien des Model-Human-Processor (MHP), dem Rationalitätsprinzip und dem Problemraumprinzip. Das Rationalitätsprinzip besagt, dass Nutzer, unter Berücksichtigung der Aufgaben-Umgebung, ihrer eigenen Fähigkeiten und Beschränkungen, effiziente Methoden entwickeln werden. Die Betonung liegt dabei auf „effizient", weil Menschen natürlicherweise danach streben, effizient zu handeln. Somit kann man die menschliche Aktivität mit einem Computersystem als „das Ausführen von Methoden um Ziele zu erreichen" betrachten. Das Problemraumprinzip setzt voraus, dass die Nutzeraktivität eine Sequenz von Aktionen ist, die einen Ausgangszustand in einen Zielzustand überführen. Die Aktionen werden dabei als Operatoren und die Sequenz dieser Operatoren als Methode bezeichnet. Diese Methoden können wiederverwendet und wiederholt ausgeführt werden, wenn eine gleiche Zielsituation bemerkt wird.

3.2.2.2 Bewertung

Auch CMN-GOMS-Modelle lassen sich recht schnell und einfach erstellen. Ihr Dateilgrad ist etwas höher als der der KLM-Modelle. Dank der Programmform sind CMN-Modelle allgemeiner, denn Selektionsregeln und Methoden ermöglichen die Verwendung eines Modells für mehrere verschiedene Aufgaben. Je nach Aufgabe kann dann ein anderer Pfad in der Programmstruktur eingeschlagen werden. Zusätzlich macht eben diese Programmform das Modell übersichtlich und strukturiert.
Gut geeignet ist CMN-GOMS für Aufgaben, bei denen die Wiederverwendbarkeit von

Methoden eine wichtige Rolle spielt. Dies könnte z.B. in Aufgaben bzgl. des Editieren von Texten der Fall sein.
Ein Nachteil dieser Variante ist, dass nicht explizit beschrieben wird, wie Methoden repräsentiert werden und wie der Mechanismus, der die Ausführung der Aufgabe steuert, aussieht. Somit sind CMN-GOMS-Modelle relativ undeutlich und unspezifiziert, verglichen mit den zwei im Folgenden beschriebenen GOMS-Techniken. Außerdem berücksichtigt CMN-GOMS keinen „kognitiven Overhead“, d.h. Zeit, die für das Manipulieren von Zielen und dem Kurzzeitgedächtnis und zum Aktivieren und Beenden von Methoden nötig ist. Des Weiteren wird auch hier davon ausgegangen, dass ein Nutzer sämtliche Interaktionen sequentiell ausführt. Parallelität von Aktionen wird also nicht unterstützt. Eine Einschätzung dieser Variante in Bezug auf diese Arbeit wird in Kapitel 4.1 vorgenommen.

3.2.2.3 Beispiel

Im Folgenden wird auch diese Technik, wie im vorherigen Kapitel KLM, an dem Beispiel mit dem Navigationsgerät angewendet. Zu sehen ist die, für ein CMN-GOMS-Modell typische, Einteilung in Ziele (Goals) mit den dazugehörigen Operatoren. Diese werden sequentiell mit den entsprechenden Ausführungszeiten für die Operatoren aufgelistet. Der Vereinfachung halber wurden hier die speziellen Werte der einzelnen Variablen bezüglich der Aufgabe teilweise bereits eingesetzt. Für die Eingabe des Zielortes ist die Selektionsregel auszuführen, die dann, je nach Buchstabenanzahl, die erforderliche Operatorenreihenfolge vorgibt. Zur Berechnung der Bearbeitungszeit wurden hier gedanklich die Buchstaben B, R, E, M, E, N eingegeben. Es werden auch hier die in der Fachliteratur üblichen englischen Bezeichner verwendet. Die hier aufgeführten Ausführungszeiten wurden u.a. von Card abgeschätzt und in (CMN83) veröffentlicht.

Beschreibung	**Dauer in Sek.**
GOAL: ROUTE ERSTELLEN UND BERECHNEN LASSEN	
. GOAL: ZIELORT EINGEBEN	
. . GOAL: EINGABE INITIALISIEREN	
. . . MOVE-CURSOR-TO-BUTTON „Ziel eingeben“	1.10
. . . VERIFY-POSITION	1.35
. . . CLICK-MOUSE-BUTTON	0.20
. . GOAL: EINGABE DURCHFÜHREN	
. . . [select: GOAL: EINGABE FORTFÜHREN...*if input is not complete*	
. . . . MOVE-CURSOR-TO-BUTTON	1.10
. . . . CLICK-MOUSE-BUTTON	0.20
. . . . GOAL: EINGABE DURCHFÜHREN	
. . . GOAL: EINGABE BEENDEN...*if input is complete*	
. . . . VERIFY-ENTRY	1.35
. . . . MOVE-CURSOR-TO-BUTTON „Enter“	1.10
. . . . VERIFY-POSITION	1.35
. . . . CLICK-MOUSE-BUTTON]	0.20
. GOAL: BEVORZUGTE STRAßENART EINGEBEN	
. . MOVE-CURSOR-TO-BUTTON „Bevorzugte Straßenart“	1.10
. . VERIFY-POSITION	1.35
. . CLICK-MOUSE-BUTTON	0.20
. . MOVE-CURSOR-TO-BUTTON „Autobahn“	1.10
. . VERIFY-POSITION	1.35
. . CLICK-MOUSE-BUTTON	0.20
. GOAL: ROUTE BERECHNEN LASSEN	
. . MOVE-CURSOR-TO-BUTTON „Route berechnen“	1.10
. . VERIFY-POSITION	1.35
. . CLICK-MOUSE-BUTTON	0.20
. . SYSTEM-RESPONSE-TIME	r
Geschätzte Gesamt-Bearbeitungszeit der Aufgabe	**22,4+r Sek.**

3.2.3 Natural GOMS Language (NGOMSL)

3.2.3.1 Konzept

Natural GOMS Language wurde Ende der 80er Jahre von David Kieras entwickelt. Es ist eine strukturierte, natürliche Sprachnotation zur Repräsentation von GOMS-Modellen und einer Prozedur, um diese zu erstellen. Diese Variante ermöglicht das Berechnen der Bearbeitungszeit einer Aufgabe, kann Vorhersagen zur Operatorsequenz machen und bietet als einzige GOMS-Variante die Möglichkeit, die Lernzeit von Methoden zu berechnen. NGOMSL-Modelle liegen in Programmform vor. So beschreiben diese Modelle Mensch-Maschine-Interaktionen, ähnlich wie CMN-GOMS, auch mit Zielen, Methoden, Operatoren und Selektionsregeln. Eine Zielhierarchie ist explizit gegeben.
Erstellt wird so ein NGOMSL-Modell durch eine top-down und breadth-first Erweiterung, d.h. es werden top-level Ziele solange in Methoden heruntergebrochen, bis diese nur noch primitive (KLM-) Operatoren enthalten. Dabei wird auf jeder neuen Ebene zunächst in die Breite konstruiert, d.h. alle Ziele einer Ebene werden ermittelt, bevor ein Ziel der nächsttieferen Ebene betrachtet wird. Die Gesamt-Bearbeitungszeit der Aufgabe wird, ähnlich wie bei CMN-GOMS, durch die Aufsummierung der Operatoren-Ausführungszeiten, unter Berücksichtigung der Zielhierarchie und Selektionsregeln, berechnet. Hinzu addiert wird hier die NGOMSL Statement Time, die durch die Anzahl ausgeführter Statements multipliziert mit 0,1 Sekunden berechnet wird. Diese 0,1 Sekunden stellen die Zykluszeit des kognitiven Prozessors dar, auf dem diese Variante basiert. Die Zeit basiert auf der Annahme, dass in einem NGOMSL Modell jedes Statement einer einzelnen Produktionsregel entspricht und der kognitive Prozessor alle 0,1 Sekunden eine neue Produktionsregel verarbeiten kann. Die Gesamt-Lernzeit einer Prozedur ist abhängig von der Anzahl der zu erlernenden Statements, aus der die Prozedur besteht, sowie von der Anzahl der Zugriffe auf das Langzeit-Gedächtnis (LTM Chunks), die für das Ausführen der Prozedur nötig sind. Als einzelne Statements gelten dabei jeder Schritt der entsprechenden Methode, jede Methoden-Aussage, jede Selektionsregel-Aussage, jede if-then Bedingung und jede else Bedingung innerhalb von Selektionsregeln. Ein LTM Chunk könnte beispielsweise das Abrufen eines Menünamens oder das Speichern eines solchen im Langzeitgedächtnis sein. Die in dieser Arbeit berechnete Lernzeit wird wie folgt aufgefasst. Es ist die Zeit, die erforderlich ist, um ein Gerät in allen seinen modellierten Funktionen fehlerfrei und zügig bedienen zu können. Dabei wird vorausgesetzt, dass der Lerner die Ausführung einzelner Operatoren, wie beispielsweise eines Tastendrucks, bereits beherrscht.
Die Methoden von NGOMSL werden mittels einer kognitiven Architektur repräsentiert: der Kognitiven Komplexitätstheorie, auch CCT abgekürzt. Diese Theorie geht von einer einfachen seriellen Architektur aus, bei der das Kurzzeitgedächtnis Produktionsregeln aus-

löst, die in einer festen Frequenz angewendet werden. Diese Regeln verändern die Inhalte des Kurzzeitgedächtnis' oder führen primitive externe Operatoren, wie z.B. einen Tastendruck, aus. GOMS Methoden werden durch Mengen von Produktionsregeln in einem festgelegten Format repräsentiert. Um prozedurales Wissen lernen zu können, müssen diese individuellen Produktionsregeln einzeln gelernt werden und genau nach dieser Idee funktioniert hier letztlich auch die Abschätzung der Lernzeit.

3.2.3.2 Bewertung

Der größte Vorteil und auch gleichzeitig die Besonderheit von NGOMSL ist sicherlich, dass diese Variante, als einzige der vier GOMS-Varianten, die Möglichkeit bietet, Vorhersagen über die Erlernzeit von Prozeduren zu machen. Somit repräsentiert sie als einzige GOMS-Variante explizit die Benutzung vom Arbeits- (auch: Kurzzeit-) und Langzeitgedächtnis. Die Programmform ermöglicht auch hier das Verwenden eines Modells für verschiedene Aufgaben, die mittels Selektionsregeln entsprechend ausgeführt werden. Gleichzeitig ist so ein NGOMSL-Modell durch die Struktur mit expliziter Zielhierarchie und Methoden und mit der zugrunde liegenden kognitiven Theorie sehr übersichtlich und zeichnet sich so durch mehr Formalität und Präzision gegenüber der CMN-Variante aus.
Aber die große Funktionalität von NGOMSL geht auch auf Kosten des Aufwands, der erforderlich ist, um solch ein komplexes Modell zu erstellen. Die Berücksichtigung der Kognitiven Komplexitätstheorie erfordert komplette und exakte Methoden mit wohlüberlegter Ziel- und Arbeitsgedächtnis-Handhabung, deren Konstruktion entsprechendes Vorwissen beim Analysten voraussetzt. Wie bei den vorherigen GOMS-Varianten wird auch hier keine mögliche Parallelität oder Unterbrechung von Aktionen berücksichtigt. Es werden nur hierarchische und sequentielle Methoden verwendet. Des Weiteren ist die Genauigkeit der Lernzeit-Vorhersage vom Stil des Analysten abhängig. Das bedeutet, je nachdem, wie der Analyst Methoden schreibt, ob er z.B. viele Unterziele verwendet oder eher mit wenigeren, allgemeineren Zielen arbeitet, oder wo und wieviele mentale Pausen er einbaut, verändert sich die Berechnung der Lernzeit. Wie die Nutzbarkeit dieser Variante bzgl. dieser Arbeit eingeschätzt wird, ist in Kapitel 4.1 nachzulesen.

3.2.3.3 Beispiel

Im Folgenden wird auch diese Technik, wie im vorherigen Kapitel CMN-GOMS, an dem Beispiel mit dem Navigationsgerät angewendet. Auch hier wurden, zwecks Vereinfachung, die speziellen Werte der einzelnen Variablen bezüglich der Aufgabe teilweise bereits eingesetzt.

NGOMSL Statements	**Ausführungen**	**Dauer in Sek.**
Method for goal: Route erstellen und berechnen lassen.	1	
Step 1. Accomplish goal: Zielort eingeben.	1	
Step 2. Accomplish goal: Bevorzugte Straßenart eingeben.	1	
Step 3. Accomplish goal: Route berechnen lassen.	1	
Step 4. Return with goal accomplished.	1	
Method for goal: Zielort eingeben.	1	
Step 1. Accomplish goal: Eingabe initialisieren.	1	
Step 2. Accomplish goal: Eingabe durchführen.	1	
Step 3. Return with goal accomplished.	1	
Method for goal: Eingabe initialisieren.	1	
Step 1. Determine position of button „Ziel eingeben“.	1	1.20
Step 2. Move cursor to button „Ziel eingeben“.	1	1.10
Step 3. Click mouse button.	1	0.20
Step 4. Return with goal accomplished.	1	
Selection rule set for goal: Eingabe durchführen.	7	
If input is not complete, Then accomplish goal: Eingabe fortführen.	6	
If input is complete, Then accomplish goal: Eingabe beenden.	1	
Return with goal accomplished.	1	
Method for goal: Eingabe fortführen.	1	
Step 1. Determine position of button.	1	1.20
Step 2. Move cursor to button.	1	1.10
Step 3. Click mouse button.	1	0.20
Step 4. Accomplish goal: Eingabe durchführen.	1	
Step 5. Return with goal accomplished.	1	
Method for goal: Eingabe beenden.	1	
Step 1. Verify entry.	1	1.20
Step 2. Determine position of button „Enter“.	1	1.20
Step 3. Move cursor to button „Enter“.	1	1.10
Step 4. Click mouse button.	1	0.20
Step 5. Return with goal accomplished.	1	
Method for goal: Bevorzugte Straßenart eingeben.	1	
Step 1. Determine position of button „Bevorzugte Straßenart“.	1	1.20
Step 2. Move cursor to button „Bevorzugte Straßenart“.	1	1.10
Step 3. Click mouse button.	1	0.20
Step 4. Determine position of button „Autobahn“.	1	1.20
Step 5. Move cursor to button „Autobahn“.	1	1.10
Step 6. Click mouse button.	1	0.20
Step 7. Return with goal accomplished.	1	

NGOMSL Statements	**Ausführungen**	**Dauer in Sek.**
Method for goal: Route berechnen lassen.	1	
Step 1. Determine position of button „Route berechnen“.	1	1.20
Step 2. Move cursor to button „Route berechnen“.	1	1.10
Step 3. Click mouse button.	1	0.20
Step 4. Response Time of the system.	1	r
Step 5. Return with goal accomplished.	1	
Geschätzte Gesamt-Bearbeitungszeit der Aufgabe	**37,9+r Sek.** (=28,6+r+9,3)	
Geschätzte Prozeduren-Lernzeit für 44 Statements	**748 Sek.** (=ca. 12,5 Min.)	

Für die Eingabe des Zielortes ist die Selektionsregel auszuführen, die dann, ähnlich wie im vorherigen Modell, je nach Buchstabenanzahl, die erforderliche Operatorenreihenfolge vorgibt. An der Anzahl der Statements ist abzulesen, wie oft die Selektionsregel und die einzelnen Methoden aufgerufen wurden. So ist ein Rückschluss zu ziehen, dass in dieser Aufgabe ein Ort mit sechs Buchstaben eingegeben wurde.
Zu sehen ist die, für ein NGOMSL-Modell typische, Einteilung in Ziele (Goals) mit den dazugehörigen Methoden und deren Operatoren. Diese werden sequentiell mit den entsprechenden Ausführungszeiten für die Operatoren aufgelistet. Neu ist hier die Angabe der Anzahl an Ausführungen pro Operator und Methode, die im Gegensatz zur Angabe der Ausführungsdauern bereits auf den Unterschied zwischen zu lernenden Statements (Berechnung der Lernzeit) und auszuführenden Operatoren (Berechnung der Bearbeitungszeit) hinweisen. Jede Methode endet mit dem Schritt „Return with goal accomplished.“. Außerdem ist nun auch Rekursion innerhalb von Methoden möglich. Dies führt zu der bei dieser Variante deutlich vertretenen Ziel- und Methodenhierarchie.
Es werden auch hier die in der Fachliteratur üblichen englischen Bezeichner verwendet. Die hier aufgeführten Werte basieren auf Berechnungen von Gong (Gon93).

3.2.4 Cognitive-Perceptual-Motor GOMS (CPM-GOMS)

3.2.4.1 Konzept

Cognitive-Perceptual-Motor-GOMS wurde Ende der 80er Jahre von Bonnie John vorgestellt. Sie gilt als die komplexeste und am schwierigsten zu verwendende GOMS-Variante. Sie erfordert ein spezifisches Analyse-Level, bei dem die primitiven Operatoren einfache wahrnehmende, kognitive und motorische Aktionen sind. Anders als die anderen GOMS-Techniken setzt CPM-GOMS nicht voraus, dass die Operatoren nacheinander ausgeführt

werden, stattdessen können wahrnehmende, kognitive und motorische Operatoren parallel zueinander ausgeführt und Ziele parallel zueinander erreicht werden, je nachdem, wie es die Aufgabe erfordert. CPM-GOMS-Modelle verwenden die Sequenzform und sind in der Lage, Vorhersagen bzgl. der Bearbeitungszeit einer bestimmten Aufgabe zu treffen. Eine explizite Darstellung von Zielen und Methoden gibt es hierbei nicht.
Um die Operatoren mit ihren Abhängigkeiten darzustellen, verwendet CPM-GOMS Zeitplan-Diagramme (engl.: schedule charts), sogenannte PERT-Charts. Deshalb wird die Abkürzung CPM nicht nur als Cognitive-Perceptual-Motor, sondern auch als Critical-Path-Method interpretiert, denn der kritische Pfad in solch einem Zeitplan-Diagramm liefert die Vorhersage der totalen Bearbeitungszeit der entsprechenden Aufgabe.
CPM-GOMS basiert auf dem Model Human Processor (MHP), einer einfachen kognitiven Architektur, die Informationsprozesse eines Menschen modelliert. Die folgende Abbildung 3.2 zeigt skizzenhaft das Prinzip des MHPs:

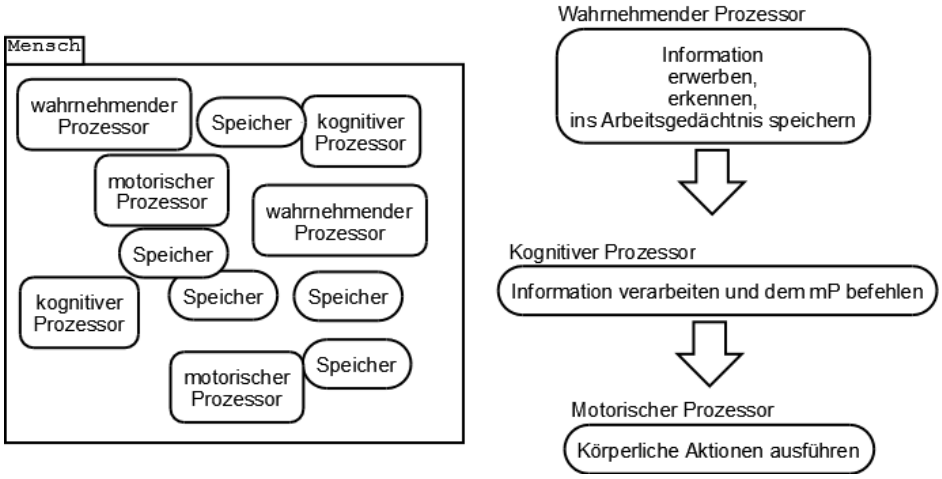

Abb. 3.2: Schemenhafte Darstellung des MHP-Prinzips

Nach diesem Modell besteht ein Mensch aus einer Menge von verschiedenen Prozessoren und Speichern. Es gibt wahrnehmende, kognitive und motorische Prozessoren. Ein kompletter Informationsfluss funktioniert folgendermaßen: Ein wahrnehmender Prozessor nimmt Information auf, erkennt sie und speichert sie dann im Arbeitsgedächtnis. Ein kognitiver Prozessor verarbeitet diese Information anschließend und erteilt einem motorischen Prozessor ggf. Befehle. Der motorische Prozessor sorgt letztlich dafür, dass, entsprechend den Befehlen, körperliche Aktionen ausgeführt werden. Intern, also innerhalb eines Prozessors, laufen die Prozesse stets sequentiell ab. Extern können die Prozesse dagegen parallel ablaufen, d.h. die einzelnen Prozessoren können parallel zueinander arbeiten.
Die CPM-GOMS-Technik verwendet MHP direkt für die Aufgabenanalyse, indem die Operatoren für die einzelnen Prozessoren und deren sequentiellen Abhängigkeiten zueinander identifiziert werden um dann anschließend die Interaktionen der zu analysierenden Aufgabe korrekt anzuordnen.

3.2.4.2 Bewertung

CPM-GOMS ist die einzige GOMS-Variante, die es ermöglicht, parallele Aktivitäten bei der Berechnung der Bearbeitungszeit zu berücksichtigen. Dadurch ist eine sehr detaillierte Modellierung einer Aufgabe möglich.
Der größte Nachteil dieser Variante ist allerdings die Schwierigkeit, mit der diese Technik zu handhaben ist und der erhebliche Aufwand, der nötig ist, um so ein Modell zu erstellen. Gerade das zugrunde liegende Prinzip des MHP erfordert ein großes Vorwissen vom Analysten, damit dieser in der Lage ist, zu identifizieren und zu beschreiben, wie wahrnehmende, kognitive und motorische Aktivitäten zeitlich koordiniert werden können. Der extrem hohe Aufwand bei der Erstellung eines Modells führt dazu, dass diese Variante nur dann angewendet werden sollte, wenn die unbestimmte Reihenfolge der Operatoren eine entscheidende Rolle für die Zeitberechnung spielt. Des Weiteren wird vorausgesetzt, dass der Nutzer, der die Aufgabe bearbeitet, sehr geschickt und erfahren im Umgang mit dem verwendeten Programm für die Aufgabe und dem Computer allgemein ist. Denn bei CPM-GOMS wird stets davon ausgegangen, dass der Nutzer alle möglichen parallel ausführbaren Aktivitäten auch tatsächlich parallel und in der schnellstmöglichen Zeit ausführt, die das MHP erlaubt. Dass dies i.d.R. eine zu optimistische Annahme ist, leuchtet ein. Die Eignung dieser Variante bzgl. dieser Arbeit wird in Kapitel 4.1 erläutert.

3.2.4.3 Beispiel

Im Folgenden wird auch diese Technik, wie im vorherigen Kapitel NGOMSL, an dem Beispiel mit dem Navigationsgerät angewendet. Allerdings ist hier auf Grund der Detailliertheit eines CPM-GOMS-Modells nur ein kleiner Ausschnitt des Beispiels modelliert, den Abbildung 3.3 zeigt.
Um ein CPM-GOMS-Modell zu erstellen, wird zunächst ein CMN-GOMS-Modell erstellt, das auf dem Aktivitäts-/Motorlevel stoppt, um dann die Operatoren dort auf ein niedrigeres Level abzusenken, wo diese Operatoren dann zu neuen Zielen werden. Die einzelnen Operatoren werden anschließend auf die drei Prozessoren des MHP abgebildet.
Auf der linken Seite des PERT-Charts sind die Prozessoren, unterteilt in „Visual Perception“, „Cognitive Operators“, „Right Hand Motor Operators“, „Left Hand Motor Operators“ und „Eye Movement Operators“, zu sehen. Ein Operator wird als ein Rechteck dargestellt. Horizontale Linien zwischen den Rechtecken veranschaulichen die Sequenz von Operatoren eines Prozessors. Ebenso weisen alle Linien, auch diagonale, zwischen Rechtecken auf sequentielle Abhängigkeiten hin. Die fünfeckigen Objekte fassen eine Operatorsequenz zu möglichen Teilzielen zusammen. Hier wird der Detailgrad deutlich, denn diese Teilziele waren in den vorherigen GOMS-Modellen einzelne Operatoren. Die untere Zahl über

den Rechtecken gibt die Ausführungsdauer des entsprechenden Operators, die obere Zahl die Gesamtausführungsdauer bis zu dem Punkt in Millisekunden wieder. Um die totale Bearbeitungszeit einer Aufgabe/Sequenz von Operatoren zu berechnen, wird der kritische Pfad ermittelt. Dies ist der Pfad mit den jeweils längsten Bearbeitungszeiten von einzelnen Operatoren. Am letzten Objekt des kritischen Pfades wird letztlich die totale Bearbeitungszeit abgelesen. Der kritische Pfad ist hier mittels einer fettgedruckten Linie markiert.

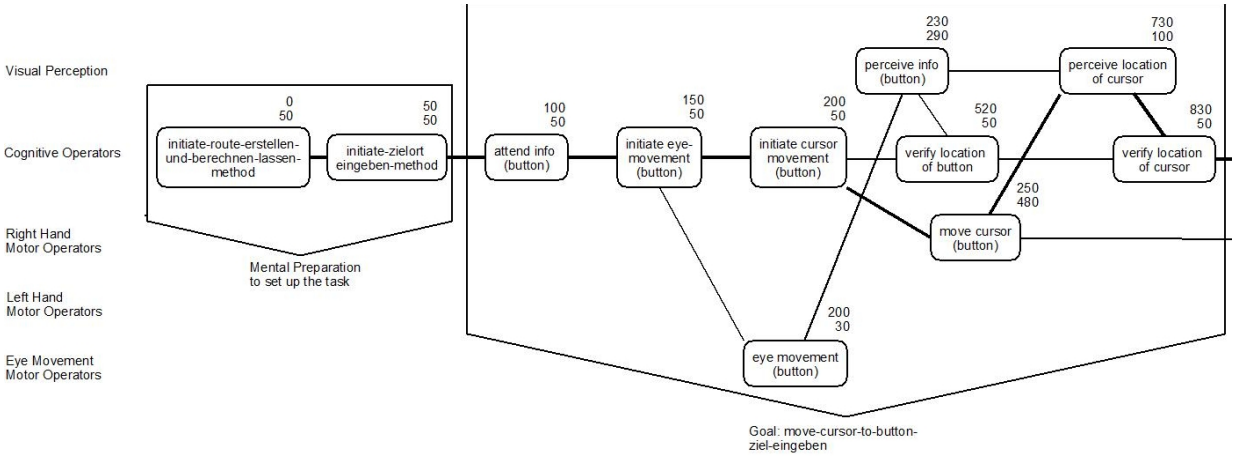

Abb. 3.3: Ausschnitt eines CPM-GOMS-Modells im PERT-Chart

Geschätzte Bearbeitungsdauer dieser Subsequenz: 880ms = 0,88s (bei CMN: 2,45s). Die hier verwendeten Operatorzeiten basieren auf Bearbeitungsdauern, geschätzt von John und Gray (JG94).

3.2.5 Erweiterungen von GOMS

Aber auch bei diesen vier Varianten bleiben einige Fragen bezüglich der Analyse offen. So gehen alle GOMS-Varianten davon aus, dass ein, im Umgang mit computergestützten Systemen, erfahrener Nutzer die zu analysierende Benutzungsoberfläche bedient. Weniger kompetente und ungeschicktere Nutzer werden von GOMS-Modellen nicht berücksichtigt. Des Weiteren beruhen alle GOMS-Varianten auf der Annahme, dass dem Nutzer keine Fehler passieren. Das ist natürlich unrealistisch. Um einiger dieser Schwächen entgegenzuwirken, wird an verschiedenen Erweiterungen von GOMS gearbeitet. Eine recht neue, im Jahr 2007 präsentierte, Erweiterung ist Multitasking GOMS. Diese Erweiterung wurde am Fraunhofer-Institut entwickelt und ermöglicht, wie der Name bereits andeutet, die Modellierung einer Nebenaufgabe, ohne dabei die Anforderungen einer Hauptaufgabe zu vernachlässigen. Mit dieser Methode kann berechnet werden, inwieweit Haupt- und Nebenaufgabe sich überlappen bzw. beeinflussen. Mit Hilfe von Multitasking-GOMS sollen

„In-Vehicle Information Systems“ bewertet werden. Auf eine Erweiterung bzgl. der Handhabung von Fehlern in GOMS-Analysen, wird im Folgenden genauer eingegangen.

Schon Card, Moran und Newell wiesen in (CMN83) auf die Notwendigkeit einer solchen Fehler-Erweiterung hin, um Aufgaben vollständig modellieren zu können. Wood griff ihren Ansatz in seiner Arbeit (Woo00, Kapitel 5) auf und entwickelte ein eigenes Konzept, um GOMS bzgl. Fehlern zu erweitern. Beide Ansätze gehen davon aus, dass ein Nutzer fünf Stufen durchläuft, wenn er einen Fehler begeht. Die erste Stufe ist die, auf der der Nutzer den Fehler macht. Darauf folgt die Erkennungsstufe, auf der dem Nutzer bewusst wird, dass er einen Fehler gemacht hat. Die nächste Stufe unterscheidet sich zwischen den beiden Ansätzen. Card, Moran und Newell sprechen an dritter Stelle von der Stufe der Rücksetzung. Dort setzt der Nutzer das Programm zurück, um eine Fehlerkorrektur zu ermöglichen. Wood dagegen bezeichnet die dritte Stufe als Identifikationsstufe, auf der der Nutzer den Fehlertyp identifiziert. Auf eine Rücksetzung verzichtet Wood. Die zwei letzten Stufen werden in beiden Ansätzen wieder gleich bezeichnet. Es sind die Korrekturstufe an vierter, und die Stufe der Wiederaufnahme an fünfter Stelle. Auf der vierten Stufe korrigiert der Nutzer die Auswirkungen des Fehlers, während er auf der fünften Stufe die normale Ausführung der Aufgabe wieder aufnimmt. Die Abweichung der dritten Stufe bei Wood basiert auf dessen Annahme, dass ein Nutzer, wenn er erkannt hat, dass er einen Fehler begangen hat, nicht gleichzeitig weiß, um was für einen Fehler es sich dabei handelt. Aus diesem Grund führt Wood die Identifikationsstufe ein. Eine Rücksetzungsstufe hält er für zu speziell. Hierbei sollte erwähnt werden, dass Card, Moran und Newell ihr Konzept auf die Arbeit mit einem Texteditor beziehen, in dem eine solche Stufe sinnvoll erscheinen mag. Um das Konzept allerdings nicht auf eine Anwendung zu reduzieren, eliminierte Wood diese Stufe.
Beide Konzepte klassifizieren zudem unterschiedliche Arten von Fehlern, die sich in den beiden Arbeiten jedoch deutlich unterscheiden. Auf eine genaue Beschreibung dieser Fehlerarten wird an dieser Stelle verzichtet, da hierfür eine ausführliche Ausweitung nötig wäre, die irrelevant für den weiteren Verlauf dieser Arbeit ist.
Die beiden Konzepte verfolgen das Ziel, die Fehlerbetrachtung, in Form von zusätzlichen Operatoren und Fehlerbehebungs-Methoden, direkt in ein fertiges[1] GOMS-Modell zu integrieren. Anhand unterschiedlicher Algorithmen werden dann Fehler vom GOMS-Modell generiert, die direkt in die Modellierung des Verlaufs der einzelnen Aufgabeninstanzen einfließen.
Während dieser Erweiterung werden kritische Methoden im Modell identifiziert, d.h. solche, die den Erfolg der Aufgabe maßgeblich beeinflussen können. Diese Methoden sollten besonders fehlertolerant sein und werden dahingehend überprüft und ggf. überarbeitet.

[1] Mit einem fertigen Modell ist hier ein solches gemeint, dass auf der Annahme fehlerfreier Ausführung erstellt wurde.

Um den Nutzer in der Fehlererkennung und -behebung zu unterstützen, werden Behebungsmethoden, Operatoren und Feedback gezielt eingesetzt.

3.2.6 Werkzeuge für GOMS-Analysen

Um GOMS-Analysen zu unterstützen und teilweise zu automatisieren, wurden bereits diverse Werkzeuge entwickelt. Im Folgenden wird ein kleiner Einblick in einige bekannte Werkzeuge gegeben.

3.2.6.1 CogTool

Das CogTool ermöglicht die Erstellung von KLM-Modellen und anschließend die automatische Berechnung von deren Bearbeitungszeiten. Entwickelt wird das Werkzeug von Bonnie E. John und weiteren Forschern der Carnegie Mellon University in Pittsburgh, USA. Zur Erstellung eines Modells wird zunächst das Design der Benutzungsoberfläche mit den dazugehörigen Transitionen zwischen den einzelnen Fenstern in Form eines Storyboards modelliert (s. Abbildung 3.4). Anschließend wird eine Aufgabe erstellt, deren Bearbeitungszeit berechnet werden soll. Darauffolgend wird zu dieser Aufgabe und dem Design ein Skript geschrieben, das schließlich die einzelnen Operatoren in Form eines KLM-Modells auflistet und die Bearbeitungszeit berechnet. Um dieses Skript erstellen zu können, muss der Analyst, wie ein potentieller Nutzer es tun würde, durch die einzelnen Bildschirmanzeigen navigieren, damit das Werkzeug die dazu nötigen Operatoren ermitteln kann.

Die Berechnungen des CogTools basieren auf der komplexen kognitiven Architektur ACT-R, auf die in Kapitel 3.1.1.3 kurz eingegangen wurde, zuzüglich des „Fitt's Law“[2]. Das Fittsche Gesetz ermöglicht die differenzierte Berechnung von motorischen Operatorzeiten. Es besagt, dass die Zeit, die erforderlich ist, um einen Mauszeiger bzw. die Hand, von einem Punkt zu einem anderen zu bewegen, abhängig von der Größe des Ziels und der Entfernung zum Ziel, z.B. einer Schaltfläche, ist. Dieses Gesetz ermöglicht dem CogTool also, eine individuelle Zeit pro motorischem Operator zu berechnen und dadurch eine genauere Gesamt-Bearbeitungszeit voherzusagen. Das CogTool beinhaltet zusätzlich ein Visualisierungswerkzeug, mit dem die genauen Schritte inkl. der berechneten Operatorzeiten, die ACT-R zur Berechnung der Bearbeitungszeit vornimmt, nachvollzogen und validiert werden können (Joh08).

Das Programm liegt aktuell in der Version 1.0b24 vom 7.9.2008 für Mac OS X (10.3 oder mehr, inkl. Power PC und IntelMacs) und Windows (Windows XP oder Windows Vista)

[2]Quelle: Briefverkehr mit Bonnie E. John am 12.9.2008

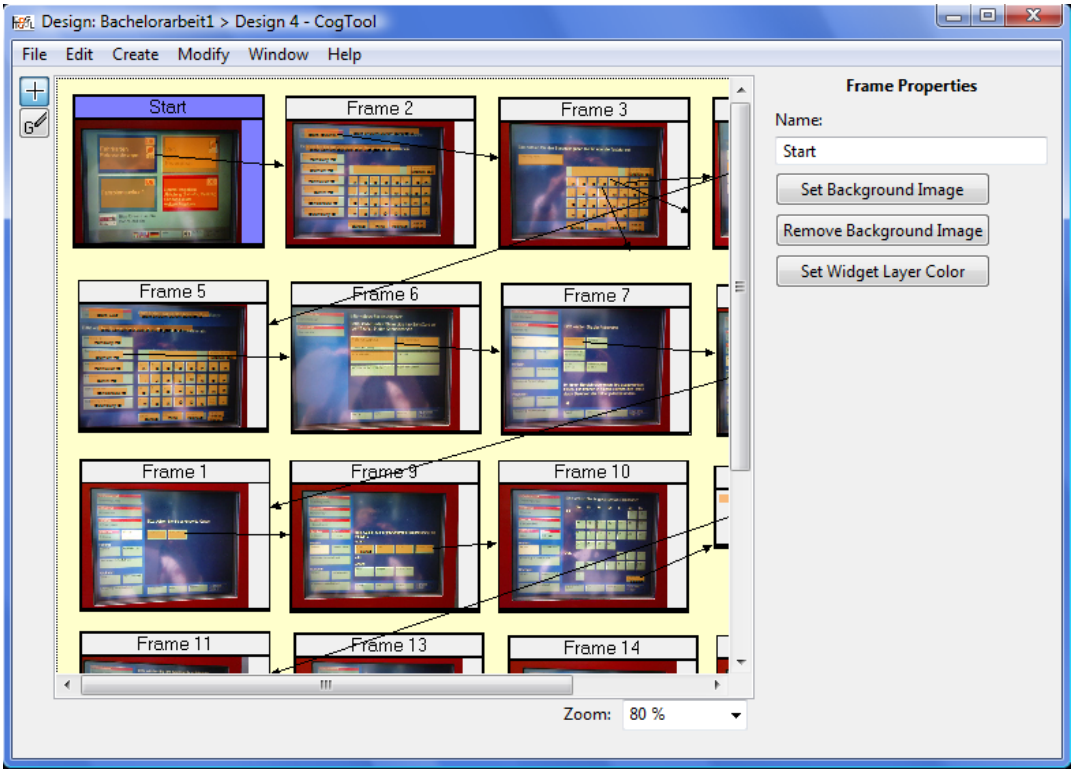

Abb. 3.4: Ausschnitt eines Storyboards im CogTool

vor. Vorgänger war die in dieser Arbeit verwendete Version 1.0b18 vom 14.2.2007. Das Werkzeug kann in beiden Versionen kostenlos von der Internetseite des CogTool Projekts (Car06) heruntergeladen werden. Dort findet sich ebenfalls ein ausführliches Handbuch, sowie ein Tutorial und weitere Publikationen zu dem Thema.

3.2.6.2 GLEAN4

Das Akronym GLEAN steht für GOMS Language Evaluation and Analysis. GLEAN ist eine Konsolenanwendung (s. Abbildung 3.5), die bezüglich eines GOMS-Modells unterschiedliche Usability-Messdaten erstellen kann. Voraussetzung dafür ist, dass im Vorfeld ein komplettes GOMS-Modell, einige Benchmarkaufgabenstellungen, sowie eine abstrakte Darstellug der Benutzungsoberfläche erstellt wurden. GLEAN erstellt daraus Messdaten wie die Lern- und Bearbeitungszeit. Das Werkzeug macht folglich GOMS-Modelle handlicher und automatisiert deren Ausführung (RY01). Es nutzt die eigene Sprache GOMSL, die eine formalisierte und maschinen-ausführbare Notation von NGOMSL ist. Wie NGOMSL bleibt auch GOMSL nah an der natürlichen (englischen) Sprache, sodass sie ohne große Einarbeitungszeit lesbar ist. Die im Vorfeld erstellten GOMS-Modelle müssen in diesem *.gomsl-Format vorliegen und können mit jedem Texteditor erstellt werden. Eine Anleitung dazu findet sich in (Kie06).

Die aktuelle Version GLEAN4 orientiert sich an dem originalen GLEAN von 1993, das

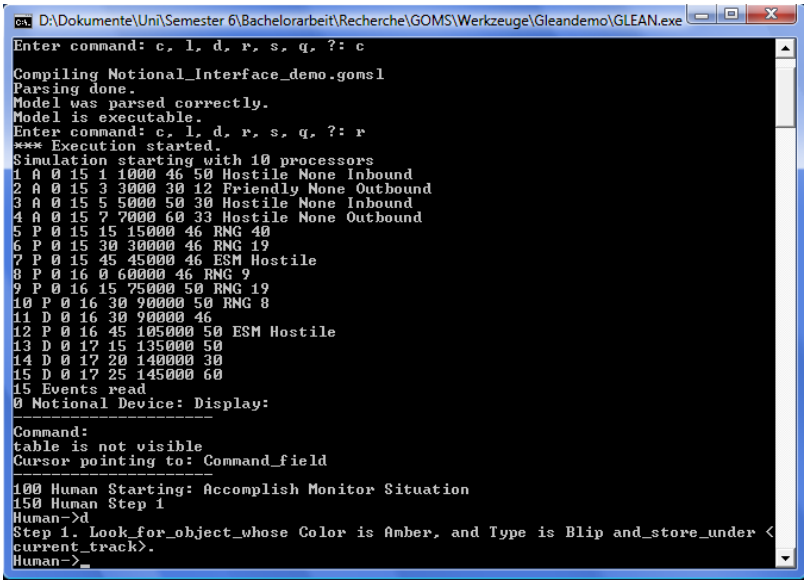

Abb. 3.5: Screenshot von GLEAN

von Scott Wood entwickelt wurde. GLEAN4 ist eine Erweiterung von GLEAN3 und wurde von David Kieras, Scott Wood, Kasem Abotel und Anthony Hornof an der University of Michigan reimplementiert und ausgearbeitet. Seit GLEAN3 wird die Entwicklung, statt vorher in LISP, in C++ vorgenommen, sodass diese Anwendung plattformunabhängig ist.

3.2.6.3 GOMSED 2.0

GOMSED 2.0 ist ein Editor zum Erstellen und Auswerten von sequentiellen GOMS-Modellen. Das bedeutet, dass sich mit GOMSED sowohl KLM- als auch NGOMSL-ähnliche Modelle erstellen lassen. Das Werkzeug ist in der Lage, die Ausführungszeiten bestimmter Aufgaben und die Lernzeiten für das gesamte Modell zu berechnen. Dabei können die Ausführungszeiten einzelner Operatoren über Parameter individuell angepasst werden, um Werte wie z.B. die Geübtheit des Nutzers mit der Maus oder die Länge der Strecke, die durch Cursor-Positionierung zurückgelegt werden muss, einfließen zu lassen. Die Abbildung 3.6 zeigt dieses Werkzeug.
Die Erstellung eines GOMS-Modells mit GOMSED beginnt mit der Konstruktion eines Aufgabenbaums. Dieser stellt hierarchisch-sequentiell den Ablauf einer Aufgabe mittels Wurzelknoten, innerer Knoten und Blättern dar. Dabei repräsentieren der Wurzelknoten das Aufgabenziel, die inneren Knoten die dazu gehörigen Unterziele und die Blätter die Operatoren. Wird als Schritt der Aufgabenbearbeitung der Wurzelknoten oder ein innerer Knoten ausgeführt, so werden dadurch auch alle Ziele aktiviert, die durch Kinder dieses Knotens dargestellt sind. Dabei wird nach der festen Reihenfolge vorgegangen, die die Baumstruktur vorgibt. Auswahlregeln werden durch Auswahlregelknoten modelliert,

deren Kinder die zur Auswahl stehenden Operatoren sind, die mit Ausführungswahrscheinlichkeiten versehen werden. Die einzelnen Parameter und Zeiten pro Operator können individuell angepasst werden. Anhand eines solchen Aufgabenbaums kann GOMSED dann die Ausführungs- und Lernzeit berechnen (Wan02).

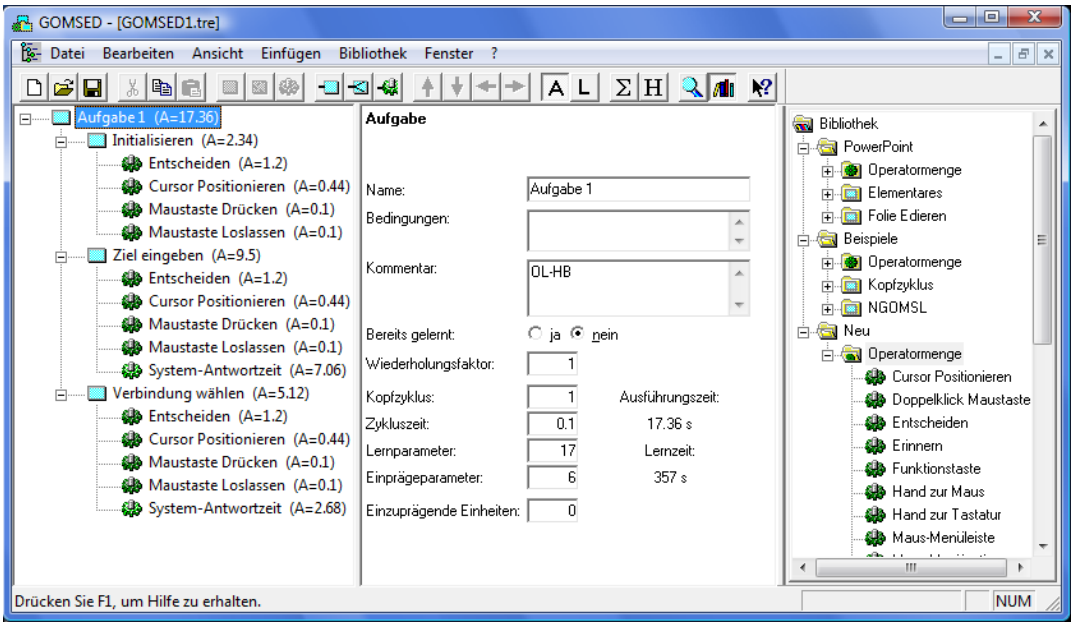

Abb. 3.6: Screenshot eines geöffneten Modells in GOMSED 2.0

Entwickelt wurde das Werkzeug von D. Feldmann und J. Wandmacher an der Technischen Universität Darmstadt. Die aktuelle Version 2.0 existiert seit 1998 und wurde für MS Windows 95 konstruiert. Diese Version kann in eigeschränkter Form kostenlos von der Internetseite der Universität (Tec04) heruntergeladen werden. Um auch die Exportfunktion nutzen zu können, muss eine kostenpflichtige Version erworben werden. Die Hilfe des Werkzeugs soll eine Einführung in die Benutzung des Editors geben. Zwar funktioniert das Werkzeug, nach eigenen Erfahrungen, auch unter MS Windows Vista, allerdings ist dort die Hilfe nicht abrufbar, da sie in einem nicht mehr unterstützten älteren Windows-Hilfeformat vorliegt. Da es außer dieser Hilfe keine weiteren Dokumente zur Einführung in das Werkzeug gibt, könnte sich die Arbeit damit unter aktuellen Windows-Versionen als schwierig erweisen. Unter Windows XP läuft das Programm inkl. der Hilfe-Bibliothek allerdings noch problemlos. Auf oben angegebener Seite wird außerdem eine Publikation zu weiterführenden Möglichkeiten und Problemen von Analysen mit dem GOMSED Editor bereitgestellt. Der GOMSED-Editor wird nicht mehr weiterentwickelt.

4 Anwendung der GOMS-Techniken am Beispiel eines Fahrkartenautomaten

Dieses Kapitel widmet sich der Analyse des Fahrkartenautomaten. Zunächst werden die Gründe für die Auswahl der GOMS-Familie als Analyseverfahren erläutert, bevor auf die Auswahl bestimmter Varianten dieser Familie eingegangen wird. Darauf folgt die Beschreibung des Vorgehens, sowie der Durchführung und Auswertung der Analyse.

4.1 Auswahl von GOMS

Die Verwendung der GOMS-Familie zur Analyse der Bedienerführung erscheint hier auf Grund mehrerer Aspekte sinnvoll. Mit Hilfe der GOMS-Varianten können u.a. die Bearbeitungs- und Lernzeiten für bestimmte Aufgaben mit einer bestimmten Benutzungsoberfläche vorhergesagt werden. Mit diesen Zeiten lassen sich Rückschlüsse auf die in Kapitel 2.1 beschriebenen Usability-Kriterien „Aufgabenangemessenheit“, „Lernförderlichkeit“ und „Erwartungskonformität“ ziehen. Während die Aufgabenangemessenheit durch die Bearbeitungszeit bewertet werden kann, liefert die Lernzeit Erkenntnisse über die Lernförderlichkeit und die Konsistenz des Geräts, die Teil der Erwartungskonformität ist. Empirische Untersuchungen können dagegen theoretisch Aussagen bzgl. aller sieben vorgestellten Usability-Kriterien treffen. Jedoch ist es wegen der fehlenden Formalität dieser Verfahren praktisch unmöglich, alle diese Kriterien in einer Analyse zu berücksichtigen bzw. die Ergebnisse der Untersuchungen im Nachhinein wieder zu rekonstruieren, um mit diesen auf Optimierungsbedürfnisse spezieller Designeigenschaften der Benutzungsoberfläche schließen zu können. Die Varianten der GOMS-Familie, stellen im Gegensatz dazu den Aufbau eines Designs in einer inspizierbaren Form dar. Dadurch ermöglichen sie neben einer späteren Wiederverwendung erfolgreicher Designaspekte auch eine genauere Spezifizierung evtl. Problemstellen einer Benutzungsoberfläche. Dies verschafft ihnen gegenüber reinen empirischen Verfahren einen entscheidenen Vorteil.

Des Weiteren ist die Einarbeitungszeit in die Konzepte der einzelnen GOMS-Varianten und der Schwierigkeitsgrad dieser recht gering. So zeigte sich in der Praxis, dass z.B. für das Erlernen der KLM- und CMN-Variante lediglich eine Unterrichtseinheit inkl. einiger Hausaufgaben ausreicht, um eigene GOMS-Modelle zu konstruieren, die bzgl. Designentscheidungen akkurate Aussagen treffen können (JK96b). Um NGOMSL zu erlernen, sind einige wenige Unterrichtseinheiten zuzüglich einiger Hausaufgaben und Feedback nötig. Für CPM-GOMS erstellten John und Gray (JG94) bereits eine Reihe von PERT-chart Vorlagen für häufige Situationen, die als Teilstücke in Modellen kombiniert werden können. Nach einigen wenigen Unterrichtseinheiten sind Schüler bereits in der Lage, existierende CPM-GOMS-Modelle korrekt zu manipulieren und durch das daraus erlangte Wissen eigene Modelle zu skizzieren. So ist es folglich auch „GOMS-Laien“ innerhalb recht kurzer Einarbeitungszeit möglich, selbst akkurate GOMS-Modelle zu erstellen.
Auch die Verwendung der GOMS-Methoden, also die Analyse an sich, erfordert einen durchaus akzeptablen Zeitrahmen. Gong zeigte in (Gon93) an einem konkreten Beispiel, dass die Verwendung von NGOMSL einen Bruchteil der Programmierzeit und wesentlich weniger Zeit beansprucht und Kosten verursacht, als z.B. eine empirische Evaluierung. Insbesondere gilt dies, wenn die Zielgruppe Experten einer bestimmten Domäne sind (Kie96). Bei einer empirischen Evaluierung ist zudem eine große Anzahl von Probanden nötig, um wirklich repräsentative Aussagen über ein System machen zu können. Für eine GOMS-Analyse dagegen reicht ein einziger Analyse-Durchlauf aus, um bereits konkrete und auch gültige Aussagen bzgl. des zu evaluierenden Systems formulieren zu können. Auch ist die Anwendung von GOMS nicht aufwendiger als die anderer formaler Methoden, wie z.B. des Cognitive Walkthrough. Diese Methode wird bereits wegen ihrer schnellen und effizienten Anwendung als „discount technique“ bezeichnet. Tatsächlich ist ein KLM-Modell fast ausschließlich die einleitende Arbeit für einen Cognitive Walkthrough, nämlich die Definition von Nutzeraktivitäten auf der Keystroke-Ebene. Auch für die Definition von CMN-GOMS- oder NGOMSL-Modellen wäre dies bereits ein wesentlicher Teil der Arbeit. Demnach sollte auch die GOMS-Modellierung als eine schnelle und effiziente Technik angesehen werden und nicht, wie fälschlicherweise oft geschehen, als extrem zeit- und arbeitsaufwendig (LR94).
Auch ist GOMS in allen Phasen der Systementwicklung anwendbar. So ermöglichen GOMS-Modelle nicht nur akkurate Aussagen über Systeme in deren Entwicklungsphase, sondern auch in deren Redesign- oder Evaluierungsphase nach der Implementierung, wie es hier der Fall ist.
Natürlich spielen auch die Kosten und benötigten Ressourcen im Rahmen dieser Bachelorarbeit eine entscheidene Rolle. Zu den wichtigsten Veröffentlichungen bezüglich der GOMS-Familie und ihrer konkreten Anwendung verschaffte insbesondere der öffentliche

FTP-Server[1] von Herrn David Kieras kostenlosen Zugang. Auch die kostenlose Nutzung des Analyse-Werkzeugs CogTool war möglich.

Die Auswahl einer geeigneten GOMS-Variante bestimmt auch gleichzeitig den Aufwand, der für die Erstellung eines Modells nötig ist. Um die richtige GOMS-Variante zu finden, sollten folgende Fragen beantwortet werden:

- Ist es ausschließlich von Interesse zu erfahren, ob zwei verschiedene Designs beide wichtige Funktionalität abdecken?
- Ist die Bearbeitungszeit eines geübten Nutzers wichtig für die langfristige Nutzung des Systems?
- Erhöht ein häufiger Benutzerwechsel den Stellenwert der Lernzeit?

In der ersten Situation reicht eine eher oberflächliche CMN-GOMS-Analyse aus, während die zweite eine, je nach Stellenwert der Parallelität, tiefergehende CMN- oder CPM-GOMS-Analyse erfordern könnte. Die dritte Situation dagegen benötigt ein umfangreiches NGOMSL-Modell (JK96b).

Begonnen wurde hier mit der Analyse anhand von KLM-Modellen, weil KLM die schnellste und einfachste Methode ist, mit der in kurzer Zeit ein Überblick über die einzelnen Aufgaben und deren Bearbeitungszeiten gewonnen werden kann. Die Definition von Operatoren auf der Keystroke-Ebene dient außerdem als Grundlage für die weiteren GOMS-Varianten, da diese dort wiederverwendet werden können. Auch deshalb lohnt es sich, mit dieser Variante einzusteigen. Wurde ein KLM-Modell erstellt, ist bereits eine Abschätzung von Schwachstellen der Benutzungsoberfläche möglich, anhand dieser dann besser entschieden werden kann, ob eine weitere Variante voraussichtlich mehr oder genauere Erkenntnisse liefern würde. Da es sich generell bei den Aufgaben, die mit einem Fahrkartenautomaten bearbeitet werden, um kurze Aufgaben handelt, ist KLM ebenfalls als geeignete Methode anzusehen.
Als weitere Variante erschien hier NGOMSL sinnvoll. An Fahrkartenautomaten herrscht ein ständiger Benutzerwechsel, wobei die unterschiedlichsten Nutzer das Gerät bedienen und jeder einzelne von ihnen dies i.d.R. lediglich ein paar Minuten macht. Dieses Nutzungsverhalten macht die Abschätzung der Lernzeit interessant, wozu von den GOMS-Varianten ausschließlich NGOMSL fähig ist.
CPM-GOMS schied hier komplett aus, da diese Methode auf Grund ihres Aufwands nur dann benutzt werden sollte, wenn die Parallelität von Aktionen entscheidend ist. Das ist hier nicht der Fall, da bei der Bedienung eines Fahrkartenautomaten hauptsächlich

[1] `ftp://ftp.eecs.umich.edu/people/kieras/`

sequentielle Operationen statt finden. Außerdem werden hier keine verschiedenen Designs von Benutzungsoberflächen verglichen, sodass sich die Erstellung eines ersten, extrem aufwendigen CPM-GOMS-Modells nicht lohnen würde, da es nicht für weitere Designs wiederverwendet werden könnte und sich so der hohe Aufwand des ersten Modells nicht durch die Anzahl weiterer Designs auszahlen könnte.
Da bereits eine KLM-Analyse durchgeführt wurde, erschien eine zusätzliche CMN-GOMS-Analyse hier unnötig, denn diese hätte vermutlich keine weiteren nennenswerten Erkenntnisse gebracht.

4.2 Vorgehen

Zur Durchführung der Analyse wurden verschiedene Vorbereitungen getroffen. Nach einer intensiven Auseinandersetzung mit dem Fahrkartenautomaten wurde ein Aufgabenkatalog erstellt, anhand dessen sowohl die GOMS-Modelle aufgebaut, als auch einige Tests mit Probanden durchgeführt wurden. Dieser Aufgabenkatalog befindet sich im ersten Anhang.
Die Aufgaben stellen fünf unterschiedliche Nutzerabsichten dar, die möglichst viele verschiedene Aufgabenfelder abdecken sollen. Dies wurde dadurch erreicht, dass zwischen kürzeren und längeren Verbindungen, der Anzahl Reisender, der Verfügbarkeit von Bahncards, der Wahl einer Rückfahrt, Platzreservierung oder Zugart, den Reisedaten, als auch den Tickettypen und Startbahnhöfen variiert wurde.
Des Weiteren wurden willkürlich fünf Probanden ausgewählt, die diese Aufgaben mit Hilfe des Fahrkartenautomaten erledigten. Zwecks späterer Analyse wurden die Aktionen aller Probanden dabei auf Video aufgezeichnet. Die Videoaufzeichnungen dienten, neben der Fixierung einiger Handlungsabläufe, der Ermittlung der Denkzeiten der Nutzer und der Antwortzeiten des Systems. In den erstellten Modellen wurden die Durchschnittswerte aller verfügbaren Aufzeichnungen verwendet. Zusätzlich zu den fünf willkürlich ausgewählten Probanden, die zufälligerweise alle ungeübt im Umgang mit dem Fahrkartenautomaten waren, wurde die Interaktion eines im Umgang mit dem Fahrkartenautomaten geübten Probanden aufgezeichnet und ausgewertet. Als geübter Proband wird hier eine Person bezeichnet, die bestens vertraut mit der Bedienung des Fahrkartenautomaten und mit den Aufgabenstellungen ist, d.h. die die Aufgaben bereits im Vorfeld mehrmals mit dem Fahrkartenautomaten bearbeitet hat, bevor die von ihr benötigten Bearbeitungszeiten gemessen wurden. Diese Werte wurden mit denen der von den GOMS-Varianten abgeschätzten Bearbeitungszeiten verglichen und so eine Validierung der verwendeten GOMS-Methoden vorgenommen. Der Vergleich mit den Werten der anderen Probanden erschien nicht sinnvoll, da diese ausnahmslos Fehler machten und deshalb Rückschritte vornahmen. Da die

GOMS-Abschätzungen auf der Annahme beruhen, dass die Nutzer keine Fehler machen, waren diese Daten nicht für solch einen Vergleich geeignet.
Um möglichst realistische Operatorzeiten verwenden zu können, wurden mit Hilfe der Videos auffallend lange (über einer Sekunde Dauer) System-Antwortzeiten ermittelt, diese Zeiten in allen verfügbaren Videoaufzeichnungen gemessen und deren Durchschnitt dann als Operatorzeit des „Wait for"-Statements an den entsprechenden Stellen verwendet. Auch Denkpausen der Probanden wurden ermittelt. Bei der Auswahl der Denkpausen wurde darauf geachtet, dass lediglich an den Stellen Denkpausen ermittelt wurden, an denen auch geübte Nutzer Denkzeit benötigen würden. Dies ist z.B. an der Stelle der Fall, an der der Nutzer die ermittelten Verbindungen präsentiert bekommt und sich für eine Verbindung entscheiden muss. Die Verbindungen variieren ständig und müssen deshalb jedes Mal neu bedacht werden, unabhängig von der Geübtheit des Nutzers mit dem Gerät.
Im Vorfeld der Analyse musste die Entscheidung für eine geeignete Analysemethode getroffen werden. Der Gedankenprozess der Methodenfindung wird in Kapitel 4.1 beschrieben.
Für die Erstellung der KLM-Modelle wurde das Werkzeug „CogTool" (s. Kapitel 3.2.6.1) verwendet. Zum Zeitpunkt der Erstellung existierte die aktuelle Version des Werkzeugs noch nicht, sodass mit der Version 1.0b18 gearbeitet wurde. Mit Hilfe des CogTools wurde pro Aufgabe zunächst ein Storyboard erstellt, das das Design der Benutzungsoberfläche und die Übergänge zwischen den einzelnen Bildschirmanzeigen modelliert. Im Anschluss wurde jeweils ein Skript geschrieben, d.h. die einzelnen Operatoren, die zur Ausführung einer bestimmten Aufgabe benötigt werden, wurden in Form eines KLM-Modells aufgelistet. Dies geschieht im CogTool durch Interaktion mit dem zuvor erstellten Design in Form des Storyboards. Die Berechnung der einzelnen Bearbeitungszeiten anhand der erstellten KLM-Modelle wurde per Hand durchgeführt. Die Begründung dieser Entscheidung wird in Abschnitt 4.3.3 erläutert.
Im nächsten Schritt wurde das NGOMSL-Modell entwickelt. Dabei wurde strikt nach dem NGOMSL-Leitfaden von Kieras (Kie96) vorgegangen. Auf die Verwendung eines Werkzeugs wurde verzichtet. Die gesichteten Werkzeuge erschienen entweder ungeeignet oder hätten erheblich längere Einarbeitungs- und Durchführungszeiten benötigt. GOMSED schien einerseits durch seine vollständig eingedeutschten Operatorenbezeichner, andererseits auf Grund der Modelldarstellung in grafischer Baumstruktur ungeeignet für eine Analyse, die möglichst nah an der in der ursprünglichen Literatur beschriebenen Methode liegen sollte. Auch das Kriterium, dass GOMSED seit 1998 nicht mehr weiterentwickelt wird, trug zum Entschluss gegen dieses Werkzeug bei. GLEAN hingegen scheint generell als ausgereiftestes Werkzeug für NGOMSL-Analysen sehr geeignet, erfordert allerdings auf Grund der eigenen Sprache GOMSL und der nicht intuitiven Bedienung beträchtliche zusätzliche Einarbeitungszeit, die im Rahmen dieser Arbeit nicht zur Verfügung stand.
Das NGOMSL-Modell wurde, wie im Konzept 3.2.3.1 beschrieben, durch eine top-down

und breadth-first Erweiterung erstellt. Das Modell durchlief mehrere Iterationen, während der u.a. die Konsistenz des Modells verbessert wurde, sowie die Abdeckung und die Korrektheit überprüft wurden. Parallel zur Erstellung des Modells fand die Formulierung der Aufgaben- und Variablenbeschreibung, sowie der Annahmen und Ermessensentscheidungen statt. Auch die Belegung der Variablen für die einzelnen Aufgabeninstanzen wurde hinzugefügt. Im Anschluss folgte die Berechnung der Bearbeitungs- und Lernzeiten nach der in (Kie96) angegebenen Formeln. Die Berechnung der Bearbeitungszeit geschah, indem die für die Aufgabeninstanzen definierten Variablenwerte in das NGOMSL-Modell eingesetzt wurden und das komplette Programm entsprechend den Werten einmal bis zum Ende ausgeführt wurde. Alle für den Programmdurchlauf notwendigen Methodenaufrufe wurden im Anschluss zur Berechnung der Bearbeitungszeit herangezogen. Die Lernzeit wurde anhand des kompletten NGOMSL-Modells berechnet, da ein Nutzer alle möglichen Methoden und Abzweigungen wissen muss, um unterschiedliche Aufgabeninstanzen durchführen zu können.

4.3 Durchführung der Analyse

Dieses Kapitel beschreibt die Durchführung der Analyse mit den zwei gewählten GOMS-Varianten KLM und NGOMSL. Alle erstellten Modelle, auf die sich die folgenden Aussagen beziehen, sind im zweiten und dritten Anhang zu finden. Dort finden sich außerdem eine Beschreibung der Annahmen, auf denen die Modelle beruhen und weitere wichtige Informationen zu den Modellen.

4.3.1 Analyse mit KLM

Grundsätzlich treten bei den KLM-Modellen zur Abwicklung des Drucks auf eine Schaltfläche drei Operatoren auf. Wie in Tabelle 4.1 zu sehen ist, wird davon ausgegangen, dass zu dieser Abwicklung zunächst ein mentaler „Think-“Operator, gefolgt von dem Lokalisieren der Schaltfläche, dargestellt mittels „Look At“, und der Hand- und Druckbewegung, bezeichnet durch „Move And Tap“, gehören. Dies beruht auf der Annahme, dass ein Nutzer, bevor er einen Druck auf eine Schaltfläche ausführt, zunächst über diese Handlung nachdenken muss. Außerdem entspricht dies den in Kapitel 3.2.1.1 vorgestellten Regeln zur Platzierung der mentalen Operatoren nach (CMN83) und wird vom Cogtool automatisch generiert, sobald ein Druck auf eine Schaltfläche simuliert wird. Die Zeit, die für die Dauer des allgemeinen „Think-“Operators angenommen wird, ist die in der Literatur (Kie01; CMN83) vorgeschlagene Durchschnittszeit für einen mentalen Operator. Hat der Nutzer über die bevorstehende Handlung nachgedacht, so lokalisiert er zunächst die

Schaltfläche, bevor er eine Handbewegung und im Anschluss eine Druckbewegung auf die Schaltfläche ausführt. Diese Operatorenreihenfolge zieht sich gehäuft durch alle erstellten KLM-Modelle, da die Drucke auf eine Schaltfläche ein Hauptbestandteil der Interaktion mit dem Fahrkartenautomaten sind.

Anzeige	**Aktion „Widget“**
Start	Think for 1.2 s
Start	Look At „Fahrkarten, Platzreservierungen“
Start	Move And Tap „Fahrkarten, Platzreservierungen“
Frame 2	Think for 1.2 s
Frame 2	Look At „Bremen Hbf“
Frame 2	Move And Tap „Bremen Hbf“

Tab. 4.1: Auzug aus dem KLM-Modell zu Aufgabeninstanz 1

Das Modell zur ersten Aufgabeninstanz unterscheidet sich in seinem Verlauf wesentlich zu den anderen KLM-Modellen. Es handelt sich dabei um eine Verbund-Verbindung, bei der der Fahrkartenautomat scheinbar ein stark verkürztes Abfrageprogramm vornimmt. Der Nutzer braucht lediglich Angaben zu Reiseziel und gewünschter Verbindung machen, bevor der Bezahlbildschirm geladen wird. Erst bei dem Bezahlbildschirm werden Angaben zu gebuchter Qualitätsklasse und Personenanzahl gemacht. Voreingestellt sind hier die Qualitätsklasse zwei und ein Einzelticket für den Nahverkehr. Möchte der Nutzer diese Daten ändern, so kann er über die entsprechenden Schaltflächen Rückschritte vornehmen. Davon wird hier aber nicht ausgegangen.
Für auffallende System-Antwortzeiten[2] bzw. Wartezeiten für den Nutzer, d.h. solche, die über einer Sekunde Dauer lagen, wurden die empirisch erhobenen Werte[3] verwendet und in die Modelle integriert. Dies geschah über den Einsatz des „Wait for-“ Operators. Die Stellen, an denen dieser Operator auftritt, werden im Folgenden erläutert. Auf Grund der Unterschiedlichkeit des ersten Modells zu den vier weiteren Modellen, wurden auch unterschiedliche System-Antwortzeiten gemessen, weshalb nachfolgend zwischen dem ersten und den weiteren Modellen unterschieden wird. Für auffallende Denkpausen der Nutzer wurden ebenfalls zusätzliche Zeiten integriert. Diese fließen durch den „Think for-“Operator in die Modelle mit ein. Das Modell zur dritten Aufgabeninstanz unterscheidet sich ebenso maßgeblich von allen anderen Modellen, da dieses den Ticketkauf eines Spezial-Angebotes abbildet. Dieses Modell benötigt keinerlei zusätzliche Warte- oder Denkzeiten.

[2]Der Begriff „System“ wird ab hier synonym zu dem Begriff „Fahrkartenautomat“ verwendet
[3]Sämtliche in dieser Arbeit empirisch ermittelten Werte wurden anhand der beschriebenen Videoaufzeichnungen erhoben, bei denen Probanden am Fahrkartenautomaten Beispielaufgaben durchführen.

4.3.1.1 Zeiten des ersten Modells

An dem Zeitpunkt, an dem der Fahrkartenautomat die verfügbaren Verbindungen ermittelt, muss der Nutzer 6,33 Sekunden auf das System warten.
Darauf folgt eine große Denkpause des Nutzers, die er benötigt, um eine Entscheidung für eine Verbindung zu treffen. Für diese Denkpause wurde eine Durchschnittszeit von 11,27 Sekunden ermittelt. Es wird davon ausgegangen, dass sowohl ungeübte als auch geübte Nutzer an dieser Stelle eine Denkpause brauchen, da die verfügbaren Verbindungen stets variieren und dadurch der Entscheidungsprozess unabhängig von der Fähigkeit des Nutzers, mit dem Fahrkartenautomaten umzugehen, ist.
Hat sich der Nutzer für eine Verbindung entschieden und die entsprechende Schaltfläche zum Auswählen der Verbindung gedrückt, so lädt das System bei dieser ersten Aufgabeninstanz sofort den Bezahlbildschirm. Dafür braucht es an dieser Stelle 1,33 Sekunden. Da bei diesem Modell Details zum gewünschten Ticket, wie oben erwähnt, nicht im Vorfeld vom Nutzer abgefragt werden, wird davon ausgegangen, dass der Nutzer an dieser Stelle die Ticketdetails verifiziert, bevor er bezahlt. Aus diesem Grund wurde hier ein zusätzlicher „Think for-“Operator eingefügt, der mit der Durchschnittszeit für mentale Aktionen von 1,2 Sekunden berechnet wird. Die Abbildung 4.1 veranschaulicht diesen Ablauf. In diesem und den beiden folgenden Ablaufdiagrammen stellen Rechtecke jeweils motorische Aktionen des Nutzers dar, während Ellipsen jeweils die aktuelle Bildschirmanzeige repräsentieren. Geschwärzte Rechtecke modellieren Abzweigungen, an denen je nach Aufgabeninstanz der weitere Verlauf bestimmt wird. Die einzelnen Pfeile zwischen Rechtecken und Ellipsen spezifizieren die Ablaufrichtung und geben ggf. Warte- oder Denkzeiten an. Wurden Teile des Ablaufs ausgelassen, so ist dies durch „[...]“ gekennzeichnet.

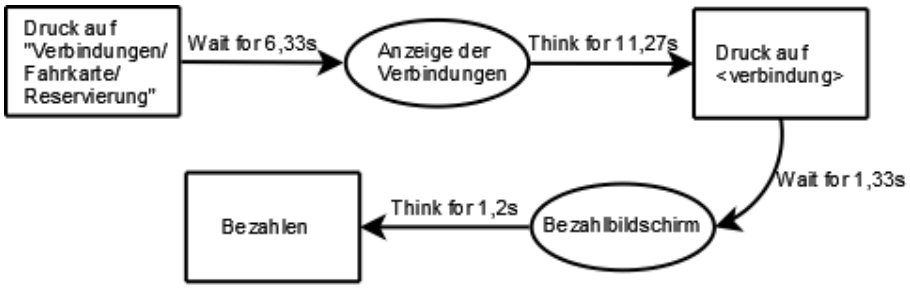

Abb. 4.1: Ablaufdiagramm des ersten Modells

4.3.1.2 Zeiten der Modelle zwei, vier und fünf

Die erste Wartezeit tritt im Verlauf der Datumsauswahl der Hinfahrt auf. Liegt das Reisedatum weder im aktuellen noch im nächsten Monat, so muss der Nutzer über eine Schaltfläche spätere Auswahlmöglichkeiten aufrufen. Nachdem diese Schaltfläche gedrückt

wurde, benötigt das System 1,16 Sekunden, bevor die nächste Bildschirmanzeige folgt und der Nutzer eine neue Aktion an dem Fahrkartenautomaten ausführen kann. Für die Rückfahrt passt sich der Automat an das gewählte Hinfahrtdatum an, sodass das Drücken der zuletzt genannten Schaltfläche nicht nötig ist und die Wartezeit entfällt.
Die Abbildung 4.2 zeigt den nachfolgend beschriebenen Ablauf anhand eines Ablaufdiagramms.

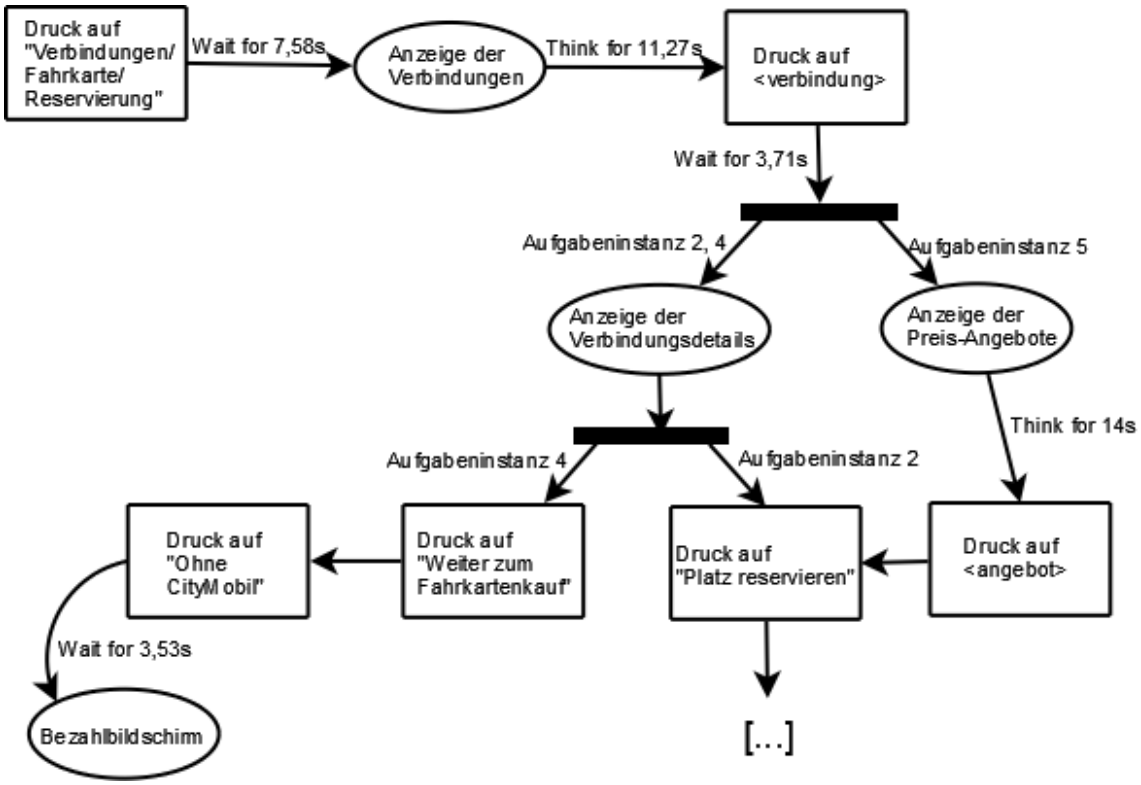

Abb. 4.2: Erstes Ablaufdiagramm der Modelle zwei, vier und fünf

Die nächste längere Rechenzeit benötigt der Fahrkartenautomat für die Ermittlung der verfügbaren Verbindungen. Hierfür wird eine Wartezeit von 7,58 Sekunden in die Modelle eingefügt. Im Anschluss wird, wie im ersten Modell, angenommen, dass ein Entscheidungsprozess des Nutzers folgt, bei dem dieser sich eine Verbindung auswählt. Dafür werden 11,27 Sekunden berechnet.
Hat sich der Nutzer für eine Verbindung entschieden und die entsprechende Schaltfläche zum Auswählen der Verbindung gedrückt, so braucht das System weitere 3,71 Sekunden, um die Verbindungsdetails zu laden und diese auf der nächsten Bildschirmanzeige anzuzeigen.
Im Falle des fünften Modells erscheint hier eine zusätzliche Bildschirmanzeige, die dem Nutzer unterschiedliche Preis-Angebote unterbreitet. Es wird angenommen, dass diese Anzeige für den Nutzer unerwartet erscheint und dieser sich zunächst mit den dargebotenen

Angeboten befassen muss, bevor er eine Entscheidung für ein bestimmtes Angebot trifft. Die dafür ermittelte Denkpause beläuft sich auf 14 Sekunden.
Bei dem vierten Modell wird dem Nutzer unmittelbar vor dem Bezahlvorgang das Zusatzangebot „CityMobil“ vorgeschlagen. Nachdem der Nutzer dieses Angebot abgelehnt hat, rechnet der Fahrkartenautomat 3,53 Sekunden, bevor der Bezahlbildschirm erscheint.

Die nachfolgend beschriebenen Abläufe sind in Abbildung 4.3 veranschaulicht.
Eine weitere zusätzlich eingefügte Wartezeit wird durch den „Wait for-“Operator nach der Wahl der Reservierung repräsentiert. Um die Reservierung zu buchen, benötigt das System 3,13 Sekunden. Falls die Reservierung nicht wie gewünscht gebucht werden konnte, sondern nur mit bestimmten Abweichungen, so wird der Nutzer darauf hingewiesen und muss sich entscheiden, ob er die Abweichungen akzeptiert oder ob er stattdessen die Reservierung rückgängig machen möchte. Für diesen Denkprozess wurde ein „Think of-“Operator von 6,4 Sekunden eingefügt. Dieser Operator wird sowohl bei der Reservierung der Rückfahrt als auch bei der der Hinfahrt verwendet. Bevor die Reservierung der Rückfahrt durchgeführt wird, muss der Nutzer dies bestätigen und 1,82 Sekunden auf das System warten. Dies wird mittels eines Wait for-“Operators modelliert.
Im Fall des zweiten Modells werden dem Nutzer zwei andere Zusatzangebote offeriert, nach deren Ablehnung eine einmalige Wartezeit von 2,13 Sekunden nötig ist, bis der Bezahlbildschirm angezeigt wird. Diese Zeit entspricht auch der letzten Wartezeit des fünften Modells, in der das System den Bezahlbildschirm lädt.

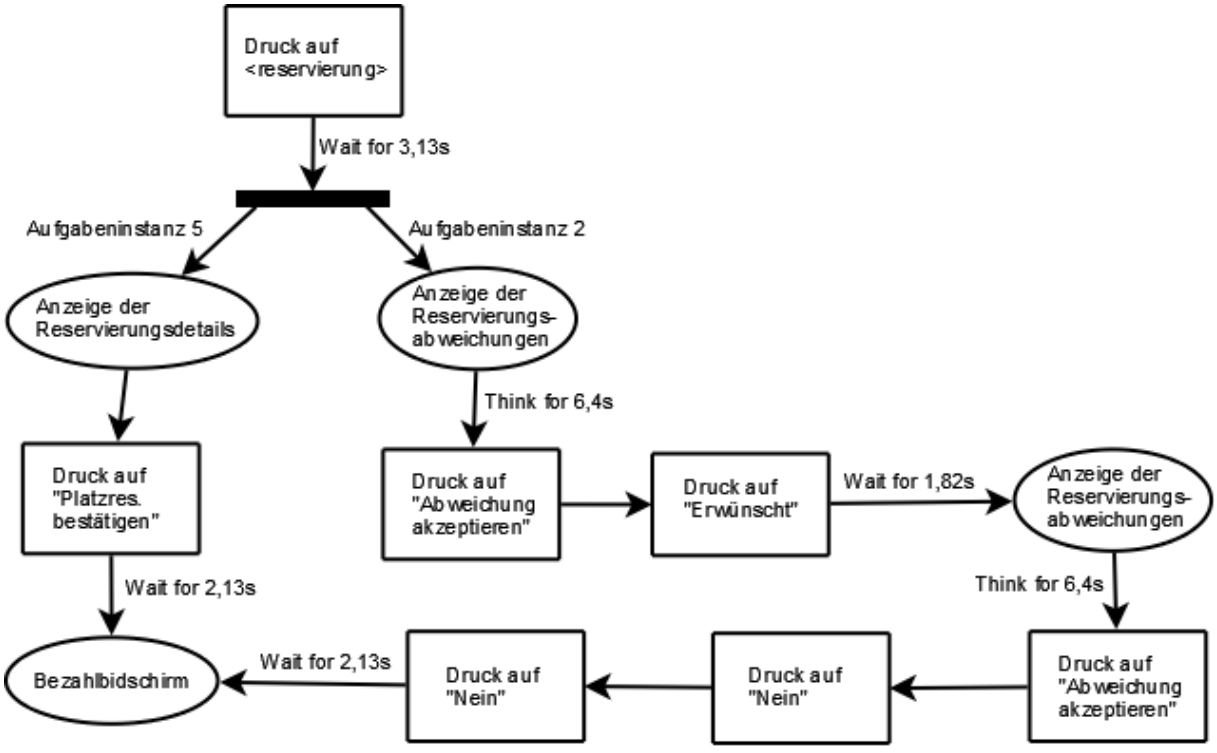

Abb. 4.3: Zweites Ablaufdiagramm der Modelle zwei, vier und fünf

Die restlichen Operatoren ergeben sich bedingt durch die erforderlichen Tastendrücke. Da bei den KLM-Modellen keine weiteren Designentscheidungen zu treffen waren, konnten die Modelle mit Hilfe oben beschriebener Informationen erstellt werden. Nach der Erstellung der Modelle wurden die Bearbeitungszeiten der einzelnen Aufgabeninstanzen berechnet.

Dafür wurden folgende Operatorzeiten verwendet[4]:

Look at $< name >$	1,2 Sekunden
Move and Tap $< object >$	1,3 Sekunden
Wait for $< duration >$ s	jeweils empirisch ermittelt
Think for $< duration >$ s	1,2 Sekunden vor einem „Move and Tap“, sonst jeweils empirisch ermittelt

Zur Berechnung der Gesamt-Bearbeitungszeit eines Modells wurden alle einzelnen Operatorzeiten addiert. Die Ergebnisse dieser Berechnungen sind im zweiten Anhang, jeweils im Anschluss an das entsprechende KLM-Modell, zu finden.

4.3.2 Analyse mit NGOMSL

4.3.2.1 Operatorenwahl und Berechnungsformeln

Die Operatorenreihenfolge, die für den Druck auf eine Schaltfläche erforderlich ist, verläuft hier ähnlich wie bei den KLM-Modellen. In diesem NGOMSL-Modell wird dieser Druck mittels einer eigenen Methode repräsentiert. Auch hier wird zunächst davon ausgegangen, dass die Schaltfläche lokalisiert wird, bevor auf die Schaltfläche gezeigt und gedrückt wird. Es folgt ein Ausschnitt aus dem NGOMSL-Modell, in dem die Methode „drücke button“ zu sehen ist, die diesen Schaltflächendruck modelliert.

Method for goal: drücke button $< buttonName >$.
Step 1. Locate object $< buttonName >$ on screen.
Step 2. Point to $< buttonName >$.
Step 3. Tap on $< buttonName >$.
Step 4. Return with goal accomplished.

Anders als bei den KLM-Modellen wurde hier auf einen allgemeinen mentalen Operator vor dem Druck auf eine Schaltfläche verzichtet. Dies ist damit zu begründen, dass in dem NGOMSL-Modell vor der Durchführung eines Schaltflächendrucks i.d.R. bereits Denkprozesse stattfinden, die mit dem Entschluss enden, die entsprechende Schaltfläche zu

[4]Diese Werte stimmen mit den Abschätzungen von Card, Moran und Newell (CMN83) überein.

drücken. Diese Entscheidungsprozesse werden so nicht in den KLM-Modellen dargestellt, da für diese nur primitive Operatoren verwendet werden und Entscheidungsprozesse, die z.B. mittels eines „Decide“-Operators in NGOMSL modelliert werden, dort auf Grund der Spezialisierung eines Modells auf eine bestimmte Aufgabe unnötig sind. Um diese Denkzeit dort aber dennoch einfließen zu lassen, wurde bei den KLM-Modellen vereinfacht ein „Think for“-Operator genutzt. Wird die Methode „drücke button“ aufgerufen, ohne dass unmittelbar zuvor Denkvorgänge modelliert wurden, so liegt dies an der Annahme, dass es sich bei diesen Vorgängen um offensichtliche, einfache Tastendrücke handelt, die keine weitere vorherige Bedenkzeit benötigen. Dies ist beispielsweise in der Methode „wähle Wochenendticket“ der Fall, in der zweimal nacheinander die Schaltfläche zur Auswahl des „Schönes-Wochenende-Ticket“ gedrückt wird. Des Weiteren orientiert sich die Operatorenreihenfolge dieses NGOMSL-Modells, sofern möglich, an den Leitlinien und den Beispielen aus (Kie96).

Die im vorherigen Kapitel 4.3.1 beschriebenen Werte für die zusätzlichen Warte- und Denkzeiten wurden auch für das NGOMSL-Modell verwendet. Das Auftreten der damit verbundenen Zusatzoperatoren ist bezogen auf den Ablauf der einzelnen Aufgabeninstanzen ebenfalls analog. Allerdings wurden für das NGOMSL-Modell weitere, vom Analysten definierte, Operatoren verwendet. Außerdem war zu beachten, dass die empirisch ermittelten Werte teilweise mehr als einem aufgeführten Operator zugewiesen werden konnten. Dies hatte zur Folge, dass die jeweiligen Operatorzeiten entsprechend angeglichen werden mussten, um doppelt berechnete Zeiten zu vermeiden. An folgendem Beispiel wird dies deutlich: Die Methode „wähle Verbindung“ beginnt mit dem mentalen Operator „Read Verbindungen value from the screen“, der standardmäßig mit 1,2 Sekunden Ausführungszeit berechnet wird. Darauf folgt ein weiterer mentaler Operator, der einen Entscheidungsprozess modelliert und dessen Ausführungszeit empirisch mit 11,27 Sekunden ermittelt wurde. Tatsächlich ist in dieser empirisch ermittelten Zeit bereits die des ersten Operators enthalten, da bei den empirischen Messungen die reine Lesezeit nicht von dem Entscheidungsprozess unterschieden werden konnte. Deshalb fließt der zweite mentale Operator in dieser Methode lediglich mit 10,07 Sekunden in die Berechnung ein. Es folgt eine Auflistung aller in diesem NGOMSL-Modell verwendeten Operatoren und ihren Ausführungsdauern.

Zur Berechnung der Bearbeitungszeit einiger ausgewählter Aufgabeninstanzen wurden folgende Operatorzeiten verwendet[5]:

Locate object $< name >$	1,2 Sekunden
Point to $< object >$	1,1 Sekunden
Tap on $< object >$	0,2 Sekunden

[5]Diese Zeiten stimmen mit den Werten des Keystroke-Level-Model von Card, Moran und Newell (CMN83) überein.

Read < *name* > value from the screen	1,2 Sekunden
Verify that < *description* >	1,2 Sekunden

Mentale Operatoren, die vom Analysten definiert wurden, wurden, sofern keine genauen Werte vorhanden waren, mit 1,2 Sekunden berechnet. Dies entspricht der Empfehlung von Kieras in (Kie96).

Empirisch ermittelt wurden die Ausführungszeiten der folgenden Operatoren. Diese variieren je nach Methode und sind deshalb direkt in das Modell eingefügt. Es gibt hierfür demnach keine allgemeingültigen Zeiten.

Make up your mind about < *decision description* >
Think about < *name* >
Wait for system for < *duration* > Sekunden

Die folgenden Operatoren wurden in diesen Modellen nur auf einer hohen Ebene eingesetzt und nicht genauer spezifiziert, da sie für die hier beschriebenen Aufgabeninstanzen nicht relevant sind und nur der Vollständigkeit halber aufgeführt wurden.

Handle not-accepted variation
Choose another date
Pay tickets
Insert < *object* >

Die Beschreibung der einzelnen Operatoren ist Teil des NGOMSL-Modells und deshalb, im Anschluss an das Modell, im dritten Anhang zu finden. Die Ergebnisse der berechneten Bearbeitungs- und Lernzeiten befinden sich ebenfalls an dieser Stelle.

Zur Berechnung der Gesamt-Bearbeitungszeit einer Aufgabeninstanz wurde folgende Formel verwendet (Kie96):

$$\begin{aligned} \textit{Execution Time} = {} & \textit{NGOMSL Statement Time} + \textit{Primitive External Operator Time} \\ & + \textit{Analyst-defined Mental Operator Time} + \textit{Waiting Time} \end{aligned}$$

Dabei bezeichnen die einzelnen Elemente der Formel die in Tabelle 4.2 aufgeführten Berechnungen.

Die einzelnen Warte-Zeiten, die zur Berechnung der Waiting Time herangezogen wurden, sind hier direkt in den einzelnen Methoden integriert, sodass diese Zeiten demnach bereits in der NGOMSL Statement Time enthalten sind und hier nicht mehr zusätzlich zur Berechnung der Gesamt-Bearbeitungszeit herangezogen werden müssen.

NGOMSL Statement Time	*= Anzahl ausgeführter Statements * 0,1 Sekunden*
Primitive External Operator Time	*= Summe der Ausführungszeiten aller primitiven externen Operatoren*
Analyst-defined Mental Operator Time	*= Summe der Ausführungszeiten aller mentalen Operatoren, die vom Analysten definiert wurden*
Waiting Time	*= Gesamtzeit, die der Nutzer auf eine Antwort des Systems warten muss.*

Tab. 4.2: Nötige Einzelberechnungen für die Berechnung der Gesamt-Bearbeitungszeit

Zur Berechnung der Lernzeit des NGOMSL-Modells wurde folgende Formel verwendet (Kie96):

Pure Learning Time = Pure Method Learning Time + LTM Item Learning Time

Dabei bezeichnen die einzelnen Elemente der Formel folgende Berechnungen.

Pure Method Learning Time	*= Learning Time Parameter * Anzahl zu erlernender NGOMSL Statements*
LTM Item Learning Time	*= 6 Sekunden * Anzahl zu speichernder LTM chunks*
Learning Time Parameter	*= 30 Sekunden nach Bovair, Kieras und Polson, 17 Sekunden nach Gong (typischer Wert)*

Die sechs Sekunden bei der LTM Item Learning Time drücken die Zeit aus, die zum Speichern eines chunks im LTM benötigt wird. Es ist der in der Literatur empfohlene und von Kieras in (Kie96) zitierte Wert, der auf Gong zurückgeht. Als LTM chunks wurden hier die einzelnen Variablenabfragen über den Operator „Retrieve from LTM“ bezeichnet. Denn dieser Operator repräsentiert die Abfrage einer Information aus dem LTM und die anschließende Zwischenspeicherung dieser, die während der Ausführung der Methoden erforderlich ist.
Der Wert des „Learning Time Parameter“ variiert je nach Spezifikationen der Lernsituation. Kieras unterscheidet zwischen zwei Ansätzen. Laut ihm nutzen Bovair, Kieras und Polson zur Berechnung der Lernzeit eine sehr strenge Trainingssituation, während Gong von einer realistischeren Trainingssituation ausgeht. Bei der strengen Trainingssituation werden dem Lerner die Methoden explizit präsentiert. Problemlöseverhalten ist dadurch nicht erforderlich. Der Lerner muss alle Methoden einzeln nacheinander solange üben, bis er diese perfekt beherrscht. Nebenbei erhält er dazu Feedback. Die Gesamt-Trainingszeit ist hier

die Summe der Zeiten, die für die einzelnen Trainings jeder Methode benötigt wurden. Bei der typischen Trainingssituation nach Gong durchläuft der Lerner eine Demonstration, in der ihm eine Beispielaufgabe visuell und verbal erklärt wird. Auch hier ist demnach kein Problemlöseverhalten nötig. Im Anschluss muss er selbstständig und ohne Korrektur oder Feedback mehrere Trainingsaufgaben durchführen. Die Gesamt-Trainingszeit berechnet sich dann aus der Summe der für die Demonstration und die Trainingsaufgaben benötigten Zeiten.
Für den Fahrkartenautomaten trifft auf Grund seiner Auslegung für Gelegenheitnutzung keine der vorgeschlagenen Lernsituationen zu. Um die Berechnung der Lernzeit dennoch durchführen zu können, wurde hier der typische Wert nach Gong verwendet, da dieser auf Grund der an einem Fahrkartenautomaten durchgeführten Aufgaben eher passend erscheint. Bei solchen Aufgaben werden Fahrkartenkäufe als Ganzes betrachtet. Das wiederholte Üben einer Teilaufgabe, wie z.B. des Reservierungsvorganges oder der Datumsauswahl erscheint eher ungeeignet, da diese Teilaufgaben je nach Gesamtaufgabe variieren können und deshalb nur im Gesamtkontext sinnvoll zu bearbeiten sind.
Des Weiteren ist zu beachten, dass gleiche oder ähnliche Statements in derselben Methode oder ähnlichen Methoden oder Selektionsregeln lediglich einmal in die Berechnung der Lernzeit einfließen. Laut Kieras kann ein Nutzer, nachdem er das erste von zwei ähnlichen Statements gelernt hat, das zweite so leicht erlernen, dass keine Lernzeit für dieses zweite Statement erforderlich ist. Von der Anzahl aller zu erlernenden Statements werden folglich jene Statements subtrahiert, die bereits gelernt wurden. Wann zwei Statements als ähnlich klassifiziert werden, beschreibt er in (Kie96, Seite 42).

4.3.2.2 Designentscheidungen

Die Erstellung eines NGOMSL-Modells erfordert auf Grund seiner Form im Vorfeld einige Designentscheidungen. Dazu gehört vor allem die Konstruktion einer Zielhierachie, deren Aufbau möglichst nah an den Vorstellungen der potentiellen Nutzer des Gerätes liegen sollte. Die Struktur der Zielhierarchie ist stark von dem Analysten abhängig, der sie erstellt. Denn für die Erstellung gibt es zwar Richtlinien (Kie96), aber keine allgemeingültigen Regeln. Dies ist einleuchtend, da jedes NGOMSL-Modell individuell gestaltet wird. Denn es orientiert sich an speziellen Kriterien wie z.B. den Vorstellungen der potentiellen Nutzer oder weiteren Annahmen über Nutzer und Gerät, die vom Analysten getroffen wurden.
Zur Konstruktion der Zielhierarchie wurde, wie in Kapitel 3.2.3.1 beschrieben, eine top-down und breadth-first Erweiterung vorgenommen. Ausgehend von dem obersten Ziel „kaufe Fahrkarten“ wurde eine Baumstruktur aufgebaut, die die Methoden in ihren einzelnen Abstraktionsebenen immer weiter herunterbricht, bis schließlich die Ebene der primitiven Operatoren erreicht wird. Unterziele entstanden häufig durch zu lange Methoden.

Es wurde versucht, die Anzahl der Steps in einer Methode unter acht zu halten. Dadurch sollte die Übersichtlichkeit erhalten bleiben. Dies hatte zur Folge, dass einige Methoden in mehrere Methoden aufgeteilt werden mussten, wodurch gleichzeitig neue Unterziele entstanden.
Da das NGOMSL-Modell allgemein formuliert sein sollte, war die Verwendung von Selektionsregeln und „Decide"-Operatoren erforderlich. Diese wurden an Stellen eingesetzt, an denen verschiedene Abzweigungen gewählt werden können. Selektionsregeln steuern den Fluss außerhalb von Methoden, d.h. sie entscheiden, welche Methode die passende ist und führen diese dann aus. „Decide-"Operatoren dagegen werden eingesetzt, um den Kontrollfluss innerhalb einer Methode zu regeln. Das bedeutet, dass diese die für die jeweilige Instanz passende Operatorenabfolge innerhalb einer Methode steuern. In den Then-Sätzen von Selektionsregeln werden ausschließlich „Accomplish goal"-Operatoren eingesetzt, während in den Then-Sätzen eines „Decide-"Operators jegliche Operatoren verwendet werden dürfen. Allerdings ist die Unterscheidung trotz dieser Regeln nicht immer eindeutig, da mit mehreren If-Then-Aussagen eine Selektionsregel nachgebildet werden kann (Kie96). Bei diesem NGOMSL-Modell treten „Decide-"Operatoren, über diese Richtlinien hinaus, auch dort auf, wo lediglich zwischen dem Drücken unterschiedlicher Schaltflächen ausgewählt wird. Wie bereits erwähnt, repräsentiert die Methode „drücke button" einen vollständigen Druck auf eine Schaltfläche auf der untersten Abstraktionsebene, dem Keystroke-Level. Gerade aus diesem Grund wurde hier von den Richtlinien abgewichen und statt einer Selektionsregel ein „Decide-"Operator verwendet. Denn die Methode repräsentiert letztlich die Einzelschritte eines Schaltflächendrucks auf der einfachen Operatorebene und ist deshalb, nach Ansicht des Analysten, eher als einfacher Operator, als als differenzierte Methode zu betrachten. Der folgende Auszug aus dem NGOMSL-Modell zeigt die Methode „initialisiere Fahrkartenkauf", die den „Decide-"Operator verstärkt in dieser Art nutzt.

Method for goal: initialisiere Fahrkartenkauf.

Step 1. Decide: If $< ticketType >$ is „normal" and button „Fahrkarten Platzreservierungen" is visible, Then accomplish goal: drücke button „Fahrkarten Platzreservierungen".
If $< ticketType >$ is „normal" and button „Fahrkartenkauf (Schritt für Schritt)" is visible, Then accomplish goal: drücke button „Fahrkartenkauf (Schritt für Schritt)".
If $< ticketType >$ is „wochenend", Then accomplish goal: drücke button „Spezial-Angebote".
If $< ticketType >$ is „verbund", Then accomplish goal: drücke button „Fahrkarten Platzreservierungen".

Step 2. Forget $< ticketType >$ and Return with goal accomplished.

Wichtig bei der Formulierung dieser Selektionsregeln und „Decide-“Sätzen ist außerdem, dass diese jeweils eindeutig ist, d.h. es darf an solch einer Abzweigung immer nur eine If-Bedingung zutreffen, denn sonst gäbe es einen Konflikt bei der Auswahl und der Programmfluss wäre unterbrochen.
Um die allgemeine Form des NGOMSL-Modells zu wahren, wurden Variablen eingeführt, die für die verschiedenen Aufgabeninstanzen entsprechend gesetzt werden. Zwecks eindeutiger Identifizierung erscheinen diese in dem Modell kursiv und sind von spitzen Klammern umschlossen. Eine komplette Auflistung aller verwendeten Variablen, deren Bedeutung und Definitionsbereich, befindet sich im dritten Anhang, im Anschluss an das NGOMSL-Modell.

Zur Konstruktion des Modells waren zahlreiche Iterationen nötig, bei denen neben der Korrektheit der Zielhierarchie und der einzelnen Methoden auch die Konsistenz dieser immer wieder überprüft und ggf. verbessert wurde. Ein Anzeichen für eine hohe Konsistenz ist die häufige Verwendung von allgemeinen anstatt speziellen Methoden. In gewissem Maße kann die Konsistenz demnach vom Analysten optimiert werden. Dies wurde hier versucht. Die Richtlinien von Kieras (Kie96, Seite 42) halfen, zu erkennen, inwieweit sich zwei Methoden ähnelten und ob sie ggf. zu einer allgemeinen Methode zusammengefasst werden konnten. Um das überprüfen zu können, wurden im Vorfeld die Methodennamen nach einem einheitlichen Schema aufgebaut. Es wurden aussagekräftige und präzise Bezeichnungen verwendet, die einheitlich in Imperativform formuliert wurden. Dies half dabei, ähnliche Methoden schneller zu erkennen.

4.3.3 Analyse mit dem CogTool

Wie bereits beschrieben, wurden die KLM-Modelle mit Hilfe des CogTools erstellt. Des Weiteren wurden die Bearbeitungszeiten der einzelnen Aufgaben durch das CogTool berechnet. Allerdings lieferte das Werkzeug Werte, die sich stark von den selbst errechneten Werten, die auf Grundlage der KLM-Richtlinien (Kie01) ermittelt wurden, unterscheiden. Dies ist folgendermaßen zu begründen.
Das CogTool verwendet andere Operatorzeiten. Es berechnet für jeden einzelnen Operator individuelle Ausführungszeiten anhand der zu Grunde liegenden kognitiven Architektur ACT-R und dem bereits erwähnten Fitt's Law. Diese fallen im Allgemeinen kürzer aus. Das Fitt's Law wurde bei den Berechnungen in dieser Arbeit aus Zeitgründen nicht angewendet. Außerdem liegen KLM-Modelle generell keiner kognitiven Architektur zu Grunde. So ist das CogTool dadurch eher nicht für „reine“ KLM-Analysen geeignet, sondern liefert Berechnungen, die auf der Erweiterung um die kognitive Architektur basieren.
Da in dieser Arbeit die quantitativen Aussagen der KLM-Modelle auf die ursprüngliche

Art, d.h. allein auf Basis des Keystroke-Level-Models ermittelt werden sollten, erschienen die Werte des CogTools ungeeignet und die Arbeit wurde auf Grundlage der per Hand berechneten Bearbeitungszeiten weitergeführt.

4.3.4 Mögliche Fehlinterpretationen

Alle empirisch ermittelten Werte können durch Messfehler oder Ungenauigkeiten in der Messung verfälscht sein. Außerdem ist bei den empirisch ermittelten Ausführungszeiten der mentalen Operatoren zu berücksichtigen, dass diese jeweils der Durchschnittswert von fünf Probanden sind, die ausnahmslos ungeübt im Umgang mit dem Fahrkartenautomat waren. Es ist also zu beachten, dass die Höhe der Messwerte ggf. durch die Erhebung weiterer Messdaten mit geübten Nutzern revidiert werden kann.
Weiterhin war anhand der empirisch gemessenen Werte festzustellen, dass das System generell nach jedem einzelnen Tastendruck eine Verzögerungszeit zwischen 0,5 und 1,3 Sekunden verursachte. Diese Zeiten variierten stark zwischen den einzelnen Schaltflächen, sodass die Verwendung eines Durchschnittwerts pro Druck auf eine Schaltfläche vermutlich zu deutlich verfälschten Ergebnissen geführt hätte[6]. Die hier erstellten Modelle beinhalten diese Wartezeiten folglich nicht. Dies ist bei der Interpretation der Zeiten zu beachten.

Insgesamt ist bei diesen Analysen zu beachten, dass sie von einem unerfahrenen Analysten durchgeführt wurden. Da sich Designentscheidungen des Analysten, wie beispielsweise der Aufbau der Zielhierarchie oder die Verwendung mentaler Operatoren, auf die berechneten Bearbeitungs- und Lernzeiten auswirken, könnte die Analyse eines Experten evtl. qualifiziertere Ergebnisse liefern. Diese Vermutung bestätigte John[7]:

> *„Also, you have to place the mental operators carefully, according to Card. Moran and Newell's rules. Kieras's rules approximate theirs. In both cases, I have done research that shows that people doing their first few models are not comfortable using those rules and insert fartoo many mental operators compared to Card, Moran, Newell & people like me and David Kieras. This was even more true if people followed Dave's rules instead of the original Card, Moran and Newell rules.“*

[6] Die Einführung eines individuellen Werts pro Schaltfläche war hier aus Zeitgründen nicht möglich und hätte die Komplexität und Unübersichtlichkeit aller erstellten Modelle unnötig erhöht.
[7] Briefverkehr mit Bonnie E. John am 12.9.2008

5 Optimierungspotentiale des Fahrkartenautomaten

In diesem Kapitel werden die Ergebnisse der Analysen hinsichtlich des Fahrkartenautomaten bewertet. Dabei werden beispielhaft Optimierungspotentiale des Fahrkartenautomaten aufgezeigt. Die Aussagen des ersten Abschnitts basieren direkt auf den erstellten KLM- und NGOMSL-Modellen und deren Berechnungen. Im zweiten Abschnitt werden Optimierungsmöglichkeiten beschrieben, die darüber hinaus im Rahmen dieser Arbeit, durch die empirischen Untersuchungen, aufgedeckt wurden.

5.1 Aussagen der Modelle

Da im Folgenden genauer auf die berechneten Bearbeitungszeiten der einzelnen Aufgaben eingegangen wird, sind diese hier zunächst tabellarisch aufgeführt.

Aufgabeninstanz	KLM	NGOMSL
1	30,30s	42,16s
2	162,10s	173,10s
3	22,20s	22,20s
4	103,75s	110,95s
5	116,15s	134,47s

Die Ausdehnung des Wertebereiches zeigt, dass die Bearbeitungszeiten zwischen den unterschiedlichen Aufgaben stark variieren. Außerdem sind die Zeiten jeweils dort geringer, wo die Aufgaben einen geringeren Umfang haben. Der Umfang der Aufgaben ist bei dem NGOMSL-Modell an der Anzahl der gesetzten Variablen, aber auch an den Werten der Variablen abzulesen. Je mehr Variablen gesetzt sind, desto spezieller ist die Aufgabe und folglich deren Umfang. Je mehr Variablen beispielsweise auf „kein(e)“, „nein“ oder „0“ gesetzt sind, desto geringer ist der Umfang dieser Aufgabe, da weniger Folge-Eingaben erforderlich sind. Zunächst ist also festzuhalten, dass die Bearbeitungszeit abhängig von dem Umfang der Aufgaben ist. Dieser Zusammenhang erscheint akzeptabel.
Die einzelnen Bearbeitungszeiten liegen zwischen 22,20 und 173,10 Sekunden. Diese Zeiten

befinden sich in einem annehmbaren Rahmen bzgl. der Aufgaben, für die das Gerät bestimmt ist. Knapp drei Minuten Bearbeitungsdauer sollten für einen komplexen Ticketkauf vom Nutzer angenommen werden. Demnach lassen die reinen Gesamt-Bearbeitungszeiten nicht auf einen akuten Optimierungsbedarf schließen. Dies schließt allerdings nicht aus, dass für Teilaufgaben, im Verhältnis zur Gesamtaufgabe, zu viel Zeit benötigt wird oder zuviele Einzelschritte erforderlich sind.

Aussagekräftiger, als die Gesamt-Bearbeitungszeiten der einzelnen Aufgaben, sind die Ausführungszeiten bestimmter Operatoren und Operatorsequenzen. Denn Sequenzen, in denen sich der Nutzer besonders lange aufhält, können auf konkrete Schwachstellen des Systems hinweisen. Deshalb lohnt es sich besonders, an diesen Stellen zu optimieren, da sich dies dadurch voraussichtlich am stärksten auf die Gesamt-Bearbeitungszeit auswirken wird.
Besonders auffällig sind die Operatorzeiten des „Wait for“-Operators. Außerdem markant sind einige Zeiten der mentalen Operatoren „Think for“ bzw. „Make up your mind about“ und „Think about“. Da diese Zeiten empirisch ermittelt wurden, werden sie im nächsten Abschnitt behandelt.
Eine Operatorsequenz, in der sich der Nutzer lange aufhält, ist die Methode „spezifiziere Bahncards“. Diese Methode modelliert die Auswahl der vorhandenen Bahncards. Sie besteht aus 12 Statements und ist damit die Methode mit der höchsten NGOMSL Statement Time in diesem NGOMSL-Modell. Darüber hinaus besteht sie aus fünf erforderlichen Tastendrücken. Hier sticht die Sequenz der Tastendrücke hervor. Es wird zwischen vier verschiedenen Bahncardtypen unterschieden. Um die vorhandenen Bahncards auszuwählen, muss über jeden möglichen Bahncardtyp eine Aussage getroffen werden. Entweder der jeweilige Typ wird abgelehnt, oder die Anzahl Karten des Typs wird eingegeben. Es ist jedoch nicht möglich, die einzelnen Auswahlen zu überspringen. Der folgende Ausschnitt aus dem NGOMSL-Modell zeigt diese Methode.

Method for goal: spezifiziere Bahncards.

Step 1. Accomplish goal: drücke button $< anzahlBahncards >$.

Step 2. Retrieve from LTM the value for $< typBahncards >$, and recall it from WM.

Step 3. Decide: If $< typBahncards >$ is 25_2, Then accomplish goal: drücke button $< anzahlBahncards >$.
Else accomplish goal: drücke button „Keine“.

Step 4. Decide: If $< typBahncards >$ is 50_2, Then accomplish goal: drücke button $< anzahlBahncards >$.
Else accomplish goal: drücke button „Keine“.

Step 5. Decide: If $< typBahncards >$ is 25_1, Then accomplish goal: drücke button $< anzahlBahncards >$.

Else accomplish goal: drücke button „Keine“.

Step 6. Decide: If < *typBahncards* > is 50_1, Then accomplish goal: drücke button < *anzahlBahncards* >.
Else accomplish goal: drücke button „Keine“.

Step 7. Forget < *typBahncards* > and Return with goal accomplished.

Eine Optimierung, bei der die Anzahl Statements und die Anzahl erforderlicher Tastendrücke reduziert werden, erscheint hier sinnvoll. Diese könnte beispielsweise darin bestehen, dass, anstelle der vier nacheinander angezeigten Auswahlmöglichkeiten mit jeweils mehreren Schaltflächen, ein Auswahlmenü, z.B. in Form einer „Aufklappliste“, verwendet wird. So würden dem Nutzer von vornherein alle möglichen Auswahlmöglichkeiten präsentiert. Zudem genügte ein Tastendruck zur Auswahl des Bahncardtyps. Zur Auswahl der Anzahl Bahncards dieses Typs wäre jeweils eine Schaltfläche pro Anzahl denkbar, wie sie derzeit verwendet werden und auf Abbildung 5.1 zu sehen sind, die diesen kompletten Auswahlvorgang zeigt. Der Tastendruck im ersten Step der NGOMSL-Methode würde komplett entfallen, da die Anzahl der Bahncards am Ende der Methode angegeben würde. Die Anzahl der erforderlichen Tastendrücke reduzierte sich dadurch von fünf auf zwei. Die Anzahl der Statements dieser Methode, exklusive der Statements der Submethoden, könnte z.B. von zwölf auf acht verringert werden. Die Bearbeitungszeit dieser Methode, inkl. der Statement Time und der Submethoden, würde, laut Berechnung mittels des NGOMSL-Modells, dadurch von 16,2 Sekunden auf 6,8 Sekunden verkürzt. Die manipulierte NGOMSL-Methode befinden sich am Ende des dritten Anhangs. Dieser Optimierungsbedarf zeigt eine Schwäche des Systems bzgl. seiner Aufgabenangemessenheit. Denn durch die Erfordernis unnötig vieler Tastendrücke wird die Erledigung der Aufgabe uneffizient.

Mentale Operatoren gehören i.d.R. zu den Operatoren, die hohe Ausführungszeiten benötigen. Kann man diese teilweise eliminieren, so würde sich dies bereits merklich auf die berechnete Gesamt-Bearbeitungszeit auswirken. Betrachtet man sowohl die KLM-Modelle, als auch das NGOMSL-Modell, so sind mehrere dieser mentalen Operatoren zu finden. Die Denkpausen, die einem Tastendruck unmittelbar vorausgehen, sind höchstens dann zu verhindern, wenn auch der Tastendruck verhindert werden kann, da es sich bei diesen Pausen um Vorbereitungen auf einen Tastendruck handelt. Anders ist dies bei zusätzlichen Denkpausen, die beispielsweise im NGOMSL-Modell durch „Read value from the screen“-Operatoren dargestellt werden. Sie repräsentieren das Lesen und Aufnehmen eines Textes und treten dort auf, wo dem Nutzer besondere, neue Informationen in Textform präsentiert werden. Dies trifft auf fünf verschiedene Methoden zu, die jeweils mit solch einem Operator beginnen. Bei den Methoden „wähle Verbindung“, „verifiziere Ticketdetails“, „wähle

Abb. 5.1: Schemenhafte Darstellung des Bahncard-Dialogs

Preis-Angebot“ und „behandle Reserv.-Abweichungen“ sind diese Operatoren vermutlich nicht zu vermeiden, da der Nutzer dort situationsbedingte, also i.d.R. neue, Informationen, wie z.B. die verfügbaren Zugverbindungen oder Preis-Angebote erhält, die ein unvermeidbarer Bestandteil eines Ticketkaufs sind. Anders verhält es sich bei der Methode „wähle Zusatzangebote“. Diese Methode wird eingesetzt, wenn zu der gewünschten Verbindung zusätzliche Angebote verfügbar sind. Dabei kann es sich um eine Versicherung, um den Erwerb von Bonuspunkten oder um ein zusätzliches Ticket für den öffentlichen Nahverkehr (CityMobil) handeln. Diese Zusatzangebote werden jeweils auf einer eigenen Bildschirmanzeige angeboten. Der Nutzer muss für jedes Angebot zunächst die Beschreibung lesen und aufnehmen und im Anschluss eine Entscheidung für oder gegen dieses Angebot treffen. Bei Aufgabeninstanz zwei nimmt nur der Teil mit der Abwicklung der zwei Zusatzangebote für den Fall, dass alle Angebote abgelehnt werden, laut des NGOMSL-Modells 12,93 Sekunden Bearbeitungszeit in Anspruch. Davon machen 2,6 Sekunden die besagten Operatoren inkl. ihrer Statement Time aus.
Um die Bearbeitungszeit dieser Abwicklung zu verkürzen, könnte für die Präsentation aller verfügbaren Zusatzangebote lediglich eine Bildschirmanzeige genutzt werden, auf der ausschließlich nach dem Wunsch der Angebote gefragt wird. Dabei sollte eine Mehrauswahl ermöglicht werden. Eine Erklärung der jeweiligen Angebote könnte durch die Betätigung einer weiteren Schaltfläche aufgerufen werden. So würde dem Nutzer ermöglicht, alle Zusatzangebote gleichzeitig abzulehnen. Außerdem könnte ein geübter Nutzer direkt die gewünschten Angebote auswählen, ohne, dass ihm zunächst Erklärungen und Informationen dazu präsentiert würden. Daneben hätten unerfahrene Nutzer jedoch immer noch die Möglichkeit, eine zusätzliche Beschreibung anzufordern, für die sie sicherlich

auch zusätzliche Bearbeitungszeit duldeten. Würde man das NGOMSL-Modell dahingehend verändern, so verkürzte sich die berechnete Bearbeitungszeit von Aufgabeninstanz zwei für diesen Auszug von 12,93 auf 10,73 Sekunden. Dabei würden die Anzahl der Statements und Tastendrücke verringert werden. Außerdem wäre lediglich ein „Read value from the screen"-Operator nötig. Der Korrektheit des gesamten Modells halber, würden hierfür ebenfalls die Methoden „behandle Versicherung", „behandle Bonusprogramm" und „behandle Angebot CityMobil" verändert werden. Alle manipulierten Methoden sind am Ende des dritten Anhangs zu finden. Diese Optimierung verbessert sowohl die Steuerbarkeit, als auch die Aufgabenangemessenheit des Systems, denn der Nutzer kann stärker in den Dialogablauf eingreifen und benötigt weniger Zeit, um die Aufgabe zu erledigen.

In Kapitel 4.3.2.2 wurde bereits auf die Konsistenz des NGOMSL-Modells eingegangen. Betrachtet man das Modell hinsichtlich seiner Konsistenz, so fallen sowohl einige mehrmals genutzte, als auch nur einmal benutzte Methoden auf. Im Folgenden werden die mehrmals genutzen Methoden als „allgemeine Methoden" und die nur einmal benutzen als „spezielle Methoden" bezeichnet. Diese Verwendung steht nicht im Zusammenhang zu der allgemeinen Form eines NGOMSL-Modells, die besagt, dass das NGOMSL-Modell nicht auf eine spezielle Aufgabeninstanz zugeschnitten ist. Insgesamt sieben Methoden und Selektionsregeln, von insgesamt 56, werden mehrmals im NGOMSL-Modell aufgerufen. Dazu zählen beispielsweise die Methoden „initialisiere Fahrkartenkauf", „gebe Ziel ein" und insbesondere „drücke button". Zu den einmal verwendeten Methoden gehören einerseits ähnliche Ziele, wie beispielsweise die Wahl des Hinfahrt- und des Rückfahrtdatums oder die Sitzplatzreservierung der Hin- und Rückfahrt, andererseits spezielle Ziele, wie „spezifiziere Fahrradauswahl" oder „wähle Bahncards". Trotz der Ähnlichkeit erstgenannter Ziele, sind unterschiedliche Methoden erforderlich, um diese zu erreichen. Bei diesen Methoden liegt es daran, dass die Eingaben bzgl. der Hinfahrt jeweils genutzt wurden, um die erforderlichen Eingaben der Rückfahrt zu verkürzen. So erscheinen bei der Wahl des Rückfahrtdatums z.B. sofort Daten, die nach dem Hinfahrtdatum liegen. Dem Nutzer werden dadurch Tastendrücke erspart, wodurch die Bearbeitungszeit der entsprechenden Aufgabe kürzer wird. Im Gegenzug verlängert sich allerdings die Lernzeit für das gesamte Modell, da statt einer Methode zwei Methoden gelernt werden müssen. Hier ist demnach abzuwägen, welche Zeiten entscheidender für die Gebrauchstauglichkeit des Geräts sind. Vermutlich trifft dies hier auf die Bearbeitungszeit zu, da der Fahrkartenautomat darauf ausgelegt ist, für kurze Aufgaben genutzt zu werden und dies auch der Erwartungshaltung der Nutzer entspricht[1]. Außerdem sind die Auswirkungen auf die Bearbeitungszeit auf Grund von Tastendrücken drastischer. Denn z.B. die Methoden „wähle anderes Hinfahrt-Datum" und „wähle anderes Rückfahrt-Datum" unterscheiden sich zwar, jedoch lediglich

[1] Alle Probanden waren sich einig, dass sie für komplexere Aufgaben, wie hier Aufgabeninstanz zwei oder fünf, nicht den Fahrkartenautomaten nutzen würden, sondern ein Service-Büro in Anspruch nähmen.

in wenigen Statements. Die Hauptteile dieser Methoden laufen nahezu analog ab. Die Auswirkungen dieser Methoden auf die Lernzeit sind folglich nicht sehr hoch, da viele gleiche Statements auftreten, die nur einmal in die Lernzeit einfließen. Allerdings gilt dies nur für die ähnlichen Methoden und nicht für die speziellen, die sich stärker auf die Lernzeit auswirken. Relativ ausgedrückt wurden 10,86% aller Statements als ähnlich oder gleich klassifiziert und deshalb nicht in die Berechnung der Lernzeit einbezogen. Zusammen mit den sieben allgemeinen Methoden und Selektionsregeln, die 12,5% aller Methoden und Selektionsregeln darstellen, deutet dies auf eine positive Tendenz bzgl. der Konsistenz der Benutzungsoberfläche hin, die allerdings nicht ausreicht, um das Modell insgesamt als konsistent bewerten zu können.
Dies wird von der berechneten Lernzeit bestätigt. Diese sagt vorher, dass ein neuer Nutzer 81,55 Minuten zuzüglich der Ausführungszeit der Trainingsaufgaben benötigen wird, um alle möglichen Schritte, Abzweigungen und Funktionen des gesamten Modells zu erlernen[2]. Diese Lernzeit ist aus zwei Perspektiven zu betrachten. Unter der Voraussetzung, dass ein Nutzer eine neue Software erlernen möchte, weil er diese nutzen und bedienen können möchte, sind etwa 80 Minuten zum Erlernen aller Funktionen und Vorgänge nicht viel. Unter der Voraussetzung, dass ein Nutzer lediglich das spezielle Ziel hat, eine Fahrkarte zu erwerben, sind ca. 80 Minuten, die er investieren muss, um zu wissen, wie er dies mit dem Fahrkartenautomaten tun kann, zu lang. Denn ein Nutzer eines solchen Automaten ist nicht daran interessiert, sämtliche Funktionen des Fahrkartenautomaten beherrschen zu können. Er hat lediglich ein Ziel vor Augen: die Fahrkarte. Aus diesem Grund wird er nicht bereit sein, diese Lernzeit im Vorfeld der Bedienung zu opfern, indem er z.B. an einem Training teilnimmt, wie es für die Lernzeitberechnung angenommen wird (s. Kapitel 4.3.2.1). Ein Fahrkartenautomat ist ein „walk-up-and-use“ System. Dies bedeutet, er ist ein gelegentlicher Gebrauchsgegenstand, der auch von Erst- und einmaligen Nutzern ohne Training effektiv bedient werden können sollte. Er sollte deshalb in seiner Bedienung selbsterklärend sein. Die berechnete Lernzeit deutet allerdings auf eine unintuitive Bedienung hin.
Ein Vergleich mit den Probandenwerten bekräftigt diese Aussage. Die ungeübten Probanden benötigten durchschnittlich etwa 13 Minuten, um alle fünf Aufgabeninstanzen einmalig zu bearbeiten. Die Probanden kannten die notwendigen Schritte nicht und lösten die Aufgaben durch Ausprobieren. Da mit den Aufgabeninstanzen alle Funktionen des erstellten NGOMSL-Modells abgedeckt werden, könnten diese Aufgabeninstanzen auch als Trainingsaufgaben fungieren. Bei der berechneten Lernzeit ist die Zeit zum Ausführen der Trainingsaufgaben nicht enthalten. Addiert man die berechneten Bearbeitungszeiten der einzelnen Aufgaben zu der Lernzeit, so ergibt sich ein Wert von 89,6 Minuten, mit

[2]In diesem NGOMSL-Modell sind nicht alle Funktionen integriert, die der Fahrkartenautomat bereitstellt, sondern nur jene, die für die Bearbeitung der hier gewählten Aufgabeninstanzen nötig sind

dem ein direkter Bezug zu den Probandenwerten möglich ist. Bezieht man die 13 Minuten auf die berechnete Lernzeit inkl. der Trainingszeit, so bedeutet dies, dass ein Lerner zwischen sechs und sieben Durchläufe der fünf Aufgaben vornehmen muss, um die Bedienung des Automaten bzgl. dieser Funktionen erlernt zu haben. Dabei ist zu beachten, dass die Lernzeitberechnung davon ausgeht, dass dem Lerner die Bedienung des Geräts im Vorfeld erklärt wird. Zeit zum Ausprobieren ist in der berechneten Lernzeit demnach nicht enthalten. Eliminierte man diese „Ausprobier-Zeit“ bei den Probanden, so würde sich die Anzahl der erforderlichen Durchläufe erhöhen. Diese Wiederholungsrate erscheint zu hoch für ein Gerät, von dem angenommen wird, dass es besonders intuitiv ist. Eine Optimierung der Lernzeit wäre folglich empfehlenswert. Da die Lernzeit maßgeblich von der Konsistenz der Bedienerführung abhängt, wird vorgeschlagen, weniger Methoden einzusetzen, die dafür allgemeiner gehalten und dadurch häufiger wiederverwendbar sind. So muss der Lerner weniger Methoden erlernen und hat trotzdem die Fähigkeit, dieselbe Aufgabenvielfalt mit dem Gerät zu bearbeiten. Durch diese Optimierung würde folglich eine Verbesserung der Erwartungskonformität und der Lernförderlichkeit des Systems erreicht.

5.2 Aussagen der empirischen Untersuchungen

Auch wenn die empirischen Untersuchungen in dieser Arbeit ausschließlich Hilfsmittel für die Erstellung detaillierter GOMS-Analysen waren und im Hintergrund dieser Analysen stehen, soll dennoch ein Einblick in einige auffallende Nutzerbeobachtungen gewährt werden.

Wie im vorherigen Abschnitt bereits erwähnt, sind die Operatorzeiten der „Wait for“-Operatoren, sowie der mentalen Operatoren „Think for“ bzw. „Make up your mind about“ und „Think about“, auffällig. Die Wartezeiten treten nicht nur an den Stellen auf, wo offensichtliche Rechenzeiten erwartet werden, wie z.B. dort, wo der Fahrkartenautomat die verfügbaren Verbindungen ermittelt. Auch nach Tastendrücken, die aus Nutzersicht keine besonderen Berechnungen erfordern sollten, treten Verzögerungszeiten von über einer Sekunde Dauer auf. Dies ist beispielsweise in der Methode „wähle anderes Hinfahrt-Datum“ der Fall. Wurde die Schaltfläche „späteres Datum“ gedrückt, so lädt das System 1,16 Sekunden, bis die nächste Bildschirmanzeige erscheint. Außerdem verursachte das System nach jedem Tastendruck eine Verzögerungszeit zwischen 0,5 und 1,3 Sekunden (s. Kapitel 4.3.4). Diese Zeiten deuten auf einen Optimierungsbedarf hin. Denn mit den derzeit verfügbaren Technologien sollte eine fühlbare Wartezeit nach jedem einzelnen Tastendruck nicht auftreten. Die Optimierung sollte hier weniger bei der Bedienerführung, sondern bei der Hardware stattfinden. Um diese Optimierung genauer spezifizieren zu können, werden Details zu den vorhandenen Systemressourcen benötigt, die im Rahmen dieser Arbeit

nicht zur Verfügung standen. Vermutlich können die Wartezeiten durch die Aufrüstung von Prozessor und Arbeitsspeicher des Fahrkartenautomaten reduziert werden.
Die drei empirisch gemessenen mentalen Operatoren sind unvermeidbar. Diese Denkzeiten sind nicht auf die Bedienerführung des Geräts zurückzuführen, da sie durch die Aufnahme aktueller Informationen entstehen, die unabdingbar sind. Somit ist hier keine Optimierung möglich.

Im vorherigen Abschnitt wurde bereits der Auswahldialog bzgl. der Bahncards erwähnt. Dazu wurden zusätzliche, empirische Beobachtungen gemacht. Wie die Zuordnung der aktuellen Anzeige „Frame“ neun bis zwölf in dem KLM-Modell zu Aufgabeninstanz zwei zeigt, erscheinen die einzelnen Auswahlmöglichkeiten zu den verschiedenen Bahncardtypen nicht gleichzeitig, sondern nacheinander. Zu Beginn dieser Sequenz wurde ausschließlich die Verfügbarkeit eines Bahncardtypen abgefragt. Nichts wies darauf hin, dass noch weitere Auswahlmöglichkeiten diesbezüglich folgen würden. Alle ungeübten Probanden waren an dieser Stelle sehr verunsichert über den weiteren Verlauf der Bedienerführung. Hier wurde folglich die Selbstbeschreibungsfähigkeit des Systems vermisst. Teilweise brachen die Probanden die Aufgabe ab, nahmen Rückschritte vor oder fuhren erst nach längeren Überlegungen mit der Aufgabe fort. Bei dem geübten Nutzer traf dieses Verhalten erwartungsgemäß nicht auf, weshalb diese Denkzeiten nicht als mentale Operatoren in die Modelle integriert wurden. Neuen Nutzern scheint dieses Auswahlverfahren jedoch erhebliche Probleme zu bereiten. Da Fahrkartenautomaten oft von ungeübten Nutzern bedient werden, wäre hier auch auf Grund dieser empirischen Beobachtungen eine Optimierung durchaus sinnvoll. Der im vorherigen Abschnitt beschriebene Optimierungsvorschlag, würde auch diesem Problem entgegenwirken.

6 Bewertung der eingesetzten Techniken

In diesem Kapitel werden die unterschiedlichen hier verwendeten GOMS-Varianten gegenübergestellt und bzgl. der Realitätsnähe der Bearbeitungszeiten, ihrer Aussagekraft und des für die Erstellung und Analyse erforderlichen Aufwands verglichen und bewertet.

6.1 Realitätsnähe der Abschätzungen

In diesem Kapitel wird die Realitätsnähe der berechneten Bearbeitungszeiten bewertet. Da für eine Einschätzung bzgl. der Realitätsnähe der berechneten Lernzeit keine Vergleichswerte vorlagen, konnte diese im Rahmen dieser Arbeit nicht vorgenommen werden. Um die Eignung der GOMS-Varianten anhand der Realitätsnähe ihrer Abschätzungen bewerten zu können, sind im Folgenden die berechneten Bearbeitungszeiten tabellarisch aufgeführt. Auch die empirisch erhobenen Zeiten des geübten Probanden werden zum Vergleich hinzugezogen, um einen direkten Bezug zu einer realistischen Situation an dem Fahrkartenautomaten herstellen zu können. Zu beachten ist bei der folgenden Validierung der Berechnungen, dass diese lediglich anhand der Werte eines Probanden vorgenommen wird. Es handelt sich dabei folglich um eine tendenzielle Einschätzung und nicht um eine repräsentative Bewertung.

Aufgabeninstanz	KLM	NGOMSL	geübter Proband
1	30,30s	42,16s	33,51s
2	162,10s	173,10s	134,78s
3	22,20s	22,20s	15,51s
4	103,75s	110,95s	84,0s
5	116,15s	134,47s	91,25s

Auffallend ist, dass die Abschätzung der Bearbeitungszeiten bei den KLM-Modellen, bis auf eine Ausnahme, stets geringer ausfallen, als bei dem NGOMSL-Modell. Dies ist u.a. darauf zurückzuführen, dass die einzelnen Statements in den KLM-Modellen, unabhängig vom Inhalt, nicht mit einer extra Zeit einfließen, wie es bei dem NGOMSL-Modell mittels

der NGOMSL Statement Time geschieht (s. Kapitel 3.2.3.1). Durch die Programmform besteht das NGOMSL-Modell zudem aus wesentlich mehr Statements, als das entsprechende KLM-Modell, was sich zusätzlich in der berechneten Bearbeitungszeit bemerkbar macht. Auch „Decide-“Operatoren, die, ähnlich wie Selektionsregeln, Abzweigungen formulieren, oder die Berücksichtigung des Kurz- und Langzeitgedächtnisses, erhöhen die Anzahl Statements in dem NGOMSL-Modell und damit auch die Bearbeitungszeit. Die Abweichungen zwischen den berechneten Bearbeitungszeiten sind allerdings nicht hoch. Jedoch nimmt das Verhältnis dieser Abweichungen mit zunehmender Länge der Aufgabeninstanz zu. Die Vorhersagen der unterschiedlichen Techniken entfernen sich also, mit zunehmender Länge der Aufgabe, weiter voneinander.

Insgesamt war ein Unterschied zwischen den Werten der GOMS-Analysen und denen des Probanden zu erwarten. Denn die Modelle bleiben eine vereinfachte Modellierung der Interaktion zwischen Mensch und Fahrkartenautomat und können deshalb nicht die exakte Zeit eines Probanden vorhersagen. Darüber hinaus kann die Abweichung weitere Gründe, wie z.B. die Wahl der Zielhierarchie, haben. Diese wird vom Analysten gewählt und ist deshalb subjektiv. Sie muss daher nicht mit der Zielhierarchie des Probanden übereinstimmen. Da die Struktur dieser Hierarchie aber bedeutende Auswirkungen auf die abgeschätzte Bearbeitungszeit hat, sind Unterschiede zwischen dieser Zeit und der des Probanden zu erwarten. Da die Anzahl der Statements nur bei NGOMSL zur Berechnung der Bearbeitungszeit herangezogen wird, sind insgesamt die Unterschiede zwischen den Werten des Probanden und den von KLM abgeschätzten Werten nicht so groß, wie die zwischen den Werten des Probanden und denen des NGOMSL-Modells. Außerdem basieren die mentalen Operatoren in den Abschätzungen auf den empirisch ermittelten Werten der ungeübten Probanden. Die Vermutung liegt nahe, dass diese Denkpausen bei dem geübten Probanden kürzer ausfallen, da dieser bereits die Darstellung der neuen Informationen kennt und sich somit auf der präsentierten Bilschirmansicht nicht zunächst orientieren muss. Ein Kriterium erhöht die tatsächliche Abweichung allerdings noch. Bei den Abschätzungen sind nicht die Wartezeiten berücksichtigt, die der Fahrkartenautomat nach jedem einzelnen Tastendruck verursacht[1]. Bei den Zeiten des Probanden sind diese jedoch enthalten.

Es scheint, als würde die höhere Komplexität des NGOMSL-Modells dazu führen, dass die Bearbeitungszeiten generell bei diesem Modell höher als bei den KLM-Modellen ausfallen. Hier bedeutet die Erhöhung der Bearbeitungszeit gleichzeitig eine weitere Entfernung von der Zeit, die der Proband tatsächlich benötigt hat. Es stellt sich folglich die Frage, ob das präzisere NGOMSL-Modell bzgl. der Bearbeitungszeiten tatsächlich auch exaktere Aussagen treffen kann, als die stärker vereinfachten KLM-Modelle. Die oben aufgeführten

[1] Diese Abweichung wurde bereits in Kapitel 4.3.4 erläutert.

Werte deuten auf eine Verneinung dieser Frage hin. Vergleicht man nur die abgeschätzten Gesamt-Bearbeitungszeiten der einzelnen Aufgaben, so sind die mit KLM-berechneten Werte realistischer, als die mittels NGOMSL berechneten Zeiten. Allerdings sind die abgeschätzten Werte insgesamt nicht zufriedenstellend. Die größte Abweichung zwischen den von KLM vorhergesagten Zeiten und denen des Probanden liegt bei ca. 27 Sekunden bei einer abgeschätzen Gesamtzeit von 162,1 Sekunden. Bei NGOMSL sind dies 40 Sekunden bei einer Aufgabe, die insgesamt mit 134,47 Sekunden abgeschätzt wurde. Diese Unterschiede verfälschen die Auswertung zu sehr. Für die kürzeren Aufgaben eins und drei sind die Abschätzungen beider Modelle jeweils nah an denen des Probanden. Allerdings liegen diese Zeiten im Bereich von 30 Sekunden oder darunter. Es scheint demnach, als würden die Abschätzungen beider Modelle für Aufgaben, die über einer Minute Dauer liegen, bereits zu ungenau werden. Daher sollten die abgeschätzten absoluten Werte kritisch betrachtet werden und folglich nicht als sekundengenaue Vorhersagen, sondern lediglich als eine Tendenz bei der Bewertung einer Benutzungsoberfläche angesehen werden.

6.2 Aussagekraft der GOMS-Analysen

Vergleicht man die Erkenntnisse der KLM-Modelle mit denen des NGOMSL-Modells, so unterscheidet sich sowohl deren Qualität, als auch deren Quantität.

- **Bearbeitungszeiten:** Mit beiden Modellen konnte die Gesamt-Bearbeitungszeit der einzelnen Aufgaben berechnet werden. Diese Zeiten ermöglichten eine grobe Überprüfung, ob die Bearbeitungszeiten der Aufgabe im erwarteten Toleranzbereich liegen oder nicht. Auch markante Ausführungszeiten von bestimmten Operatoren konnten aus den Modellen herausgefiltert werden. Allerdings ist die Existenz dieser auffälligen Operatoren nicht auf die Fähigkeiten der Modelle zurückzuführen, sondern auf die empirischen Untersuchungen, anhand der diese Werte ermittelt wurden.
- **Operatorsequenzen:** Einzelne Operatorsequenzen konnten zwar mit beiden Varianten betrachtet werden, allerdings erwies sich die Zuweisung der Sequenzen zu bestimmten Stellen im gesamten Aufgabenverlauf mit den KLM-Modellen als schwierig. Denn bei diesen wurden lediglich einige wenige Operatoren zur Beschreibung der Aufgaben verwendet, während die NGOMSL-Modelle aus konkret formulierten Zwischenzielen und individueller bezeichneten Operatoren aufgebaut sind. Durch die dadurch entstehende Struktur ist in dem NGOMSL-Modell jederzeit ersichtlich, in welchem Teil der Aufgabe die betrachtete Operatorsequenz liegt. Dies erleichterte die Lokalisierung von auffälligen Operatorsequenzen innerhalb des Gesamtkontextes deutlich. Die NGOMSL Statement Time der Sequenzen erwies sich darüber hinaus

als hilfreiches Maß für die Länge und Komplexität einer Methode und konnte zusätzlich zur Ermittlung zeitintensiver Methoden herangezogen werden.

- **Lernzeit:** Aussagen über die Lernzeit kann ausschließlich das NGOMSL-Modell treffen. Da die Lernzeit auch ein Maß für die Konsistenz des Modells und der Bedienerführung ist, gilt dies ebenso dafür. Mittels der Lernzeit konnte auf eine unintuitive Bedienung geschlossen werden, die vermutlich insbesondere von der überwiegenden Inkonsistenz der Methoden verursacht wird. Im Gegensatz dazu sagen KLM-Modelle nichts über Lernzeit und Konsistenz des Modells und damit auch der Bedienerführung aus.
- **Dokumentation:** Alle hier erstellten GOMS-Modelle repräsentieren den Handlungsablauf zur Bearbeitung bestimmter Aufgaben. Die Modelle stellen eine Art Verlaufsprotokoll dar, d.h. sie sind eine Art Anleitung, die beschreibt, welche einzelnen Schritte nacheinander vorgenommen werden müssen, um eine bestimmte Aufgabe zu lösen. Das NGOSML-Modell ist in dieser Anleitung jedoch präziser. Die konkrete Zielformulierung und Aufteilung in einzelne Ziele innerhalb einer Zielhierarchie, erleichtert die Erstellung von Dokumentation und Handbuch. Ein fertiges NGOMSL-Modell kann deshalb auch als eine Art Betriebsanleitung in die Dokumentation eines Gerätes einfließen. Die Unterteilung in Ziele und Unterziele kann verwendet werden, um die Dokumentation zu gliedern und Stichworte für einen Glossar oder eine Suche herauszufiltern.

Anhand dieser Kriterien soll nun zusammenfassend geklärt werden, ob die in dieser Arbeit angewendeten Varianten der GOMS-Familie tatsächlich dabei helfen, Schwächen in der Usability eines Geräts, insbesondere dessen Bedienerführung, aufzudecken.
Vor allem die Betrachtung einzelner Operatorsequenzen und Methoden bzgl. ihrer Bearbeitungszeit hilft dabei, konkrete Mängel in der Bedienerführung eines Systems zu enttarnen und daraus Optimierungsvorschläge zu entwickeln. Allein diese Fähigkeit von NGOMSL-Modellen macht sie für eine Usability-Bewertung empfehlenswert. KLM-Modelle können dies theoretisch auch, praktisch jedoch ist es mit ihnen kaum durchführbar, da durch fehlende Zielformulierungen die Zuordnung zum Kontext innerhalb der Aufgabe fehlt. Darüber hinaus können mit den Aussagen über Lernzeit und Konsistenz Rückschlüsse auf die Intuition der Bedienerführung gezogen werden. Der Grad der Intuition einer Bedienerführung wirkt sich auf die Gebrauchstauglichkeit dieser aus. Somit wäre diese ein weiteres Kriterium, das für die Aussagekraft eines NGOMSL-Modells bzgl. Usability spricht. Da KLM-Modelle als Aussagen hauptsächlich die berechneten Gesamt-Bearbeitungszeiten einer Aufgabe liefern, die eher als Trend zu bewerten sind, sind sie weniger hilfreich, um konkrete Schwächen einer Bedienerführung aufzudecken, da, wie bereits erläutert, die Aussagekraft der berechneten Gesamt-Bearbeitungszeiten in diesem Zusammenhang nicht

hoch ist. Sie sind dagegen in der Lage, eine Gesamttendenz bzgl. der Bearbeitungszeit einer Aufgabe anzugeben. Aber für Vergleichsschätzungen scheinen sie aussagekräftiger und überzeugender sein zu können. Denn dort ist das Verhältnis der berechneten Bearbeitungszeiten zueinander entscheidend, nicht aber sind es die konkreten Werte. Da die Zeiten auf Grund gleichen Vorgehens berechnet wurden, werden die Vergleiche realistischer, als die konkreten Zeiten, sein. Auch NGOMSL-Modelle können vermutlich nützliche Aussagen über den Vergleich von Designalternativen treffen.
Zusammenfassend ist festzuhalten, dass beide GOMS-Varianten hilfreich bei der Bewertung der Usability sein können, wobei KLM allerdings weniger für die Einschätzung einzelner Benutzungsoberflächen, sondern mehr für die von Designalternativen geeignet ist.

6.3 Aufwand der GOMS-Analysen

Vergleicht man die KLM-Modelle mit dem NGOMSL-Modell, so wird augenblicklich ein großer Unterschied deutlich. Pro Aufgabeninstanz wurde ein KLM-Modell erstellt. Allerdings gibt es nur ein NGOMSL-Modell für alle Aufgabeninstanzen. Dies ist darauf zurückzuführen, dass erstere Modelle stets speziell formuliert sind, während letzteres durch die Programmform mit Zielen und Selektionsregeln allgemeiner formuliert ist und dadurch für unterschiedliche Aufgabeninstanzen verwendet werden kann. Gerade aus diesem Grund wird das NGOMSL-Modell allerdings auch wesentlich umfangreicher, als die KLM-Modelle.
Die Erstellung der KLM-Modelle nahm weniger Zeit in Anspruch, als die des NGOMSL-Modells. Da für die KLM-Modelle keine Zielhierarchie oder Selektionsregeln konstruiert werden mussten, waren weniger Iterationen nötig, um die Modelle in ihre endgültige Form zu bringen. Anders verhielt es sich mit der Erstellung des NGOMSL-Modells. Bis die Zielhierarchie in ihrer endgültigen Form war, waren viele Iterationen nötig. Oft ergaben sich neue Unterziele, weil die Anzahl Statements pro Zielmethode zu hoch wurde. Auch die Formulierung der Selektionsregeln erforderte mehr Zeit, da diese eindeutig sein mussten und zugleich alle möglichen Fälle abdecken sollten. Letztlich erforderte auch die Konsistenzüberprüfung weitere Iterationen. So wurde darauf geachtet, Methoden ähnlich aufzubauen, einheitliche Methodenbezeichner zu verwenden und soweit es ging, ähnliche Methoden zu einer zusammenzufassen um eine höchstmögliche Wiederverwendbarkeit von Methoden zu ermöglichen. Auf Grund der Sequenzform der KLM-Modelle stellten sich diese letzteren Überlegungen dort nicht. Dies führte wesentlich schneller zu der Konstruktion ausgereifter Modelle.
Die Berechnung der Bearbeitungszeiten erfolgte sowohl bei den KLM-Modellen, als auch bei dem NGOMSL-Modell per Hand. Bei den KLM-Modellen konnten, dank der Sequen-

form, die einzelnen Operatoren direkt abgelesen und deren einzelne Ausführunsgzeiten zur Gesamt-Bearbeitungszeit einer Aufgabe aufsummiert werden. Für die Berechnung anhand des NGOMSL-Modells war zunächst eine Protokollierung des Methodenablaufs für jede einzelne Aufgabeninstanz erforderlich. Im Anschluss wurden die Statements ausgezählt und die einzelnen Operatorzeiten aufsummiert. Nach Einsetzen in die entsprechende Formel ergab sich dann die abgeschätzte Bearbeitungszeit der einzelnen Aufgaben. Für die Berechnung der Lernzeit wurden alle Statements des NGOMSL-Modells gezählt und die LTM Chunks ermittelt. Anschließend erfolgte die Berechnung mittels der vorgestellten Formel. Der Aufwand hierfür war also wesentlich höher.

6.4 Zusammenfassende Gegenüberstellung von KLM und NGOMSL

Um die KLM-Variante und die NGOMSL-Variante zu bewerten, wird im Folgenden eine Gegenüberstellung von Aufwand und Aussagekraft inkl. der Realitätsnähe vorgenommen. Setzt man den Aufwand, der für die Erstellung der verschiedenen Modelle nötig war, in Bezug zu deren Nutzen, der hier in der Aussagekraft und der Realitätsnähe der Abschätzungen gesehen wird, so schneiden beide GOMS-Varianten insgesamt gut ab. Mit Hilfe der KLM-Modelle konnten schnell quantitative Aussagen über bestimmte Aufgabeninstanzen getroffen werden, wobei die Schnelligkeit zu Lasten des Detailgrads ging. Dennoch ist die Aussagekraft dieser Modelle nicht zu verkennen. Die Bearbeitungszeit einer Aufgabe, berechnet mit einfachen Operatoren auf der Keystroke-Ebene, kann bereits wertvolle Hinweise auf die Effizienz der Benutzungsoberfläche geben und den Entschluss für weitere Verfahren zur Bewertung der Usability gezielt veranlassen, auch wenn sie nur als Trend zu bewerten ist. Das NGOMSL-Modell erforderte deutlich mehr Zeit, um in einen ausgereiften und korrekten Zustand zu gelangen. Die Anforderungen an den Analysten waren deutlich höher, da dieser sich selbst eine realistische Zielhierarchie mit den dazugehörigen Selektionsregeln überlegen, und dabei nach gewissen Vorgaben vorgehen musste. Auch die Berechnung der Abschätzungen wurde vom Analysten durchgeführt und erforderte einen zusätzlichen Aufwand. Als Ergebnis erhielt man bei der Analyse mittels NGOMSL allerdings nicht nur quantitative Abschätzungen zu Bearbeitungs- und Lernzeiten, sondern darüber hinaus auch eine klare strukturierte Anleitung, wie ein Nutzer vorgehen muss, um bestimmte Aufgaben zu erreichen. Auch die Konsistenz des Modells war, im Gegensatz zu KLM, bei NGOMSL prüfbar und lieferte wertvolle Hinweise auf die Gestaltung der Methoden und damit des Systems.
Zusammenfassend ist festzuhalten, dass beide hier eingesetzten Varianten ein angemessenes Kosten-Nutzen-Verhältnis aufweisen. KLM-Modelle sind schnell erstellt und deren

Bearbeitungszeiten schnell berechnet. Sie geben bereits eine objektive Auskunft über quantitative Messgrößen und können als Vorarbeit für die Erstellung weiter Modelle, z.B. eines NGOMSL-Modells wie in dieser Arbeit, dienen. NGOMSL-Modelle erfordern deutlich mehr Erstellungs- und Auswertungsaufwand, gehen in ihrer Aussagekraft durch die Möglichkeit einer Lernzeitvorhersage und die Verwendung einer Zielhierachie aber auch weit über die KLM-Modelle hinaus und sind außerdem wesentlich präziser formuliert und dadurch leichter nachvollzieh- und erweiterbar.

Steht die Zeit zur Erstellung und Auswertung eines NGOMSL-Modells zur Verfügung, wird geraten, diese einer KLM-Analyse vorzuziehen, sofern es gewünscht ist, konkrete Optimierungsbedürfnisse einer Benutzungsoberfläche aufzudecken. Sollen dagegen unterschiedliche Designs verglichen werden, um das effizientere Design zu ermitteln, so scheint eine KLM-Analyse durchaus ausreichend und geeignet zu sein, vorausgesetzt, die Lernzeit ist nicht erforderlich.

6.5 Eignung der Techniken für die Analyse des Fahrkartenautomaten

Nachfolgend wird beurteilt, ob die in dieser Arbeit angewendeten GOMS-Techniken tatsächlich für die Analyse des Fahrkartenautomaten geeignet waren.
Die in Kapitel 5 vorgestellten Optimierungsmöglichkeiten des Fahrkartenautomaten zeigen, dass mit Hilfe der hier angewendeten Analyseverfahren konkrete Schwachstellen in der Bedienerführung des Automaten erkannt wurden. Außerdem ließen sich diese so genau klassifizieren, dass spezielle Optimierungsempfehlungen abgeleitet werden konnten. Dies spricht eindeutig dafür, dass die Anwendung der GOMS-Varianten, insbesondere von NGOMSL, nützliche Bewertungen bzgl. der Gebrauchstauglichkeit des Automaten liefern konnte. Lediglich die Verwendung der Lernzeitabschätzung war hier kritisch zu betrachten, da es sich, wie bereits in Kapitel 5 erläutert, bei dem Fahrkartenautomaten um ein „walk-up-and-use“ System handelt, auf das die vorgestellten Lernsituationen nicht passen. Dadurch kann es zu irreführenden Vorhersagen kommen, vor allem zu solchen, die zu niedrig berechnet sind.

6.6 Grenzen von KLM und NGOMSL

In den vorherigen Abschnitten wurden die Aussagen, die mit den GOMS-Varianten KLM und NGOMSL getroffen werden können, diskutiert. Daneben gibt es Kriterien, die nicht

mittels dieser Analyseverfahren untersucht werden können, deren Auswertungen aber hilfreiche Erkenntnisse über die Usability einer Benutzungsoberfläche liefern könnten.

Da mittels der GOMS-Verfahren lediglich Routineaufgaben beschrieben werden können, die von geübten Nutzern ausgeführt werden, ergeben sich bereits zwei Kriterien, die GOMS nicht betrachten kann. Ersteres ist das explorierende Verhalten eines Nutzers. Es wäre interessant zu erfahren, wie das Problemlöseverhalten des Nutzers aussieht und wie es sich in der Ausführungszeit einer Aufgabe bemerkbar macht. GOMS-Analysen setzen voraus, dass der Nutzer alle notwendigen Operatoren bereits kennt. Doch wie der Nutzer zu diesem Wissen gekommen ist, wird nicht berücksichtigt oder modelliert. Da jedoch jedem Operator, den der Nutzer kennt, ein explorierendes Verhalten vorausgehen muss, erscheint es wichtig, mehr über diesen Vorgang zu erfahren. Dieser Bereich ist bereits ein aktives Forschungsgebiet der Kognitionspsychologie.
Das zweite Kriterium ist die Fehlerbetrachtung. Nicht nur neue Nutzer eines Systems machen Fehler, sondern auch Experten. Denn Fehler liegen in der Natur des Menschen. Es stellt sich die Frage, wie Fehler sich auf die weitere Handlung eines Nutzers auswirken und, inwiefern sie sich bei der Bearbeitungszeit einer Aufgabe bemerkbar machen.
Darüber hinaus sind die mentalen Prozesse des Nutzers stark vereinfacht. Lediglich einige nicht spezifizierte mentale Operatoren, denen bestimmte Ausführungszeiten zugewiesen werden, modellieren komplexe Denkvorgänge. Die menschlichen Denkprozesse sind in der Realität jedoch wesentlich komplizierter, sodass die Beschreibung eines solchen Vorgangs allein durch einen oberflächlichen Begriff, wie z.B. „Denkpause“, nicht ausreicht.
Des Weiteren macht GOMS keine Aussagen über die Nutzerzufriedenheit. Eine Benutzungsoberfläche kann durch GOMS als konsistent und schnell zu bedienen bewertet werden, dennoch kann dieselbe so unansprechend gestaltet oder auch fehleranfällig sein, dass sie den Nutzer verärgert und abschreckt.

In Kapitel 3.2.5 wurde auf ein Erweiterungskonzept von GOMS hinsichtlich der Fehler-Berücksichtigung eingegangen, bei dem zusätzliche Fehlerbehebungs-Methoden direkt in ein GOMS-Modell integriert werden. Im nächsten Kapitel wird ein Ansatz vorgestellt, der mit Hilfe von empirischen Untersuchungen versucht, Fehler in GOMS-Analysen zu berücksichtigen. Der zusätzliche Einsatz empirisch ermittelter Erkenntnisse und Werte in einer GOMS-Analyse hat sich während dieser Arbeit als positiv erwiesen. Denn dadurch konnten die GOMS-Modelle spezieller auf das zu untersuchende System zugeschnitten werden, wodurch u.a. eine exaktere Berechnung der Bearbeitungszeiten möglich war. Auf der Grundlage dieser Erfahrungen wurde das im nächsten Kapitel folgende Konzept entwickelt.

7 Erweiterung der GOMS-Varianten

In diesem Kapitel wird der Ansatz einer Erweiterung von GOMS hinsichtlich der Berücksichtigung von Bedienfehlern vorgestellt. Diese Erweiterung ermöglicht es, Bedienfehler in GOMS-Analysen einzubeziehen, ohne dass die zuvor erstellten GOMS-Modelle überarbeitet werden müssen. Dies geschieht mit Hilfe empirischer Untersuchungen, deren Ergebnisse in die Abschätzungen der GOMS-Modelle einfließen.

7.1 Voraussetzungen

Der hier vorgestellte Ansatz beruht auf der Integration empirisch ermittelter Werte in eine GOMS-Analyse. Das bedeutet, dass dieses Verfahren lediglich für die Analyse existierender Systeme verwendet werden kann. Zumindet muss das System insoweit existieren, als dass ein Prototyp mit vollem Funktionsumfang zur Verfügung steht.
Des Weiteren wird vorausgesetzt, dass im Vorfeld ein vollständiges GOMS-Modell, auf der Annahme einer fehlerfreien Bedienung, erstellt wurde. Dieses Konzept ist nicht auf eine GOMS-Variante beschränkt.

7.2 Konzept

Der erste Schritt dieser Erweiterung ist die Definition der Zielgruppe. Es muss geklärt werden, auf welche Zielgruppe das zu untersuchende Gerät ausgerichtet ist. Im Anschluss wird die Zielgruppe in sinnvolle, unterschiedliche Nutzergruppen unterteilt. Dabei kann, je nachdem, wie es für das Gerät sinnvoll erscheint, z.B. nach Alter, Geschlecht, Bildungsgrad, Beruf oder explizit dem Geübtheitsgrad mit dem Umgang von computergestützten Geräten unterschieden werden. Diese Nutzergruppen können dazu verwendet werden, eine Unterscheidung hinsichtlich der Wahrscheinlichkeit von Bedienfehlern vorzunehmen.
Im zweiten Schritt folgt die empirische Untersuchung. Zu jeder Nutzergruppe werden Probanden gesucht, die unterschiedliche Aufgaben an dem zu untersuchenden Gerät durchführen sollen. Dabei sollte auf eine repräsentative Anzahl von Probanden geachtet werden.

Um diese Anzahl zu bestimmen, können unterschiedliche Berechnungsverfahren angewendet werden[1]. Bei den Aufgaben sollte es sich um dieselben Aufgaben handeln, die die einzelnen Aufgabeninstanzen zu dem erstellten GOMS-Modell darstellen. Voraussetzung dieser Aufgaben ist, dass sie den Funktionsumfang des Geräts weitestgehend abdecken. Weiterhin muss beachtet werden, dass alle Probanden aus den verschiedenen Nutzergruppen mit den gleichen Aufgaben konfrontiert werden. Nur so wird sichergestellt, dass die Ergebnisse der Untersuchungen vergleichbar sind. Ein Beispiel für solche Aufgaben sind jene, die für die, im Rahmen dieser Arbeit, durchgeführten GOMS-Analysen erstellt wurden. Die Aufgaben sollten insgesamt einen zeitlichen Rahmen von etwa 15 Minuten pro Proband und Aufgabenblock nicht überschreiten. Dies dient dazu, die empirischen Untersuchungen in einem realisierbaren Rahmen zu halten und darüber hinaus, zeitlich bedingte Konzentrationsschwächen der Probanden zu umgehen.
Die Probanden werden bei der Durchführung der Aufgaben beobachtet. Es werden die einzelnen Fehler pro Aufgabe gezählt, die die Probanden verursachen. Anschließend wird die durchschnittliche Fehleranzahl pro Aufgabe, getrennt nach Nutzergruppen, ermittelt. Diese wird im Folgenden als Fehlerrate bezeichnet.
Es folgt die Einbeziehung der Fehlerrate in die GOMS-Analyse. Dazu muss zunächst geklärt werden, inwieweit sich die Bearbeitungszeit einer Aufgabe verändert, wenn ein Fehler auftritt. Es muss demnach festgelegt werden, wieviel Bearbeitungszeit pro Fehler beansprucht wird. Card, Moran und Newell haben Experimente durchgeführt, in denen sie Nutzer diesbezüglich beoachtet haben und schreiben dazu in (CMN83, Seite 176f):

> *„Overall, about 26% of the total time spent in all the experimental tasks is due to error. [...,] and errors doubled the time to perform the tasks in which they occured (from 12.5 sec to 24.4 sec).“*

Diese Zeiten sollen hier als Grundlage verwendet werden. Allerdings wurden diese Werte anhand von Text-Editier-Aufgaben ermittelt und sind deshalb evtl. unpassend für das zu untersuchende Gerät. Um allgemeingültigere Werte zu erhalten, sind weitere Untersuchungen diesbezüglich wünschenswert. Da auf diese derzeit jedoch nicht zurückgegriffen werden kann, wird hier angenommen, dass das Auftreten eines Fehlers in einer Aufgabe die Bearbeitungszeit dieser Aufgabe verdoppelt. Das bedeutet also, dass die Fehlerrate pro Aufgabe multipliziert mit zwei den Faktor darstellt, mit dem die Bearbeitungszeit dieser Aufgabe multipliziert wird, um diese Zeit bzgl. der Fehlerberücksichtigung anzugleichen. Im Folgenden wird dies in einer Formel veranschaulicht.

Sei B' die Bearbeitungszeit pro Aufgabe inkl. Fehlerbetrachtung. Des Weiteren sei B die Bearbeitungszeit pro Aufgabe exkl. Fehlerbetrachtung. Außerdem sei F die Fehlerrate. Dann gilt: $B' = B * F * 2$

[1] s. z.B. http://www.uni-giessen.de/mossig/Minimaler%20Stichprobenumfang%20-%20Script.pdf

Demnach würde sich die Bearbeitungszeit einer Aufgabe verdoppeln, wenn in dieser ein Fehler auftritt, vervierfachen, wenn zwei Fehler auftreten, versechsfachen wenn drei Fehler auftreten und so fort. Mit Hilfe dieser Formel wird für jede Aufgabe pro Nutzergruppe die Bearbeitungszeit inkl. Fehlerzeit berechnet.

Mittels dieser Erweiterung kann eine quantitative Angleichung der Bearbeitungszeiten verschiedener Aufgaben vorgenommen werden. Durch die Angleichung wird sichtbar, wie sich das Auftreten von Fehlern auf die Bearbeitungszeit der einzelnen Aufgaben auswirkt. Durch die empirische Bestimmung der aufgetretenen Fehler pro Aufgabe erhält der Analyst bereits im Vorfeld einen Eindruck von der generellen Fehlerempfänglichkeit des Geräts. Die Beobachtungen können darüber hinaus dazu verwendet werden, die konkreten Stellen in der Bedienerführung, an denen Fehler auftreten, zu identifizieren. Zusammen mit den angeglichenen Bearbeitungszeiten können dann, über die ursprünglichen Aussagen der GOMS-Analyse hinaus, beispielsweise Optimierungspotentiale bzgl. der Fehleranfälligkeit und Unterstützung bei der Fehlerbehebung ermittelt werden. Außerdem ermöglicht die Unterteilung in unterschiedliche Nutzergruppen eine differenzierte Bewertung des Geräts hinsichtlich dessen Eignung für bestimmte Nutzergruppen. Es könnte z.B. festgestellt werden, dass das Gerät für bestimmte Nutzergruppen auf Grund einer zu hohen Fehlerrate prinzipiell ungeeignet ist. Für diese Nutzergruppen müsste dann ggf. ein Ersatz für das Gerät gefunden oder eine Neukonstruktion dessen vorgenommen werden, sofern die Nutzergruppen fester Bestandteil der Zielgruppe bleiben sollen.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

In den ersten drei Kapiteln der vorliegenden Arbeit wurden die wesentlichen Grundlagen der Usability-Bewertung interaktiver Systeme dargestellt. Insbesondere wurde hier die GOMS-Familie behandelt. Auf der Basis dieses Wissens wurde in dieser Arbeit der komplette Prozess einer Usability-Bewertung anhand eines Fahrkartenautomaten aufgezeigt. Nach der Darlegung der Entscheidung für geeignete Analyseverfahren, wurde auf die Durchführung der Analyse eingegangen. Hierbei wurden auch nötige Designentscheidungen des Analysten, wie die Konstruktion einer angemessenen Zielhierarchie, beschrieben.
Anhand der Ergebnisse der Analyse wurden anschließend konkrete Optimierungspotentiale des Fahrkartenautomaten aufgezeigt. Es wurden zwei konkrete Problemstellen, sowie ein allgemeiner Optimierungsbedarf identifiziert. Anhand dieser Optimierungsbedürfnisse wurden konkrete Optimierungsvorschläge abgeleitet. Die konkreten Problemstellen wurden durch die Inspektion einzelner Operatorsequenzen und deren Ausführungs- und Lernzeiten ermittelt. In beiden Fällen wurden Schwächen bzgl. der Aufgabenangemessenheit des interaktiven Systems deutlich, denn es waren zuviele Tastendrücke und Denkpausen für die Interaktionen nötig. Durch zusätzliche empirische Beobachtungen konnten zudem Optimierungen bzgl. der Selbstbeschreibungsfähigkeit und der Steuerbarkeit des Systems vorgeschlagen werden. Es zeigte sich, dass den Probanden teilweise der weitere Verlauf der Bedienerführung unklar war oder, dass sie die Aktivitäten des Systems nicht mehr steuern konnten. Der allgemeine Optimierungsbedarf bezog sich auf die Konsistenz des Systems und damit auf die Usability-Kriterien Erwartungskonformität und Lernförderlichkeit.
Eine kritische Hinterfragung bzgl. der Eignung der eingesetzten Techniken folgte. Dabei wurden diese Techniken sowohl allgemein bzgl. ihres Wertes bei der Usability-Bewertung, als auch speziell hinsichtlich ihrer Eignung für diese Analyse, beurteilt. Die Angaben zu dem unterschiedlichen Aufwand von KLM- und NGOMSL-Analysen bestätigten sich während dieser Arbeit insofern, als dass für erstere ein geringer und für letzere ein deutlich höherer Aufwand nötig war. Unerwartet stark fiel dagegen die Beobachtung hinsichtlich

der Einflussnahme des Analysten auf die Ergebnisse der NGOMSL-Analyse aus. Designentscheidungen des Analysten bzgl. der Zielhierarchie, des Einsatzes mentaler Operatoren oder der Klassifizierung ähnlicher Statements wirkten sich stark auf die berechneten Bearbeitungs- und Lernzeiten des Modells aus. Dadurch wurde aufgezeigt, dass eine qualifizierte NGOMSL-Analyse entsprechende Erfahrung und Professionalität beim Analysten voraussetzt. Denn auch die Bewertung und Einordnung der quantitativen Aussagen erwies sich für einen unerfahrenen Analysten teilweise als schwierig.
Deutlich herausgestellt hat sich bei den Analysen außerdem, dass die Zielhierarchie des NGOMSL-Modells entscheidend dazu beigetragen hat, genaue Problemstellen des Systems zu orten und daraus konkrete Optimierungsvorschläge zu entwickeln. Die Abschätzungen der Modelle erwiesen sich als geeignet, um generelle Tendenzen bzgl. der Effizienz des Systems vorherzusagen. Da die Aussagekraft der KLM-Modelle hauptsächlich auf diesen Abschätzungen basiert, wurde vorgeschlagen, dieses Verfahren eher für Vergleichsschätzungen unterschiedlicher, als für die Bewertung eines Designs einzusetzen. Darüber hinaus erwies sich die KLM-Analyse als gute Grundlage für die Erstellung eines NGOMSL-Modells. Die aufgezeigten Grenzen der verwendeten Techniken rundeten diese Bewertung ab und bildeten die Grundlage des vorgestellten Erweiterungskonzeptes, das im Rahmen dieser Arbeit entwickelt wurde. Dieses Erweiterungskonzept ermöglicht die Einbeziehung von Bedienfehlern in quantitative Abschätzungen von GOMS-Modellen mit Hilfe von empirischen Untersuchungen. So kann die Auswirkung von Bedienfehlern auf die Usability eines Geräts berechnet werden. Darüber hinaus können durch die empirischen Untersuchungen zusätzliche Erkenntnisse, wie die allgemeine Fehlerempfänglichkeit eines Geräts oder die Eignung für bestimmte Nutzergruppen, gewonnen werden. Auch die Identifikation konkreter Fehlerstellen im System ist hiermit denkbar.
Die Anwendung dieses Erweiterungskonzepts könnte zukünftig als hilfreiche Ergänzung zu GOMS-Analysen dienen.

8.2 Ausblick

In Kapitel 6.6 wurde auf die Grenze der GOMS-Techniken hingewiesen, keine Aussagen über das explorierende Verhalten von Nutzern treffen zu können. Auch ist die Modellierung der mentalen Prozesse bei allen GOMS-Modellen sehr oberflächlich. Das Wissen über die genauen Vorgänge, die bei explorierendem Verhalten und Denkvorgängen geschehen, könnte weiteren Aufschluss darüber geben, wie Benutzungsoberflächen gebrauchstauglicher gestaltet werden können, indem die Fähigkeiten und internen Prozesse des Nutzers berücksichtigt werden. Aufbauend auf dieser Arbeit, könnte z.B. in einer Diplomarbeit untersucht werden, inwiefern kognitive Architekturen mit Task-Analyse-Methoden wie

GOMS verknüpft werden können, um präzisere Aussagen hinsichtlich der Usability eines interaktiven Systems treffen zu können.

Ferner wäre es für zukünftige NGOMSL-Analysen denkbar, statt der ursprünglichen NGOMSL-Schreibweise die formalisierte Notation GOMSL (s. Kapitel 3.2.6.2) zu verwenden. Kieras wies darauf hin, dass diese Notation in bestimmten Modellierungsfragen exakter ist und deshalb ebenso dann hilfreich sein kann, wenn zur Ausführung des erstellten GOMSL-Modells nicht das dafür vorgesehene Werkzeug GLEAN verwendet wird[1]. In seinem Leitfaden zur Erstellung von GOMSL-Modellen (Kie06) erklärt er, wie solche GOMSL-Modelle konstruiert werden.

Bei der Vorstellung des Erweiterungskonzeptes wurde bereits darauf hingewiesen, dass zusätzliche empirische Untersuchungen es ermöglichten, exaktere Aussagen über die Fehlerauswirkung hinsichtlich eines speziellen Geräts treffen zu können. Dies betrifft insbesondere die zusätzliche Bearbeitungszeit, die ein Fehler verursacht. Es wäre wünschenswert, hier auf Zeiten zurückgreifen zu können, die nicht speziell auf Text-Editier-Aufgaben ausgelegt sind, sondern vor allem aktuelle Anwendungen mit ausgereiften grafischen Benutzungsoberflächen berücksichtigen. Solche Werte würden dabei helfen, die quantitative Angleichung der mit GOMS berechneten Bearbeitungszeiten präziser vornehmen zu können um daraus qualifiziertere Bewertungen hinsichtlich des untersuchten Geräts abgeben zu können.

Die hier durchgeführten Analysen betreffend, würden weitere Modellierungsaspekte die Genauigkeit der Vorhersagen verbessern bzw. zusätzliche Erkenntnisse über den Fahrkartenautomaten liefern. Da diesen Aspekten aus Zeitgründen nicht nachgegangen werden konnte, werden diese im Folgenden als Ausblick für weitere Arbeiten vorgestellt.

- Die Integration des Fitt's Law (s. Kapitel 3.2.6.1) in die Berechnung der Bearbeitungszeiten erscheint sinnvoll. Durch die Berechnung spezieller Operatorzeiten würden die Gesamt-Bearbeitungszeiten der berechneten Aufgaben an Realitätsnähe zunehmen. Dadurch nähme ebenfalls die Aussagekraft aller GOMS-Modelle zu, insbesondere die der KLM-Modelle (s.o.).
- Die in Kapitel 4.3.2.1 aufgeführten „High-Level"-Operatoren, könnten auf die Keystroke-Ebene heruntergebrochen werden. Dadurch würde ebenfalls der Bezahlvorgang des Ticketkaufs modelliert, was zu weiteren Erkenntnissen bzgl. des Fahrkartenautomaten führen könnte. Im Zuge dessen wäre ebenfalls die Erweiterung des Aufgabenkatalogs und damit verbundene weitere Berechnungen sinnvoll, um die Auswirkungen der neu spezifizierten Operatoren unmittelbar sichtbar zu machen.

[1] Quelle: Briefverkehr mit D. Kieras am 16.9.2008

- Die hier entwickelten Modelle beschränken sich auf die Funktionalität, Fahrkarten zu kaufen. Wie in Kapitel 2.2 erwähnt, stellt der Automat darüber hinaus weitere Funktionen zur Verfügung. Um ein Gesamtbild des Geräts hinsichtlich seiner, mit GOMS messbaren, Gebrauchstauglichkeit zu erhalten, wäre die Modellierung sämtlicher Funktionen notwendig.
- Darüber hinaus fanden die empirischen Untersuchungen in dieser Arbeit nur nebensächlich statt. Die Erfassung weiterer Probandendaten würde, im angemessenen Rahmen, eine repräsentative Validierung der durch die Analyseverfahren berechneten Werte ermöglichen.

Literaturverzeichnis

[BGJK04] Brix, Johanna ; Granilo, Anida ; Jäger, Maresa ; Kapeller, Simone: *Cognitive Walkthrough.* Version: 2004. http://www.evaluationstechniken.de/index.php?page=abgr_04.html, Abruf: 28. Juli 2008

[BH00] Baggen, Robert ; Hemmerling, Sabine: Evaluation und Benutzbarkeit von Mensch-Maschine-Systemen. In: *Mensch-Maschine-Systemtechnik*, Symposion publishing, 2000, S. 233–284

[BN02] Bachem, Guido ; Nettingsmeier, Jörg: *Modelle des Anwenders im Softwaredesign - (2) Kognitive Modelle.* Version: 2002. http://www.collide.info/Lehre/DidaktikSeminar/BuA/page01.php, Abruf: 28. Juli 2008

[Car06] Carnegie Mellon University: *The CogTool Project.* Version: 2006. http://www.cs.cmu.edu/~bej/cogtool/index.html, Abruf: 13. September 2008

[CMN83] Card, Stuart K. ; Moran, Thomas P. ; Newell, Allen: *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, 1983. – ISBN 0–89859–859–1

[DAT08] DATech Deutsche Akkreditierungsstelle Technik: *Leitfaden Usability.* Version: 2008. http://www.datech.de/share/files/Leitfaden-Usability.pdf, Abruf: 29. September 2008

[Dei07] Deil, Manuel: *User Interface Façades: freigestaltbare Benutzeroberflächen.* Version: 2007. http://medien.informatik.uni-ulm.de/lehre/courses/ss07/ProseminarMensch-Computer-Interaktion/deil-proseminar2007.pdf, Abruf: 27. September 2008

[Fra] Fraunhofer Institut Angewandte Informationstechnik: *Usability 1x1.* http://www.fit-fuer-usability.de/1x1/uebersicht.html, Abruf: 6. Juli 2008

[Fra08] Fraunhofer Institut Arbeitswirtschaft und Organisation: *Konzepte für den kundenfreundlichen Fahrscheinautomaten der Zukunft.* Version: 2008. http://www.hci.iao.fraunhofer.de/de/projekte/konzepte_fur_den_

kundenfreundlichen_fahrscheinautomaten_der_zukunft/index.html, Abruf: 28. September 2008

[Gei07] Geis, Thomas: *(Die neue) DIN EN ISO 9241-110.* Version: September 2007. http://www.fit-fuer-usability.de/1x1/knigge/110.html, Abruf: 5. Juli 2008

[Gon93] Gong, Richard J.: *Validating and refining the GOMS model methodology for software user interface design and evaluation.* Version: 1993. http://portal.acm.org/citation.cfm?id=194400, Abruf: 20. Juli 2008

[Gre98] Green, Thos: *Interaction as an action language.* Version: 1998. http://homepage.ntlworld.com/greenery/workStuff/res-inter.html, Abruf: 28. Juli 2008

[Hof05] Hofmann, Britta: *Einführung in die ISO 9241-10.* Version: März 2005. http://www.fit-fuer-usability.de/1x1/knigge/einfuehrung.html, Abruf: 5. Juli 2008

[JG94] John, Bonnie E. ; Gray, Wayne D.: *GOMS Analyses for Parallel Activities.* Version: 1994. http://portal.acm.org/citation.cfm?id=259963.260522, Abruf: 19. Juli 2008

[JK94] John, Bonnie E. ; Kieras, David E.: *The GOMS Family of Analysis Techniques: Tools for Design and Evaluation.* Version: August 1994. ftp://ftp.eecs.umich.edu/people/kieras/GOMS/John-Kieras-TR94.pdf, Abruf: 17. April 2008

[JK96a] John, Bonnie E. ; Kieras, David E.: *The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast.* Version: 1996. ftp://ftp.eecs.umich.edu/people/kieras/GOMS/Compare-GOMS.pdf, Abruf: 12. September 2008

[JK96b] John, Bonnie E. ; Kieras, David E.: *Using GOMS for User Interface Design and Evaluation: Which Technique?* Version: Juni 1996. ftp://ftp.eecs.umich.edu/people/kieras/GOMS/Which-GOMS.pdf, Abruf: 12. September 2008

[Joh08] John, Bonnie E.: *CogTool User Guide.* Version: September 2008. http://www.cs.cmu.edu/~bej/cogtool/CogToolUserGuide-1_0b24.pdf, Abruf: 12. September 2008

[Jun02] Jung, Raphael: *Formale Bewertung der Benutzungskomplexität bildschirmgestützter Informations- und Unterhaltungssysteme im Kraftfahrzeug.* VDI Verlag, 2002. – ISBN 3–18–301122–0

[Kie96] Kieras, David: *A Guide to GOMS Model Usability Evaluation using NGOMSL.* Version: 1996. ftp://ftp.eecs.umich.edu/people/kieras/GOMS/NGOMSL_Guide.pdf, Abruf: 6. August 2008

[Kie01] Kieras, David: *Using the Keystroke-Level Model to Estimate Excecution Times.* Version: 2001. ftp://ftp.eecs.umich.edu/people/kieras/GOMS/KLM.pdf, Abruf: 8. September 2008

[Kie06] Kieras, David: *A Guide to GOMS Model Usability Evaluation using GOMSL and GLEAN4.* Version: 2006. ftp://ftp.eecs.umich.edu/people/kieras/GOMS/GOMSL_Guide.pdf, Abruf: 5. August 2008

[KS89] Kantorowitz, Eliezer ; Sudarsky, Oded: *The Adaptable User Interface.* Version: 1989. http://www.cs.technion.ac.il/~kantor/publications/AdaptableUserInterface89.pdf, Abruf: 27. September 2008

[Kug05] Kugelmeier, Dorothea: *Benutzer-orientierte Gestaltung interaktiver Systeme gemäß der Norm ISO 13407.* Version: März 2005. http://www.fit-fuer-usability.de/1x1/entwicklung/iso.html, Abruf: 5. Juli 2008

[Lin07] Lindstädt, Hagen: *Problemlösen und Verstehen bei ökonomischen Agenten - Eine Gegenüberstellung ökonomischer und kognitionspsychologischer Modelle regelbasierten Entscheidens.* Version: 2007. http://www.ibu.uni-karlsruhe.de/rd_download/SVZ_Problemloesen_und_Verstehen.pdf, Abruf: 29. Juli 2008

[LR94] Lewis, Clayton ; Rieman, John: *TASK-CENTERED USER INTERFACE DESIGN: A Practical Introduction.* Version: 1994. ftp://ftp.cs.colorado.edu/pub/distribs/clewis/HCI-Design-Book/, Abruf: 20. Juli 2008

[Lü05] Lüdtke, Andreas: *Kognitive Analyse Formaler Sicherheitskritischer Steuerungssysteme auf Basis eines integrierten Mensch-Maschine-Modells.* Aka, 2005. – ISBN 3–89838–288–5

[Nie93] Nielsen, Jakob: *Usability Engineering.* Academic Press, 1993. – ISBN 0125184050

[RY01] Ritter, Frank ; Young, Richard: *Embodied Models as Simulated Users: Introduction to this Special Issue on Using Cognitive Models to Improve Interface Design.* Version: 2001. http://www.cs.ucl.ac.uk/staff/r.m.young/publications/01.ijhcs-intro.pdf, Abruf: 5. August 2008

[Tec04] Technische Universität Darmstadt: *GOMSED*. Version: 2004. http://www1.tu-darmstadt.de/fb/fb3/psy/kogpsy/indexgoms.htm, Abruf: 29. September 2008

[Wan02] Wandmacher, Jens: *GOMS-Analysen mit GOMSED*. Version: Mai 2002. http://www1.tu-darmstadt.de/fb/fb3/psy/kogpsy/GOMS-Analysen%20mit%20%20GOMSED.pdf, Abruf: 26. Juli 2008

[Woo00] Wood, Scott D.: *Extending GOMS to Human Error and Applying it to Error-Tolerant Design. Doctoral Dissertation*. Version: 2000. http://portal.acm.org/citation.cfm?id=932797, Abruf: 22. September 2008

Teil I

Anhang: Aufgabenkatalog

Aufgaben

Für alle Aufgaben gilt:

- Es sollen Fahrkarten gekauft werden.
- Treten unerwartete Ereignisse oder Meldungen auf, sollen diese ignoriert werden.
- Am wichtigsten ist Zeit/Datum des Zuges, am zweitwichtigsten der günstigste Preis.
- Ende der Bearbeitung ist unmittelbar vor dem Bezahlvorgang.

1. **Profil: Spontaner Einkaufstrip**
    - 1 Fahrkarte von Oldenburg nach Bremen Hbf
    - Zeit: ab sofort, dann auch den erstmöglichen Zug wählen
    - In der 2. Klasse
    - Ohne Bahncard
    - Nur Nahverkehr (alle Züge außer ICE/EC/IC)
    - Keine Rückfahrt
    - Keine Platzreservierung

2. **Profil: Geschäftsreise**
    - 2 Fahrkarten von Oldenburg nach München-Hauptbahnhof
    - Für den 1.9.08: erster Zug ab 10 Uhr
    - 1. Klasse
    - Mit 2 Bahncards 50 für die 1. Klasse
    - Alle Verkehrsmittel (inkl. ICE)
    - Rückfahrt am 3.9.08: erster Zug ab 18 Uhr
    - Platzreservierung für Hin- und Rückfahrt jeweils wenn möglich: Abteil, Plätze am Fenster, Handybereich; evtl. Abweichungen akzeptieren

3. **Profil: Wochenendtrip**

   - Ein Schönes-Wochenende-Ticket (für max. 5 Personen)
   - Für den nächsten Samstag
   - Mit 3 Fahrrädern

4. **Profil: Geplante Kurzreise von anderem Bahnhof**

   - 1 Fahrkarte von Leer nach Bremen Hbf
   - Für den 1.9.08: erster Zug ab 10 Uhr
   - In der 2. Klasse
   - Ohne Bahncard
   - Nur Nahverkehr (alle Züge außer ICE/EC/IC)
   - Keine Rückfahrt
   - Keine Platzreservierung

5. **Profil: Besuch in Kleinstadt**

   - 1 Fahrkarte von Oldenburg nach Willingen
   - Für den 1.9.08: erster Zug ab 10 Uhr
   - In der 2. Klasse
   - Ohne Bahncard
   - Alle Verkehrsmittel (inkl. ICE)
   - Keine Rückfahrt
   - Platzreservierung wenn möglich: Abteil, Platz am Fenster; evtl. Abweichungen akzeptieren

Teil II

Anhang: KLM-Modelle

Aufgabeninstanz 1: Spontaner Einkaufstrip

Anzeige	**Aktion „Widget“**	**Bemerkungen**
Start	Think for 1.2 s	
Start	Look At „Fahrkarten, Platzreservierungen“	
Start	Move And Tap „Fahrkarten, Platzreservierungen“	
Frame 2	Think for 1.2 s	
Frame 2	Look At „Bremen Hbf“	
Frame 2	Move And Tap „Bremen Hbf“	
Frame 2	Wait for 6.33 s	System ermittelt Verbindungen
Frame 3	Think for 11.27 s	Nutzer entscheidet sich für eine Verbindung
Frame 3	Look At „Verbindung 1“	
Frame 3	Move And Tap „Verbindung 1“	
Frame 3	Wait for 1.33 s	System lädt Bezahlbildschirm
Bezahlen	Think for 1.2 s	Nutzer verifiziert Ticketdetails
Bezahlen		Hier folgt der Bezahlvorgang

Berechnete Bearbeitungszeit: 30,30 Sekunden.[2]

[2]Berechnete Bearbeitungszeit durch „CogTool“: 16,52 Sekunden.

Aufgabeninstanz 2: Geschäftsreise

Anzeige	**Aktion „Widget“**	**Bemerkungen**
Start	Think for 1.2 s	
Start	Look At „Fahrkarten, Platzreservierungen“	
Start	Move And Tap „Fahrkarten, Platzreservierungen“	
Frame 2	Think for 1.2 s	
Frame 2	Look At „M“	
Frame 2	Move And Tap „M“	
Frame 3	Think for 1.2 s	
Frame 3	Look At „München-“	
Frame 3	Move And Tap „München-“	
Frame 4	Think for 1.2 s	
Frame 4	Look At „Hauptbahnhof“	
Frame 4	Move And Tap „Hauptbahnhof“	
Frame 5	Think for 1.2 s	
Frame 5	Look At „Fahrkartenkauf“	
Frame 5	Move And Tap „Fahrkartenkauf“	
Frame 6	Think for 1.2 s	
Frame 6	Look At „mehrere Erwachsene“	
Frame 6	Move And Tap „mehrere Erwachsene“	
Frame 7	Think for 1.2 s	
Frame 7	Look At „2“	
Frame 7	Move And Tap „2“	
Frame 8	Think for 1.2 s	
Frame 8	Look At „2 BahnCards“	
Frame 8	Move And Tap „2 BahnCards“	
Frame 9	Think for 1.2 s	
Frame 9	Look At „keine“	
Frame 9	Move And Tap „keine“	Nutzer hat keine Bahncard 25, 2. Klasse
Frame 10	Think for 1.2 s	
Frame 10	Look At „keine“	
Frame 10	Move And Tap „keine“	Nutzer hat keine Bahncard 50, 1. Klasse
Frame 11	Think for 1.2 s	
Frame 11	Look At „keine“	
Frame 11	Move And Tap „keine“	Nutzer hat keine Bahncard 25, 1. Klasse
Frame 12	Think for 1.2 s	
Frame 12	Look At „2“	
Frame 12	Move And Tap „2“	Nutzer wählt 2 Bahncards 50, 1. Klasse

Fortsetzung auf nächster Seite

Fortsetzung von vorhergehender Seite

Anzeige	Aktion „Widget“	Bemerkungen
Frame 13	Think for 1.2 s	
Frame 13	Look At „1. Klasse“	
Frame 13	Move And Tap „1. Klasse“	
Frame 14	Think for 1.2 s	
Frame 14	Look At „anderes Datum“	
Frame 14	Move And Tap „anderes Datum“	
Frame 15	Think for 1.2 s	
Frame 15	Look At „später“	
Frame 15	Move And Tap „später“	
Frame 16	Think for 1.2 s	
Frame 16	Look At „1“	
Frame 16	Move And Tap „1“	Nutzer wählt 1. September für Hinfahrt
Frame 17	Think for 1.2 s	
Frame 17	Look At „10 Uhr“	
Frame 17	Move And Tap „10 Uhr“	Nutzer wählt 10 Uhr als Abfahrtzeit
Frame 18	Think for 1.2 s	
Frame 18	Look At „alle inkl. ICE“	
Frame 18	Move And Tap „alle inkl. ICE“	Nutzer wählt Zugart
Frame 19	Think for 1.2 s	
Frame 19	Look At „anderes Datum“	
Frame 19	Move And Tap „anderes Datum“	
Frame 20	Think for 1.2 s	
Frame 20	Look At „3“	
Frame 20	Move And Tap „3“	Nutzer wählt 3. September für Rückfahrt
Frame 21	Think for 1.2 s	
Frame 21	Look At „18 Uhr“	
Frame 21	Move And Tap „18 Uhr“	Nutzer wählt 18 Uhr als Abfahrtzeit
Frame 22	Think for 1.2 s	
Frame 22	Look At „alle inkl. ICE“	
Frame 22	Move And Tap „alle inkl. ICE“	Nutzer wählt Zugart
Frame 23	Think for 1.2 s	
Frame 23	Look At „Verbindungen/Fahrkarte/ Reservierungen“	
Frame 23	Move And Tap „Verbindungen/Fahrkarte/ Reservierungen“	
Frame 23	Wait for 7.58 s	System ermittelt Verbindungen

Fortsetzung auf nächster Seite

Fortsetzung von vorhergehender Seite

Anzeige	Aktion „Widget“	Bemerkungen
Frame 24	Think for 11.27 s	Nutzer entscheidet sich für eine Verbindung
Frame 24	Look At „Verbindung 2“	
Frame 24	Move And Tap „Verbindung 2“	
Frame 24	Wait for 3.71 s	System lädt Verbindungsdetails
Frame 25	Think for 1.2 s	
Frame 25	Look At „Platz reservieren“	
Frame 25	Move And Tap „Platz reservieren“	Nutzer wählt Sitzplatzreservierung
Frame 26	Think for 1.2 s	
Frame 26	Look At „Abteil“	
Frame 26	Move And Tap „Abteil“	Nutzer wählt Abteil
Frame 27	Think for 1.2 s	
Frame 27	Look At „2 Plätze am Fenster“	
Frame 27	Move And Tap „2 Plätze am Fenster“	Nutzer wählt Fensterplätze
Frame 28	Think for 1.2 s	
Frame 28	Look At „Handy-Bereich“	
Frame 28	Move And Tap „Handy-Bereich“	Nutzer wählt Handybereich im ICE
Frame 28	Wait for 3.13 s	System führt Reservierung durch
Frame 29	Think for 6.4 s	Nutzer denkt über Abweichungen nach
Frame 29	Look At „Abweichung akzeptieren“	
Frame 29	Move And Tap „Abweichung akzeptieren“	Nutzer akzeptiert Abweichungen
Frame 30	Think for 1.2 s	
Frame 30	Look At „Erwünscht“	
Frame 30	Move And Tap „Erwünscht“	Nutzer wählt Reservierung für Rückfahrt
Frame 30	Wait for 1.82 s	System führt Reservierung durch
Frame 31	Think for 6.4 s	Nutzer denkt über Abweichungen nach
Frame 31	Look At „Abweichung akzeptieren“	
Frame 31	Move And Tap „Abweichung akzeptieren“	Nutzer akzeptiert Abweichungen
Frame 32	Think for 1.2 s	
Frame 32	Look At „Nein“	
Frame 32	Move And Tap „Nein“	Nutzer lehnt Bonusprogramm ab

Fortsetzung auf nächster Seite

Fortsetzung von vorhergehender Seite

Anzeige	**Aktion „Widget“**	**Bemerkungen**
Frame 33	Think for 1.2 s	
Frame 33	Look At „Nein“	
Frame 33	Move And Tap „Nein“	Nutzer lehnt Versicherung ab
Frame 33	Wait for 2.13 s	System lädt Bezahlbildschirm
Bezahlen		Hier folgt der Bezahlvorgang

Berechnete Bearbeitungszeit: 162,10 Sekunden.[3]

[3]Berechnete Bearbeitungszeit durch „CogTool“: 81,58 Sekunden.

Aufgabeninstanz 3: Wochenendtrip

Anzeige	**Aktion „Widget“**	**Bemerkungen**
Start	Think for 1.2 s	
Start	Look At „Spezial Angebote“	
Start	Move And Tap „Spezial Angebote“	
Frame 2	Think for 1.2 s	
Frame 2	Look At „Schönes-Wochenende-Ticket“	Nutzer wählt Wochenend-Ticket
Frame 2	Move And Tap „Schönes-Wochenende-Ticket“	
Frame 3	Think for 1.2 s	
Frame 3	Look At „Schönes-Wochenende-Ticket“	
Frame 3	Move And Tap „Schönes-Wochenende-Ticket“	Nutzer bestätigt Wochenend-Ticket
Frame 4	Think for 1.2 s	
Frame 4	Look At „Samstag“	
Frame 4	Move And Tap „Samstag“	Nutzer wählt Samstag
Frame 5	Think for 1.2 s	
Frame 5	Look At „Mit Fahrradkarte“	
Frame 5	Move And Tap „Mit Fahrradkarte“	Nutzer wählt Fahrradkarte
Frame 6	Think for 1.2 s	
Frame 6	Look At „3“	
Frame 6	Move And Tap „3“	Nutzer wählt 3 Fahrräder
Bezahlen		Hier folgt der Bezahlvorgang

Berechnete Bearbeitungszeit: 22,20 Sekunden.[4]

[4]Berechnete Bearbeitungszeit durch „CogTool“: 10,42 Sekunden.

Aufgabeninstanz 4: Geplante Kurzreise von anderem Bahnhof

Anzeige	**Aktion „Widget“**	**Bemerkungen**
Start	Think for 1.2 s	
Start	Look At „Fahrkarten, Platzreservierungen“	
Start	Move And Tap „Fahrkarten, Platzreservierungen“	
Frame 2	Think for 1.2 s	
Frame 2	Look At „Start“	
Frame 2	Move And Tap „Start“	
Frame 3	Think for 1.2 s	
Frame 3	Look At „L“	
Frame 3	Move And Tap „L“	
Frame 4	Think for 1.2 s	
Frame 4	Look At „E“	
Frame 4	Move And Tap „E“	
Frame 5	Think for 1.2 s	
Frame 5	Look At „E“	
Frame 5	Move And Tap „E“	
Frame 6	Think for 1.2 s	
Frame 6	Look At „Leer (Ostfriesland)“	
Frame 6	Move And Tap „Leer (Ostfriesland)“	
Frame 7	Think for 1.2 s	
Frame 7	Look At „Bremen“	
Frame 7	Move And Tap „Bremen“	
Frame 8	Think for 1.2 s	
Frame 8	Look At „Fahrkartenkauf“	
Frame 8	Move And Tap „Fahrkartenkauf“	
Frame 9	Think for 1.2 s	
Frame 9	Look At „1 Erwachsener“	
Frame 9	Move And Tap „1 Erwachsener“	
Frame 10	Think for 1.2 s	
Frame 10	Look At „keine BahnCard“	
Frame 10	Move And Tap „keine BahnCard“	
Frame 11	Think for 1.2 s	
Frame 11	Look At „2. Klasse“	
Frame 11	Move And Tap „2. Klasse“	
Frame 12	Think for 1.2 s	
Frame 12	Look At „anderes Datum“	
Frame 12	Move And Tap „anderes Datum“	
Frame 13	Think for 1.2 s	
Frame 13	Look At „später“	
Frame 13	Move And Tap „später“	
Frame 13	Wait for 1.16 s	System lädt

Fortsetzung auf nächster Seite

Fortsetzung von vorhergehender Seite

Anzeige	Aktion „Widget“	Bemerkungen
Frame 14	Think for 1.2 s	
Frame 14	Look At „1“	
Frame 14	Move And Tap „1“	
Frame 15	Think for 1.2 s	
Frame 15	Look At „10 Uhr“	
Frame 15	Move And Tap „10 Uhr“	
Frame 16	Think for 1.2 s	
Frame 16	Look At „nur Nahverkehr“	
Frame 16	Move And Tap „nur Nahverkehr“	
Frame 17	Think for 1.2 s	
Frame 17	Look At „keine“	
Frame 17	Move And Tap „keine“	
Frame 18	Think for 1.2 s	
Frame 18	Look At „Verbindungen/Fahrkarte/ Reservierungen“	
Frame 18	Move And Tap „Verbindungen/Fahrkarte/ Reservierungen“	
Frame 18	Wait for 7.58 s	System ermittelt Verbindungen
Frame 19	Think for 11.27 s	Nutzer entscheidet sich für eine Verbindung
Frame 19	Look At „Verbindung 2“	
Frame 19	Move And Tap „Verbindung 2“	
Frame 19	Wait for 3.71 s	System lädt Verbindungsdetails
Frame 20	Think for 1.2 s	
Frame 20	Look At „Weiter zum Fahrkartenkauf“	
Frame 20	Move And Tap „Weiter zum Fahrkartenkauf“	
Frame 21	Think for 1.2 s	
Frame 21	Look At „Ohne CityMobil“	
Frame 21	Move And Tap „Ohne CityMobil“	Nutzer lehnt Zusatzangebot ab
Frame 21	Wait for 3.53 s	System lädt Bezahlbildschirm
Bezahlen		Hier folgt der Bezahlvorgang

Berechnete Bearbeitungszeit: 103,75 Sekunden.[5]

[5]Berechnete Bearbeitungszeit durch „CogTool“: 50,61 Sekunden.

Aufgabeninstanz 5: Besuch in Kleinstadt

Anzeige	**Aktion „Widget“**	**Bemerkungen**
Start	Think for 1.2 s	
Start	Look At „Fahrkarten, Platzreservierungen“	
Start	Move And Tap „Fahrkarten, Platzreservierungen“	
Frame 2	Think for 1.2 s	
Frame 2	Look At „W“	
Frame 2	Move And Tap „W“	
Frame 3	Think for 1.2 s	
Frame 3	Look At „I“	
Frame 3	Move And Tap „I“	
Frame 4	Think for 1.2 s	
Frame 4	Look At „L“	
Frame 4	Move And Tap „L“	
Frame 5	Think for 1.2 s	
Frame 5	Look At „L“	
Frame 5	Move And Tap „L“	
Frame 6	Think for 1.2 s	
Frame 6	Look At „Willingen“	
Frame 6	Move And Tap „Willingen“	
Frame 7	Think for 1.2 s	
Frame 7	Look At „Fahrkartenkauf“	
Frame 7	Move And Tap „Fahrkartenkauf“	
Frame 8	Think for 1.2 s	
Frame 8	Look At „1 Erwachsener“	
Frame 8	Move And Tap „1 Erwachsener“	
Frame 9	Think for 1.2 s	
Frame 9	Look At „keine BahnCard“	
Frame 9	Move And Tap „keine BahnCard“	
Frame 10	Think for 1.2 s	
Frame 10	Look At „2. Klasse“	
Frame 10	Move And Tap „2. Klasse“	
Frame 11	Think for 1.2 s	
Frame 11	Look At „anderes Datum“	
Frame 11	Move And Tap „anderes Datum“	
Frame 12	Think for 1.2 s	
Frame 12	Look At „später“	
Frame 12	Move And Tap „später“	
Frame 12	Wait for 1.16 s	System lädt
Frame 13	Think for 1.2 s	

Fortsetzung auf nächster Seite

Fortsetzung von vorhergehender Seite

Anzeige	Aktion „Widget“	Bemerkungen
Frame 13	Look At „1“	
Frame 13	Move And Tap „1“	
Frame 14	Think for 1.2 s	
Frame 14	Look At „10 Uhr“	
Frame 14	Move And Tap „10 Uhr“	
Frame 15	Think for 1.2 s	
Frame 15	Look At „alle“	
Frame 15	Move And Tap „alle“	
Frame 16	Think for 1.2 s	
Frame 16	Look At „Verbindungen/Fahrkarte/ Reservierungen“	
Frame 16	Move And Tap „Verbindungen/Fahrkarte/ Reservierungen“	
Frame 16	Wait for 7.58 s	System ermittelt Verbindungen
Frame 17	Think for 11.27 s	Nutzer entscheidet sich für eine Verbindung
Frame 17	Look At „Verbindung 2“	
Frame 17	Move And Tap „Verbindung 2“	
Frame 17	Wait for 3.71 s	System lädt Verbindungsdetails
Frame 18	Think for 14.0 s	Nutzer entscheidet sich für ein Angebot
Frame 18	Look At „Dauer-Spezial“	
Frame 18	Move And Tap „Dauer-Spezial“	Nutzer wählt Dauer-Spezial
Frame 19	Think for 1.2 s	
Frame 19	Look At „Platz reservieren“	
Frame 19	Move And Tap „Platz reservieren“	Nutzer wählt Sitzplatzreservierung
Frame 20	Think for 1.2 s	
Frame 20	Look At „Abteil“	
Frame 20	Move And Tap „Abteil“	Nutzer wählt Abteil
Frame 21	Think for 1.2 s	
Frame 21	Look At „Platz am Fenster“	
Frame 21	Move And Tap „Platz am Fenster“	Nutzer wählt Fensterplatz
Frame 21	Wait for 3.13 s	System führt Reservierung durch
Frame 22	Think for 1.2 s	
Frame 22	Look At „Platzres. bestätigen“	
Frame 22	Move And Tap „Platzres. bestätigen“	Nutzer bestätigt Reservierung

Fortsetzung auf nächster Seite

Fortsetzung von vorhergehender Seite

Anzeige	**Aktion „Widget“**	**Bemerkungen**
Frame 22	Wait for 2.13 s	System lädt Bezahlbildschirm
Bezahlen		Hier folgt der Bezahlvorgang

Berechnete Bearbeitungszeit: 116,15 Sekunden.[6]

[6]Berechnete Bearbeitungszeit durch „CogTool“: 66,58 Sekunden.

Annahmen über die KLM-Modelle

- Der Nutzer kennt die korrekte Schreibweise der Verbindungsorte.
- Der Standort der Automaten ist Oldenburg (Oldb).
- Aufgabeninstanz eins gilt für den Fall, dass eine Verbindung im „Verbundtarif“ ausgewählt wird.
- Der Monat September war bei der Datumsauswahl Hinfahrt noch nicht sichtbar, weil die Tests bereits im Juli stattfanden.
- Der Nutzer akzeptiert jegliche Reservierungsabweichungen.
- Der Nutzer möchte keine Zusatzangebote wahrnehmen.

Teil III

Anhang: NGOMSL-Modell

Lesehinweis: Die einzelnen Methoden des folgenden NGOMSL-Modells sind hierarchisch nach ihren Aufruf-Ebenen geordnet. D.h. es wird auf der obersten Zielebene mit dem Top-Level-Ziel „kaufe Fahrkarten“ begonnen. Dann folgen alle Methoden der nächsten Aufruf-Ebene, geordnet nach ihrer Aufrufreihenfolge innerhalb der einzelnen Methoden. Dies setzt sich fort bis zur untersten Ebene des Modells.

Method for goal: kaufe Fahrkarten.
Step 1. Retrieve from LTM the value for $< ticketType >$, and recall it from WM.
Step 2. Accomplish goal: wähle Fahrkartenkauf.
Step 3. Return with goal accomplished.

Selection Rule set for goal: wähle Fahrkartenkauf.
If $< ticketType >$ is „normal“, Then Accomplish goal: führe normalen Fahrkartenkauf durch.
If $< ticketType >$ is „wochenend“, Then Accomplish goal: kaufe Wochenendticket.
If $< ticketType >$ is „verbund“, Then Accomplish goal: führe schnellen Verbund-Fahrkartenkauf durch.
Return with goal accomplished.

Method for goal: führe normalen Fahrkartenkauf durch.
Step 1. Accomplish goal: gebe Grunddaten ein.
Step 2. Accomplish goal: gebe Zusatzdaten ein.
Step 3. Accomplish goal: gebe Daten und Verkehrsmittel ein.
Step 4. Accomplish goal: wähle Angebot und Platzreservierung.
Step 5. Return with goal accomplished.

Method for goal: kaufe Wochenendticket.
Step 1. Accomplish goal: wähle Wochenendticket.
Step 2. Accomplish goal: gebe Daten des Wochenendtickets ein.
Step 3. Return with goal accomplished.

Method for goal: führe schnellen Verbund-Fahrkartenkauf durch.
Step 1. Accomplish goal: initialisiere Fahrkartenkauf.
Step 2. Accomplish goal: gebe Ziel ein.
Step 3. Wait for system for 6,33s
Step 4. Accomplish goal: wähle Verbindung.
Step 5. Accomplish goal: verifiziere Ticketdetails.
Step 6. Accomplish goal: bezahle Tickets.
Step 7. Return with goal accomplished.

Method for goal: gebe Grunddaten ein.
Step 1. Accomplish goal: initialisiere Fahrkartenkauf.
Step 2. Retrieve from LTM the value for <start>, and recall it from WM.
Step 3. Decide: If $< start >$ is „Oldenburg Olb“, Then forget $< start >$ and Goto Step 4.
Else Accomplish goal: ändere Startbahnhof.
Step 4. Accomplish goal: gebe Ziel ein.
Step 5. Accomplish goal: initialisiere Fahrkartenkauf.
Step 6. Return with goal accomplished.

Method for goal: gebe Zusatzdaten ein.
Step 1. Accomplish goal: wähle Reisende.
Step 2. Accomplish goal: wähle Bahncards.
Step 3. Accomplish goal: wähle Klasse.
Step 4. Return with goal accomplished.

Method for goal: gebe Daten und Verkehrsmittel ein.
Step 1. Accomplish goal: wähle Datum Hinfahrt.
Step 2. Accomplish goal: wähle Verkehrsmittel Hinfahrt.
Step 3. Accomplish goal: wähle Datum Rückfahrt.
Step 4. Decide: If $< tagR >$ is „kein“, Then Goto Step 5.
Else Accomplish goal: wähle Verkehrsmittel Rückfahrt.
Step 5. Accomplish goal: ermittle Verbindungen.
Step 6. Return with goal accomplished.

Method for goal: wähle Angebot und Platzreservierung.
Step 1. Accomplish goal: wähle Verbindung.
Step 2. Wait for system for 2,38s
Step 3. Accomplish goal: prüfe Existenz von Preis-Angebot.
Step 4. Accomplish goal: prüfe Möglichkeit einer Reservierung.
Step 5. Decide: If $< reservierung >$ is „keine“, Then Goto Step 6.
If $< tagR >$ is „kein“, Then forget $< tagR >$ and Goto Step 6.
Else Accomplish goal: wähle Sitzplatzreservierung Rückfahrt.
Step 6. Accomplish goal: wähle Zusatzangebote.
Step 7. Accomplish goal: bezahle Tickets.
Step 8. Return with goal accomplished.

Method for goal: wähle Wochenendticket.
Step 1. Accomplish goal: initialisiere Fahrkartenkauf.
Step 2. Accomplish goal: drücke button „Schönes-Wochenende-Ticket“.
Step 3. Accomplish goal: drücke button „Schönes-Wochenende-Ticket“.
Step 4. Return with goal accomplished.

Method for goal: gebe Daten des Wochenendtickets ein.
Step 1. Retrieve from LTM the value for $< tagGueltig >$, and recall it from WM.
Step 2. Decide: If $< tagGueltig >$ is „samstag“, Then Accomplish goal: drücke button „Samstag“.
If $< tagGueltig >$ is „sonntag“, Then Accomplish goal: drücke button „Sonntag“.
Else choose another date.
Step 3. Forget $< tagGueltig >$ and Accomplish goal: spezifiziere Fahrradauswahl.
Step 4. Accomplish goal: bezahle Tickets.
Step 5. Return with goal accomplished.

Method for goal: initialisiere Fahrkartenkauf.

Step 1. Decide: If $< ticketType >$ is „normal“ and button „Fahrkarten Platzreservierungen“ is visible, Then Accomplish goal: drücke button „Fahrkarten Platzreservierungen“.
If $< ticketType >$ is „normal“ and button „Fahrkartenkauf (Schritt für Schritt)“ is visible, Then Accomplish goal: drücke button „Fahrkartenkauf (Schritt für Schritt)“.
If $< ticketType >$ is „wochenend“, Then Accomplish goal: drücke button „Spezial-Angebote“.
If $< ticketType >$ is „verbund“, Then Accomplish goal: drücke button „Fahrkarten Platzreservierungen“.

Step 2. Forget $< ticketType >$ and Return with goal accomplished.

Method for goal: gebe Ziel ein.

Step 1. Retrieve from LTM the value for $< ziel >$, and recall it from WM.
Step 2. Accomplish goal: treffe Auswahl für Zieleingabe.
Step 3. Forget $< ziel >$ and Return with goal accomplished.

Method for goal: wähle Verbindung.

Step 1. Read Verbindungen value from the screen and retain it in WM.
Step 2. Recall Verbindungen value from WM and Make up your mind about the Verbindung for 10,07s and retain decision in WM.
Step 3. Recall decision from WM and Accomplish goal: drücke button $< verbindung >$.
Step 4. Wait for system for 1,33s
Step 5. Forget Verbindungen value, forget decision and Return with goal accomplished.

Method for goal: verifiziere Ticketdetails.

Step 1. Read Ticketdetails value from the screen and retain it in WM.
Step 2. Retrieve from LTM the value for $< start >$ and $< klasse >$ and $< anzahlReisende >$, and recall them from WM.
Step 3. Verify that Ticketdetails $< start >< ziel >< klasse >< anzahlReisende >$ are correct.
Step 4. Forget Ticketdetails value, forget $< ziel >$, forget $< start >$, forget $< klasse >$, forget $< anzahlReisende >$ and Return with goal accomplished.

Method for goal: bezahle Tickets.

Step 1. Pay Tickets.
Step 2. Return with goal accomplished.

Method for goal: ändere Startbahnhof.

Step 1. Accomplish goal: drücke button „Start Oldenburg Olb“.
Step 2. Accomplish goal: treffe Auswahl für Starteingabe.
Step 3. Forget $< start >$ and Return with goal accomplished.

Method for goal: wähle Reisende.

Step 1. Retrieve from LTM the value for $< anzahlReisende >$, and recall it from WM.
Step 2. Accomplish goal: treffe Auswahl über Anzahl Reisender.
Step 3. Forget $< anzahlReisende >$ and Return with goal accomplished.

Method for goal: wähle Bahncards.

Step 1. Retrieve from LTM the value for $< anzahlBahncards >$, and recall it from WM.

Step 2. Decide: If $< anzahlBahncards >$ is 0, Then Accomplish goal: drücke button „Keine“.
Else Accomplish goal: spezifiziere Bahncards.

Step 3. Forget $< anzahlBahncards >$ and Return with goal accomplished.

Method for goal: wähle Klasse.

Step 1. Retrieve from LTM the value for $< klasse >$, and recall it from WM.

Step 2. Accomplish goal: drücke button $< klasse >$.

Step 3. Forget $< klasse >$ and Return with goal accomplished.

Method for goal: wähle Datum Hinfahrt.

Step 1. Retrieve from LTM the value for $< tagH >$, and recall it from WM.

Step 2. Retrieve from LTM the value for $< zeitH >$, and recall it from WM.

Step 3. Decide: If $< tagH >$ is „heute“, Then Accomplish goal: drücke button „heute“.
If $< tagH >$ is „morgen“, Then Accomplish goal: drücke button „morgen“.
If $< zeitH >$ is „ab sofort“, Then Accomplish goal: drücke button „ab sofort“.
Else Accomplish goal: wähle anderes Hinfahrt-Datum.

Step 4. Forget <zeitH> and Return with goal accomplished.

Method for goal: wähle Verkehrsmittel Hinfahrt.

Step 1. Retrieve from LTM the value for $< zugH >$, and recall it from WM.

Step 2. Decide: If $< zugH >$ is „alle“, Then Accomplish goal: drücke button „alle inkl. ICE“.
If $< zugH >$ is „alle außer ICE“, Then Accomplish goal: drücke button „alle außer ICE“.
If $< zugH >$ is „nur Nahverkehr“, Then Accomplish goal: drücke button „alle außer ICE/EC/IC“.

Step 3. Forget $< zugH >$ and Return with goal accomplished.

Method for goal: wähle Datum Rückfahrt.

Step 1. Retrieve from LTM the value for $< tagR >$, and recall it from WM.

Step 2. Decide: If $< tagR >$ is „kein“, Then Accomplish goal: drücke button „Keine Rückfahrt“.
If $< tagR >$ is $< tagH >$, Then Accomplish goal: drücke button „am Tag der Hinfahrt“.
Else Accomplish goal: wähle anderes Rückfahrt-Datum.

Step 3. Forget $< tagH >$ and Return with goal accomplished.

Method for goal: wähle Verkehrsmittel Rückfahrt.

Step 1. Retrieve from LTM the value for $< zugR >$, and recall it from WM.

Step 2. Decide: If $< zugR >$ is „alle“, Then Accomplish goal: drücke button „alle inkl. ICE“.
If $< zugR >$ is „alle außer ICE“, Then Accomplish goal: drücke button „alle außer ICE“.
If $< zugR >$ is „nur Nahverkehr“, Then Accomplish goal: drücke button „alle außer ICE/EC/IC“.

Step 3. Forget $< zugR >$ and Return with goal accomplished.

Method for goal: ermittle Verbindungen.
Step 1. Accomplish goal: drücke button „Verbindungen/Fahrkarte/Reservierung“.
Step 2. Wait for system for 7,58s
Step 3. Return with goal accomplished.

Selection rule set for goal: prüfe Existenz von Preis-Angebot.
If Preis-Angebot value is visible, Then Accomplish goal: wähle Preis-Angebot.
Else Accomplish goal: prüfe Möglichkeit einer Reservierung.
Return with goal accomplished.

Selection rule set for goal: prüfe Möglichkeit einer Reservierung.
If button „Weiter zum Fahrkartenkauf“ is visible, Then Accomplish goal:
führe Kauf fort, wenn Platzreservierung nicht möglich and return with goal accomplished.
Else Accomplish goal: wähle Sitzplatzreservierung Hinfahrt.
Return with goal accomplished.

Method for goal: wähle Sitzplatzreservierung Rückfahrt.
Step 1. Retrieve from LTM the value for $< reservierungR >$, and recall it from WM.
Step 2. Accomplish goal: treffe Auswahl über Rückfahrt.
Step 3. Forget $< reservierung >$, forget $< reservierungR >$ and Return with goal accomplished.

Method for goal: wähle Zusatzangebote.
Step 1. Read Angebot value from the screen and retain it in WM.
Step 2. Recall Angebot value from WM and Decide: If Angebot Reiseschutz-Versicherung is visible, Then Accomplish goal: behandle Versicherung.
If Angebot bahn.comfort und bahn.bonus is visible, Then Accomplish goal: behandle Bonusprogramm.
If Angebot CityMobil is visible, Then Accomplish goal: behandle Angebot CityMobil.
Else wait for system for 2,13s
Step 3. Forget Angebot value and Return with goal accomplished.

Method for goal: drücke button $< buttonName >$.
Step 1. Locate object $< buttonName >$ on screen.
Step 2. Point to $< buttonName >$.
Step 3. Tap on $< buttonName >$.
Step 4. Return with goal accomplished.

Method for goal: spezifiziere Fahrradauswahl.
Step 1. Retrieve from LTM the value for $< fahrradAnzahl >$, and recall it from WM.
Step 2. Decide: If $< fahrradAnzahl >$ is 0, Then Accomplish goal: drücke button „Keine Fahrradkarte“ and Goto Step 5.
Else Goto Step 3.
Step 3. Accomplish goal: drücke button „Mit Fahrradkarte“.
Step 4. Accomplish goal: drücke button $< fahrradAnzahl >$.
Step 5. Forget $< fahrradAnzahl >$ and Return with goal accomplished.

Selection rule set for goal: treffe Auswahl für Zieleingabe.
If $< ziel >$ is visible on a button on the left, Then
Accomplish goal: gebe Ziel über Button ein.
If $< ziel >$ is not visible on a button on the left, Then Accomplish goal: gebe Ziel über Tastatur ein.
If $< ziel >$ is input completely, Then Return with goal accomplished.
Return with goal accomplished.

Selection rule set for goal: treffe Auswahl für Starteingabe.
If $< start >$ is visible on a button on the left, Then Accomplish goal: gebe Start über Button ein.
If $< start >$ is not visible on a button on the left, Then
Accomplish goal: gebe Start über Tastatur ein.
If $< start >$ is input completely, Then Return with goal accomplished.
Return with goal accomplished.

Selection rule set for goal: treffe Auswahl über Anzahl Reisender.
If $< anzahlReisende >$ is > 1, Then Accomplish goal: wähle Anzahl der Reisenden aus.
Else Accomplish goal: drücke button „1 Erwachsener“.
Return with goal accomplished.

Method for goal: spezifiziere Bahncards.
Step 1. Accomplish goal: drücke button $< anzahlBahncards >$.
Step 2. Retrieve from LTM the value for $< typBahncards >$, and recall it from WM.
Step 3. Decide: If $< typBahncards >$ is 25_2, Then Accomplish goal: drücke button $< anzahlBahncards >$.
Else Accomplish goal: drücke button „Keine“.
Step 4. Decide: If $< typBahncards >$ is 50_2, Then Accomplish goal: drücke button $< anzahlBahncards >$.
Else Accomplish goal: drücke button „Keine“.
Step 5. Decide: If $< typBahncards >$ is 25_1, Then Accomplish goal: drücke button $< anzahlBahncards >$.
Else Accomplish goal: drücke button „Keine“.
Step 6. Decide: If $< typBahncards >$ is 50_1, Then Accomplish goal: drücke button $< anzahlBahncards >$.
Else Accomplish goal: drücke button „Keine“.
Step 7. Forget $< typBahncards >$ and Return with goal accomplished.

Method for goal: wähle anderes Hinfahrt-Datum.
Step 1. Retrieve from LTM the value for $< monatH >$, and recall it from WM.
Step 2. Accomplish goal: drücke button „anderes Datum“.
Step 3. Decide: If $< tagH >$ and $< monatH >$ is visible, Then Accomplish goal: drücke button $< tagH >$.
Else Accomplish goal: drücke button „späteres Datum“ and wait for system for 1,16s and Goto Step 3.
Step 4. Accomplish goal: drücke button $< zeitH >$.
Step 5. Forget $< monatH >$ and Return with goal accomplished.

Method for goal: wähle anderes Rückfahrt-Datum.

Step 1. Retrieve from LTM the value for $< zeitR >$, and recall it from WM.
Step 2. Retrieve from LTM the value for $< monatR >$, and recall it from WM.
Step 3. Accomplish goal: drücke button „anderes Datum“.
Step 4. Decide: If $< tagR >$ and $< monatR >$ is visible, Then Accomplish goal: drücke button $< tagR >$.
Else Accomplish goal: drücke button „späteres Datum“ and wait for system for 1,16s and Goto Step 4.
Step 5. Accomplish goal: drücke button $< zeitR >$.
Step 6. Forget $< zeitR >$, forget $< monatR >$ and Return with goal accomplished.

Method for goal: wähle Preis-Angebot.

Step 1. Read Preis-Angebot value from the screen and retain it in WM.
Step 2. Recall Preis-Angebot value from WM and Make up your mind about the Preis-Angebot for 12,8s and retain decision in WM.
Step 3. Recall decision from WM and Accomplish goal: drücke button $< angebot >$.
Step 4. Forget Preis-Angebot value, forget decision and Return with goal accomplished.

Method for goal: führe Kauf fort, wenn Platzreservierung nicht möglich.

Step 1. Accomplish goal: drücke button „Weiter zum Fahrkartenkauf“.
Step 2. Accomplish goal: wähle Zusatzangebote.
Step 3. Accomplish goal: bezahle Tickets.
Step 4. Return with goal accomplished.

Method for goal: wähle Sitzplatzreservierung Hinfahrt.

Step 1. Retrieve from LTM the value for $< reservierung >$, and recall it from WM.
Step 2. Decide: If $< reservierung >$ is „keine“, Then Accomplish goal: drücke button „keine Platzreservierung (Hin- und Rückfahrt)“.
If $< reservierung >$ is „keineHin“, Then Accomplish goal: drücke button „keine Platzreservierung für die Hinfahrt“.
Else Accomplish goal: spezifiziere Reservierung.
Step 3. Return with goal accomplished.

Selection rule set for goal: treffe Auswahl über Rückfahrt.

If $< reservierungR >$ is „nein“, Then Accomplish goal: drücke button „Keine Platzres. Rückfahrt“.
If $< reservierungR >$ is „ja“, Then Accomplish goal: bestätige Platzreservierung der Rückfahrt.

Return with goal accomplished.

Method for goal: behandle Versicherung.

Step 1. Decide: If you want to contract insurance, Then Accomplish goal: drücke button „Ja“.
Else Accomplish goal: drücke button „Nein“.
Step 2. Wait for system for 2,13s
Step 3. Return with goal accomplished.

Method for goal: behandle Bonusprogramm.
Step 1. Decide: If you want to collect bonuses with your Bahncard, Then Accomplish goal: drücke button „Ja“ and insert your Bahncard.
Else Accomplish goal: drücke button „Nein“.
Step 2. Accomplish goal: wähle Zusatzangebote.
Step 3. Return with goal accomplished.

Method for goal: behandle Angebot CityMobil.
Step 1. Decide: If you want to have CityMobil, Then Accomplish goal: drücke button „Mit CityMobil“.
Else Accomplish goal: drücke button „Ohne CityMobil“.
Step 2. Wait for system for 3,53s
Step 3. Return with goal accomplished.

Method for goal: gebe Ziel über Button ein.
Step 1. Accomplish goal: drücke button $< ziel >$.
Step 2. Accomplish goal: treffe Auswahl für Zieleingabe.
Step 3. Return with goal accomplished.

Method for goal: gebe Ziel über Tastatur ein.
Step 1. Accomplish goal: drücke button $< Buchstabe >$.
Step 2. Accomplish goal: treffe Auswahl für Zieleingabe.
Step 3. Return with goal accomplished.

Method for goal: gebe Start über Button ein.
Step 1. Accomplish goal: drücke button $< start >$.
Step 2. Accomplish goal: treffe Auswahl für Starteingabe.
Step 3. Return with goal accomplished.

Method for goal: gebe Start über Tastatur ein.
Step 1. Accomplish goal: drücke button $< Buchstabe >$.
Step 2. Accomplish goal: treffe Auswahl für Starteingabe.
Step 3. Return with goal accomplished.

Method for goal: wähle Anzahl der Reisenden aus.
Step 1. Accomplish goal: drücke button „Mehrere Erwachsene“.
Step 2. Accomplish goal: drücke button $< anzahlReisende >$.
Step 3. Return with goal accomplished.

Method for goal: spezifiziere Reservierung.
Step 1. Accomplish goal: wähle Reservierungsraum.
Step 2. Accomplish goal: wähle Reservierungsplatz und -bereich.
Step 3. Accomplish goal: schließe Reservierung ab.
Step 4. Return with goal accomplished.

Method for goal: bestätige Platzreservierung der Rückfahrt.

Step 1. Accomplish goal: drücke button „Platzreservierung erwünscht“.
Step 2. Wait for system for 1,82s
Step 3. Accomplish goal: prüfe Existenz von Abweichungen.
Step 4. Return with goal accomplished.

Method for goal: wähle Reservierungsraum.

Step 1. Accomplish goal: drücke button „Platz reservieren“.
Step 2. Retrieve from LTM the value for $< reserv\_raum >$, and recall it from WM.
Step 3. Decide: If $< reserv\_raum >$ is „Großraum“, Then Accomplish goal: drücke button „Großraum“.
If $< reserv\_raum >$ is „Großraum am Tisch“, Then Accomplish goal: drücke button „Großraum am Tisch“.
If $< reserv\_raum >$ is „Kleinkindabteil“, Then Accomplish goal: drücke button „Kleinkindabteil“.
Else Accomplish goal: drücke button „Abteil“.
Step 4. Return with goal accomplished.

Method for goal: wähle Reservierungsplatz und -bereich.

Step 1. Retrieve from LTM the value for $< reserv\_platz >$, and recall it from WM.
Step 2. Decide: If $< reserv\_platz >$ is „Fenster“, Then Accomplish goal: drücke button „Platz am Fenster“.
Else Accomplish goal: drücke button „Platz am Gang“.
Step 3. Decide: If button „Platzres. bestätigen“ or button „Abweichung akzeptieren“ is visible, Then Goto Step 6.
Else Goto Step 4.
Step 4. Retrieve from LTM the value for $< reserv\_extra >$, and recall it from WM.
Step 5. Decide: If $< reserv\_extra >$ is „Ruhebereich im ICE“, Then Accomplish goal: drücke button „Ruhebereich im ICE“.
If $< reserv\_extra >$ is „Handybereich im ICE“, Then Accomplish goal: drücke button „Handybereich im ICE“.
Else Accomplish goal: drücke button „keine Präferenz“.
Step 6. Return with goal accomplished.

Method for goal: schließe Reservierung ab.

Step 1. Wait for system for 3,13s
Step 2. Accomplish goal: prüfe Existenz von Abweichungen.
Step 3. Forget $< reserv\_raum >$, forget $< reserv\_platz >$, forget $< reserv\_extra >$ and Return with goal accomplished.

Selection rule set for goal: prüfe Existenz von Abweichungen.

If there are Abweichungen, Then Accomplish goal: behandle Reserv.-Abweichungen.
Else Accomplish goal: drücke button „Platzres. bestätigen“.

Return with goal accomplished.

Method for goal: behandle Reserv.-Abweichungen.

Step 1. Read Verbindung value from the screen and retain it in WM.

Step 2. Recall Verbindung value from WM and Think about Abweichungen for 5,2s and retain result in WM.

Step 3. Recall result from WM and Decide: If you think that Abweichungen is ok, Then Accomplish goal: drücke button „Abweichung akzeptieren“.
Else Handle not-accepted variation.

Step 4. Forget Verbindung value, forget result and Return with goal accomplished.

Vom Analysten definierte Operatoren

Retrieve from LTM the value for < *name* >, and recall it from WM	Zugriff auf einen Aufgabenparameter, bezeichnet durch < *name* >, mit anschließendem Speichern im und Aufruf dieses aus dem Kurzzeitgedächtnis WM
Read < *name* > value from the screen	Lesen von Buchstaben auf dem Bildschirm, die einen Wert für einen Parameter < *name* > bereitstellen und anschließende Speicherung dieses im Kurzzeitgedächtnis WM.
Make up your mind about < *decision description* >	Komplizierter Entscheidungsprozess, dessen Ergebnis im WM gespeichert wird.
Think about < *name* >	Über etwas nachdenken und die Entscheidung im WM speichern
Handle not-accepted variation	Nicht-akzeptierte Abweichungen behandeln
Locate object < *name* >	Lokalisieren eines Objektes, z.B. einer Schaltfläche
Point to < *object* >	Auf ein lokalisiertes Objekt zeigen
Tap on < *object* >	Auf dem Touchscreen etwas antippen, z.B eine Schaltfläche
Wait for system for < *duration* > s	Antwortzeit des Systems nach einer Aktion
Verify that < *description* >	eine Eingabe verifizieren, bevor sich auf diese festgelegt wird; Feedback-Information aufnehmen
Insert < *object* >	Etwas in den Automaten einführen bzw. einwerfen
Pay tickets	Tickets bezahlen
Choose another date	Ein anderes Datum für das Wochenendticket wählen

Aufgabenbeschreibung und Variablen:

Das Ziel ist es, Fahrkarten zu kaufen.
Folgende Parameter treten während der unterschiedlichen Aufgabeninstanzen auf:

$< ticketType >$ ist die Art der Tickets
$< tagGueltig >$ ist der Tag, an dem ein Schönes-Wochenende-Ticket gültig ist
$< fahrradAnzahl >$ ist die Anzahl der im Zug mitgeführten Fahrräder
$< start >$ ist der Startbahnhof der Reise
$< ziel >$ ist der Zielbahnhof der Reise
$< anzahlReisende >$ ist die Anzahl der reisenden Erwachsenen
$< klasse >$ ist die Qualitätsklasse, in der gereist werden möchte
$< tagH >$ ist der Tag der Hinfahrt
$< zeitH >$ ist die Zeit für die Hinfahrt, ab der gereist werden möchte
$< monatH >$ ist der Monat der Hinfahrt
$< tagR >$ ist der Tag der Rückfahrt
$< zeitR >$ ist die Zeit für die Rückfahrt, ab der gereist werden möchte
$< monatR >$ ist der Monat der Rückfahrt
$< zugH >$ ist die gewünschte Zuggattung für die Hinfahrt
$< zugR >$ ist die gewünschte Zuggattung für die Rückfahrt
$< anzahlBahncards >$ ist die Anzahl vorhandener Rabattkarten, sogenannter Bahncards
$< typBahncards >$ ist der Rabatt-Typ der vorhandenen Bahncards in der Form $< Prozentsatz\_Klasse >$
$< reservierung >$ spezifiziert, ob und welche Reservierungen gewünscht sind.
$< reservierungR >$ klärt, ob Reservierung der Rückfahrt gewünscht ist
$< reserv\_raum >$ ist der gewünschte Raum bei der Platzreservierung
$< reserv\_platz >$ ist der gewünschte Sitzplatz bei der Platzreservierung
$< reserv\_extra >$ ist eine gewünschte Lautstärkesituation bei der Platzreservierung

Definitionsbereiche

$< ticketType >$	normal, wochenend, verbund
$< tagGueltig >$	samstag, sonntag, anders
$< fahrradAnzahl >$	0-5
$< zugH >/< zugR >$	alle, alle außer ICE, nur Nahverkehr
$< anzahlReisende >$	1-5
$< tagH >$	heute, morgen, 1-31
$< tagR >$	kein, $< tagH >$, 1-31
$< monatH >/< monatR >$	Januar, Februar, März, April, Mai, Juni, Juli, August, September, Oktober, November, Dezember
$< zeitH >/< zeitR >$	ab sofort, 0:00, 1:00, 2:00, 3:00, 4:00, 5:00, 6:00, 7:00, 8:00, 9:00, 10:00, 11:00, 12:00, 13:00, 14:00, 15:00, 16:00,17:00, 18:00, 19:00, 20:00, 21:00, 22:00, 23:00
$< klasse >$	1, 2
$< ziel >$	[alle Buchstabenkombinationen]
$< start >$	[alle Buchstabenkombinationen]
$< anzahlBahncards >$	0, 1, 2
$< typBahncards >$	25_1, 25_2, 50_1, 50_2
$< reservierung >$	keine, keineHin, ja
$< reservierungR >$	nein, ja
$< reserv\_raum >$	Großraum, Großraum am Tisch, Abteil, Kleinkindabteil
$< reserv\_platz >$	Fenster, Gang
$< reserv\_extra >$	Ruhebereich im ICE, Handybereich im ICE, keine

Annahmen und Ermessensentscheidungen:

- Bei der Auswahl zu Beginn wird „Fahrkartenkauf (Schritt für Schritt)“ ausgewählt.
- Der Nutzer besitzt höchstens zwei Bahncards. Alle Bahncards sind vom selben Typ.
- Der Nutzer kennt die korrekte Schreibweise der Verbindungsorte.
- Reisende sind immer erwachsen.
- Der Nutzer macht keine Fehler: Einige Schaltflächen, die z.B. Rückschritte, Hilfe, eine andere Sprache oder besondere Funktionen ermöglichen, wurden hier nicht modelliert, da diese für die hier vorgesehenen Aufgabeninstanzen irrelevant sind. Dies ist unter anderem so, weil hier keine Fehlererkennung und -bearbeitung der Nutzer berücksichtigt wird.
- Der Standort der Automaten ist Oldenburg (Oldb).
- Aufgabeninstanz eins gilt für den Fall, dass eine Verbindung im „Verbundtarif“ ausgewählt wird.
- Wenn keine Platzreservierung möglich ist, wird der Wert der Variablen $< reservierung >$ ignoriert und in einer gesonderten Methode fortgefahren.
- Der Monat September war bei der Datumsauswahl Hinfahrt noch nicht sichtbar, weil die Tests bereits im Juli stattfanden.
- Der Nutzer akzeptiert jegliche Reservierungsabweichungen.
- Der Nutzer möchte keine Zusatzangebote wahrnehmen.
- Die Variable $< reservierungR >$ wird nur spezifiziert, wenn eine Rückfahrt überhaupt erwünscht ist.

Aufgabeninstanzen:

Belegung der Variablen für Aufgabeninstanz 1:
$< ticketType >$ = verbund
$< start >$ = Oldenburg Oldb
$< ziel >$ = Bremen
$< anzahlReisende >$ = 1
$< klasse >$ = 2

Belegung der Variablen für Aufgabeninstanz 2:
$< ticketType >$ = normal
$< zugH >$ = alle
$< zugR >$ = alle
$< anzahlReisende >$ = 2
$< tagH >$ = 1
$< tagR >$ = 3
$< monatH >$ = September
$< monatR >$ = September
$< zeitH >$ = 10:00
$< zeitR >$ = 18:00
$< klasse >$ = 1
$< start >$ = Oldenburg Oldb
$< ziel >$ = München-Hauptbahnhof
$< reservierung >$ = ja
$< reservierungR >$ = ja
$< reserv\_raum >$ = Abteil
$< reserv\_platz >$= Fenster
$< reserv\_extra >$= Handybereich im ICE
$< anzahlBahncards >$ = 2
$< typBahncards >$ = 50_1

Belegung der Variablen für Aufgabeninstanz 3:
$< ticketType >$ = wochenend
$< tagGueltig >$ = samstag
$< fahrradAnzahl >$ = 3

Belegung der Variablen für Aufgabeninstanz 4:
$< ticketType >$ = normal
$< zugH >$ = nur Nahverkehr
$< anzahlReisende >$ = 1
$< tagH >$ = 1
$< tagR >$ = kein
$< monatH >$ = September
$< zeitH >$ = 10:00
$< klasse >$ = 2
$< start >$ = Leer
$< ziel >$ = Bremen
$< reservierung >$ = keine

$< anzahlBahncards > = 0$

Belegung der Variablen für Aufgabeninstanz 5:
$< ticketType > =$ normal
$< zugH > =$ alle
$< anzahlReisende > = 1$
$< tagH > = 1$
$< tagR > =$ kein
$< monatH > =$ September
$< zeitH > =$ 10:00
$< klasse > = 2$
$< start > =$ Oldenburg Oldb
$< ziel > =$ Willingen
$< reservierung > =$ ja
$< reserv\_raum > =$ Abteil
$< reserv\_platz >=$ Fenster
$< reserv\_extra >=$ Ruhebereich im ICE
$< anzahlBahncards > = 0$

Berechnete Bearbeitungszeiten der einzelnen Aufgabeninstanzen

Abschätzung der Bearbeitungszeit für Aufgabeninstanz 1:

70 Statements x 0,1 Sekunden = 7,0 Sekunden
35,16 Sekunden Operatorenausführungs- + Wait-Zeiten
7,0 + 35,16 Sekunden

= 42,16 Sekunden

Abschätzung der Bearbeitungszeit für Aufgabeninstanz 2:

446 Statements x 0,1 Sekunden = 44,6 Sekunden
128,50 Sekunden Operatorenausführungs- + Wait-Zeiten
44,6 + 128,50 Sekunden

= 173,1 Sekunden ≙ 2,89 Minuten

Abschätzung der Bearbeitungszeit für Aufgabeninstanz 3:

72 Statements x 0,1 Sekunden = 7,2 Sekunden
15,0 Sekunden Operatorenausführungs- + Wait-Zeiten
7,2 + 15,0 Sekunden

= 22,20 Sekunden

Abschätzung der Bearbeitungszeit für Aufgabeninstanz 4:

300 Statements x 0,1 Sekunden = 30,0 Sekunden
80,95 Sekunden Operatorenausführungs- + Wait-Zeiten
30,0 + 80,95 Sekunden

= 110,95 Sekunden ≙ 1,85 Minuten

Abschätzung der Bearbeitungszeit für Aufgabeninstanz 5:

339 Statements x 0,1 Sekunden = 33,9 Sekunden
100,48 Sekunden Operatorenausführungs- + Wait-Zeiten
33,9 + 100,48 Sekunden

= 134,47 Sekunden ≙ 2,24 Minuten

Abschätzung der Lernzeit für das NGOMSL-Modell:

Anzahl aller Statements:

313 Statements

Anzahl zu erlernender Statements (Subtraktion der ähnlichen bzw. gleichen Statements):

279 Statements

Pure Method Learning Time = 17 Sekunden x 279 Statements

= 4743 Sekunden ≙ 79,05 Minuten

LTM Item Learning Time = 6 Sekunden x 25 LTM chunks

Pure Learning Time = 4743 Sekunden + 150 Sekunden
= **4893 Sekunden ≙ 81,55 Minuten**

Manipulierte Methoden des NGOMSL-Modells

Method for goal: spezifiziere Bahncards.

Step 1. Retrieve from LTM the value for $< typBahncards >$, and recall it from WM.

Step 2. Decide: If $< typBahncards >$ is 25_2, Then Accomplish goal: drücke button „25 2. Klasse“.
If $< typBahncards >$ is 50_2, Then Accomplish goal: drücke button „50 2. Klasse“.
If $< typBahncards >$ is 25_1, Then Accomplish goal: drücke button „25 1. Klasse“.
If $< typBahncards >$ is 50_1, Then Accomplish goal: drücke button „50 1. Klasse“.

Step 3. Accomplish goal: drücke button $< anzahlBahncards >$.

Step 4. Forget $< typBahncards >$ and Return with goal accomplished.

Method for goal: wähle Zusatzangebote.

Step 1. Decide: If no Angebot is visible, wait for 2,13s and Goto Step 7.

Step 2. Read Angebot value from the screen and retain it in WM.

Step 3. Recall Angebot value from WM and Decide: If Angebot Reiseschutz-Versicherung is visible, Then Accomplish goal: behandle Versicherung.

Step 4. If Angebot bahn.comfort und bahn.bonus is visible, Then Accomplish goal: behandle Bonusprogramm.

Step 5. If Angebot CityMobil is visible, Then Accomplish goal: behandle Angebot CityMobil and Goto Step 7.

Step 6. Forget Angebot value and Wait for system for 2,13s

Step 7. Return with goal accomplished.

Method for goal: behandle Versicherung.

Step 1. Decide: If you want to contract insurance, Then Accomplish goal: drücke button „Ja“.

Step 2. Return with goal accomplished.

Method for goal: behandle Bonusprogramm.

Step 1. Decide: If you want to collect bonuses with your Bahncard, Then Accomplish goal: drücke button „Ja“ and insert your Bahncard.

Step 2. Return with goal accomplished.

Method for goal: behandle Angebot CityMobil.

Step 1. Decide: If you want to have CityMobil, Then Accomplish goal: drücke button „Mit CityMobil“.

Step 2. Wait for system for 3,53s

Step 3. Return with goal accomplished.

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe.

Oldenburg, den 29. September 2008

Jutta Fortmann