Prognose von Bedienungsfehlern durch Simulation der Entstehung gelernter Sorglosigkeit bei der Pilot-Cockpit Interaktion

Andreas Lüdtke & Claus Möbus
Carl von Ossietzky Universität, Angewandte Informatik, Abteilung Lehr- und Lernsysteme

1. Einleitung

Über die letzten 20 Jahre ist die Unfallrate der weltweiten kommerziellen Luftfahrt mit 2 bis 3 Unfällen pro 1 Millionen Starts [6] relativ konstant geblieben. Im selben Zeitraum hat sich die Anzahl weltweiter Starts von 8 Millionen auf 15 Millionen Starts pro Jahr fast verdoppelt. Der Flugverkehr wird weiter anwachsen, was zwangsläufig zu einer Steigerung der absoluten Unfallzahl führt. Das Vertrauen der Öffentlichkeit in die Sicherheit von Flugreisen und damit verbunden die Zukunft der Luftfahrtindustrie ist somit in Gefahr, wenn die Unfallrate nicht reduziert wird. Menschliches Versagen ist die Hauptursache in über 60% der Unfälle [18], deshalb muss hier angesetzt werden, um die Unfallrate zu reduzieren. Bereits 1975 wurde menschliche Fehlverhalten auf einer IATA-Konferenz in Istanbul als die letzte herausfordernde Grenze der Flugsicherheit („the last frontier in Aviation Safety“) bezeichnet. Daran hat sich bis heute nichts geändert. Dies belegt auch eine Untersuchung der Pilot-Cockpit Interaktion im Auftrag der FAA [6]. Dabei wurden 166 Problembereiche identifiziert und insgesamt eine "klobige" Automatisierung (clumsy automation [19]) attestiert, die komplexe und dynamische Cockpitsysteme zur Folge hat, bei deren Bedienung sogenannte Zusammenbrüche in der Mensch-Maschine Interaktion auftreten, wobei der Pilot nicht mehr nachvollziehen kann, was das System tut [17]. Voraussetzung für das Nachvollziehen des Systemverhaltens ist die Mode-Awareness. Das Flugzeug verhält sich je nach aktivem Modus unterschiedlich. In diesem Beitrag wird Mode Awareness als wichtiger Bestandteil der Situation Awareness behandelt. Das konstruktive Ziel der hier vorgestellten Arbeiten ist es, eine Möglichkeit zu bieten, bereits bei der Entwicklung von Cockpitsystemen, Stellen im Design aufzudecken, die für dei Aufrechterhaltung der Mode Awareness kritisch sind. Es entsteht ein Cognitive Engineering Tool[1)], das formale Systemdesigns analysiert und potentielle Bedienungsfehler prognostiziert. Zu Beginn der Arbeiten wurde untersucht, wodurch der Verlust der Mode Awareness begründet sein kann. Die hier vertretende Fehlererklärungshypothese berücksichtigt einerseits bestimmte Eigenschaften des Designs und andererseits kognitive Prozesse wie „gelernte Sorglosigkeit". Deren Entstehung wurde auf Basis empirischer Untersuchungen in einem Piper Cheyenne PA42 IIIA Full-Motion Simulator bei der Lufthansa Verkehrsfliegerschule modelliert. Der folgenden Beitrag beschreibt die Verwendung des Modells zur Prognose von Bedienungsfehlern. Dies soll dazu führen, dass in Zukunft bei der Entwicklung von Automatisierungssystemen Bedienungsfehler aufgrund des Verlustes der Mode Awareness durch geeignete technische Maßnahmen weitestgehend verhindert wird. Auf diese Weise wird dann im Sinne eines menschen-zentrierten Designprozesses [3] die Technik dem Menschen angepasst.

2. Bedienungsfehler verursacht durch Designstrukturen und gelernte Sorglosigkeit

In der bereits oben angeführten Studie des Federal Aviation Association (FAA) Human Factors Teams [6] wurde insbesondere der Verlust der Mode Awareness als Problembereich herausgestellt:

> „Issue 095: **Mode awareness may be lacking**.
> Pilots may not be able to tell what mode or state the automation is in, how it is configured, what it is doing, and how it will behave. This may lead to reduced situation awareness and errors."

Der Verlust der Mode Awareness kann unterschiedliche Gründe haben. In diesem Beitrag wird die mangelnde Abstimmung zwischen Gerätedesign und den Eigenschaften der menschlichen Kognition als Ursache untersucht.

[1)] Die Bezeichnung "Cognitive Engineering Tool" steht für Softwaresysteme, die Entwickler von Automatisierungssystemen dabei unterstützen, ihre Systeme unter Berücksichtigung kognitiver Prinzipien zu entwickeln (siehe [20]).

Problematische Strukturen in modusbasierten Designs: Die Weichenstruktur

Unter einem Modus kann man ein bestimmtes Verhalten des Gerätes verstehen. Beispielsweise steuern Autopiloten im "Altitude Hold"-Modus das Flugzeug so, dass die momentane Höhe konstant beibehalten wird. Ein Modus beschreibt aber nicht nur ein charakteristisches Verhalten, sondern darüber hinaus auch die Reaktion des Gerätes auf Benutzereingaben, die durchaus modusabhängig sein kann. Mit Hilfe von Modi ist es auf der einen Seite möglich, die Systeme flexibel zu gestalten, so dass eine große Anzahl unterschiedlicher Funktionen und Funktionsvariationen zur Durchführung einer Aufgabe angeboten werden. Auf der anderen Seite müssen die Piloten die automatischen Systeme sehr gut beherrschen und während des Fluges ständig überwachen, um stets das Verhalten nachvollziehen und vorhersagen zu können. Degani [4] weist darauf hin, dass bestimmte Eigenschaften von Designs zum Verlust der Mode Awareness beitragen können. Er nennt diese Eigenschaften Strukturelemente (structural elements). Leveson et al. [8] beschreiben aufbauend auf den Arbeiten von Degani sechs Kategorien von Strukturelementen. Zwei davon werden in diesem Beitrag aufgegriffen: Indirekte Modusübergänge und inkonsistentes Systemverhalten, deren Kombination hier als Weichenstruktur bezeichnet werden soll.

Indirekte Modusübergänge. Ein neuer Modus kann entweder vom Benutzer direkt über entsprechende Bedienelemente oder vom Gerät indirekt, d.h. ohne direkte Intervention von Seiten des Benutzers, aktiviert werden. Indirekte Modusübergänge werden beispielsweise beim Erreichen einer bestimmten Flughöhe ausgelöst. In Abb. 1 kann nach A_1 ein indirekter Modusübergang von M_2 nach M_3 stattfinden. Zu einem Problem kann diese Struktur werden, wenn der Benutzer einen Übergang nicht bemerkt und sich das Gerätes inkonsistent verhält.

Inkonsistentes Verhalten. Ein modusbasiertes System verhält sich inkonsistent, wenn der auf eine Benutzeraktion folgende Modus (Nachfolgemodus) in Abhängigkeit des vorher aktiven Modus (Vorgängermodus) unterschiedlich sein kann. In Abb. 1 geht das System nach der Aktion A_2 entweder in Modus M_4 oder M_5 über. Somit gibt es zwei mögliche Nachfolgemodi für A_2. Dies ist unkritisch, wenn das Manöver in beiden Fällen problemlos erfolgreich abgeschlossen werden kann. Steht jedoch M_4 für eine erfolgreiche Manöverdurchführung und M_5 für einen Misserfolg, dann ist das inkonsistente Verhalten in Abb. 1 problematisch.

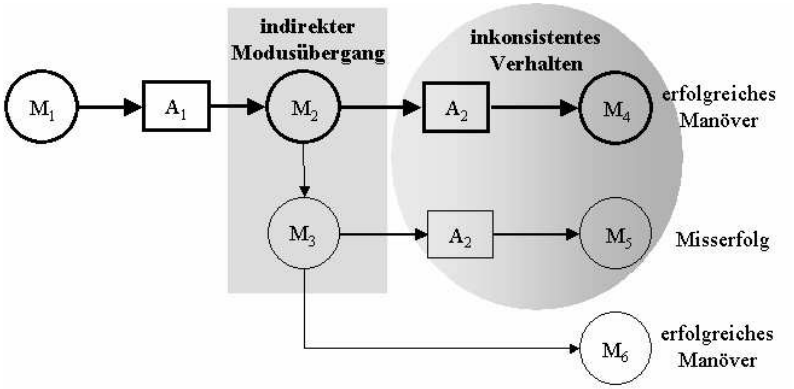

Abb. 1: Weichenstruktur: Indirekter Modusübergang und inkonsistentes Verhalten

Das inkonsistente Verhalten bedingt, dass der Benutzer nach dem indirekten Modusübergang das Manöver anders fortführen muss als vorher. Vorher muss er noch A_2 durchführen, während er sich nachher passiv verhalten muss. Es gibt also zwei unterschiedliche Variationen desselben Manövers, die anfangs identisch sind und nach A_1 verzweigen. Die Gefahr ist, dass der Benutzer den Übergang zu M_3 nicht bemerkt und sich wie in M_2 verhält. Die Situation wird noch erheblich verschärft, wenn die Gestaltung der Modusanzeigen im Cockpit ähnlich und dadurch nur schwer zu unterscheiden ist.

Die in Abb. 1 dargestellte Kombination von indirektem Moduswechsel und inkonsistentem Verhalten soll als Weichenstruktur bezeichnet werden.

Gelernte Sorglosigkeit als Gefahr für die Mode Awareness

Um die Gefahr von Weichenstrukturen einzuschätzen, muss bewertet werden, mit welcher Wahrscheinlichkeit der Benutzer eine obligatorische Modusüberprüfung unterlässt und sich dem falschen

Modus entsprechend verhält. Aus welchen Gründen kann es passieren, dass Modusüberprüfungen unterbleiben? Die Theorie der gelernten Sorglosigkeit [5] liefert einen Erklärungsansatz: Wenn sich jemand wiederholt sicherheitswidrig verhält und dennoch keine negativen Konsequenzen erlebt, entwickelt er eine Haltung derart „alles ist gut und wird auch gut bleiben". Diese Haltung wird als gelernte Sorglosigkeit (GS) bezeichnet und führt zu einer mangelnden Bereitschaft, Informationen zu beschaffen. Im Zusammenhang mit modusbasierten Systemen bedeutet dies, dass ein Benutzer, der das Gerät mehrmals erfolgreich anwendet und dabei wichtige Modusüberprüfungen unterlässt, Sorglosigkeit lernt und es bald gar nicht mehr für notwendig erachtet, modusabhängig zu agieren, weil sowieso der gewohnte Modus aktiv ist. Bestimmte Bediensequenzen werden bildlich gesprochen zu „Trampelpfaden“, die routinemäßig beschritten werden. Eine Weichenstruktur ist in diesem Sinne kritisch, wenn der zugehörige indirekte Moduswechsel im Regelfall erst spät, nachdem genügend Zeit blieb, die gewohnten Manöveraktionen durchzuführen, und im eher seltenen Fall bereits früh eintritt, so dass plötzlich eine alternative Manöverfortsetzung korrekt wäre. Aufgrund der GS kann es passieren, dass der frühe Moduswechsel nicht bemerkt und die Aktionssequenz wie gewohnt fortgesetzt wird. In Abb. 1 ist der „Trampelpfad“ fett gekennzeichnet. In der Terminologie Reasons [14] gelangt der Benutzer nach A_1 an einen kritischen Entscheidungspunkt.

Die in unseren Arbeiten vertretene Fehlererklärungshypothese lautet:

> *Durch den Flugbetrieb bildet sich GS, die, bei Vorhandensein bestimmter Strukturelemente, in „minimal" abweichenden Situationen zu einer Gefahr werden kann, wobei die Auftretenswahrscheinlichkeit eines gefährdenden Bedienungsfehlers von der Erfolgswahrscheinlichkeit des („Routine"-)Verhaltens, der momentanen Arbeitsbelastung (Workload), der Seltenheit sowie der Qualität der Abweichung abhängig ist.*

Diese Hypothese ist die theoretische Grundlage für die weiter unten beschriebene Analyse formaler Designs. Im derzeitigen Status wird das Vorhandensein von Weichenstrukturen im Systemdesign und die Herausbildung von Routineverhalten aufgrund der Seltenheit abweichender Situationen berücksichtigt. Für den letzteren Aspekt wurde ein Pilotenmodell implementiert, das unter bestimmten Umständen Sorglosigkeit lernt und resultierende Bedienungsfehler produziert. Da das Pilotenmodell eine zentrale Rolle bei der Designanalyse spielt, ist es notwendig, die Validität der Modellaussagen zu prüfen. Diese Aufgabe wird in anschließenden Arbeiten wiederum bei der Lufthansa Verkehrsfliegerschule durchgeführt. Erste empirische Hinweise für die Anwendbarkeit des Modells wurden bereits in [10] dargelegt.

Die in der Hypothese genannte Qualität abweichender Situationen, die maßgeblich auf die Gestaltung der Anzeigen im Cockpit zurückzuführen ist, wird in der Analyse noch ausgespart. Erfolgswahrscheinlichkeit und Workload stehen ebenfalls noch aus, wobei bereits daran gearbeitet wird, die Erfolgswahrscheinlichkeit basierend auf dem von Anderson [1] in ACT-R realisierten Ansatz zu integrieren.

3. Beispiele für Bedienungsfehler

In einer bei der Lufthansa Verkehrsfliegerschule durchgeführten Studie wurde das Verhalten von Pilotenschülern in einem Piper PA42 Cheyenne IIIA Full-Motion Simulator beobachtet und ausgewertet [10]. Obwohl dieses Flugzeug nicht zu den modernsten gehört, sind die prinzipiellen Fehlermechanismen mit denen in modernen Flugzeugen vergleichbar. Dies belegen u.a. sowohl Studien von Sarter und Woods, die die Mensch-Maschine-Interaktion in Glass-Cockpits der frühen Generation [15] und nachfolgenden Flugzeugen wie dem A320 [16] untersuchten, als auch von Palmer [13], der in einer Simulatorstudie 22 Linien-Crews ein realistisches Flugszenario in einer DC-9 und MD-88 fliegen ließ.

Die während unserer empirischen Studie beobachteten Bedienungsfehler bezogen sich vorrangig auf den Einsatz des Autopiloten für den gestuften Sink- und Steigflug (Step-Descend und Step-Climb). Dabei erhalten die Piloten einen Funkspruch vom Fluglotsen mit der Freigabe (Clearance) einer bestimmten Höhe. Der Pilot hat dann die Aufgabe, auf diese Höhe zu sinken bzw. zu steigen. Solche Clearances werden erteilt, um das Flugzeug nach dem Start gestuft auf die Reisehöhe bzw. vor der Landung gestuft auf die Anflughöhe zu leiten. Je nach Verkehrslage ist der Sink- bzw. Steigflug mehr oder weniger kontinuierlich. Folgendes Szenario schildert einen Bedienungsfehler, der wiederholt zu beobachten war (vgl. Abb. 2):

Kurz nach dem Start wurde die Freigabe, auf 4000 Fuß zu steigen, erteilt. Der Pilot stellte den ALERTER auf 4000 und betätigte den Knopf (ALTS_BUTTON) zur Aktivierung des „Altitude Selected“-Modus (ALTS_M), in dem die eingestellte Höhe „scharf geschaltet“ ist. Anschließend erhöhte der Pilot

die vertikale Geschwindigkeit (VS für Vertical Speed) mittels des Knopfes (ETRIM_BUTTON) zur Trimmung des Höhenruders, um die für einen Steigflug empfohlene VS von 2000 ft/min zu erreichen. Dabei gilt es zu beachten, dass die Geschwingkeit (IAS für Indicated Airspeed) nicht 160 kn unterschreitet, denn bei erhöhter VS sinkt die IAS. Um dies sicherzustellen, wird empfohlen, den Modus (IAS_M) zur automatischen Stabilisierung der Geschwindigkeit zu aktivieren, sobald 160 kn erreicht sind. Während der Pilot auf die sinkende IAS achtete, kam das Flugzeug der freigegebenen Höhe bereits sehr nahe (der sog. Lead Point wurde erreicht), so dass der Autopilot automatisch in den „Capture"-Modus (ALTC_M) wechselte. In diesem Modus wird die VS automatisch verringert, so dass das Flugzeug sanft auf die gewünschte Höhe einschwebt. Als letztlich 160 kn IAS erreicht waren und der Pilot den IAS-Modus aktivierte, hatte der Übergang in den „Capture"-Modus bereits stattgefunden. Überraschend für den Piloten schwebte das Flugzeug nicht auf 4000 ft ein, sondern stieg weiter. Das Betätigen des IAS-Knopfes (IAS_BUTTON) in der Capture-Phase hatte das Manöver abgebrochen. Der IAS-Modus ist in der Capture-Phase nicht erlaubt. Das Problem in diesem Fall war, dass der Übergang in die „Capture"-Phase außergewöhnlich früh und vom Piloten unbemerkt stattfand.

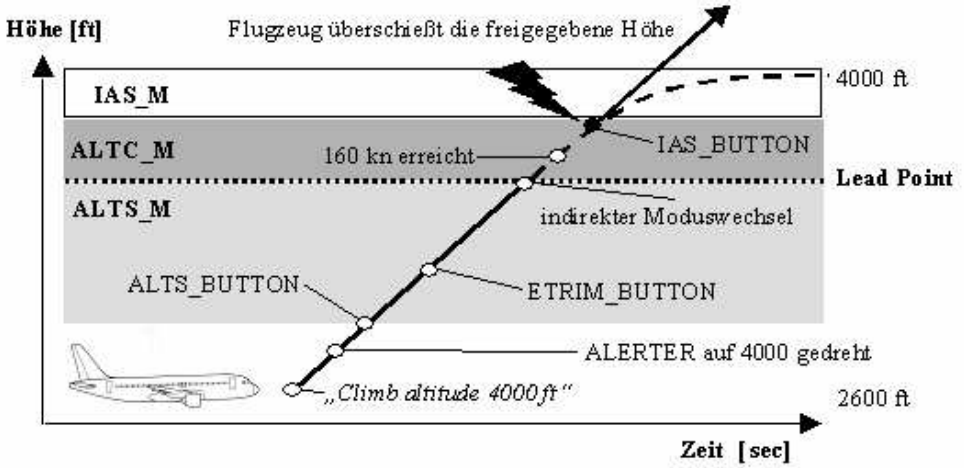

Abb. 2: Trajektorie eines Step-Climb Manövers mit Bedienungsfehler

Wie im folgenden Abschnitt herausgearbeitet wird, sind die Voraussetzungen für die Anwendung unserer Fehlererklärungshypothese in diesem Szenario vorhanden: Der Autopilot reagiert nach dem indirekten Modusübergang anders als vorher, d.h. er verhält sich inkonsistent. Im Flugbetrieb begegnet dem Piloten das eine Verhalten bedeutend häufiger als das andere, wodurch gelernte Sorglosigkeit entstehen kann. Eine alternative Erklärung kann bspw. sein, dass die Probanden ein fehlerhaftes mentales Modell der Bedienung hatten. Um dies auszuschließen, wurde vor den Simulatorstudien eine Befragung zu den Modusübergängen und der korrekten Bedienung durchgeführt. Dabei zeigte sich deutlich, dass die Probanden ein korrektes mentales Modell hatten. Sie wussten also, welche Handlungen in welchen Modi zu den gewünschten Zielzuständen führen. Sie sind grundsätzlich auch in der Lage dieses Wissen anzuwenden, was sich im Step-Climb zeigt, wenn eine Freigabe für eine sehr nahe Höhe (Höhenunterschied 500 ft) erteilt wird. In diesen Szenarien agierten die Probanden sehr vorsichtig und begangen in der Regel keinen Fehler. Der Fall, dass das Bedienungswissen zu Beginn des Simulatortrainings noch nicht optimiert ist, so dass Fehler deshalb entstehen, weil umfangreiche Überlegungen in zu kurzer Zeit angestellt werden müssen, wird im Modell durch deduktive Wissensoptimierung berücksichtigt. Sorglosigkeit kommt nur dann als Erklärung in Frage, wenn der Benutzer genügend Zeit zur korrekten und sorgfältigen Manöverdurchführung hat und trotzdem die Modusüberprüfung an entscheidenden Stellen unterlässt.

4. Prognose von Bedienungsfehlern

Die Prognose von Bedienungsfehlern basiert auf unserer Fehlererklärungshypothese und berücksichtigt im derzeitigen Stand das Vorhandensein von Weichenstrukturen im Systemdesign und die Herausbildung von Routineverhalten aufgrund der Seltenheit abweichender Situationen. In einer Vorstufe wird analysiert, ob in einem gegebenen formalen Design Weichenstrukturen vorhanden sind. Anschließend wird die Anwendung des Designs in realistischen Szenarien mit Hilfe eines Pilotenmodells simuliert. Die Prognose orientiert sich jeweils an einem vorgegebenen Manöver und wird somit grundsätzlich für jedes Manöver separat durchgeführt.

4.1 Vorstufe

Die Vorstufe dient zur Identifikation von Weichenstrukturen, also von indirekten Moduswechseln in Kombination mit inkonsistentem Systemverhalten. Sie wird direkt auf dem Systemmodell durchgeführt. Dabei wird vorausgesetzt, dass die Funktionalität auf einer Abstraktionsebene beschrieben ist, in der sich das Gerät auch später den Piloten darstellt. Wie bei Leveson [8] und Degani [4] wird in dem hier vorgestellten Verfahren zu diesem Zweck eine Beschreibung der Moduslogik verwendet. Das Verfahren arbeitet auf Zustandsautomaten (Statecharts), die mit dem Modellierungstool Statemate [7] erstellt wurden. In Abb. 3 ist ein vereinfachter Ausschnitt der Moduslogik-Rekonstruktion des Autopiloten (AP) der Piper PA42 Cheyenne IIIA abgebildet, die speziell zur Evaluierung der hier vorgestellten Verfahren angefertigt wurde.

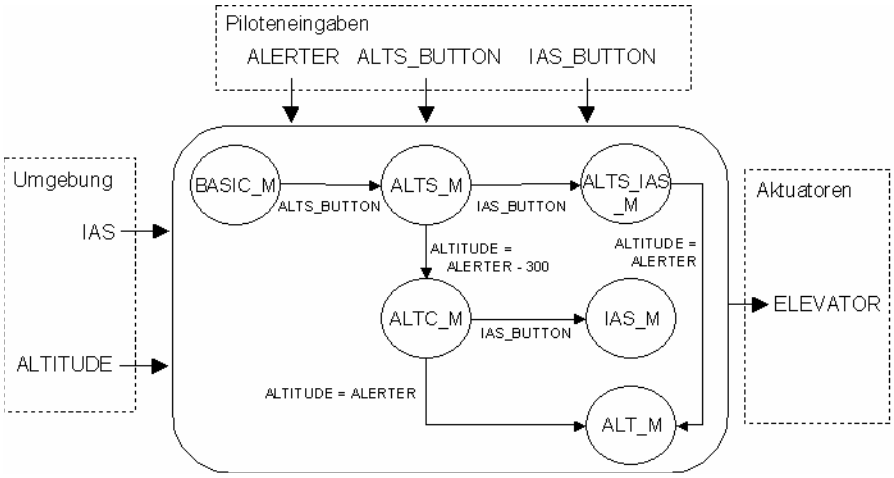

Abb. 3: Rekonstruktion der AP-Moduslogik (vereinfachter Ausschnitt)

Neben dem Design des Systems wird eine einfache formale Beschreibung des zu untersuchenden Manövers vorausgesetzt. Die notwendigen Angaben spezifizieren einen Startmodus (M_S), einen Endmodus (M_E) und eine Sequenz von Aktionen (A_1, .., A_n), die notwendig ist, um M_S in M_E zu überführen. Diese Angaben für das Step-Climb Manöver lauten wie folgt:

M_S = BASIC_M
- A_1 = ALERTER (Freigegebene Höhe in den Alerter eindrehen)
- A_2 = ALTS_BUTTON (Eingedrehte Höhe „scharf schalten“)
- A_3 = ETRIM_BUTTON (Vertikale Geschwindigkeit – VS - erhöhen)
- A_4 = IAS_BUTTON (Geschwindigkeit - IAS - stabilisieren)

M_E = ALT_M

Zur Analyse wird in dem Systemmodell zu Beginn M_S aktiv gesetzt. Anschließend werden sukzessiv die Manöveraktionen an den entsprechenden Designeingängen (Piloteneingaben) angelegt, wodurch direkte und indirekte Moduswechsel ausgelöst werden. Zum Schluss befindet sich das System in einem Modus, der im Falle einer erfolgreichen Manöverdurchführung mit M_E übereinstimmen muss. Mit Hilfe dieser Vorgehensweise ist es möglich, folgende Fragestellungen zu untersuchen:

1. Müssen für die zum Manöver gehörenden Aktionen überhaupt unterschiedliche Vorgängermodi betrachtet werden, z.B. aufgrund möglicher indirekter Moduswechsel?
2. Gibt es Aktionen, die in unterschiedlichen Vorgängermodi zu unterschiedlichen Nachfolgemodi führen?
3. Kann das Manöver in den unterschiedlichen Nachfolgemodi jeweils mit denselben Aktionen erfolgreich abgeschlossen werden?

Zur Beantwortung dieser Fragen wird die Aktionssequenz in unterschiedlichen Variationen „durchgespielt“, so dass unterschiedliche Modusabfolgen und somit unterschiedliche Vorgängermodi für die einzelnen Aktionen auftreten können. Unterschiedliche Modusabfolgen sind grundsätzlich möglich wenn

- ein Manöver in einem bestimmten Modus durch parallele Handlungen verzögert und anschließend weitergeführt wird, wodurch indirekte Moduswechsel stattfinden können, die sonst erst zu einem späteren Zeitpunkt oder gar nicht eintreten würden,

- ein Manöver in einem bestimmten Modus abgebrochen und in diesem erneut begonnen werden muss. Dieser Fall tritt bspw. ein, wenn der Pilot eine neue Höhen-Freigabe erhält, während er bereits eine vorher erhaltene Höhe anfliegt.

Bzgl. des ersten Falls sei gesagt, dass derzeit Interferenzen zwischen dem zu untersuchenden Manöver und parallelen Handlungen nur auf zeitlicher Basis betrachtet werden und der Aspekt ausgespart bleibt, dass sich durch parallele Handlungen weitere Modi einstellen können, deren Einfluss ebenfalls untersucht werden müsste.

Neben dem sukzessiven Anlegen von Benutzeraktionen kommt eine weitere Technik zur Anwendung: Model Checking [2]. Model Checking ist eine Methode zur formalen Verifikation endlicher Zustandsautomaten. Dabei wird mittels eines symbolischen Algorithmus der gesamte Zustandsraum des Systems traversiert und geprüft, ob eine bestimmte Eigenschaft erfüllt ist oder nicht. Die Eigenschaft wird meist durch eine temporale Logik spezifiziert. Bei der Traversierung wird auch der gesamte Wertebereich der Umgebungsinputs vollständig durchlaufen, ohne dass ein Umgebungsmodell gegeben sein muss. Dies ist notwendig, um indirekte Moduswechsel aufzudecken, die bspw. auftreten, weil bestimmte Inputs aus der Umgebung (in Abb. 3 IAS und ALTITUDE) Schwellenwerte überschreiten. Model Checking ist ebenfalls notwendig, um zu bestimmten, ob die Durchführung einer Aktionssequenz letztendlich zu M_E führt (Fragestellung 3. von oben), denn auch hierbei muss die Fortentwicklung bestimmter Inputs aus der Umgebung betrachtet werden.

Im folgenden werden die einzelnen Schritte der Vorstufe erläutert:

1. Schritt: Im ersten Schritt wird die Aktionssequenz des Manövers ohne Verzögerung und Abbruch „durchgespielt“ und jeweils für jede Aktion protokolliert, welcher Folgemodus sich unmittelbar einstellt.

2. Schritt: Anschließend wird das Manöver sukzessiv nach jeder Aktion abgebrochen und erneut begonnen. Dabei wird geprüft, ob eine der Manöveraktionen aufgrund des neuen Vorgängermodus in einem anderen Folgemodus landet.

3. Schritt: Schließlich wird nach jeder Aktion mit Hilfe von Model Checking geprüft, ob ein indirekter Moduswechsel auftreten kann. Zu diesem Zweck wird jeweils vom aktuellen Modus ausgehend geprüft, ob irgendein anderer Modus ohne neue Manöveraktionen erreichbar ist. Diese Art der Analyse wird als Erreichbarkeits-Analyse [2] bezeichnet. Wenn die Möglichkeit eines indirekter Moduswechsels besteht, wird die nächste Aktion der Sequenz nach dem Wechsel, also verzögert angelegt.

4.Schritt: Die zweite Anwendung von Model Checking, ebenfalls eine Erreichbarkeits-Analyse, ist im Anschluss an jeden der ersten drei Schritte notwendig, um zu prüfen, ob das Manöver Erfolg hatte oder nicht.

Als Ergebnis liegt für jede Aktion (A_i) eine Menge von 3-Tupeln vor: { (M_{V1}, A_i, M_{N1}),..., (M_{Vn}, A_i, M_{Nn}) }. M_{vj} repräsentiert jeweils einen Vorgängermodus und M_{nj} den resultierenden Nachfolgemodus. Auf Basis dieser Tupel können die Fragestellungen 1 und 2 von oben beantwortet werden. Informationen für Fragestellung 3 liefert der vierte Schritt. Somit liegen alle Informationen zur Identifizierung von Weichenstrukturen vor.

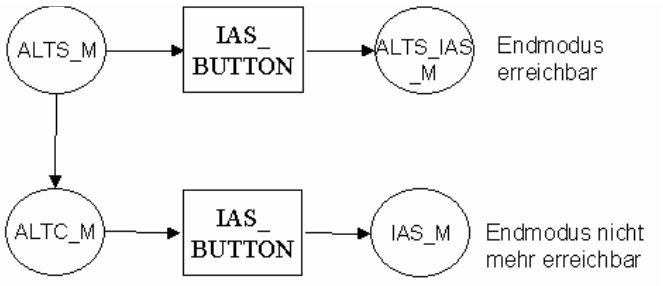

Abb. 4: In AP-Design identifizierte Weichenstruktur

Das Verfahren wurde mit einer eigens angefertigten Rekonstruktion der Moduslogik des Piper PA42 Cheyenne IIIA Autopiloten getestet. Bei der Analyse dieses Designs konnte die in Abb. 4 gezeigte Weichenstruktur aufgedeckt werden. Wird der IAS-Knopf in ALTS_M gedrückt, ist daraufhin ALTS/IAS_M aktiv, d.h. die in den Alerter eingestellte Höhe ist aktiviert und zusätzlich wird die IAS

stabilisiert. Drückt der Pilot den IAS-Knopf in ALTC_M, d.h. in der Einschwebephase, ist anschließend der Modus IAS_M aktiv, in dem die eingestellte Höhe keine Berücksichtigung mehr erfährt und insofern auch nicht mehr angeflogen wird. Der AP stabilisiert dann lediglich noch die IAS. Zwischen ALTS_M und ALTC_M gibt es einen indirekten Modusübergang, der ausgelöst wird, wenn die Entfernung zur im Alerter eingestellten Höhe (ALERTER) nur noch 300 ft beträgt.

Auf Basis der identifizierten Weichenstrukturen kann das Manöver verfeinert werden. Dies soll mittels Bedienungsregeln geschehen, wobei an den Stellen, wo Weichen möglich sind, notwendige Modusüberprüfungen eingefügt werden. Die erste Regel beschreibt die Teilziele, die zur Durchführung des Manövers realisiert werden müssen.

Wenn eine freigegebene Höhe angeflogen werden soll,
Dann stelle diese Höhe in den AP ein,
aktiviere die eingestellte Höhe,
erhöhe die VS,
stabilisiere die IAS.

Die folgenden Regeln geben jeweils an, durch welche Aktionen, die Teilziele erreicht werden:

BR1: Wenn die Höhe in den AP eingestellt werden soll,
Dann drehe sie in den ALERTER ein.

BR2: Wenn die ALTITUDE aktiviert werden soll,
Dann drücke den ALTS_BUTTON.

BR3: Wenn die VS erhöht werden soll,
Dann drücke den ETRIM_BUTTON.

BR4: Wenn die IAS stabilisiert werden soll,
Dann stelle zunächst sicher, dass ALTC_M noch nicht aktiv ist,
drücke den IAS_BUTTON.

Die letzte Regeln berücksichtigt die identifizierte Weichenstruktur.

4.2 Simulation

Mit Hilfe der verfeinerten Bedienungsregeln sollte dann eine sichere und zierorientierte Bedienung des Gerätes möglich sein. Fraglich ist allerdings, ob die Regeln anfällig sind für GS. Um dies herauszufinden, wird eine Simulation durchgeführt, die ein Pilotenmodell, das Systemdesign, ein Flugzeugmodell, ein Umgebungsmodell und einen Szenariengenerator umfasst. Innerhalb dieser Umgebung interagiert der simulierte Pilot mit dem System in unterschiedlichen Szenarien und lernt dabei den routinemäßigen Umgang. Das Systemdesign ist dabei dasselbe wie bereits in der Vorstufe. Im Rahmen der Simulation wird die Berücksichtigung weiterer Informationen ermöglicht: Einerseits die Häufigkeit von Manövern und andererseits auf Seite des Piloten die Beachtung von Modusabzeigen und das Erlernen von Routineabläufen.

Das Pilotenmodell

Das Pilotenmodell umfasst eine Wissensbasis, in der Bedienungsregeln gespeichert sind, und ein kognitives System, das die Anwendung und Modifikation der Regeln ermöglicht. Zu Beginn der Simulation werden die Bedienungsregeln, die als Ergebnis der Vorstufe vorliegen, in die Wissensbasis geladen. Das Regelformat orientiert sich an dem GMR-Ansatz (Goals Means Relations), der bereits im ABSYNT-Projekt [12] zur Modellierung eines Lernprozesses vom Novizen zum Experten im Bereich der funktionalen Programmierung verwendet wurde. Die Adaption des ursprünglichen Ansatzes auf die Pilotendomäne ist in [9] beschrieben.

Jede Regel (Abb. 5) besteht aus einem Regelkopf (links vom Pfeil) und einem Regelkörper (rechts vom Pfeil). Der Regelkopf enthält genau ein Ziel(Kreis)-Mittel(Rechteck)-Paar, im Körper kann eine Konjunktion solcher Paare enthalten sein. Eine Bedienungsregel (z.B. BR1 - BR4 von oben) wird u.U. durch mehrere GMR-Regeln implementiert. BR4 wird durch die Regeln R1 (Abb. 5) und R2 (Abb. 6) implementiert. R1 kann wie folgt verbalisiert werden:

Regelkopf:

Wenn es das Hauptziel ist, die IAS zu stabilisieren, dann führe die entsprechende Aktion aus.

Regelkörper:

- Verfolge das Ziel, die Modusanzeige zu prüfen.
- Verfolge dann das Ziel, den IAS_BUTTON zu betätigen. Die zugehörige Aktion ist notwendig zur Erfüllung des Hauptziels

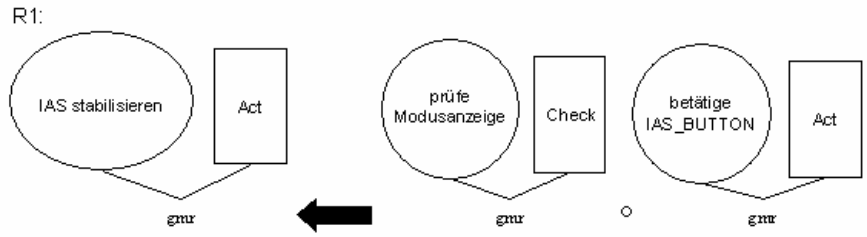

Abb. 5: Implementierung der Bedienungsregel BR4 im GMR-Format – Teil 1

Bei der Manöverdurchführung werden die angewendeten Regeln in einem Trace gespeichert. Nach jedem Manöver setzt ein Lernprozess ein, der die Entstehung von Routine und gelernter Sorglosigkeit durch Regelkomposition nachbilden soll. Dabei werden Regeln, die wiederholt in der gleichen Reihenfolge angewendet wurden, miteinander zu einer Regel verschmolzen. Das Verfahren der Regelkomposition lässt sich mit Hilfe der Schnittregel beschreiben:

$$\frac{(F \leftarrow P \wedge C) \qquad (P' \leftarrow A)}{(F \leftarrow A \wedge C)\sigma}$$

Hierbei werden Regeln als Hornklauseln interpretiert. Die beiden oberen Klauseln verschmelzen zu der unteren, wobei *P*, *P*' und *F* Atome und *A*, *C* Konjunktionen von Atomen sind. *P* und *P*' müssen unifizierbar sein, d.h. sie müssen mindestens eine gemeinsame Instand besitzen. σ steht für die Anwendung der allgemeinsten gemeinsamen Instanz (most general unifier). Zu Illustration soll R1 mit R2 verschmolzen werden. R2 lässt sich wie folgt lesen:

Regelkopf:

Wenn es das Hauptziel ist, die Modusanzeige zu prüfen, dann stelle sicher, dass ALTC_M (Capture Modus) noch nicht aktiv ist.

Regelkörper:

Der Autopilot befindet sich noch nicht im Capture-Modus.

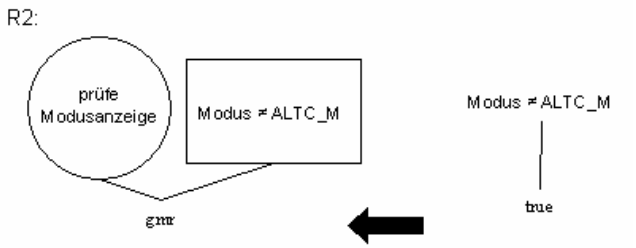

Abb. 6: Implementierung der Bedienungsregel BR4 im GMR-Format – Teil 2

Der Regelkopf von R2 und die zweite GMR im Regelkörper von R1 können miteinander unifiziert werden. Unter Anwendung der Schnittregel ergibt sich das Kompositum in Abb. 7:

Regelkopf:

Wenn es das Hauptziel ist, die IAS zu stabilisieren dann führe die entsprechende Aktion aus.

Regelkörper:

- Der Autopilot befindet sich noch nicht im Capture-Modus.
- Verfolge dann das Ziel, den IAS_BUTTON zu betätigen. Die zugehörige Aktion ist notwendig zur Erfüllung des Hauptziels

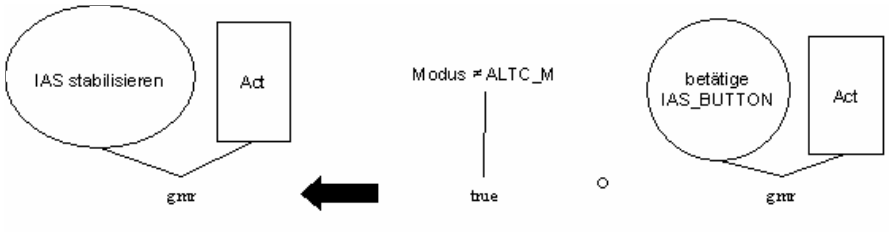

Abb. 7: Regelkompositum aus R1 und R2

Die Komposition modelliert einen Piloten, der annimmt, der Autopilot befindet sich noch nicht im „Capture"-Modus, ohne dies anhand der Modusanzeige zu überprüfen, denn durch die Komposition wurde diese Prüfhandlung eliminiert. Durch Eliminierung von Prüfhandlungen wird die Entstehung GS simuliert.

Die Anwendung von Komposita ist mit bestimmten empirischen Constraints [11, 12] verbunden, die sich in Performanzunterschieden zur Anwendung nicht kompositionierter Regeln niederschlagen. In [10] wird anhand einer Reihe von Manöverprotokollen eines exemplarischen Pilotenschülers gezeigt, dass sich solche Performanzunterschiede im Laufe der Schulung einstellen und dass gegen Ende der Schulung Bedienungsfehler auftraten, die sich aus empirisch begründeten Komposita ableiten lassen. Diesen Beobachtungen liefern erste Hinweise zur Anwendbarkeit des Pilotenmodells.

Im aktuellen Zustand der Implementierung wird lediglich die Manöverfrequenz als Kriterium für Kompositionsbildung herangezogen: Komposita werden gebildet, wenn ein Manöver wiederholt mit denselben Regeln durchgeführt werden konnte. Weitere Faktoren, die berücksichtigt werden sollen sind:

- Manöverkontinuität: Konnte das Manöver ohne Unterbrechung durch andere Manöver durchgeführt werden? In diesem Fall werden eher Komposita gebildet als nach unterbrochenen Durchführungen.
- Manöverregularität: Wird das Manöver mit einer gewissen Regelmäßigkeit oder in unregelmäßigen Abstände durchgeführt? Im ersten Fall ist die Bildung von Komposita wahrscheinlicher.

Diese Zusammenhänge sind hypothetisch und müssen in der kognitionspsychologischen Grundlagenforschung untersucht werden.

Der Simulationsablauf

Um die Entstehung von GS zu simulieren, wird das zu untersuchende Manöver mehrmals in unterschiedlichen Flugsituationen simuliert. In Abb. 8 ist die Architektur der Simulationsumgebung zu sehen. Die einzelnen Komponenten sind in unterschiedlichen Implementierungssprachen realisiert und durch einen JAVA-Simulationskern miteinander gekoppelt. Im folgenden wird beschrieben, welche Rolle die einzelnen Komponenten spielen:

1. Szenariengenerator produziert Initialzustand:
   Der Initialzustand umfasst Flugdaten für das Flugzeugmodell, wie Höhe, Angezeigte und Vertikale Geschwindigkeit, für das Umgebungsmodell, wie Tageszeit, Sicht und Wetter und für den AP den aktuellen Modus. Der Szenariengenerator basiert auf einem Modell, das für jeden relevanten Initialzustand die Wahrscheinlichkeit repräsentiert, mit der das Manöver in diesem beginnt. Derzeit wird ein solches Modell für Step-Descent und Step-Climb Manöver auf Basis von Interviews mit Fluglotsen aufgebaut.

2. Pilot interagiert mit Autopilot unter Berücksichtigung der Bedienungsregeln:
   Das Pilotenmodell prüft auf Basis der Bedienungsregeln die aktuellen Werte der Cockpitdisplays und generiert ggf. Aktionen. Für die Umsetzung der Aktionen sorgen das Autopilotenmodell, das in einem Statemate internem Format verwendet wird, sowie das Flugzeug- und Umgebungsmodellmodell, die beide unter Verwendung einer kommerziell erhältlichen Flugsimulatorsoftware bereitgestellt werden. Das Manöver ist beendet, wenn sich das Gerät in M_E befindet oder wenn ein Abbruchkriterium erfüllt ist. Im letzteren Fall wird es als Misserfolg gewertet. Abbruchkriterien müssen durch den Benutzer auf Basis von Flugdaten für jedes Manövers definiert werden.

3. Szenariengenerator streut Ereignisse ein:
   Ereignisse können bspw. Höhenfreigaben der Fluglotsen sein. Zur Repräsentation von Ereignissen wird wiederum auf Basis von Fluglotseninterviews ein probabilistisches Modell erstellt, in

dem für jedes Ereignis die Auftretenswahrscheinlichkeit in Abhängigkeit unterschiedlicher Flugsituationen angegeben werden muss.

4. Nach Ende des Manövers wird der Lernprozess angestoßen:
   Das Pilotenmodell prüft, ob die Voraussetzungen für die Bildung von Regelkompositionen vorhanden sind und führt diese ggf. durch. Die Implementierung des Pilotenmodells ist in Prolog erfolgt.

5. Der Ablauf beginnt erneut bei 1.

Durch Inspektion der Regelbasis ist es möglich zu prüfen, ob notwendige Prüfhandlungen eliminiert wurden.

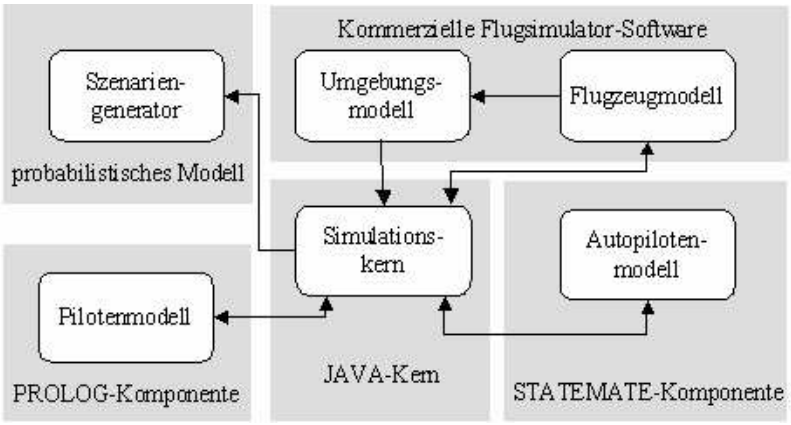

Abb. 8: Architektur der Simulationsumgebung

Bei der Simulation des Step-Climb Manövers unterscheiden sich die generierten Initialzustände derzeit lediglich in der Flughöhe. Gleich zu Beginn gibt der Szenariengenerator eine Höhenfreigabe, die basierend auf einer rudimentären probabilistischen Verteilung der in der Realität vorkommenden Höhenunterschiede produziert wird. Weitere Ereignisse werden noch nicht eingestreut. In dieser Konstellation ergibt sich, dass der indirekte Modusübergang, der in der Vorstufe identifiziert wurde, in den meisten Fällen erst eintritt, nachdem das Pilotenmodell genügend Zeit[2] hatte, den IAS_BUTTON zu betätigen. Zu Erinnerung: Vor dem indirekten Modusübergang führt das Betätigen des Knopfes zum erfolgreichen Abschluss des Manövers, nachher muss sich der Pilot passiv verhalten, denn dieselbe Aktion führt zum Abbruch des Manövers. Dies ergibt sich aus dem in der Vorstufe aufgedeckten inkonsistenten Verhalten des Autopiloten. Zu Beginn beinhaltet das Pilotenmodell die aus der Vorstufe abgeleiteten Bedienungsregeln und verhält sich somit in beiden Fällen korrekt. Je häufiger allerdings das Betätigen des IAS_BUTTONs zum Erfolg führt, desto häufiger werden die oben angeführten Regeln R1 und R2 nacheinander ausgeführt und deshalb kompositioniert. Wie bereits oben gezeigt, führt dies zur Elimination der Modusüberprüfung, so dass gelernte Sorglosigkeit entsteht und das Pilotenmodell in diesem Zustand den indirekten Modusübergang nicht mehr bemerken wird.

Wenn Komposita gebildet werden, bleiben die Ursprungsregeln weiterhin im Modell enthalten. Um die Auswahl zwischen den Regelvarianten weitgehend realistisch vorzunehmen, muss die Arbeitslast (Workload) des Piloten Berücksichtigung finden. In Phasen mit hoher Arbeitslast ist es wahrscheinlicher, dass Routinehandlungen vorgenommen und damit Regelkomposita angewendet werden. Weiterhin werden Regeln umso wahrscheinlicher ausgewählt, je häufiger sie in der Vergangenheit erfolgreich angewendet wurden und je höher ihr Nutzen ist. Entsprechende Parameter wurden von Anderson [1] im ACT-R System definiert und sollen in das Pilotenmodell übernommen werden. Im aktuellen Zustand werden grundsätzlich Regelkomposita gegenüber den Ursprungsregeln bevorzugt angewendet.

5. Zusammenfassung und Ausblick

[2)] Mit jeder Regel ist eine Ausführungszeit verbunden, die auf Basis von Grundlagenliteratur über kognitive Prozesse und Motoraktivitäten abgeleitet wurden.

In diesem Beitrag wurde ein zweistufiges Verfahren zur Analyse formaler Systemdesigns auf potentielle Bedienungsfehler vorgestellt. In einer Vorstufe werden strukturelle Schwächen im Design aufgedeckt, welche dann in einer Simulationsumgebung unter Berücksichtigung des operationellen Umfelds näher untersucht werden. Die Simulation umfasst neben dem Systemdesign ein Flugzeugmodell, ein Umgebungsmodell, einen Szenariengenerator zur Repräsentation des Flugbetriebs und ein Pilotenmodell. Das Verfahren wird an einer Rekonstruktion der Moduslogik des Autopiloten der Piper PA42 Cheyenne IIIA getestet. Derzeit ist die Simulationsumgebung noch auf dieses Design zugeschnitten. Es wird jedoch eine Umgebung angestrebt, in der zur Analyse eines neuen Systemdesigns lediglich die Bedienungsregeln auf Basis der Ergebnisse der Vorstufe angepasst werden müssen und die restlichen Modelle unverändert angewendet werden können. Zu diesem Zweck bedarf es einer umfassenden Modellierung des Flugbetriebs und der Verwendung eines möglichst allgemein anwendbaren Flugzeugmodells, das dem Zweck der Bedienungsfehleranalyse genügt. Die kognitive Schicht des Pilotenmodells, d.h. die Regelanwendung und -modifikation, ist bereits im jetzigen Zustand allgemein gehalten und nicht auf ein bestimmtes Systemdesign zugeschnitten. Eine Validierung der Prognosen des Pilotenmodells findet in Zusammenarbeit mit der Lufthansa Verkehrsfliegerschule in Bremen statt. Dabei werden die im Beitrag genannten Modellerweiterungen unter Berücksichtigung der Validierungsergebnisse realisiert.
Um die Allgemeingültigkeit des vorgestellten Verfahrens sicherzustellen, wird eine weitere Fallstudie auf Basis der A320 Moduslogik durchgeführt werden.

Literatur

[1] Anderson, J.R. and Lebiere, C.: The atomic components of thought Mahwah, NJ: Lawrence Erlbaum Associates, 1998.

[2] Bienmüller, T., Damm, W., Klose, J. und Wittke, H.: Formale Analyse und Verifikation von Statemate Entwürfen. it+ti Informationstechnik und technische Informatik, 43(1), S. 29-34, 2001.

[3] Billings, C.E.: Aviation Automation: The Search for a Human-Centered Approach. Mahwah, NJ : Lawrence Erlbaum Associates 1997.

[4] Degani, A.: Modeling Human-Machine Systems: On Modes, Error, and Patterns of Interaction. Ph.D. Thesis. Georgia Institute of Technology 1996.

[5] Frey, D. und Schulz-Hardt, S.: Eine Theorie der gelernten Sorglosigkeit. In H. Mandl (Hrsg.), Bericht über den 40. Kongress der Deutschen Gesellschaft für Psychologie, S. 604-611. Göttingen, Bern, Toronto, Seattle: Hogrefe Verlag für Psychologie 1996.

[6] Funk, K., Lyall, B., Wilson, J., Vint, R., Niemczyk, M., Suroteguh, C. und Owen, G.: Flight Deck Automation Issues. In The International Journal of Aviation Psychology, Bd. 9, Nr. 2, S. 109-123, 1999.

[7] Harel, D. und Politi, M.: Modeling Reactive Systems with Statecharts: The Statemate Approach. McGraw-Hill Companies 1996.

[8] Leveson, N.G., Pinnell, L.D., Sandys, S.D., Koga, S. und Reese, J.D.: Analysing Software Specifications for Mode Confusion Potential. In C.W. Johnson (Hrsg.), Proceedings of the Workshop on Human Error and System Development, Technical Report GAAG-TR-97-2, S. 132 - 146, Glasgow : Glasgow Accident Analysis Group, 1997.

[9] Lüdtke, A., Möbus, C. und Thole, H.J.: Cognitive Modelling Approach to Diagnose Over-Simplification in Simulation-Based Training, in: St. A. Cerri, G. Gouarderes & F. Paraguacu (eds), Intelligent Tutoring Systems, Proceedings of the 6th International Conference, ITS2002, S. 496 – 506. Berlin: Springer, 2002.

[10] Lüdtke, A. und Möbus, C.: Prognose von Bedienungsfehlern durch Routinebildung in teilautonomen Systemen: Konzept und empirische Untersuchung, in: R. Marzi, V. Karavezyris, H.H. Erbe & K.P. Timpe (Hrsg.): Bedienen und Verstehen: 4. Berliner Werkstatt Mensch-Maschine-Systeme 2001, S. 164 – 184. Düsseldorf: VDI Verlag GmbH, 2002.

[11] Möbus, C. und Schröder, O.: Zur Modellierung des Wissenserwerbs als deduktive und induktive Wissensveränderung. In F. Klix, H. Spada (Hrsg.), Enzyklopädie der Psychologie, Themenbereich C: Theorie und Forschung, Serie II: Kognition, Band G: Wissenspsychologie, S. 403-456. Göttingen: Hogrefe Verlag 1998.

[12] Möbus, C., Schröder, O. und Thole, H.J.: Online Modeling the Novice-Expert Sift in Programming Skills on a Rule-Schema-Case Partial Order. In K.F. Wender, F. Schmalhofer und H.-D. Böcker (Hrsg.), Cognition and Computer Programming, Ablex Series in Computational Sciences, S. 63-105. Norwood, N.J.: Ablex, 1995.

[13] Palmer, E.: "Oops, it didn't arm." - A case study of two automation surprises. In R.S. Jensen und L.A. Rakovan (Hrsg.), Proceedings of the Eighth International Symposium on Aviation Psychology, Columbus, OH : Aviation Psychology Laboratory 1995.

[14] Reason, J.: Action not as planned. In G. Underwood und R. Stevens (Hrsg.), Aspects of consciousness, S. 67-89. London: Academic Press.
[15] Sarter, N. B. und Woods, D.D.: How in the World Did We Ever Get into That Mode? Mode error and Awareness in Supervisory Control. In Human Factors, Bd. 37, Nr. 1, S. 5-19, 1995.
[16] Sarter, N. B. und Woods, D.D. Team Play with a Powerful and Independent Agent: Operational Experience and Automation Surprises on the Airbus A-320. In Human Factors, Bd. 39, Nr. 4, S. 553-569, 1997.
[17] Sarter, N.D., Woods, D.D. und Billings, C.E.: Automation Surprises. In G. Salvendy (Hrsg.), Handbook of Human Factors/Ergonomics, 2. Auflage, S.1926-1943. New York: Wiley 1997.
[18] Statistical Summary of Commercial Jet Aircraft Accidents. Worldwide Operations 1956-2000. Hrsg. Boeing Commercial Airplane Group. Seattle, Washington: Airplane Safety 2000.
[19] Wiener, E.L.: Human Factors of Advanced Technology ("Glass Cockpit") Transport Aircraft. Technical Report. 117528. Moffett Field, CA: NASA Ames Research Center 1989.
[20] Woods, D. D. und Roth, E. M.: Cognitive Engineering: Human Problem Solving with Tools. In Human Factors, Bd. 30, Nr. 4, S. 415-430, 1988.