ENTWICKLUNG EINER KOGNITIVEN SYSTEMARCHITEKTUR MIT ZENTRALER ONTOLOGIE UND SPEZIFISCHEN ALGORITHMEN

S. Brüggenwirth, Fraunhofer-Institut für Hochfrequenzphysik und Radartechnik,
Fraunhoferstr.20, 53343 Wachtberg, Deutschland

Zusammenfassung

Es wird die Entwicklung einer generischen, kognitiven Systemarchitektur basierend auf dem Rasmussen-Schema menschlicher kognitiver Leistungen beschrieben. Hauptaugenmerk liegt auf der konsequenten Verwendung einer zentral hinterlegten Wissensbasis und spezifischen, darauf operierenden Verarbeitungsalgorithmen zur Abbildung kognitiver Subfunktionen. Diese umfassen Inferenz-, Such- sowie Constraint-Satisfaction-basierte Methoden. Das zur Validierung entwickelte Softwareframework COSA² wurde zur UAV-Missionsplanung eingesetzt. Im Bereich der sensornahen Informationsverarbeitung werden Ansätze aus dem kognitiven Radar erläutert.

1. EINLEITUNG

Der Einzug komplexer Automation revolutionierte eine Vielzahl klassischer Industriezweige. Insbesondere die Luftfahrtbranche zeigte sich, etwa durch die Einführung des Zwei-Personen-Cockpits oder zukünftiger, unbemannter Luftfahrzeuge, immer wieder als Triebfeder innovativer Automatisierungskonzepte. Die Rolle des menschlichen Bedieners im Arbeitsprozess verschiebt sich hierbei von der Übernahme einfacher, manueller Steuerungsaufgaben hin zur kognitiv anspruchsvolleren Überwachung der Automatisierungsfunktionen.

Das Forschungsfeld der kognitiven Automation [1] befasst sich mit der Ausgestaltung solcher hochautomatisierter, Mensch-Maschine-Systeme, welche durch die zunehmende Berücksichtigung der höheren kognitiven Fähigkeiten des Menschen wesentlich durch Erkenntnisse der kognitiven Psychologie und Methoden der Robotik und künstlichen Intelligenz gekennzeichnet ist.

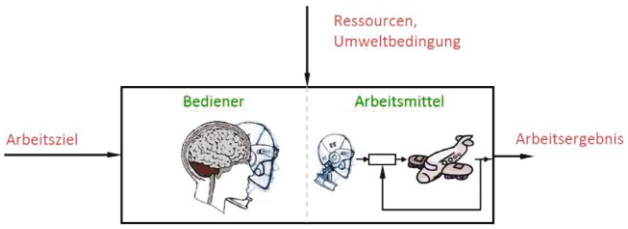

BILD 1. Dual-Mode Konzept der kognitiven Automation

Nach dem Dual-Mode Konzept (BILD 1) der kognitiven Automation beruht die Umsetzung auf ziele-basierten, intelligenten Software-Agenten (dargestellt durch einen Roboterkopf) zur Erhöhung des Automationsgrades konventioneller Komponenten oder zur Unterstützung des Bedieners als Assistenzsystem. Eine maßgebliche Herausforderung bei der realen Umsetzung kognitiv automatisierter Systeme stellt die Verarbeitung verrauschter Sensordaten dar, welche in Unsicherheit bei der Schätzung des Umweltzustandes resultiert und bei der Implementierung einer kognitiven Systemarchitektur berücksichtigt werden muss.

In Kapitel 2 wird das Rasmussen-Schema menschlicher kognitiver Leistungen als Grundlage für die Entwicklung einer kognitiven Systemarchitektur vorgestellt. Kapitel 3 befasst sich mit der Abbildung einzelner, kognitiver Subfunktionen auf Verarbeitungsalgorithmen. In Kapitel 4 wird eine Anwendung der realisierten Systemarchitektur COSA² (COgnitive System Architecture with Centralized Ontology and Specific Algorithms) im Bereich der UAV-Missionsplanung beschrieben. Des Weiteren wird eine sensornahe Erweiterung des Konzepts auf das kognitive Radar erläutert. Kapitel 5 schließt den Artikel mit einer Zusammenfassung ab.

2. ARCHITEKTURKONZEPT

Die Abbildung höherer kognitiver menschlicher Fähigkeiten wurde zum einen in der Psychologie und Kognitionswissenschaft [2] durch ganzheitliche, Prozessmodelle menschlicher Kognition untersucht. Symbolbasierte, integrierte kognitive Architekturen wie SOAR [3] und ACT-R [4] dienen der Simulation und experimentellen Überprüfung wesentlicher Fragestellungen wie dem Aufbau von Kurz- und Langzeitspeicher, die Repräsentationsform der Wissenselemente, sowie die funktionalen Prozesse, die auf diesen Strukturen operieren, inklusive der Lernprozesse, die sie modifizieren. In diesem Artikel wird das im folgenden Abschnitt beschriebene Rasmussen-Schema als übergeordnetes Prozessmodell gewählt.

Wesentlichen Einfluss bei der Abbildung kognitiver Fähigkeiten hatte auch die Informatik [5]. Ein Fokus liegt hierbei in der Entwicklung effizienter Algorithmen, mit dem Ziel realzeitlicher Verarbeitung, kurzen Reaktionszeiten, sowie der formalisierten Beschreibung ihrer Eigenschaften [6]. Insbesondere die in der Robotik häufig angewandten, geschichteten Architekturmodelle [7] sowie die in Kapitel 3 beschrieben Methoden der künstlichen Intelligenz spielen bei der Entwicklung der Systemarchitektur COSA² eine große Rolle. Im Gegensatz zu den meisten Robotik-Architekturen soll hierbei jedoch der ganzheitliche Anspruch einer integrierten kognitiven Architektur durch die Verwendung einer zentralen Wissensbasis gewahrt bleiben.

2.1. Das Rasmussen-Schema

Das Rasmussen-Schema [8] ist ein aus der Kognitionswissenschaft heraus motiviertes Modell menschlichen Verhaltens, das ursprünglich zur ergonomischen Analyse und Auslegung von Mensch-Maschine-Systemen entwickelt wurde. Es basiert auf den drei Regulationsebenen des fertigkeitsbasierten, regelbasierten und wissensbasierten Verhaltens, die je nach Situation und Trainingsstand zum Teil gleichzeitig zur Anwendung kommen (BILD 2).

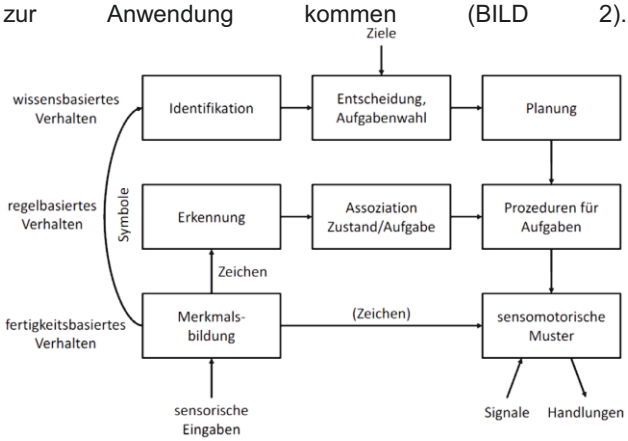

BILD 2. Regulationsebenenmodell menschlicher Informationsverarbeitung. [8] zitiert nach [9]

Das fertigkeitsbasierte Verhalten beschreibt unbewusst ablaufende, hochautomatisierte sensomotorische Fähigkeiten des Menschen. Diese hochintegrierte kontinuierliche Regulationsebene verläuft ohne Zuteilung von Aufmerksamkeitsressourcen. Ausgelöst werden diese Bewegungsmuster entweder durch aus den sensorischen Eingaben durch Merkmalsbildung erzeugten Zeichen oder durch Prozeduren der nächsthöheren Ebene.

Regelbasiertes Verhalten beschreibt die Aktivierung solcher Prozedursequenzen in bekannten, häufig wiederkehrenden Situationen. Wird eine solche Situation durch Zeichen der Merkmalsbildung erkannt, und kann sie mit einer gespeicherten Aufgabe assoziiert werden, so resultiert dies in der Ausführung der zugeordneten Prozedur unter mäßiger Belastung der Aufmerksamkeitsressourcen. Das Erlernen dieser Regeln erfolgt entweder durch eigene Erfahrung mit erfolgreichen Verhaltensmustern in der jeweiligen Situation oder durch Austausch bzw. Instruktionen durch andere Personen.

In unbekannten Situationen, d.h. Situationen für die kein regelbasiertes Verhalten möglich ist, kommt die höchste Ebene, das wissensbasierte und an expliziten Zielen orientierte Verhalten zum Tragen. Diese Art der semantikbehafteten Deliberation bietet größtmögliche Flexibilität der Handlungen, beansprucht jedoch auch die Aufmerksamkeits- und Verarbeitungsressourcen maximal.

2.2. Das interpretierte Rasmussen-Schema

Im Kontext der „*Dual-Mode Cognitive Automation*“ [1] schlagen Onken & Schulte eine interpretierte Variante des klassischen Rasmussen-Schemas BILD 2 unter Berücksichtigung informationsverarbeitender Aspekte vor (BILD 3).

Die vorgeschlagenen Änderungen stellen eine mit etablierten informationstechnischen Begriffen konsistente Darstellung sicher. So steht der wissensverarbeitende Charakter im Vordergrund. Dies kommt insbesondere in der generischen Darstellung der innerhalb des grauen Rechtecks benannten kognitiven Subfunktionen in BILD 4 zum Ausdruck.

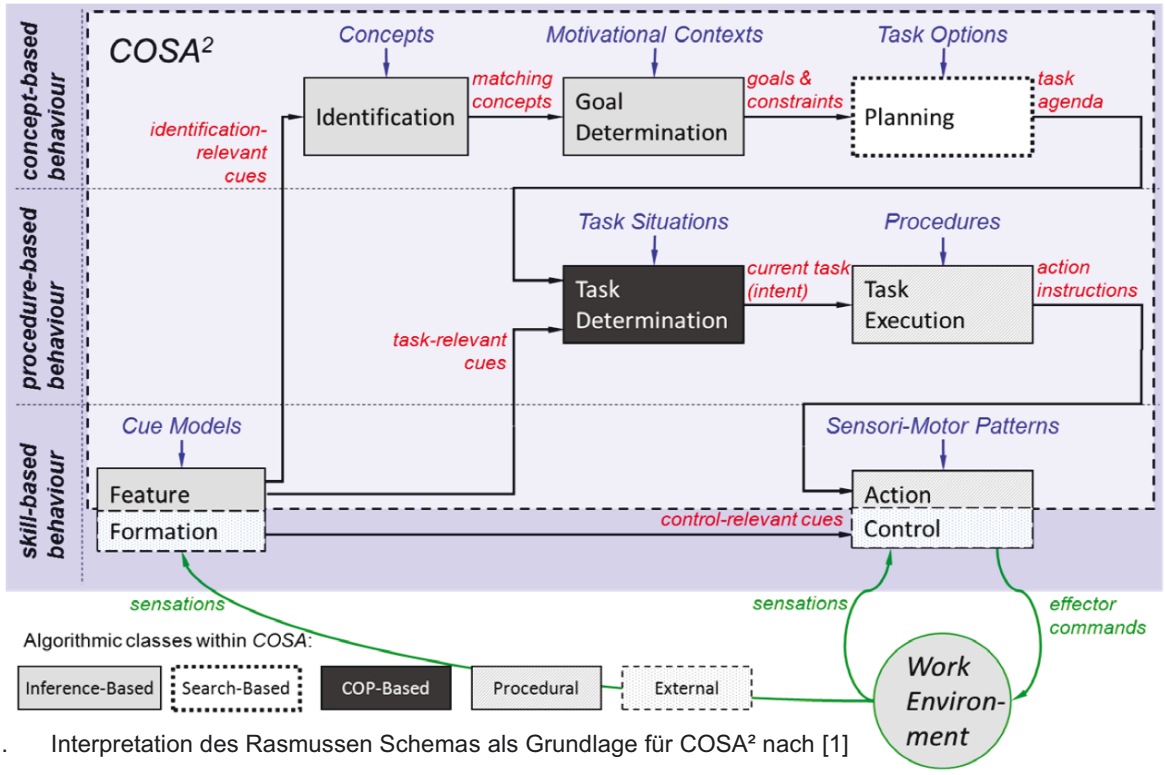

BILD 3. Interpretation des Rasmussen Schemas als Grundlage für COSA² nach [1]

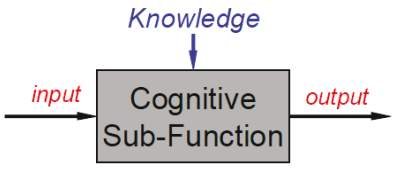

BILD 4. Formale Beschreibung kognitiver Subfunktionen ([1], S. 67)

Das in blau dargestellte *a-priori* Wissen bezeichnet implizit oder explizit vorhandenes Hintergrundwissen. Dieses wird in Form von semantisch codierten „Chunks“ im Langzeitspeicher abgelegt und über die Verarbeitungsdauer als statisch angesehen. In rot dargestellt hingegen ist das *situative Wissen*, das als Ein- und Ausgangsdatum der Subfunktion dient.

Die konsequente Anwendung dieser Darstellungsweise führte zur Umbenennung der Subfunktionen „*Prozeduren für Aufgaben*“ zu „*Task Execution*“ und „*sensomotische Muster*“ zu „*Action Control*“, da diese eher Wissen als Funktionen bezeichnen. Die Subfunktionen „*Erkennung*“ und „*Assoziation Zustand/Aufgabe*“ wurden zur „*Task Determination*“ kombiniert.

Um Verwechslungen mit den in der Informatik etablierten Begriffen wissensverarbeitendes und regelbasiertes System zu vermeiden und stattdessen die primär relevanten Wissenselemente zu betonen, wurden zudem die Abstraktionsebenen umbenannt. Im Folgenden werden die Ebenen mit Bezug auf das interpretierte Rasmussen-Schema entsprechend als *fertigkeitsbasiert* (engl. *skill-based*), *prozedurbasiert* (engl. *procedure-based*) und *konzeptbasiert* (engl. *concept-based*) bezeichnet.

2.2.1. Feature Formation

Auf der unterbewussten, fertigkeitsbasierten Ebene ist die „*Feature Formation*“ Subfunktion besonders relevant. Ihre Aufgabe ist die Abstraktion von kontinuierlichen Sinneseindrücken hin zu räumlich-zeitlichen „*cues*“ (dt. in etwa Signale, Zeichen).

Analog zur Unterscheidung zwischen Signalen, Zeichen und Symbolen im klassischen Rasmussen-Modell kann ein Sinneseindruck je nach Kontext unterschiedlich interpretiert werden:

- „*Control-relevant-cues*“ (auf der fertigkeitsbasierten Ebene) zur direkten Steuerung sensomotorischer Muster.
- „*Task-relevant cues*“ (auf der prozedurbasierten Ebene) aktivieren gespeicherte Prozeduren. Dabei wird zwischen aufgrund der „*Task Agenda*“ erwarteten *cues* (Typ 1) und unvorhergesehen Ereignissen (Typ 2) unterschieden.
- „*Identification-relevant cues*“ (auf der konzeptbasierten Ebene) treten schließlich beim aufmerksamen Beobachten des Arbeitsprozesses bzw. der Reflexion über Vorgänge in der Umgebung auf. Hierbei werden die wahrgenommenen Sinneseindrücke zu einem oder mehreren, gleichzeitig auftretenden Symbolen verarbeitet.

2.2.2. Action Control

Die fertigkeitsbasierte Subfunktion „*Action Control*“ dient der Aussteuerung unmittelbarer „*sensomotorischer Muster*“. Diese erlernten Muster dienen der Kontrolle des Bewegungsapparats unter Berücksichtigung verschiedener Regel- und Feedbackschleifen. Diese Handlungen werden durch „*action instructions*“ oder durch „*control-relevant cues*“ aus der fertigkeitsbasierten Ebene ausgelöst.

2.2.3. Task Determination

Die prozedurbasierte Subfunktion „*Task Determination*“ kontrolliert die Ausführung der aktuellen Handlungen (der sog. „*tasks*“) in bekannten Aufgabensituationen. In BILD 5 ist hierzu der durch die Wahrnehmungen der „*Feature Formation*“ Subfunktion aufgespannte „*situational feature space*“ symbolisch dargestellt.

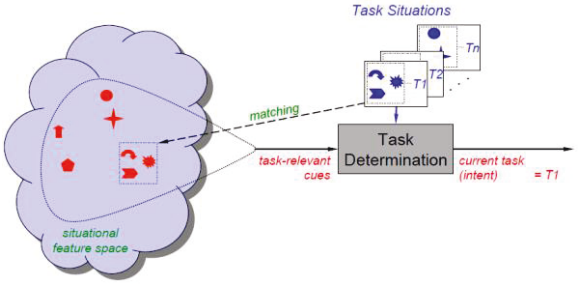

BILD 5. Kognitive Subfunktion „Task Determination“ ([1], S. 70)

Er beinhaltet bekannte *cues*, aus denen im Zuge eines Mustererkennungsprozesses die aktuellen „*task-relevant cues*“ erkannt werden. Sollten mehrere *task-relevant cues* gleichzeitig aktiviert sein, so muss priorisiert werden. Obwohl Ziele auf dieser Ebene nicht explizit berücksichtig werden, so sind sie doch implizit aufgrund der ursprünglichen Lernphase der Reaktion vorhanden.

2.2.4. Task Execution

Sobald die aktuelle Aufgabe bestimmt ist, kann mit der Durchführung begonnen werden. Die Hauptfunktionalität dieser Subfunktion („*Task Execution*“) ist es daher, die abstrakt formulierte Aufgabe der prozedurbasierten Ebene auf das Granularitätslevel einzelner Aktionen (sog. „*actions*“) herunterzubrechen. Hierbei handelt es sich um „kochrezeptartige“ Sequenzen (sog. „*Procedures*“), die je nach Trainingslevel sehr umfänglich sein können.

2.2.5. Identification

Wie alle Subfunktionen auf der wissensbasierten Ebene kommt die Subfunktion „*Identification*“ in unbekannten Situationen (d.h. Situationen für die kein *task-relevant cue* bekannt ist) zum Einsatz, um aus bekanntem a-priori Wissen eine geeignete Handlungsstrategie herzuleiten. Hierzu dienen zunächst die aus dem „*situational feature space*“ der „*Feature Formation*“ Subfunktion erkannten „*identification-relevant cues*“. Diese aktivieren hierbei das in Form von „*Concepts*“ hinterlegte a-priori Wissen. Durch die Assoziation dieses „Hintergrundwissens“ werden neue Informationen über die Umgebung inferriert und stellen als „*Matching Concepts*“ die Grundlage für die Situationsbewertung dar.

2.2.6. Goal Determination

Die Subfunktion „*Goal Determination*“ vergleicht anschließend das in Form von „*matching concepts*“ abstrahierte Situationsverständnis mit gewünschten Zielständen, abgeleitet aus dem Wissen über „*Motivational Contexts*“. Die „*Goal Determination*“ Subfunktion nimmt eine Mittlerrolle zwischen der „*Identification*“ und der „*Planning*“-Subfunktion war, da Abweichungen der Situation vom gewünschten Sollzustand die Aktivierung eines Zieles (genau diesen wieder zu erreichen) nach sich zieht.

2.2.7. Planning

Der durch „*Goal Determination*“ angezeigten Abweichungen der wahrgenommenen Ist-Situation von der gewünschten Soll-Situation kann durch die Ausführung von Handlungen entgegengewirkt werden. Die Aufgabe der Subfunktion „*Planning*“ besteht darin, eine geeignete Sequenz an Handlungen (die sog. „*task agenda*“) zu erzeugen, um die Umwelt in einen Zustand zu überführen, in dem die „*Goal Determination*“ Subfunktion keine Zieleverletzung mehr erkennt. Die generell ausführbaren Handlungsmöglichkeiten entstammen dem Wissen über „*Task Options*“.

BILD 6. Projektionsbasierter Planungsalgorithmus

Wie in BILD 6 gezeigt, wird der Anspruch der zentralen Wissensrepräsentation durch die Wiederverwendung des a-priori Wissens der Subfunktionen *Identification* und *Goal Determination* sichergestellt. Eine im aktuellen Zustand ausführbare *Task Option* wird als *Task Candidate* eine oder mehrere Zeitschritte die Zukunft projiziert. Der daraus entstehende Zustand wird mit Hilfe der *Identification* und *Goal Determination* Subfunktionen interpretiert und auf Zielverletzungen bewertet bis eine zielführende Handlungssequenz gefunden wurde. Diese wird als *task agenda* abgespeichert und dient anschließend der „*Task Determination*“ Subfunktion mit den zugehörigen Typ 1 cues als Eingabe zur sukzessiven Durchführung der Handlungen.

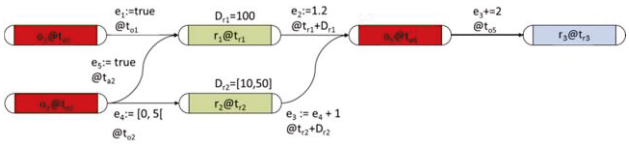

BILD 7. Berechnete Struktur einer Task Agenda

In BILD 7 ist die Struktur einer umfangreicheren Task Agenda gezeigt. In rot dargestellt sind zwei parallele Handlungen (o_1 und o_2) die zu den (noch nicht festgelegten) Zeitpunkten t_{o1} und t_{o2} ausgeführt werden. Die ausgehenden Kanten bezeichnen die erwarteten Effekte der Handlung (e_1= e_5:=*true;* e_5:=*[0,5[*) welche wiederum in der Projektionsphase die in grün dargestellten Identifikationsregeln r_1 und r_2 (mit Dauer D_{r1}=*100 bzw.* D_{r2}=*[10,50]*) auslösen. Danach kann Handlung o_5 ausgeführt werden, welche über Regel r_3 in einen Zielzustand führt.

3. IMPLEMENTIERUNG COSA²

Das im vorigen Kapitel dargelegte Prozessmodell auf Grundlage des interpretierten Rasmussen-Schemas, sowie die Verarbeitungsschritte durch die einzelnen kognitiven Subfunktionen müssen zur Umsetzung in einer kognitiven Systemarchitektur auf spezifische Algorithmen abgebildet werden. Um zudem dem Anspruch einer zentralen Wissensbasis zu genügen, ist es erforderlich, eine Wissensrepräsentationsform zu verwenden, die von allen Subfunktionen konsistent verwendet werden kann.

Die Wahl hängt vor allem von der Modellierung der Umgebungsbedingungen ab, in denen die kognitive Systemarchitektur operieren soll. Erfassen die Sensoren zu jedem Zeitpunkt alle für die Entscheidung relevanten Daten, so wird die Umgebung als *vollständig beobachtbar* bezeichnet, ansonsten als *partiell beobachtbar*. In einer *deterministischen* Umgebung ergibt sich der nachfolgende Umgebungszustand deterministisch aus dem aktuell erfassten und der vom Agenten durchgeführten Handlung, andernfalls gilt die Umgebung als *stochastisch*. Ist die Umgebung nur partiell beobachtbar, so kann eine Handlung als stochastisch erscheinen. In einer *diskreten* Umgebung werden die Zustände der Umgebung über diskrete Zustandsgrößen beschrieben, andernfalls heißt die Umgebung *kontinuierlich*.

Die in diesem Artikel beschriebene Implementierung COSA² zielt auf den (einfachsten) Fall, einer voll beobachtbaren, deterministischen, diskreten Umgebung. Dieser Anwendungsfall ist in den folgenden Tabellen jeweils hellgrau hinterlegt. Über geschickte Modellierung (etwa von Wahrscheinlichkeitsverteilungen) bzw. die Wahl alternativer Algorithmen bzw. Speicherformen kann das Konzept jedoch auch auf komplexere Umgebungsbedingungen übertragen werden. Fett gedruckt sind Verfahren die im kognitiven Radar Anwendung finden, und somit repräsentativ für eine stochastische Umgebung mit realen Sensordaten sind.

3.1.1. Repräsentationsformen der Speicher

In TAB 1 sind verschiedene gebräuchliche Repräsentationsformen für Kurz- und Langzeitspeicher in kognitiven Architekturen aufgezeigt.

Name	*Ausdrucks-stärke*	*Verständ-lichkeit*	*Umgebung*	*Beispiel*
Prädikaten-logik	+	+	Deterministisch	BDI, Prolog
Fuzzy-Logik	+	-	Deterministisch, Partiell beobachtbar, Stochastisch	(Zadeh, 1965)
Graph	-	+	Deterministisch	COSA, SOAR
Ontologie	+	+	Deterministisch	OWL, OWL2
KNN	+	-	Deterministisch, Partiell beobachtbar, Stochastisch	PDP+

TAB 1. Qualitativer Vergleich zwischen Speicherrepräsentationsformen

Prädikatenlogik stellt eine in der Mathematik, Philosophie und Informatik vielfach verwendete und etablierte Repräsentationsform dar. Dementsprechend sind die Verbreitung sowie die Verständlichkeit durch menschliche Nutzer und Wissensingenieure hoch. Die

Ausdrucksstärke ist ebenfalls hoch, da auch Quantoren unterstützt werden. Prädikatenlogik stellt die klassische Repräsentationsform symbolbasierter Ansätze dar, und ist somit primär für deterministische, vollständig beobachtbare Umgebungen geeignet.

Fuzzylogik [10] stellt eine Erweiterung der klassischen Logik dar, und ist somit mindestens so ausdrucksstark wie die zugrunde liegende Aussagen- bzw. Prädikatenlogik. Die Verständlichkeit ist jedoch eingeschränkt, da der unscharfe Modellierungsprozess nicht immer eindeutig ist. Dies jedoch macht die Fuzzylogik auch für partiell beobachtbare bzw. stochastische Umgebungen einsetzbar.

Graphen stellen ebenfalls eine in der Informatik häufig verwendete Wissensrepräsentationsform dar. Sie sind ebenfalls intuitiv handhabbar. Die Ausdrucksstärke ist jedoch geringer, da sich nativ zunächst nur Beziehungen zwischen Knoten in Form von gerichteten und gewichteten Kanten darstellen lassen. Somit liegt eine Eignung für deterministische, vollständig beobachtbare Umgebungen vor.

Ontologiesprachen wie OWL [11] sind in ihrer Ausdrucksstärke häufig konfigurierbar und umfassen meist die Ausdrucksstärke der Prädikatenlogik. Die Verwendung von Ontologien ist gut verständlich und in deterministischen, vollständig beobachtbaren Umgebungen anwendbar.

Künstliche neuronale Netze [12] sind in der Lage, beliebige logische sowie arithmetische Funktionen abzubilden. Die Effizienz des Ansatzes hängt jedoch stark von der gewählten Netzwerktopologie ab. Die subsymbolische Repräsentationsform ist somit nicht intuitiv verständlich und die Topologie häufig schwer zu erfassen bzw. zu modellieren. Aufgrund der Unterstützung statistischer, datengetriebener Lernverfahren und der Generalisierungsfähigkeit eignet sich der Ansatz sehr gut für stochastische und partiell beobachtbare Umgebungen.

3.1.2. Methoden zur Situationsbewertung

In TAB 2 sind verschiedene gebräuchliche Methoden zur Situationsbewertung angegeben.

Name	*Speicher*	*Verständ-lichkeit*	*Umgebung*	*Beispiel*
Muster-Erkennung /CBR	KNN, Fuzzy-Logik	-	Partiell beobachtbar, Stochastisch	Bild-/ Signal-verarbei-tung
Model-checking	Ontologie, Prädikaten-logik	+	Deterministisch	BDI, Prolog
Produk-tionen-systeme	Graphen, Prädikaten-logik	+	Deterministisch	COSA, SOAR, ACT-R
Unsicheres Schliessen	Wahr-scheinlich-keitsver-teilungen	0	Partiell beobachtbar, Stochastisch	Bayes'sch es Netz, MLE,MAP Kalman Filter
Unscharfes Schliessen	Fuzzy Logik	0	Partiell beobachtbar, Stochastisch	[10]

TAB 2. Qualitativer Vergleich zwischen Methoden zur Situationsbewertung

Mustererkennung bzw. CBR [13] eignet sich als Verfahren zur Symbolisierung von subsymbolischen Sensordaten. Je nach verwendetem Klassifikator kommen künstliche neuronale Netze, Fuzzy Logik, k-nächste-Nachbarn, Entscheidungsbäume oder Ähnliches zum Einsatz. In Verbindung mit KNN repräsentiert die Netztopologie den Langzeitspeicher, die Aktivierungspotentiale den Arbeitsspeicher. Aufgrund der subsymbolischen Natur kommen statistische, datengetriebene Lernverfahren zum Einsatz, die für den menschlichen Bediener schwer nachvollziehbar sind, aber auch für den Einsatz in partiell beobachtbaren oder stochastischen Umgebungen geeignet sind.

Das Konzept des Model Checking wird meist mit Prädikatenlogik oder Ontologien eingesetzt. Der Langzeitspeicher enthält das Weltmodell durch Formeln, die im jeweiligen logischen Kalkül repräsentiert sind, der Arbeitsspeicher Formeln, die Wahrnehmungen beschreiben oder Variablen binden. Die Methode zeichnet sich durch eine solide formale-logische Beschreibung und somit eine hohe Verständlichkeit aus. Sie ist für deterministische Umgebungen anwendbar. Model Checking greift auf effiziente Algorithmen zurück, welche ausschließlich relevante Fakten inferrieren und kommt zur Situationsbewertung in vielen BDI-Agentenarchitekturen zum Einsatz [14].

Produktionensysteme im forward-chaining Modus expandieren hingegen alle ableitbaren Fakten [15]. Der Langzeitspeicher wird durch Produktionsregeln gebildet, der Arbeitsspeicher durch Graphen oder Logik. Die Methode ist ebenfalls formal-logisch eindeutig beschreibbar und intuitiv verständlich. Sie eignet sich für deterministische Umgebungen. Produktionensysteme bilden die Grundlage vieler integrierter kognitiver Architekturen.

Unsichere Schlussverfahren basieren auf Methoden der Schätztheorie [16] und eigenen sich somit für partiell beobachtbare oder stochastische Umgebungen. Ein im Arbeitsspeicher hinterlegter Messwert entstammt somit einer Wahrscheinlichkeitsverteilung. Das im Langzeitspeicher abgelegte Hintergrundwissen kann etwa in Form von Bayes'schen Netzen [17] oder durch das Dynamikmodells eines Kalman Filters repräsentiert sein.

Der Langzeitspeicher unscharfer Schlussverfahren wird durch unscharfe Schlussregeln repräsentiert, der Arbeitsspeicher durch Formeln in Fuzzylogik. Die Modellierung des fuzzylogischen Kalküls bzw. Formeln erfolgt empirisch und ist somit nicht intuitiv oder formal-logisch herleitbar. Der Ansatz eignet sich für partiell beobachtbare oder stochastische Umgebungen.

3.1.3. Methoden zur Handlungsplanung

In TAB 3 sind verschiedene gebräuchliche Methoden zur Handlungsplanung angegeben.

Der Ansatz des klassischen Planens [18] basiert auf heuristischer Vorwärtssuche und bietet maximale Flexibilität bei der Lösungsfindung. Da der Suchraum jedoch nicht vorstrukturiert ist, kann er exponentiell groß sein. Der Suchvorgang endet erst mit dem Auffinden einer validen oder der optimalen Lösung [19]. Der Arbeitsspeicher wird durch den Startzustand gekennzeichnet, der Langzeitspeicher durch das Domänenmodell in STRIPS, ADL oder PDDL [5]. Die symbolische Beschreibung und der resultierende Plan

sind gut verständlich. Der Ansatz eignet sich für deterministische Umgebungen.

Name	*Speicher*	*Verständ-lichkeit*	*Umgebung*	*Beispiel*
Klassisches Planen	Graphen, Prädikaten-logik	0	Deterministisch	PDDL, FF
Anytime-Planung	Graphen, Prädikaten-logik	0	Deterministisch	PDDL, LPG
Means-Ends	Graphen, Prädikaten-logik	+	Deterministisch	GPS, COSA
HTN	Graphen, Prädikaten-logik	+	Deterministisch	SHOP
Entscheiden unter Unsicherheit	Wahrschein-lichkeitsver-teilungen	-	Partiell beobachtbar, Stochastisch	MDP, POMDP, PPDDL

TAB 3. Qualitativer Vergleich zwischen Methoden zur Handlungsauswahl

Die Anytime-Planung [20] unterscheidet sich von der klassischen Planung durch die frühe Ausgabe einer gültigen Lösung, die sukzessive in der Planqualität verbessert wird. Meist kommen lokale Suchverfahren zum Einsatz, d.h. die Lösung ist nicht deterministisch reproduzierbar.

Die Means-Ends-Planung [2] zeichnet sich durch einen vorstruktuierten Lösungsraum aus, der somit formal-logisch einfacher zu erfassen und gut verständlich ist. Der Arbeitsspeicher beinhaltet den Anfangszustand, der Langzeitspeicher den Means-Ends-Graphen. Nachteilig ist beschrieben, die Means-Ends Methode als zu starr erwies.

Hierarchische Task-Netzwerke [21] bieten durchalternative Task-Zerlegungen etwas mehr Flexibilität bei der Ausführung. Gleichzeitig ist der Lösungsraum stärker vorstruktuiert als bei der klassischen Planung und somit formal-logisch besser erfassbar und gutverständlich. Der Arbeitsspeicher beinhaltet den Anfangszustand, der Langzeitspeicher die HTN-Struktur. Der Ansatz eignet sich für deterministische Umgebungen.

Das Entscheiden unter Unsicherheit berücksichtigt stochastische Resultate einer Handlung. Mathematisch erfolgt die Modellierung als Markov-Entscheidungs-Problem (MDP) oder partiell beobachtbares MDP (POMDP). Eine etwas komfortablere Notation bietet die probabilistische Planungssprache PPDDL [22]. Die Modellierung der probabilistischen Zusammenhänge ist meist nicht offensichtlich und der Suchraum größer als beim klassischen Planen. Der Ansatz ist jedoch auch für partiell beobachtbare oder stochastische Umgebungen geeignet.

3.1.4. Methoden zur Planausführung

In TAB 4 sind verschiedene gebräuchliche Methoden zur Planausführung angegeben.

Constraint Satisfaction / Optimisation Methoden (CSP / COP) verwenden effiziente lineare Optimierungsmethoden zum Lösen von Scheduling Problemen. Die Notation erfolgt meist in Intervall-Logik und ist somit gut verständlich. Die Methode bietet durch das kontinuierliche Scheduling Robustheit bei der Ausführung in stochastischen oder partiell beobachtbaren Umgebungen. Größere Abweichungen bei der Ausführung erfordern jedoch meist eine komplette Neuplanung.

Name	*Speicher*	*Verständ-lichkeit*	*Umgebung*	*Beispiel*
CSP/COP	Graphen, Prädikaten-logik	+	Deterministisch, Partiell beobachtbar, Stochastisch	EUROPA
RAPs	Graphen, Prädikaten-logik	+	Deterministisch, Partiell beobachtbar, Stochastisch	[23]
Planaus-führungs-systeme	Graphen, Prädikaten-logik	+	Deterministisch, Partiell beobachtbar, Stochastisch	PRS, Plexil

TAB 4. Qualitativer Vergleich zwischen Methoden zur Planausführung

Reactive Action Packages (RAPs [23]) bieten Detailumplanung und Ausführungsmonitoring für stochastische oder partiell beobachtbare Umgebungen. Das Konzept basiert auf Abstraktion, indem verschiedene Ausführungsalternativen modelliert werden. Die Modellierung entspricht prädikatenlogischen Planungssprachen und ist somit gut verständlich.

Komplexere Planausführungssysteme wie PRS oder PLEXIL [24] basieren meist auf dem Konzept der RAPs. Die Ausdrucksstärke der Planausführungssprachen ist hierbei noch erhöht, sodass die Herausforderung in guter Performanz und formal-logisch eindeutig definierter Ausführungssemantik liegt.

3.1.5. Weiterschaltung der Task Agenda

Bei der Implementierung COSA² wurden CSP/COP Methoden zur Planausführung verwendet. Als Constraints werden die Abhängigkeiten der Task Agenda codiert, als Entscheidungsvariablen die Ausführungszeiten bzw. weitere anpassbare Parameter. Das CSP/COP wird mit jedem Zyklus durch den kommerziellen Solver CPlex [25] für die aktuelle Situation neu gelöst.

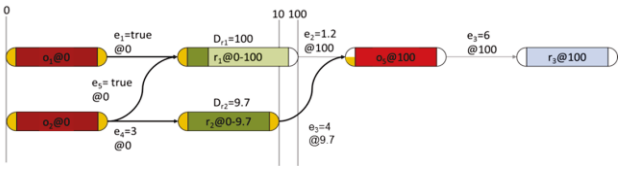

BILD 8. Berechnete Task Agenda während der Ausführung

Handlungen bzw. Regeln werden ihrer Reihenfolge in der Agenda entsprechend *aktiviert*. Dies stellt sicher, dass alle Strukturbedingungen der Agenda transitiv eingehalten werden. Eine aktivierte Handlung bzw. Regel ist durch einen ausgefüllten orangen Halbkreis an der linken Seite des Rechtecks gekennzeichnet (BILD 8).

Zur Weiterschaltung einer Handlung aus der „*task agenda*“ müssen drei Bedingungen erfüllt sein:

1. Die Vorbedingungen der Handlung müssen in der aktuellen Situation erfüllt sein.

2. Der berechnete Ausführungszeitpunkt muss verstrichen sein.

3. Die Handlung muss aktiviert sein.

Neben der konzeptbasierten automatischen Planung der „*task agenda*“ können auch reaktive, d.h. vorgefertigte und meist kurze Handlungssequenzen abgearbeitet werden. Diese werden als „*task-relevant cues*“ vom Typ 2 spezifiziert.

Da sich reaktives Verhalten auf die aktuelle Situation bezieht, ist es unmittelbar anzuwenden und unterbricht somit, wie in BILD 9 gezeigt, die langfristige Ausführung der konzeptbasierten „*task agenda*“. In der Abbildung wurde durch einen „*task-relevant cue*“ vom Typ 2 in der „*Feature Formation*“ Subfunktion reaktiv die Sequenz $\langle o_3, r_2 \rangle$ ausgelöst.

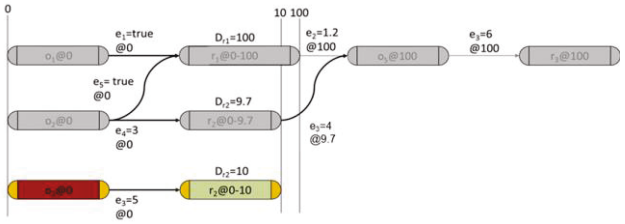

BILD 9. Berechnete Task Agenda während der Ausführung

Deren Ausführung unterscheidet sich nicht vom beschriebenen Vorgehen des vorigen Kapitels. Sobald die Sequenz durchgeführt ist, schaltet das System zurück in den konzeptbasierten Modus und versucht die Ziele zu erreichen.

3.1.6. Synthese zur Gesamtarchitektur

Der in BILD 10 gezeigte Zustandsautomat repräsentiert den Zustand des kognitiven Prozesses. Dieser durchläuft, wie oben beschrieben, sequenziell die einzelnen kognitiven Subfunktionen.

Der Prozess startet mit dem Einlesen der Eingangsdaten aus der Netzwerkschnittstelle (*Read_Input*). Diese Daten werden an einen dezidierten Inputknoten im Arbeitsspeicher geschrieben. Anschließend beginnt der komplett in COSA² implementierte Inferenzalgorithmus. Dieser realisiert die Subfunktionen *Feature_Formation, Identification* und *Goal Determination*.

Nachdem die letzte *Motivational Contexts*-Regel zur Instanziierung der Ziele überprüft wurde, wird anhand von Zustandsvariablen die Durchführung der Planungsfunktion überprüft. Eine existierende Agenda kann aus folgenden drei Gründen ungültig werden:

1. Änderung der aktiven Ziele,
2. Aktivierung reaktiven Verhaltens, oder
3. unvorhergesehene Entwicklungen bei der Planausführung (keine Lösung des CSP/COP).

Existiert keine valide Agenda, so muss neu geplant werden. Dazu wird zunächst überprüft, ob sich das System gerade im reaktiven Modus befindet, d.h. mit der Abarbeitung prozedurbasierter Handlungssequenzen beschäftigt ist. Ist dies nicht der Fall, und im aktuellen Zustand sind Ziele verletzt, so tritt das System in den Zustand Plan_PDDL_Planning ein und startet einen externen PDDL-Planer. Tritt hierbei ein Fehler auf, oder kann das Problem nicht gelöst werden, so stoppt die Ausführung des kognitiven Prozesses.

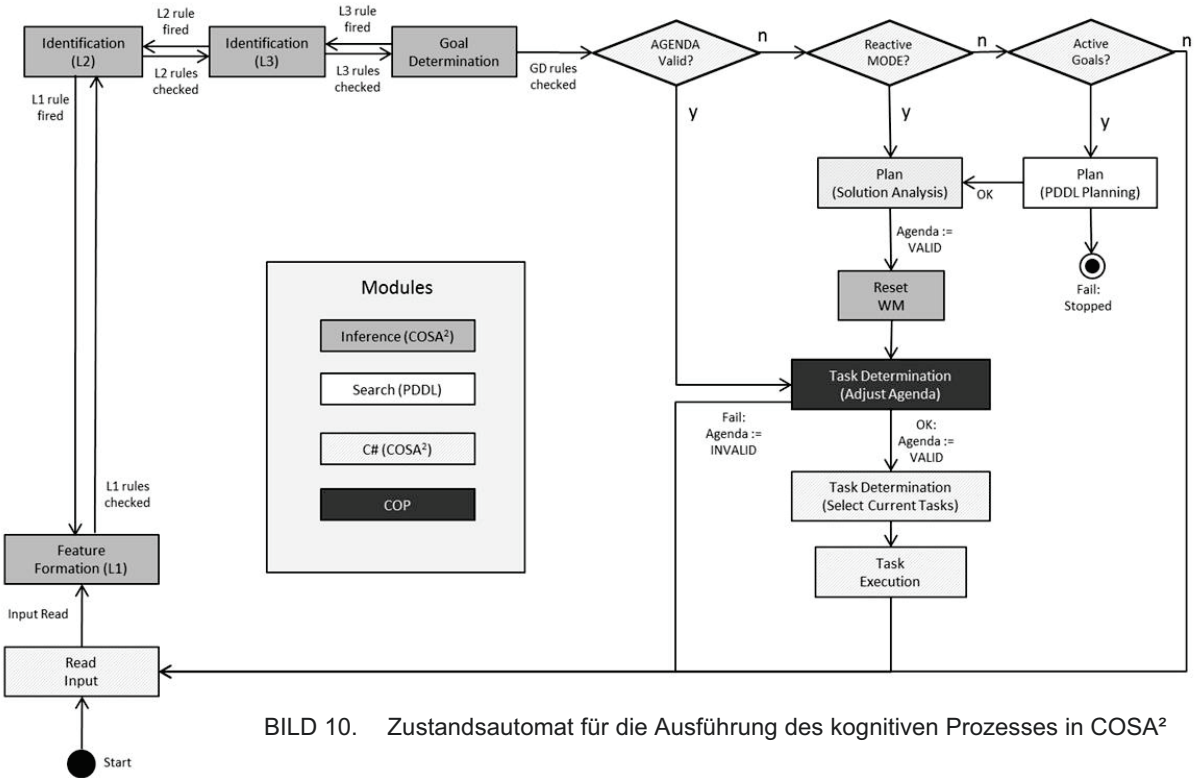

BILD 10. Zustandsautomat für die Ausführung des kognitiven Prozesses in COSA²

4. ANWENDUNGEN

Zur Validierung des Konzeptes durch den Prototypen der kognitiven Systemarchitektur COSA² für deterministische Umgebungen wurden verschiedene Anwendungen im Bereich UAV-Grundsystemmanagement [26] und Missionsplanung [27] evaluiert. Im Folgenden wird zusätzlich ein Ausblick auf die Anwendung des Konzeptes auf die stochastische Umgebung realer Sensordaten am Beispiel des kognitiven Radars gegeben.

4.1. UAV-Missionsmanagement

Die Flugerprobung erfolgte auf dem Gelände der Bundeswehr Universität München mit dem Motosegler „Graphite“ mit 1 kg Nutzlast (BILD 11). COSA² wurde im Payload Modul auf einem Intel Dual Core Atom 330 Board betrieben um einen nach unten gerichteten, mit 1080 Pixel auflösenden Aufklärungssensor zu steuern.

BILD 11. COSA² Flugversuche an der UniBwM. Oben: Graphite Flugversuchsträger. Unten: Mobile Bodenkontrollstation [27].

Das Missionsziel bestand in der auftragsbasierten Aufklärung verschiedener Zielkoordinaten (POIs) sowie der Übermittlung der Aufklärungsbilder an die Bodenkontrollstatation (BKS) über WLAN.

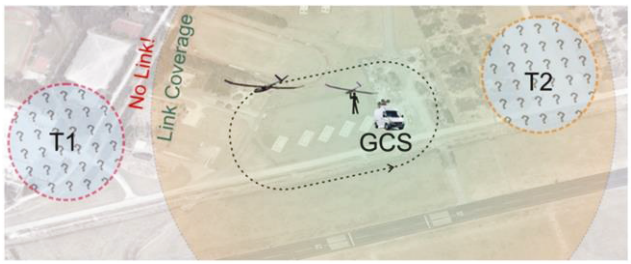

BILD 12. Flugversuchszone der UniBwM von etwa 1000m x 500m [27]

Das UAV wurde dabei an der BKS aus der Hand gestartet und dann ferngesteuert auf ca. 200 m Höhe gebracht. Anschließend wurde der Autopilot aktiviert, um wegpunktbasiert über der BKS zu kreisen und die kognitive Automation zu starten. Nun wurden verschiedene Objekte aufgeklärt, die sich entweder innerhalb der Funkreichweite der BKS (T2) oder außerhalb (T1) befanden (BILD 12).

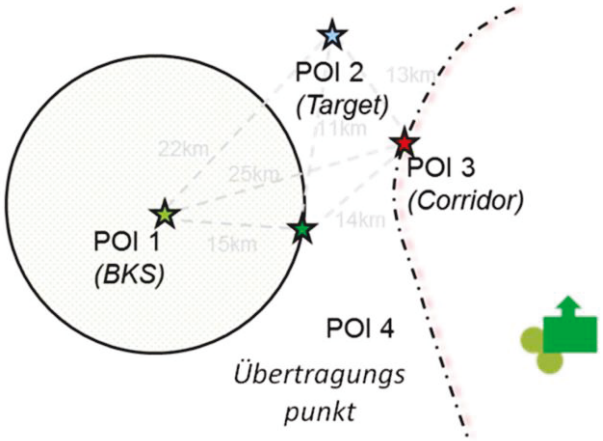

BILD 13. Geometrische Anordnung für eine auftragsbasierte Aufklärungsmission

BILD 13 zeigt die geometrische Anordnung eines Missionsszenarios mit vier POIs, die dynamisch zur Missionszeit in den Arbeitsspeicher der Architektur geladen werden können. POI1 stellt hierbei die Position der BKS dar, POI2 das Ziel, POI3 einen Transit-Korridor und POI4 einen möglichen Datenübertragungspunkt.

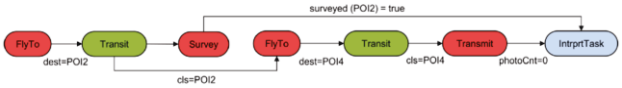

BILD 14. Verifikation der „task agenda“ für Anwendungsfall „POI2 Aufklären“

In BILD 14 ist eine daraus geplante Task Agenda gezeigt, welche den Missionsablauf strukturiert. Im Laufe des Flugversuches wurden ebenfalls komplexere Geometrien, sowie verschiedene Fehlerbedingungen und Änderungen im Missionsablauf simuliert die eine online Umplanung erforderlich machten.

In BILD 15 ist die 3D Flugtrajektorie des Beispiels gezeigt, Hier wurde ein Objekt im Bereich T1 ohne Funkverbindung aufgeklärt und zunächst entsprechend zwischengespeichert. Anschließend programmierte die kognitive Architektur den nächstliegenden Wegpunkt, für den Funkempfang prädiziert wurde und begann mit der Datenübertragung sobald die Richtantenne die Funkverbindung wieder hergestellt hatte.

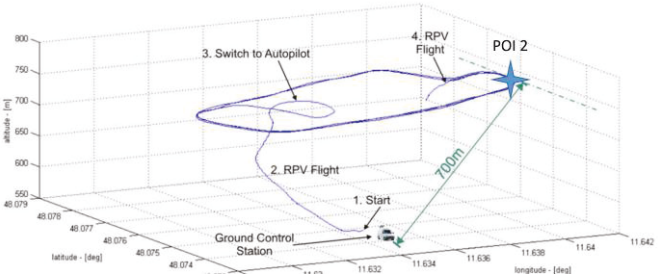

BILD 15. 3D Flugtrajektorie der auftragsbasierten Aufklärungsmission [27]

4.2. Kognitives Radar

Das in BILD 3 gezeigte Prozessmodell als Grundlage einer kognitiven Systemarchitektur lässt sich auf die sensornahe Anwendung des kognitiven Radars übertragen [29]. Als aktiver Sensor erlaubt ein Radarsystem die flexible Anpassung der Sensorsystemparameter (insbesondere der ausgesandten Wellenform) an Umgebungsbedingungen.

Der fertigkeitsbasierten Ebene entspricht somit die Adaption kontinuierlicher Parameter, etwa der Sendeleistung an das aktuelle Signal zu Rausch Verhältnis.

Im Speicher müssen nun Wahrscheinlichkeitsverteilungen der geschätzten Umgebungsparameter abgelegt werden (wie z.B. Mittelwert und Varianz der Zielentfernung, Geschwindigkeit und Winkel als Kanten eines Graphen).

Zur Abstraktion der Sensordaten durch die *Feature Formation* Subfunktion kommen maschinelle Lernverfahren wie KNN zur Klassifizierung von Objekten zum Einsatz. Zur Situationsbewertung durch a-priori Wissen kommen Bayes'sche Netze oder Kalman Filter in Frage.

Im Bereich der Handlungsplanung ist ebenfalls der stochastische Charakter bzw. die partielle Beobachtbarkeit der Umgebung zu berücksichtigen. Es kommen MDP/POMDP Verfahren zum Einsatz, wie heute auch schon beim Task-Scheduling von AESA-Multifunktionsradaren üblichen. Bisher wenig untersucht ist die konzeptbasierte Ebene, welche Missionsziele sowie externe Faktoren (z.B. Plattform-Parameter) berücksichtigt. Ebenfalls wenig erforscht ist das Zusammenspiel der Einzelfunktion hin zu einer integrierten, kognitiven Radarsystemarchitektur.

5. ZUSAMMENFASSUNG UND AUSBLICK

Es wurde ein generisches Konzept für eine kognitive Systemarchitektur basierend auf dem interpretierten Rasmussen Modell vorgestellt. Je nach Umgebungsbedingungen sind für die softwaretechnische Umsetzung verschiedene Anforderungen an Speicherformen und Algorithmen zu stellen.

Für den Fall einer deterministischen Umgebung basiert die Implementierung des Prototypen COSA² auf einem regelbasierten Produktionensystem zur Situationsbewertung, heuristischen Suchverfahren zur Handlungsplanung sowie Constraint-Satisfaction-basierten Algorithmen zur Planausführung.

Das Konzept konnte durch den Prototypen für Anwendungen aus der UAV-Missions- und Sensorplanung validiert werden.

Zukünftige Weiterentwicklungen sollten auf reale Sensordaten und stochastische Umgebungen ausgerichtet sein. Wie am Beispiel des kognitiven Radars erläutert, zeigt sich, dass das grundlegende Architekturkonzept hierfür geeignet ist, jedoch Modifikationen in den Speicherformen und Verarbeitungsalgorithmen erforderlich sind.

[1] Onken, R., & Schulte, A. (2010). System-Ergonomic Design of Cognitive Automation: DualMode Cognitive Design of Vehicle Guidance and Control Work Systems.Heidelberg: Springer.

[2] Anderson, J. R. (1996). Kognitive Psychologie (2. Ausg.). (J. Grabowski, & R. Graf, Übers.) Heidelberg, Berlin, Oxford: Spektrum Akademischer Verlag,.

[3] Laird, J., Jones, R., & Rosenbloom, P. (1987). Soar: An Architecture for General Intelligence. In Artificial Intelligence, 33(S. 1-64).

[4] Anderson, J. R. (1990). The adaptive character of thought.Psychology Press.

[5] Russel, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach(3rd. Edition Ausg.). New Jersey: Prentice Hall.

[6] Wooldridge, M. (2008). An introduction to multiagent systems.Wiley.

[7] Gat, E. (1998). On three-layer architectures. Artificial intelligence and mobile robots, S. 195-210.

[8] Rasmussen, J. (1983). Skills, Rules, and Knowledge; Signals, Signs, and Symbols, and other Distinctions in Human Performance Models. IEEE Transactions on Systems, Man and Cybernetics, 13(3), S. 257-266.

[9] Johannsen, G. (1993). Mensch-Maschine-Systeme.Berlin: Springer.

[10] Zadeh, L. A. (1965). Fuzzy sets. Information and control, 8(3), S. 338-353.

[11] OWL Working Group. (2013). OWL2 Web Ontology Language: Document Overview. Abgerufen am 06. 12. 2013 von http://www.w3.org/TR/owl2-overview/

[12] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4), S. 115-133.

[13] Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications, 7(1), S. 39-59

[14] Rao, A., & Georgeff, M. (1995). BDI Agents: From Theory to Practice. Proceedings of the First International Converence on Multi-Agent Systems.San Francisco, CA, USA, 12.-14. Juni 1995: ICMAS.

[15] Puppe, F. (1991). Einführung in Expertensysteme.Berlin: Springer.

[16] Yaakov, B. S., Li, X. R., & Thiagalingam, K. (2001). Estimation with applications to tracking and navigation. New York: Johh Wiley and Sons, 245.

[17] Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. Artificial Intelligence, 29(3), S. 241-288.

[18] Gerevini, A., & Long, D. (2005). The Language of the Fifth International Planning Competition. Plan constraints and preferences in PDDL3.Brescia, Italy: Technical

[19] Hoffmann, J., & Nebel, B. (2001). The FF Planning System: Fast Plan Generation Through Heuristic Search. Journal of Artificial Intelligence Research, 14, S. 253-302.

[20] Gerevini, A., & Serina, I. (2002). LPG: A Planner Based on Local Search for Planning Graphs with Action Costs. Proeedings on AIPS, (S. 281-290). Report, Department of Electronics for

Automation, University of Brescia, Italy.
[21] Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. Artificial Intelligence, 5(2), S. 115-135.
[22] Younes, H., & Littman, M. L. (2004). PPDDL1. 0:The language for the probabilistic part of IPC-4. Proceedings on International Planning Competition.
[23] McDermott, D. (1991). A reactive plan language.Technical Report TR-864, Yale University.
[24] Verma, V., Estlin, T., Jonsson, A., Pasareanu, C., Simmons, R., & Tso, K. (2005). Plan execution interchange language (PLEXIL) for executable plans and command sequences. Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space.Munich, Germany: iSAIRAS.
[25] IBM Corporation. (2013). IBM Cplex Optimizer. Abgerufen am 20. 12. 2013 von http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
[26] Brüggenwirth, S., Pecher, W., & Schulte, A. (2011). Design Considerations for COSA². IEEE Symposium Series on Computational Intelligence.Paris: IEEE SSCI.
[27] Böhm, F., Clauss, S., Brüggenwirth, S., & Schulte, A. (2012). Cognitive UAV Ressource Management Allowing Task-based Mission Execution Under Data Link Limitations. Proceedings of the 31st Digital Avionics Systems Conference (DASC).Williamsburg.
[28] Brüggenwirth, S., & Schulte, A. (2012). COSA² – A Cognitive System Architecture with Centralized Ontology and Specific Algorithms. IEEE International Conference on Systems, Man, and Cybernetics.Seoul, Korea: IEEE SMC.
[29] Ender, J. & Brüggenwirth, S. (2015). "Cognitive Radar-Enabling Techniques for Next Generation Radar Systems.", Proceedings of the International Radar Symposium, Dresden