Universität der Bundeswehr München
Fakultät für Luft- und Raumfahrttechnik
Institut für Systemdynamik und Flugmechanik

KOGNITIVE AUTOMATION ZUR KOOPERATIVEN UAV-FLUGFÜHRUNG

Dipl.-Ing. (FH) Claudia Meitinger

Vollständiger Abdruck der bei der
Fakultät für Luft- und Raumfahrttechnik
der Universität der Bundeswehr München
zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

eingereichten Dissertation

Vorsitzender: Prof. Dr.-Ing. Helmut Rapp
1. Berichterstatter: Prof. Dr.-Ing. Axel Schulte
2. Berichterstatter: Prof. Dr.-Ing. Frank Thielecke

Diese Dissertation wurde am 3. April 2008 bei der Universität der Bundeswehr München, 85577 Neubiberg, eingereicht und durch die Fakultät für Luft- und Raumfahrttechnik am 23. April 2008 angenommen.

Tag der Prüfung: 9. Juli 2008

Hilgertshausen, 17. September 2008

Vorwort

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftliche Mitarbeiterin am Institut für Systemdynamik und Flugmechanik der Fakultät für Luft- und Raumfahrttechnik an der Universität der Bundeswehr München.

Mein besonderer Dank gilt Prof. Dr.-Ing. Axel Schulte, der es mir ermöglichte, in einem spannenden Themengebiet wissenschaftlich zu arbeiten. Seine kontinuierliche Unterstützung durch wertvolle Kommentare und Anregungen, aber auch bei der Durchführung meiner Promotionseignungsprüfung, hat diese Arbeit ebenso erst möglich gemacht wie das Bereitstellen eines familienfreundlichen Arbeitsplatzes. Auch das vertrauensvolle Arbeitsverhältnis und die Gelegenheit, in einem internationalen Umfeld tätig zu sein sowie durch „Sperrfeuer“ wertvolle Einblicke in Projektakquisition und -durchführung zu erlangen, haben mir eine sehr schöne Zeit am Institut beschert.

Herzlich möchte ich mich auch bei Prof. Dr.-Ing. Frank Thielecke für die Übernahme des Koreferats bedanken.

Herrn Dr.-Ing. Werner Fohrer danke ich ganz besonders für die sorgfältige Durchsicht des Manuskripts. Allen meinen Kollegen, die nach und nach zum Team gestoßen sind, namentlich Mike Kriegel, Diana Donath, Andreas Benzler, Felix Maiwald, Alexander Matzner, Gregor Jarasch, Johann Uhrmann, Andreas Rauschert und Ruben Strenzke, möchte ich für die gute Zusammenarbeit und das angenehme Klima danken. Besonders zu Beginn meiner Zeit am Institut standen mir Dr.-Ing. Henrik Putzer und Dr.-Ing. Andreas Frey jederzeit für fachliche und organisatorische Fragen zur Verfügung, obwohl sie selbst mehr als genug zu tun hatten.

Dem Stammpersonal danke ich für die Unterstützung bei administrativen Vorgängen aller Art, das Einrichten von Laborarbeitsplätzen und die nutzungsgerechte Anfertigung diverser Hardware trotz unzureichender Spezifikation.

Besonders bedanken möchte ich mich auch bei den Piloten und Waffensystemoffizieren, die es mit großem persönlichen Engagement trotz enger zeitlicher Rahmenbedingungen einrichten konnten, an der Systemevaluierung mitzuwirken und sehr wertvolle Ergebnisse zu liefern.

Meinen Eltern schulde ich großen Dank dafür, dass sie mir jederzeit ermöglicht haben, meinen Weg zu gehen und mich bei all meinen Entscheidungen unterstützt haben. Zusammen mit meiner Schwester haben sie mir stets ein intaktes Familienumfeld bereitet, das heute von einem freundschaftlichen Umgang und viel Spaß miteinander geprägt ist.

Dr. med. Katharina Gockel und Daniela Uhrmann danke ich für den aufmunternden Beistand und die entspannenden und spaßigen „Mädelsabende“.

Allen voran jedoch bedanke ich mich bei meinem Mann Michael dafür, dass er mir immer den Rücken freigehalten und geduldig alle Höhen und Tiefen mit mir durchlebt hat. Zuletzt möchte ich auch unserer Tochter Johanna dafür Danke sagen, dass sie sich nicht an der Berufstätigkeit ihrer Mutter stört und mit ihrem Lachen und ihrer liebevollen Art unser Leben bereichert.

Kurzfassung

Sowohl Menschen als auch Maschinen müssen sich in immer komplexer werdenden Arbeitsumgebungen zurechtfinden und dabei insbesondere auch mit anderen Menschen und/oder Maschinen zusammenwirken, um die zu bearbeitenden Aufgabenstellungen erfolgreich meistern zu können.

Ziel der vorliegenden Arbeit ist es, zunächst maschinelle Fähigkeiten zur Kooperation auf wissensbasierter Verhaltensebene umzusetzen, um damit die Grundlage für Mensch-Maschine Kooperation auf Basis eines umfassenden Verständnisses der Gesamtsituation zu schaffen. Hierbei spielt insbesondere die Repräsentation expliziter Handlungsziele eine zentrale Rolle, die es ermöglicht, auch dann sinnvolles Verhalten zu zeigen, wenn für die aktuelle Situation keine vordefinierten Verhaltensregeln zur Verfügung stehen.

Ausgangspunkt für die weiteren Betrachtungen ist eine Arbeitssystem-Konfiguration im Anwendungsgebiet der Führung mehrerer UAVs, in der ein menschlicher Operateur einem Team aus UAVs Teilaufträge zuweisen kann, die diese eigenständig in Kooperation bearbeiten. Dabei wird jedes UAV von einer so genannten künstlichen kognitiven Einheit mit kooperativen Fähigkeiten geführt.

Im Hinblick auf eine Integration menschlicher Teammitglieder in das maschinelle Team wird für die Umsetzung rationalen, zielgerichteten Verhaltens im Rechner, d.h. die künstlichen kognitiven Einheiten, der Theorieansatz des Kognitiven Prozesses zugrunde gelegt. Dieser unterstützt insbesondere die Nachbildung der wissensbasierten Verhaltensebene des Menschen auf Basis einer expliziten Repräsentation von Handlungszielen. Den zweiten Teil des Konzepts bilden Methoden zur Kooperation, wobei vor allem die Teilaspekte der Koordination und Kommunikation im Team von Interesse sind. Die Konzeptanteile „künstliche Kognition“ und „Kooperation“ werden zusammengeführt, indem Kooperation hinsichtlich der Merkmale analysiert wird, die gemäß dem Kognitiven Prozess notwendig sind, um wissensbasiertes Verhalten darzustellen. Wesentlich ist hierbei vor allem die Identifikation von Zielen von Kooperation wie zum Beispiel das Vermeiden redundanter Aufgabenbearbeitung durch mehrere Teammitglieder.

Die Implementierung eines Funktionsprototyps, d.h. konkret einer künstlichen kognitiven Einheit mit kooperativen Fähigkeiten, erfolgt auf Grundlage der kognitiven Systemarchitektur COSA und der Programmiersprache CPL. Erstere stellt applikationsunabhängige Anteile des Kognitiven Prozesses in einem Framework zur Verfügung, während die Programmiersprache CPL die Modellierung von Wissen auf Basis mentaler Begriffe wie zum Beispiel Zielen erlaubt.

Abschließend wird der implementierte Funktionsprototyp anhand eines Anwendungsbeispiels evaluiert. Hierbei handelt es sich um eine kooperative SEAD-/Attack-Mission, bei der fünf UAVs gemeinsam ein Ziel bekämpfen sollen. In einem ersten Schritt wird dabei nachgewiesen, dass die geforderte Funktionalität mit dem gewählten Ansatz abgebildet werden kann. In einem zweiten Schritt werden die Errungenschaften im Hinblick auf die Realisierung von Kooperation auf wissensbasierter Verhaltensebene dargestellt. Schließlich erfolgt eine Bewertung der für maschinelle Kooperation ausgelegten künstlichen kognitiven Einheiten hinsichtlich deren Eignung für Mensch-Maschine Kooperation.

Inhaltsverzeichnis

1 Einleitung

Unbemannte Fluggeräte (UAVs – *Uninhabited Aerial Vehicles*) spielen sowohl in der militärischen als auch in der zivilen Luftfahrt eine immer bedeutsamere Rolle. Das Spektrum, in dem UAVs eingesetzt werden, reicht dabei im militärischen Bereich vom Träger für Aufklärungssensoren über den Einsatz als Relaisstation bis hin zum Kampfeinsatz, während im zivilen Sektor verschiedenste Anwendungen wie zum Beispiel Grenzüberwachung, Vermisstensuche oder die Erfassung und Beobachtung flächenhafter Daten in der Präzisionslandwirtschaft diskutiert werden.

Um sich ergänzende Fähigkeiten verschiedener UAVs nutzen, Redundanzen schaffen und die Missionseffizienz zum Beispiel durch die Möglichkeit zu höherer Gebietsabdeckung bei der Aufklärung steigern zu können, wird im militärischen Umfeld insbesondere der Einsatz mehrerer UAVs gemeinsam mit bemannten Kräften angestrebt (vgl. [Schulte, 2006]). Hierbei ergeben sich vielfältige technologische Herausforderungen, die von der Flugregelung der genutzten Plattformen über das Bereitstellen von Datenlinkverbindungen und Missionsmanagementfunktionalitäten an Bord der UAVs bis hin zur ergonomischen Gestaltung der Hard- und Software einer geeigneten Kontrollstation für einen Operateur reichen.

Im Zuge der Erhöhung der Führungsspanne (= Verhältnis Anzahl UAVs zu Anzahl Operateure) auf einen Wert größer Eins, welche zum Beispiel bei der luftgestützten Führung mehrerer UAVs notwendig sein dürfte, ist eine Anpassung des Abstraktionsgrades der Interaktion zwischen Operateur und UAV(s) nötig. Wurden anfangs UAVs von Operateuren, die am Boden die Rolle einen dislozierten Piloten einnahmen, ferngesteuert (RPVs – *Remotely Piloted Vehicles*), so wurde im Zuge der Nutzung von UAVs im Rahmen relevanter Anwendungen und mit fortschreitender Automatisierung der Abstraktionsgrad der Anweisungen schrittweise erhöht: auf Steuereingaben folgten Autopilotenkommandos, Flugpläne und Missionspläne. Hierdurch konnte insbesondere die benötigte Bandbreite des Führungsdatenlinks reduziert und die Abhängigkeit von dessen Integrität verringert werden. All diesen Arten von Vorgaben ist allerdings gemeinsam, dass der Operateur das UAV anweist, zu einem genau definierten Zeitpunkt eine genau definierte Handlung auszuführen. Der Operateur gibt dem UAV also vor, *wie* es eine Aufgabe durchführen soll, für die er es einsetzt (vgl. [Frampton & Keirl, 2006]) und überwacht dessen Aktionen im Sinne klassischer *supervisory control* (vgl. [Sheridan, 1992]). Dieser Ansatz bringt tendenziell eine hohe Arbeitsbelastung für den Operateur mit sich, da dieser für jedes UAV dessen Mission detailliert planen und dabei möglichst viele Eventualitäten berücksichtigen muss, um eine Umplanung im Flug – sofern diese überhaupt möglich ist – zu vermeiden und bei zeitkritischen Ereignissen nicht der Gefahr ausgesetzt zu sein, nicht mehr rechtzeitig reagieren zu können. Um dennoch eine Verbesserung der Führungsspanne erreichen, die Abhängigkeit von der Qualität des Datenlinks verringern und die Flexibilität der UAVs durch die Fähigkeit, an Bord auf unvorhergesehene Ereignisse reagieren zu können, erhöhen zu können [Frampton & Keirl, 2006], beschäftigen sich aktuelle Forschungsarbeiten einschließlich der hier vorgestellten mit der Entwicklung von Technologien, die die Vorgabe von Missionszielen an UAVs (*Was* soll getan werden?) erlauben. Damit kann der Operateur in der Rolle eines *mission managers* statt eines *UAV pilots* agieren [Frampton & Keirl, 2006].

Die skizzierte Veränderung der Interaktion zwischen Operateur und UAV(s) ist jedoch abhängig von der Weiterentwicklung vieler Aspekte, die von der Erweiterung der Fähigkeiten der UAVs was Missionsmanagement eines einzelnen und kooperative Fähigkeiten mehrerer UAVs betrifft bis hin zur Einführung geeigneter Unterstützungs- bzw. Assistenzsysteme für den Operateur reichen.

In diesem Umfeld untersuchen viele Forschungsgruppen die Gestaltung der Mensch-Maschine-Schnittstelle zum Beispiel hinsichtlich einer Verbesserung des Situationsbewusstseins des Operateurs durch die Darstellung synthetischer Informationen zusätzlich zu tatsächlich gewonnenen Sensordaten [Draper et al., 2006], Unterstützung bei zeitverzögerter Sensorsteuerung durch entsprechende Displaygestaltung [van Breda & van Erp, 2006] oder der optimalen Darstellung nebenläufiger Aufgaben mehrerer UAVs [Cummings & Mitchell, 2005].

Auch Technologien, die den gemeinsamen Einsatz mehrerer UAVs ermöglichen, werden von einer Vielzahl von Forschern betrachtet. So nutzen zum Beispiel [Dargar et al., 2002] Methoden der Netzwerk-Optimierung, welche in eine hierarchisch aufgebaute Architektur integriert sind, um Aufgaben in einem Team von UAVs zu verteilen. Diese Architektur wird beispielsweise im Umfeld der kooperativen Zielklassifizierung durch UAVs eingesetzt [Chandler et al., 2002]. Auch [Beard et al., 2002] setzten mathematische Verfahren wie Optimierung und Voronoi-Graphen ein, um eine UAV-Mission durchführen zu können, die in die Teilprobleme „kooperative Zielzuweisung“, „koordinierter Intercept“, „Routenplanung“, „Trajektorienplanung“ und „Trajektorienverfolgung“ unterteilt wird. Andere Arbeiten betrachten das Einnehmen einer räumlich und zeitlich optimierten Sensorkonfiguration, so dass mehrere UAVs, die über unterschiedliche Sensoren verfügen, zum Beispiel Bildverarbeitungsaufgaben kooperativ lösen können [Parunak et al., 2003]. Die optimale Verteilung einer bestimmten Anzahl von UAVs über einem (urbanen) Gebiet sowie die Identifikation und Verfolgung von Bodenzielen ist Gegenstand von [Hegazy et al., 2005]. Weitere, verschiedene Arbeiten im Umfeld kooperativer UAVs unter Verwendung regelungstechnischer Ansätze werden in [IEEE, 2007] vorgestellt. All diese Arbeiten betrachten allerdings lediglich das Zusammenwirken mehrerer UAVs zumeist auf Basis von Methoden der mathematischen Optimierung sowie deren Anwendung in der Regelungstechnik, ohne diese in den Kontext einer Mission und der Führung durch einen menschlichen Operateur zu stellen. Ob eine solche isolierte Betrachtung von Einzelaspekten Probleme im Gesamtsystem Mensch-UAVs vermeiden kann wie sie im Rahmen der Untersuchung von Zwischenfällen beim Einsatz von UAVs identifiziert wurden und bei denen der „Faktor Mensch“ eine große Rolle spielt (vgl. [Williams, 2004][Tvaryanas et al., 2005][Williams, 2006][Tvaryanas & Thompson, 2006]) ist fraglich, so dass im Folgenden nicht weiter auf die genannten Arbeiten eingegangen wird.

Trotz der allgemein anerkannten Wichtigkeit der Betrachtung des Gesamtsystems „Mensch-UAVs“ existieren nur wenige Forschungsprojekte, welche mehrere Aspekte des Komplexes „Führung mehrerer UAVs“ berücksichtigen. Als ein Vertreter sei hier ein Projekt des britischen Verteidigungsministeriums genannt, in welchem die Firma QinetiQ unter anderem die Aspekte Gesamtsystemarchitektur, Mensch-Maschine-Schnittstelle und UAV-UAV-Kooperation auf Basis einer Agentenarchitektur betrachtet [Walker, 2006][Howitt & Richards, 2006]. Dieses Vorhaben wird in Abschnitt 3.2.3 näher erläutert.

Aktuelle Forschungsarbeiten an der Professur für Flugmechanik und Flugführung der Universität der Bundeswehr München, an der auch die vorliegende Arbeit entstand, in-

tegrieren verschiedene Aspekte des Zusammenwirkens von Mensch und Maschine bzw. Automation, indem das so genannte Arbeitssystem als übergeordneter Theorieansatz verwendet wird [Onken & Schulte, in Vorb.]. Bei der Führung mehrerer UAVs werden dabei als wesentliche Automationsfunktionalitäten zum einen kooperative Fähigkeiten von UAVs und zum anderen Assistenzsysteme identifiziert [Kriegel et al., 2007]. Letztere unterstützen den Operateur bei seiner Aufgabe der Überwachung eines hoch komplexen technischen Systems wie es kooperative UAVs darstellen. Hierbei können neben dem Paradigma der Überwachung verschiedene Arten von kooperativen Beziehungen zwischen Subsystemen des Arbeitssystems identifiziert werden. So findet zum einen Kooperation zwischen maschinellen Komponenten, d.h. den UAVs, statt, die damit in die Lage versetzt werden, als Team Aufgaben zu bearbeiten, welche ihnen vom Operateur zugewiesen werden. Zum anderen wird ein partnerschaftliches Verhältnis zwischen dem Operateur und dem ihm zur Seite stehenden Assistenzsystem etabliert, welche als Team die Aufgaben des Operateurs, d.h. insbesondere die Führung der UAVs in einem gegebenen Missionskontext, bearbeiten. Gegenstand der vorliegenden Arbeit ist die Betrachtung des Aspekts UAV-UAV-Kooperation. Hierzu wird jedes UAV mit einer so genannten *Künstlichen Kognitiven Einheit* (*Artificial Cognitive Unit, ACU*) ausgestattet, welche ein technisches System darstellt, das in der Lage ist, das jeweilige UAV samt vorhandener Subsysteme zu führen und darüber hinaus mit anderen ACUs in einem Team zusammenzuarbeiten und somit ein gemeinsames Ziel zu erreichen.

Als Methode für die Entwicklung von ACUs wird dabei die so genannte *Kognitive Automation* verwendet, welche die Nachbildung einzelner Aspekte menschlicher Kognition zum Ziel hat. Im Rahmen der vorliegenden Arbeit wird insbesondere rationale Entscheidungsfindung auf wissensbasierter Verhaltensebene betrachtet. Wissensbasiertes Verhalten kommt bei Menschen immer dann zum Einsatz, wenn sie sich in Situationen befinden, die in der vorliegenden Konfiguration noch nicht bekannt sind und daher auch nicht bekannt ist, was in dieser Situation als nächstes zu tun ist. Es zeichnet sich insbesondere dadurch aus, dass die Entscheidung, welche Handlungsschritte als nächstes auszuführen sind, an übergeordneten, explizit repräsentierten Handlungszielen ausgerichtet ist. Damit wird selbst in unbekannten Situationen sinnvolles, d.h. an Zielen orientiertes, Verhalten möglich. Die Stärke dieses Ansatzes kommt vor allem dann zum Tragen, wenn Automation in komplexen Situationen verschiedene Handlungsalternativen zur Verfügung hat, wie es zum Beispiel bei der kooperativen Missionsdurchführung mehrerer UAVs der Fall ist. Hierbei muss nicht für jede mögliche Konfiguration der Gesamtsituation im Entwicklungsprozess eine auszuführende Aufgabe spezifiziert werden. Vielmehr wird die Automation befähigt, ein Verständnis der vorherrschenden Situation aufzubauen, zu erkennen, welche Ziele in dieser Situation relevant sind, und einen Weg zu finden, wie diese Ziele erreicht werden können.

Im folgenden Kapitel 2 wird zunächst das Arbeitssystem als übergeordneter Theorieansatz vorgestellt, mit dem komplexe Mensch-Maschine-Systeme betrachtet werden können, in denen Maschinen auch Teilaufgaben übernehmen, die klassischerweise Menschen vorbehalten waren. Nach der Anwendung auf die Führung mehrerer UAVs werden kooperative Beziehungen im Arbeitssystem charakterisiert und verschiedene Möglichkeiten aufgezeigt, wie Automation und insbesondere kognitive, kooperative Automation in das Arbeitssystem eingeführt werden kann.

Kapitel 3 stellt zu Beginn kognitive Architekturen als Ausgangspunkt für die Realisierung kognitiver Automationsfunktionen vor, die menschliche Kognition mit ihren Stärken und Schwächen nachbilden. Anschließend werden die wissensbasierte, regelbasierte und fertigkeitsbasierte Verhaltensebene des Menschen nach [Rasmussen, 1983] erläu-

tert. Im Hinblick auf die Umsetzung wissensbasierten Verhaltens im Rechner werden informationstechnisch motivierte Ansätze vorgestellt, die sich auf rationale Entscheidungsfindung konzentrieren und die Berücksichtigung explizit repräsentierter Handlungsziele erlauben. Detailliert wird dabei auf den Kognitiven Prozess eingegangen, welcher die Grundlage für diese Arbeit bildet. Danach werden Kooperation und ihre Implikationen Koordination und Kommunikation betrachtet und ihre Rolle bei der Entwicklung kooperativer künstlicher kognitiver Einheiten charakterisiert.

Kapitel 4 beschreibt vor dem Hintergrund allgemeingültiger Vorgehensweisen zur Softwareentwicklung zunächst die KP-Methode als Vorgehensweise zur Gewinnung von Wissensmodellen, welche den Kognitiven Prozess als allgemeingültige Theorie menschlichen rationalen Verhaltens in den Kontext einer konkreten Applikation stellen. Als Ergebnis einer Anwendung dieser Methode werden solche Wissensmodelle abgeleitet und beschrieben, die kooperatives kognitives Verhalten ermöglichen und im weiteren Verlauf der Arbeit verwendet werden.

Kapitel 5 stellt die Implementierung des erläuterten Konzepts auf Basis der kognitiven Systemarchitektur COSA vor. Diese stellt ein Framework für die Entwicklung kognitiver Systeme auf Basis des Kognitiven Prozesses bereit, welches unter anderem einen anwendungsunabhängigen Prozessor für die Umsetzung von Wissen in Verhalten und die Programmiersprache CPL für die Implementierung von Applikationswissen umfasst. Nach einer kurzen Einführung in COSA und CPL liegt der Schwerpunkt des Kapitels auf der Beschreibung des Zusammenwirkens der implementierten Wissensmodelle, welche die Grundlage für kooperative Fähigkeiten von UAVs bilden.

Kapitel 6 beschreibt anhand eines Anwendungsbeispiels – einer vereinfachten, militärischen SEAD/Attack-Mission – die Ergebnisse der Erprobung des in Kapitel 5 vorgestellten Prototyps unter drei verschiedenen Gesichtspunkten. Zunächst erfolgt eine funktionale Evaluierung, indem verschiedene Aspekte zielebasierter Kooperation anhand relevanter Teilprobleme nachgewiesen werden und gezeigt wird, dass diese kooperativen Einzelfähigkeiten in einer umfassenden Mission zusammenwirken können. Danach wird dargestellt, wie sich die Realisierung der wissensbasierten Verhaltensebene im Rechner auf das Verhalten der betrachteten UAVs in unterschiedlichen Situationen auswirkt und welche Erkenntnisse hieraus gewonnen werden können. Abschließend erfolgt eine Bewertung der im Rahmen dieser Arbeit entwickelten ACUs im Hinblick auf deren Eignung für Mensch-Maschine-Kooperation. Hierzu werden Ergebnisse aus einer Experimentalkampagne in Form von Anforderungen an zukünftige ACUs zusammengefasst, welche in der Lage sein sollen, mit menschlichen Teammitgliedern zu kooperieren.

Kapitel 7 schließlich fasst die Arbeit zusammen und gibt einen Ausblick auf zukünftige Arbeiten im Bereich wissensbasierter Kooperation auf Basis künstlicher Kognition im Kontext des Arbeitssystems.

2 Analyse des Arbeitssystems „Führung mehrerer UAVs“

Das Paradigma des klassischen Mensch-Maschine-Systems (vgl. z.B. [Johannsen, 1993]) setzt seine Schwerpunkte auf die Betrachtung von Mensch und Maschine mit klar voneinander getrennten Aufgabenbereichen, welche in einem hierarchischen Verhältnis zueinander stehen, d.h. dass der Mensch die Maschine im Sinne von *supervisory control* [Sheridan, 1992] überwacht. Im Rahmen der Untersuchung von Benutzerschnittstellen für Systeme, in denen der Mensch Maschinen für dedizierte Aufgabenstellungen nutzt, ist diese Betrachtungsweise daher gut geeignet. Sie stößt allerdings an ihre Grenzen, wenn es um die Beschreibung des Zusammenwirkens menschlicher und maschineller Komponenten geht, deren Fähigkeiten überlappen und deren Zusammenarbeit sich weg vom klassischen Rollenverständnis der Überwachung zum Beispiel in Richtung eines partnerschaftlichen Verhältnisses entwickelt. Für die Beschreibung solcher Systeme eignet sich das Arbeitssystem, welches Verantwortlichkeiten für Teilaspekte des zielgerichteten Zusammenwirkens von Mensch und Maschine im Arbeitsprozess stärker betont als deren Interaktion und letztlich auch die Abbildung verschiedener Rollen auf Mensch oder Maschine.

Im Folgenden wird zunächst der Theorieansatz des Arbeitssystems kurz vorgestellt und anschließend erläutert, wie kognitive und kooperative Automation in das Arbeitssystem eingeführt werden kann. Nach einer Charakterisierung derartiger Automation werden Abhängigkeiten zwischen einzelnen Komponenten des Arbeitssystems beschrieben, wobei besonders auf kooperative Beziehungen eingegangen wird. Schließlich werden verschiedene Arbeitssystemkonfigurationen im Umfeld der UAV-Führung, insbesondere der Führung mehrerer UAVs mit einer Führungsspanne größer Eins, diskutiert.

2.1 Kognitive Automation im Arbeitssystem

2.1.1 Das Arbeitssystem

Zur Beschreibung und Analyse von Mensch-Maschine-Systemen wird im Folgenden das Arbeitssystem [REFA, 1984] verwendet, wie es bereits von [Onken, 2002] und [Meitinger & Schulte, 2006b] im Bereich der Flugführung angewendet wurde. Nach DIN 33400 wird der Begriff des Arbeitssystems wie folgt definiert:

> *Ein Arbeitssystem dient der Erfüllung einer Arbeitsaufgabe; hierbei wirken Mensch und Arbeitsmittel im Arbeitsablauf am Arbeitsplatz in einer Arbeitsumgebung unter den Bedingungen dieses Arbeitssystems zusammen. (zitiert nach [von Rosenstiel et al., 2005])*

Abbildung 2-1 stellt ein Arbeitssystem mit seinen Bestandteilen dar. Wesentliches Element ist der Mensch als *Bediener*, dessen Hauptaufgaben darin bestehen, Entscheidungen im Hinblick auf das Erreichen des *Arbeitsziels* zu treffen, auftretende Probleme zu lösen und die zur Verfügung stehenden *Arbeitsmittel* zu bedienen, d.h. einzusetzen und zu überwachen. Darüber hinaus kann er sich unter Umständen das zu erreichende *Arbeitsziel* selbst vorgeben oder modifizieren [von Rosenstiel et al., 2005]. Letztere Eigenschaft des Arbeitssystems ist nach [Onken & Schulte, in Vorb.] das charakteristische Merkmal eines autonomen Systems. Die Funktion des Bedieners eines Arbeitssystems

kann nach [Onken, 2002] auch durch mehrere Menschen wahrgenommen werden, die in diesem Fall ein Team bilden. Im Hinblick auf eine weitere Integration nicht nur zusätzlicher Menschen sondern auch technischer Systeme, die Aufgaben eines Bedieners übernehmen (vgl. Abschnitt 2.1.2), wird im weiteren Verlauf dieser Arbeit statt des Begriffs „Bediener“ der von [Onken & Schulte, in Vorb.] eingeführte, umfassendere Ausdruck *Operating Force* verwendet.

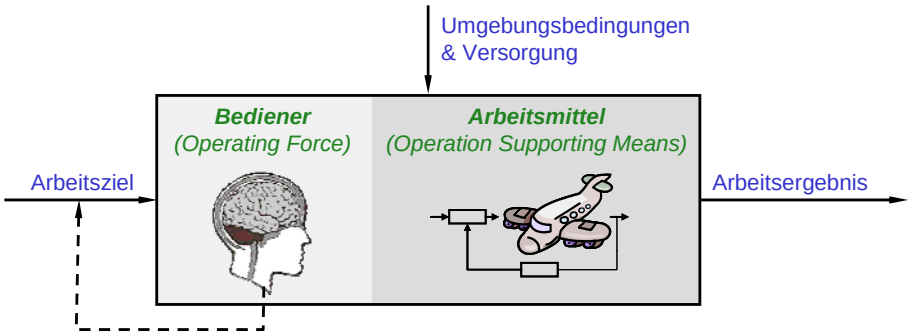


Abbildung 2-1: Arbeitssystem

Die *Arbeitsmittel* (von [Onken & Schulte, in Vorb.] als *Operation Supporting Means* bezeichnet) umfassen sämtliche dem Bediener zur Verfügung stehende technische Ausrüstung. Darunter fallen auch alle vorhandenen Automationsfunktionen, welche klar definierte Teilaufgaben ausführen können, die von der *Operating Force* vorgegeben werden (siehe [Onken & Schulte, in Vorb.] für mehr Details).

Informationen über das *Arbeitsziel*, das in Form einer Arbeitsaufgabe oder eines Auftrags formuliert werden kann, *„beinhalten insbesondere Angaben über das Arbeitsergebnis“ [von Rosenstiel et al., 2005]*. Dabei muss zwischen der objektiv gestellten Aufgabe, die eine Eingangsgröße in das Arbeitssystem darstellt und häufig als (Arbeits-) auftrag bezeichnet wird, und der durch den Bediener wahrgenommenen Aufgabe unterschieden werden [von Rosenstiel et al., 2005].

Das *Arbeitsergebnis* soll bzw. muss unter Berücksichtigung vorherrschender *Umgebungsbedingungen* erreicht werden. Hierunter fallen zur Verfügung stehende Informationen und Ressourcen ebenso wie Rahmenbedingungen und Veränderungen in der Umgebung.

2.1.2 Einführung zusätzlicher Automation in das Arbeitssystem

Betrachtet man verschiedene Möglichkeiten, zusätzliche Automation in das Arbeitssystem einzuführen, so ergeben sich nach [Schulte et al., 2008] und [Onken & Schulte, in Vorb.] insgesamt vier Fälle, nämlich die Einführung konventioneller oder kognitiver Automation auf Seiten der Arbeitsmittel oder der *Operating Force* (vgl. Abbildung 2-2).

Die erste Zeile repräsentiert hierbei die klassische Vorgehensweise bei der Einführung zusätzlicher Komponenten in das Arbeitssystem derart, dass entweder zusätzliche Automationsfunktionen auf Seiten der Arbeitsmittel eingeführt werden (z.B. Hinzunahme eines Flugmanagementsystems) oder aber die *Operating Force* durch das Einbeziehen eines zusätzlichen Menschen verstärkt wird (z.B. Hinzunahme eines zusätzlichen Operateurs bei der Führung mehrerer UAVs). Das Ergänzen eines zusätzlichen menschlichen Bedieners stellt natürlich keine Automation dar, bietet jedoch eine sehr intuitive Analogie zu der weiter unten beschriebenen Ergänzung der *Operating Force* durch Automation, da sich ähnliche Fragestellungen hinsichtlich der Zusammenarbeit innerhalb der *Operating Force* ergeben. Während zusätzliche Automation auf Seiten der Arbeitsmittel

ebenso wie bereits vorhandene Automation von der *Operating Force* überwacht werden muss und somit das hierarchische Verhältnis erhalten bleibt, ergibt sich bei der Ergänzung eines zusätzlichen Bedieners die Forderung nach Kooperation zwischen den Elementen der *Operating Force*, die im Hinblick auf das Erreichen des Arbeitsziels zusammenarbeiten müssen.

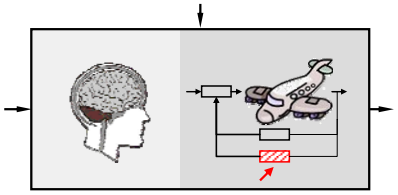
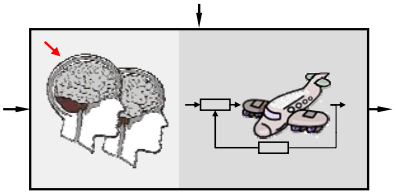
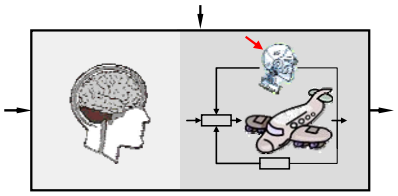
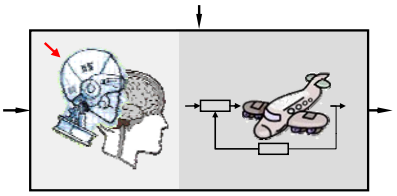
	Überwachung von Arbeitsmitteln	*Kooperation mit Bediener*
Konventionelle Automation		
Kognitive Automation		

Abbildung 2-2: Einführung zusätzlicher Automation in das Arbeitssystem (vgl. [Schulte et al., 2008])

Wird nun nicht konventionelle sondern kognitive Automation in Form so genannter künstlicher kognitiver Einheiten (*Artificial Cognitive Units*, ACUs) in das Arbeitssystem eingeführt, so ergeben sich analoge Veränderungen: Zusätzliche kognitive Automation auf Seiten der Arbeitsmittel muss von der *Operating Force* überwacht werden, während zusätzliche Automation auf Seiten der *Operating Force* mit den anderen Elementen der *Operating Force* kooperieren muss. Diese beiden Möglichkeiten der Einführung kognitiver Automation in das Arbeitssystem werden im Sinne von [Onken & Schulte, in Vorb.] in den folgenden Abschnitten detailliert. Demnach werden künstliche kognitive Einheiten auf Seiten der *Operation Supporting Means* als *Supporting ACUs* bezeichnet (Abschnitt 2.1.2.1), während ACUs, die Teil der *Operating Force* sind, *Operating ACUs* genannt werden (Abschnitt 2.1.2.2). Anschließend wird kognitive Automation als Ansatz zur Realisierung von ACUs sowie der Vorteil ihres Einsatzes im Gegensatz zu konventioneller Automationstechnologie beleuchtet (Abschnitt 2.1.2.3).

Die im Folgenden vorgestellte Sichtweise stellt nur eine Möglichkeit dar, künstliche Kognition im Kontext von Mensch-Maschine-Systemen zu betrachten. Daneben existieren viele weitere Arbeiten in diesem Umfeld, die sich zum Teil mit einer Arbeitssystembetrachtung in die folgende Klassifikation einordnen lassen. So fördert beispielsweise die DFG (Deutsche Forschungsgemeinschaft) seit kurzem das Exzellenzcluster CoTeSys (*Cognition for Technical Systems*) [Buss et al., 2007][CoTeSys Homepage], worin sich eine Teilgruppe mit interaktiven Fähigkeiten kognitiver technischer Systeme befassen wird. Weiterhin erstreckt sich das Forschungsgebiet der kognitiven Assistenzsysteme auf verschiedenste Anwendungsgebiete, worunter auch die Fahrzeug- und Prozessführung fällt (z.B. [Stütz & Schulte, 2000][Frey et al., 2001][Taylor, 2001]). In Kapitel 3 werden darüber hinaus relevante Anwendungen im Kontext der Flugführung vorgestellt, deren Schwerpunkt nicht auf der Unterstützung eines Operateurs, sondern auf der tatsächlichen Führung eines oder mehrerer Luftfahrzeuge liegt. Im Einzelnen sind dies die Systeme TacAir-Soar (vgl. Abschnitt 3.2.1.2), STEAM (vgl. Abschnitt 3.2.1.2), Surrogate UAV (vgl. Abschnitt 3.2.3.1) und COSY^{flight} (vgl. Abschnitt 3.2.3.2).

2.1.2.1 *Supporting ACUs* und semi-autonome Systeme

Die Einführung künstlicher kognitiver Einheiten in das Arbeitssystem auf Seiten der Arbeitsmittel als *Supporting ACUs*, welche durch ihren Einsatz den übergeordneten Arbeitsprozess unterstützen bzw. ermöglichen, erfolgt vor dem Hintergrund immer komplexer werdender Aufgabenstellungen wie der eingangs genannten Führung mehrerer UAVs durch wenige Operateure. Da in diesem Kontext auch die Übernahme von Aufgaben, die bisher bemannten Luftfahrzeugen vorbehalten waren, durch UAVs diskutiert wird, ist es also notwendig, Automationsfunktionen (hier: *Supporting ACUs*) verfügbar zu machen, die auch in komplexen Umgebungen sinnvoll und zielgerichtet agieren können. Durch die stetige Weiterentwicklung von Automationstechnologien ist Automation heute in der Lage, in diesem Umfeld benötigte höhere geistige Fähigkeiten des Menschen wie zum Beispiel Entscheidungen zu treffen oder Probleme zu lösen nachzubilden (vgl. [Hollnagel & Woods, 2005]). Gleichzeitig muss derartige Automation immer komplexer werden. Aus dem von Ashby formulierten und von [Hollnagel & Woods, 2005] in ähnlichem Zusammenhang angewendeten *Law of Requisite Variety* folgt nämlich, dass wirksame Regulierung eines Systems nur dann möglich ist, wenn die Komplexität der regulierenden Einheit mindestens der Komplexität des Systems entspricht, das durch den zu Grunde liegenden Prozess und darauf wirkende Störgrößen beschrieben wird (vgl. [Hollnagel & Woods, 2005], S. 40f. und [Ashby, 1956]).

Die Einführung von *Supporting ACUs* in ein Arbeitssystem erfolgt üblicherweise im Rahmen eines ***semi-autonomen Systems*** (vgl. Abbildung 2-3), da eine *Supporting ACU* ähnlich wie der Mensch als Bediener eines Arbeitssystems nicht sämtliche bereits konventionell vorhandenen und zur Durchführung diverser Aufgaben nötigen Automationsfunktionen nachbilden soll, sondern über solche Automation verfügen können soll. Im Allgemeinen besteht ein semi-autonomes System sowohl aus konventionellen Automationsfunktionen, die genau definierte Aufgabenstellungen bearbeiten können, als auch einer *Supporting ACU*, die Querverbindungen zwischen einzelnen Aufgaben herstellt und die zur Verfügung stehenden Automationsfunktionen im Hinblick auf die zu bearbeitende, übergeordnete Aufgabenstellung einsetzt und konfiguriert.

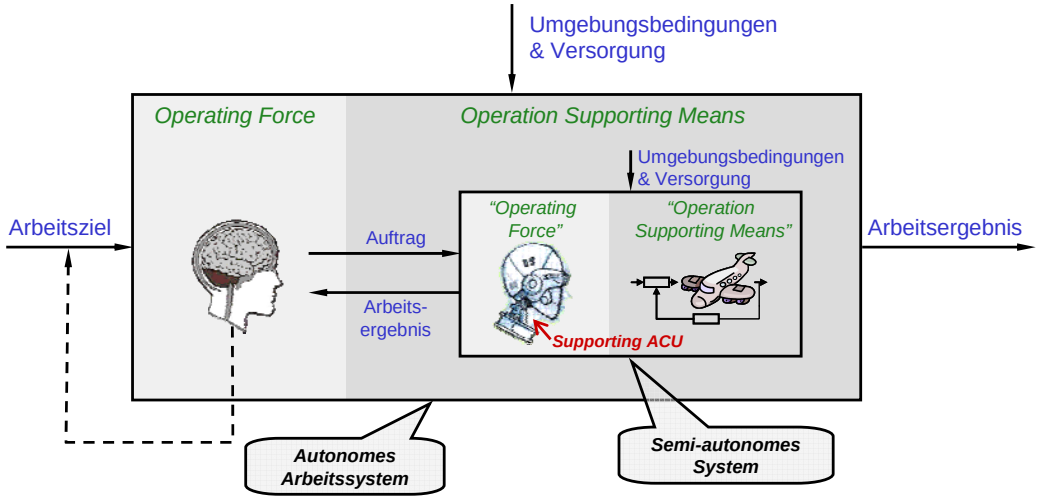


Abbildung 2-3: Semi-autonomes System als Arbeitsmittel eines übergeordneten Arbeitssystems

Ein semi-autonomes System gleicht im Aufbau einem Arbeitssystem, bei dem eine oder mehrere *Supporting ACUs* verschiedene andere technische Systeme einsetzen, um eine vorgegebene Aufgabe zu bearbeiten (vgl. [Kriegel & Schulte, 2006][Kriegel et al., 2007][Schulte et al., 2008]). An dieser Stelle sei darauf hingewiesen, dass es sich bei einem semi-autonomen System nicht um ein Arbeitssystem handelt, da es keinen Men-

schen als Teil einer *Operating Force* gibt. Vielmehr muss ein semi-autonomes System immer als Arbeitsmittel in ein übergeordnetes Arbeitssystem eingebettet werden, da technische Systeme zum einen nur dann sinnvoll sind, wenn sie von Menschen genutzt, d.h. überwacht und geführt, werden können. Zum anderen soll sich ein semi-autonomes System seine Arbeitsziele nicht vorgeben können, so dass es letztlich seine Aufgaben immer von einer übergeordneten Instanz zugewiesen bekommen muss. In dieser Konfiguration hat die *Operating Force* des Arbeitssystems, in das ein semi-autonomes System eingebettet ist, mächtige Arbeitsmittel zur Verfügung, die ihnen zugewiesene Aufgaben weitgehend selbständig bearbeiten. Somit werden mit dem gesamten Arbeitssystem gegebenenfalls auch Aufgabenstellungen durchführbar, die mit bisher verfügbaren Technologien nicht möglich waren.

Die Einsatzmöglichkeiten semi-autonomer Systeme können in diesem Umfeld weiter ausgebaut werden, indem *Supporting ACUs* als Entscheidungsinstanz semi-autonomer Systeme zur Kooperation miteinander befähigt werden. Konkrete Konfigurationen, die solcher Fähigkeiten bedürfen, werden am Beispiel der Führung mehrerer UAVs in Abschnitt 2.3.3 diskutiert, wobei die Realisierung einer derartigen Konfiguration, die ein Team aus *Supporting ACUs* beinhaltet, Gegenstand der vorliegenden Arbeit ist.

2.1.2.2 *Operating ACUs* und Assistenzsysteme

Wird kognitive Automation nicht auf Seiten der Arbeitsmittel, sondern auf Seiten der *Operating Force* in das Arbeitssystem eingeführt, so werden die entsprechenden künstlichen kognitiven Einheiten *Operating ACUs* genannt, da sie eine aktive Rolle beim Erreichen des Arbeitsziels übernehmen. Dass eine *Operating ACU* das übergeordnete Arbeitsziel kennt und versucht, dieses in Kooperation mit einem oder auch mehreren menschlichen Operateuren und ggf. anderen *Operating ACUs* zu erreichen, unterscheidet eine *Operating* von einer *Supporting ACU*. Dazu kann sie im Prinzip über alle vorhandenen Arbeitsmittel verfügen, d.h. unter Umständen auch *Supporting ACUs* bzw. semi-autonome Systeme einsetzen.

Eine *Operating ACU* kann in diesem Zusammenhang zwei verschiedene Aufgabenschwerpunkte haben [Onken & Schulte, in Vorb.]. Ersetzt sie ein Mitglied eines menschlichen Teams innerhalb der *Operating Force* (*substitution,* vergleichbar einem Übergang von rechts oben, d.h. rein menschliches Team, nach rechts unten, d.h. menschlicher Operateur und ACU, in Abbildung 2-2), so besteht ihre Hauptaufgabe in der Durchführung von Teilaufgaben im Rahmen des Arbeitsziels. Wird sie jedoch zusätzlich zu einem Menschen in die *Operating Force* eingebracht (*supplementation,* vergleichbar einem Übergang von links oben, d.h. einzelner menschlicher Operateur, nach rechts unten, d.h. menschlicher Operateur und zusätzliche ACU, in Abbildung 2-2) ist sie als klassisches Assistenzsystem im Sinne der Arbeiten von [Prévôt et al., 1995] und [Walsdorf et al., 1997] ausgeprägt. Dieses hat die situationsangepasste Unterstützung des Operateurs bei der Durchführung seiner Teilaufgaben im Rahmen der Erreichung des Arbeitsziels als Hauptaufgabe. Onken konkretisiert diesen Sachverhalt im Kontext der bemannten Luftfahrt in Form zweier Grundforderungen an Assistenzsysteme. Sie sagen aus, dass (1) die Aufmerksamkeit der Crew auf die objektiv dringlichste Aufgabe gelenkt werden muss und (2) für den Fall einer Überbeanspruchung der Crew die vorherrschende Situation mit technischen Hilfsmitteln in eine handhabbare überführt werden muss [Onken, 1994] (vgl. auch [Schulte et al., 2008] und [Onken & Schulte, in Vorb.] für eine differenziertere Diskussion dieser Sachverhalte).

2.1.2.3 Kognitive Automation

Sowohl *Supporting* als auch *Operating ACUs* stellen eine Ausprägung *kognitiver Automation* im Arbeitssystem dar. Im Gegensatz zu konventioneller Automation ist diese in der Lage, im Gesamtkontext einer Aufgabenstellung zielgerichtet zu handeln:

> *As opposed to conventional automation, cognitive automation works on the basis of comprehensive knowledge about work process objectives and goals on all goal hierarchy levels, pertinent task options and necessary data describing the current situation in the work process. Therefore, cognitive automation is prime-goal-oriented. [Onken, 2002]*

Der Einsatz kognitiver Automation im Arbeitssystem verspricht insbesondere durch deren Berücksichtigung expliziter Handlungsziele, Probleme zu vermeiden wie sie im Rahmen der Nutzung konventioneller Automationstechnologie in der bemannten Luftfahrt beobachtet wurden [Billings, 1997]. Diese Probleme werden von [Billings, 1997] hauptsächlich auf vier Automationseigenschaften zurückgeführt, nämlich

- *complexity* – Details der Automation sind auf Grund des hohen Funktionsumfangs und der Systemkomplexität vom Bediener nur schwer zu verstehen.
- *brittleness* – Jenseits der Betriebsgrenzen ist das Verhalten der Automation undefiniert oder falsch.
- *opacity* – Der Operateur hat ein unvollständiges oder fehlerhaftes mentales Modell der Automation, die ihm überdies nicht mitteilt, was sie gerade macht.
- *literalism* – Automation verfügt lediglich über im Entwicklungsprozess programmierte Funktionen und führt exakt aus, was ihr seitens des Designers und des Operateurs vorgegeben wird, ohne dessen Sinnhaftigkeit im Kontext der vorherrschenden Situation zu überprüfen.

Wie mehrfach angedeutet wird die Komplexität eines Arbeitssystems mit der Einführung zusätzlicher und damit auch kognitiver Automation zunächst weiter erhöht und dadurch das Problem der *complexity* verschärft. Dieser Effekt kann per se nicht vermieden werden, da es einer gewissen Systemkomplexität bedarf, um in einer komplexen Umgebung sinnvoll agieren zu können (vgl. [Hollnagel & Woods, 2005], S. 40f.). Durch die Berücksichtigung expliziter Ziele kann jedoch die Handlungsweise einer in sich komplexen ACU für einen Operateur verständlich gemacht werden, da Ziele auf einer abstrakten Ebene Verhalten strukturieren. Zur Verdeutlichung soll hier die Metapher von Herbert Simon [Simon, 1969][Augier & Feigenbaum, 2003] einer Ameise am Strand aufgegriffen werden, deren Ziel es ist, eine weit entfernte Futterquelle zu erreichen. Da sie viele Sandkörner umgehen muss, ist der Weg der Ameise durch viele kleine Richtungsänderungen gekennzeichnet, d.h. das beobachtbare Verhalten ist – hervorgerufen durch die Komplexität der Umgebung – komplex. Dennoch lässt sich dieses Verhalten nachvollziehen, wenn bekannt ist, dass die Ameise die beiden Ziele „Futterquelle erreichen“ und „Nicht mit Sandkorn kollidieren“ verfolgt.

Auch dem Problem der *brittleness* kann durch die Orientierung an Zielen entgegengewirkt werden. Sie bewirkt nämlich, dass kognitive Automation nicht über Betriebsgrenzen im Sinne numerischer Parameter verfügt und sich undefiniert oder sogar kontraproduktiv verhält, wenn es diese Grenzen verlässt, sondern sich auch in Situationen, die vom Entwickler so nicht vorgesehen waren, zielgerichtet verhält. Somit ergibt sich im Rahmen der Möglichkeiten einer ACU stets sinnvolles Verhalten. Tritt also eine Situation ein, die von vorhandenem Wissen nicht abgedeckt wird, fällt kognitive Auto-

mation nicht in einen undefinierten Betriebszustand, den sie nicht mehr verlassen kann, sondern setzt ihr vorhandenes Wissen weiterhin bestmöglich ein. Im Falle einer günstigen Änderung der Situation kann sie dann unter Umständen wieder ihr gesamtes Leistungsspektrum ausschöpfen.

Weiterhin verfügt kognitive Automation über die Fähigkeit zur Selbsterklärung, eine Eigenschaft, die zur Minderung der mit dem Faktor *opacity* verbundenen Probleme konventioneller Automation beiträgt. Auch diese Erklärungskomponente lässt sich im Wesentlichen auf das Vorhandensein expliziter Ziele zurückführen, und zwar sowohl hinsichtlich des Stattfindens der Selbsterklärung als auch des Inhalts. So lassen sich für den Fall einer *Operating ACU* Ziele definieren, die auf Basis eines Modells des Operateurs hinterlegen, welche Informationen sie ihm auf welche Art und Weise wann darbieten kann bzw. muss. Dabei kann der momentane Zustand des Menschen hinsichtlich seiner mentalen Ressourcen und aktuellen Aufgabenstellungen, die er im Rahmen des Arbeitsziels bearbeiten muss berücksichtigt werden. Inhalt der Selbsterklärung sind dann im Allgemeinen andere Ziele, die angeben, welche momentane Absicht die ACU verfolgt. Eine *Supporting ACU* verfügt nicht über ein Modell des Operateurs, jedoch kann sie in die Lage versetzt werden, auf Anfragen hinsichtlich ihrer aktuellen Absichten zu reagieren, und durch das Wissen über ihre Ziele entsprechend zu antworten.

Da kognitive Automation in die Lage versetzt werden kann, nicht nur vorgegebene Teilaufgaben abzuarbeiten, sondern vor dem Hintergrund eines umfassenden Situationsverständnisses auch übergeordnete Zielsetzungen zu verfolgen, können Probleme im Umfeld von *literalism* vermieden werden. So wissen *Operating ACUs* um das übergeordnete Arbeitsziel und damit in Zusammenhang stehende Aufgaben. Daher können sie ihre Fähigkeiten flexibel im Hinblick auf dieses Arbeitsziel einsetzen und gleichzeitig bewerten, wie Teilaufgaben, die ihr von einem Operateur zugewiesen werden, sich auf das Erreichen des Arbeitsziels auswirken. Somit kann die Problematik entschärft werden, dass Teilaufgaben, die einer *Operating ACU* vom Operateur korrekt zugewiesen und abgearbeitet werden, die Erfüllung des Arbeitsziels behindern. Eine *Supporting ACU* weiß nicht um das Arbeitsziel des übergeordneten Arbeitssystems und kann daher nicht die Sinnhaftigkeit von zugewiesenen Teilaufgaben im Hinblick auf das Arbeitsziel bewerten. Allerdings können hier andere übergeordnete Zielsetzungen wie zum Beispiel Flugsicherheit hinterlegt werden, vor deren Hintergrund die Zuweisung von Aufgaben kritisch betrachtet werden kann.

2.2 Beziehungen zwischen Arbeitssystem-Komponenten

Mit der Einführung kognitiver Automation in das Arbeitssystem ergeben sich neben der Überwachung der Arbeitsmittel durch die *Operating Force* im Sinne von *supervisory control* (vgl. [Sheridan, 1992]) weitere strukturelle Beziehungen zwischen Komponenten des Arbeitssystems, welche in Abbildung 2-4 dargestellt sind. Hierbei handelt es sich um Beziehungen kooperativer Aufgabenteilung sowohl innerhalb der *Operating Force* als auch der *Operation Supporting Means*.

Auf Seiten der *Operating Force* treten je nachdem, welche Komponenten hier vorhanden sind, verschiedene Arten von Kooperation zwischen Mensch(en) und/oder *Operating ACU(s)* auf, um gemeinsam das Arbeitsziel erreichen zu können. Im Einzelnen sind dies *Mensch-Mensch-Kooperation* (z.B. Cockpit-Crew im Linienflugzeug), *Mensch-ACU-Kooperation* (z.B. Pilot eines einsitzigen Kampfflugzeugs, welcher durch ein Assistenzsystem unterstützt wird) und *ACU-ACU-Kooperation* zwischen *Operating ACUs*. Je nach Fähigkeiten und Autorität der einzelnen Elemente der *Operating Force*

kann die Kooperation zwischen ihnen unterschiedlich ausgeprägt sein. So unterscheidet sich die Zusammenarbeit zweier Piloten mit gleicher Ausbildung von der eines Piloten und eines Assistenzsystems, welches lediglich über die Fähigkeit verfügt, in bestimmten Situationen unterstützende Hinweise zu geben. Die Einflüsse der Teamstruktur auf kooperative Zusammenarbeit werden in Abschnitt 3.3.2 genauer betrachtet.

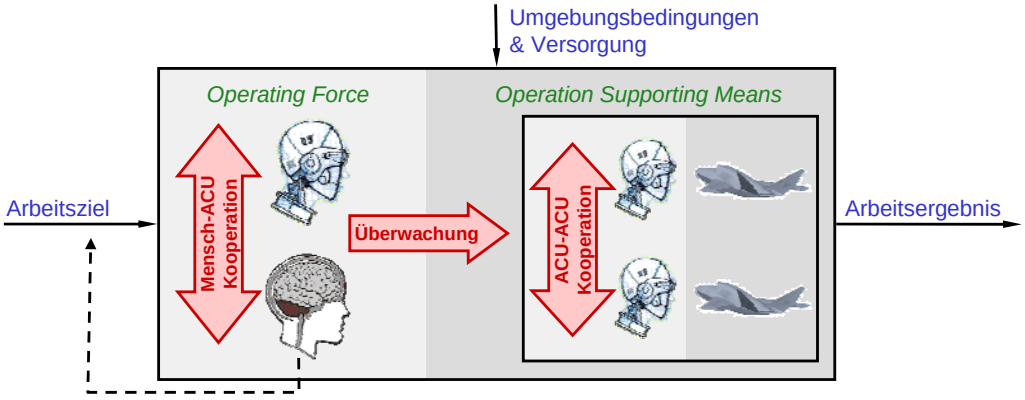


Abbildung 2-4: Beziehungen zwischen Arbeitssystem-Komponenten

Während innerhalb der *Operating Force* die genannten unterschiedlichen Arten von Kooperation auftreten können, beschränkt sich Kooperation im Bereich der Arbeitsmittel auf *ACU-ACU-Kooperation* zwischen *Supporting ACUs*, sofern diese gemeinsam eine von der *Operating Force* zugewiesene Aufgabe bearbeiten sollen (z.B. Kooperation zwischen mehreren UAVs, vgl. Abschnitt 2.3.3). Die Entwicklung solcher kooperativer *Supporting ACUs* im Umfeld kooperativer UAV-Missionen ist Gegenstand der vorliegenden Arbeit, so dass im Folgenden hierfür relevante Arbeitssystem-Konfigurationen vorgestellt werden.

2.3 Aufbau des Arbeitssystems „Führung mehrerer UAVs“

2.3.1 Führung eines UAVs

Die Führung eines UAVs durch einen Operateur erfolgt meist ähnlich der Flugführung eines bemannten Luftfahrzeugs durch einen Piloten, welcher über diverse Anzeigen Informationen über den Zustand seines Flugzeugs erhält und mit Hilfe unterschiedlicher Eingabegeräte Kommandos auf verschiedenen Abstraktionsebenen an entsprechende Automationsfunktionen wie Autopilot oder Flugmanagementsystem (FMS) absetzen kann. Dennoch ergeben sich auch einige entscheidende Unterschiede. Da sich ein UAV-Operateur nicht wie ein Pilot an Bord des Luftfahrzeugs befindet, das er führt, werden im Fall der UAV-Führung zusätzlich Datenfunk-Verbindungen für die Datenübertragung zwischen Bedien- und Anzeigegeräten und den entsprechenden UAV-Subsystemen benötigt.

Abbildung 2-5 stellt ein entsprechendes Arbeitssystem dar, in dem dem Operateur in der Rolle des Bedieners als Arbeitsmittel eine Bodenkontrollstation und ein UAV zur Verfügung stehen, die über eine Datenfunk-Verbindung Informationen austauschen können.

Bei der UAV-Führung wird hierbei versucht, die Abstraktionsebene der Interaktion so hoch wie möglich zu wählen, um so die Menge an Daten, die über eine Datenfunk-Verbindung übertragen werden muss, ebenso wie die Frequenz, mit der Daten gesendet und empfangen werden müssen, möglichst zu minimieren. Damit sinken die Anforderungen hinsichtlich der verfügbaren Bandbreite und Laufzeiteigenschaften wodurch letztlich eine Erhöhung der Flugsicherheit angestrebt wird.

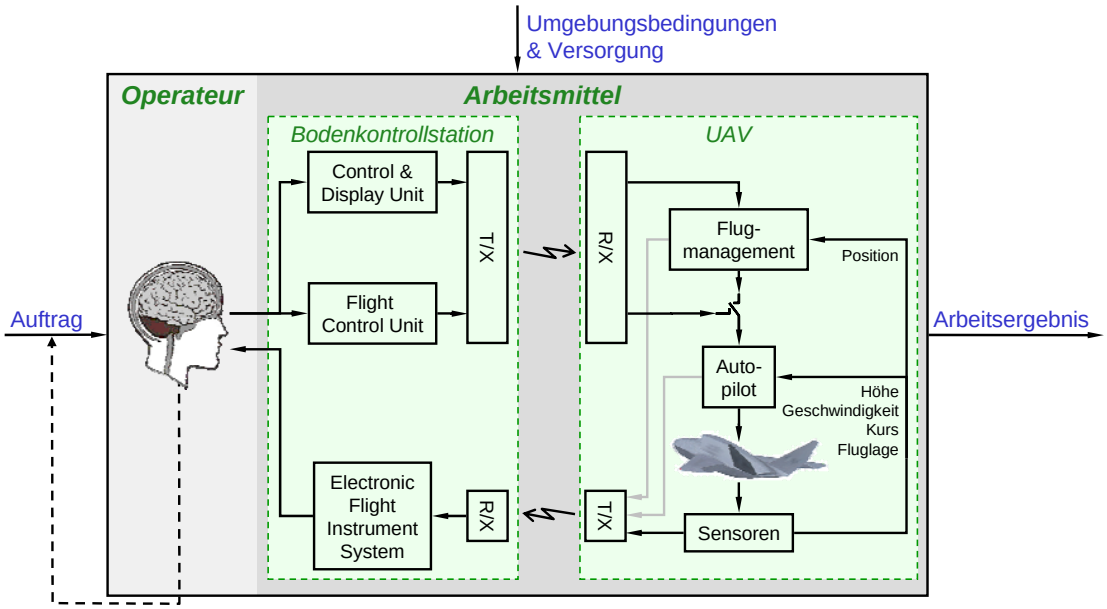


Abbildung 2-5: Aufbau des Arbeitssystems „Führung eines UAVs“

Oft wird in diesem Zusammenhang als Abstraktionsebene die Vorgabe eines Missionsplans gewählt, d.h. eine Liste anzufliegender Wegpunkte, welche mit Angaben versehen sind, welche missionsrelevanten Aktionen (z.B. Einsatz eines bildgebenden Sensors) an den einzelnen Punkten auszuführen sind. Während ein solcher Missionsplan durchgeführt wird, besteht die Aufgabe des Operateurs in der Überwachung des UAVs auf Basis zur Verfügung gestellter Daten (z.B. Videobild, Positions- und Lagedaten) im Hinblick auf das Arbeitsziel. Dabei kann es nötig werden, auf Grund geänderter Rahmenbedingungen wie zum Beispiel einer geänderten Bedrohungslage den Missionsplan anzupassen. Außerdem müssen übergeordnete Zielsetzungen wie zum Beispiel Flugsicherheit berücksichtigt und in diesem Zusammenhang entsprechende Maßnahmen ergriffen werden, um beispielsweise Kollisionen mit Gelände, Hindernissen oder anderen Luftfahrzeugen zu vermeiden, sofern es hierfür keine bordseitigen Automationsfunktionen gibt.

Als Beispiel für ein derartiges System sei hier die Führung des Barracuda-UAVs der Firma EADS-MAS angeführt. Diesem wird ein Missionsplan mit alternativen Routen vorgegeben, dessen Durchführung mit Hilfe von Kommandos wie „Take Off“, „Take Off Abort“, „Return to Base“ oder „Turn“ von einem Operateur am Boden gesteuert wird [Moormann et al., 2007].

2.3.2 Führung mehrerer UAVs

Sollen mehrere UAVs durch einen Operateur geführt werden, so ändert sich das Arbeitssystem „Führung eines UAVs“ zunächst lediglich derart, dass dem Operateur statt eines UAVs nun mehrere UAVs zur Verfügung stehen (siehe Abbildung 2-6). Diese kann er entsprechend ihrer Fähigkeiten unter Verwendung einer Kontrollstation einsetzen, um einen gegebenen Auftrag zu erfüllen. Hierbei erweitert sich sein Aufgabenspektrum im Vergleich zur Konfiguration mit einem UAV insofern, als er nicht nur einem UAV Vorgaben zum Beispiel in Form eines Missionsplans machen und dessen Durchführung wie oben beschrieben überwachen muss, sondern dies in mehreren nebenläufigen Prozesssträngen gleichzeitig tut. Außerdem muss der Operateur entscheiden, in welche Teilaufgaben ein Auftrag wie zum Beispiel die Aufklärung eines Gebietes unterteilt werden kann, welches UAV welche Teilaufgabe übernehmen soll (z.B. Sektor

eines Aufklärungsgebietes) und wie es diese Teilaufgabe bearbeiten kann (z.B. Vorgabe eines Suchmusters). An die Bearbeitung der Aufgaben schließt sich außerdem die Fusion der Ergebnisse an, da der Operateur als Bediener in dieser Konfiguration die einzige Instanz im Arbeitssystem ist, die Entscheidungen im Hinblick auf das Arbeitsziel treffen kann und einen Überblick über Teilaufgaben und deren Abhängigkeiten voneinander hat.

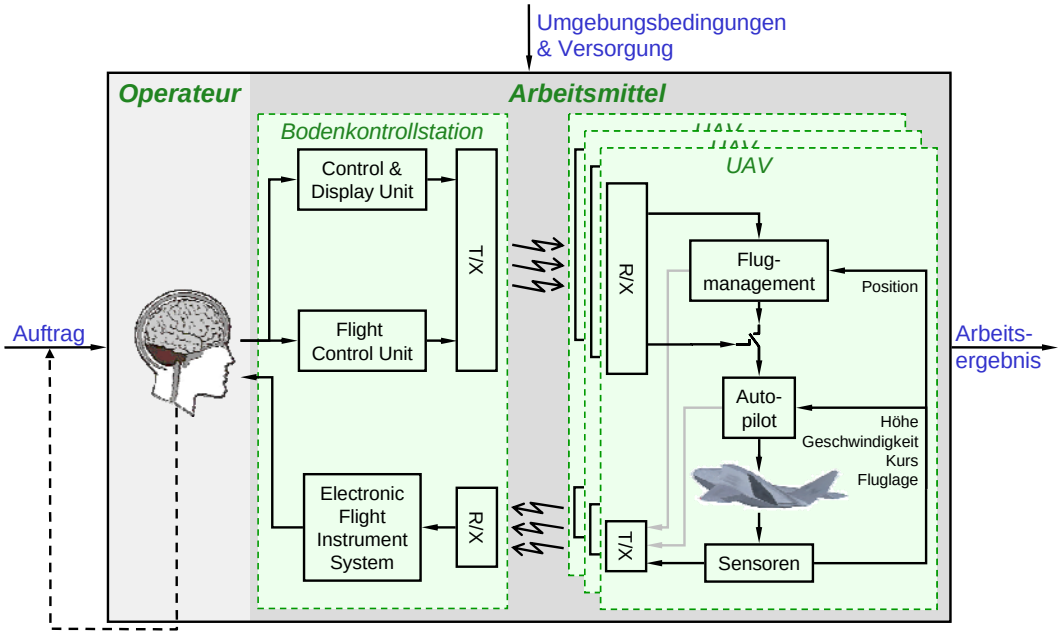


Abbildung 2-6: Arbeitssystem „Führung mehrerer UAVs“

Neben den in Abbildung 2-6 dargestellten klassischen bordseitigen Automationsfunktionen der UAVs sind im Hinblick auf die Führung mehrerer UAVs zusätzliche Funktionalitäten zum Beispiel in den Bereichen Kollisions- und Hindernisvermeidung, Formationsflug und Ansteuerung auf hohem Abstraktionsniveau nützlich [Ryan et al., 2004]. In diesem Umfeld existiert eine Reihe von Arbeiten unterschiedlicher Technologiereife, die diese Aspekte zumeist mit Hilfe von Funktionen zur automatisierten Routen- bzw. Trajektorienplanung und -verfolgung adressieren (z.B. [Beard et al., 2002] [Dogan, 2003][Gu et al., 2004][Kuwata et al., 2007][Leuthäusser & Raupp, 1991] [Moitra et al., 2003][Pettersson & Doherty, 2004][Rathinam et al., 2004][Shima et al., 2007], vgl. auch Seite 2). Die vorliegende Arbeit geht davon aus, dass Funktionalitäten in den Bereichen Kollisions- und Hindernisvermeidung sowie Formationsflug und automatisierte Routenplanung für einzelne und mehrere UAVs verfügbar sind und konzentriert sich auf den Aspekt der Führung mehrerer UAVs auf hohem Abstraktionsniveau. Diese erfolgt allerdings nicht auf Basis konventioneller Ansätze, die wie oben angedeutet meist im Umfeld von Routen- und Trajektorienplanung bzw. -verfolgung angesiedelt sind, sondern nutzt kognitive Automation, um UAVs zur Kooperation untereinander zu befähigen und die in Abschnitt 2.1.2.3 erläuterten Probleme zu vermeiden.

2.3.3 Einführung kognitiver Automation

Um nun den Operateur im Arbeitssystem „Führung mehrerer UAVs“ zu unterstützen, wird im Folgenden zusätzliche, kognitive Automation in das im vorigen Abschnitt skizzierte Arbeitssystem eingeführt (vgl. Abschnitt 2.1.2). Hierbei werden verschiedene Möglichkeiten der Einführung von ACUs vorgestellt und charakterisiert (vgl. auch [Kriegel et al., 2007][Onken & Schulte, in Vorb.]).

Zunächst können die UAVs in semi-autonome Systeme eingebettet werden, die je durch eine *Supporting ACU* bedient werden (siehe Abbildung 2-7). Diese *Supporting ACU* hat Zugriff auf notwendige Informationen und kann auf verschiedenen Eingriffsebenen das UAV steuern bzw. über Funkstrecken mit dem Operateur kommunizieren. An dieser Stelle im System finden sich im Bereich realisierter UAVs auch heute schon Funktionalitäten wie Missionsmanagement (z.B. WASLA-HALE, weitreichende abstandsfähige signalerfassende luftgestützte Aufklärung – High Altitude Long Endurance, [Stütz et al., 2001][Adam et al., 2003][Stütz, 2004]; ARTIS, Autonomous Rotorcraft Testbed for Intelligent Systems [Dittrich et al., 2003][Thielecke et al., 2004][Adolf, 2007]) und Health Monitoring (z.B. Barracuda, [Jarasch et al., 2006]), die jedoch nicht die zuvor formulierten Anforderungen an kognitive Automation berücksichtigen.

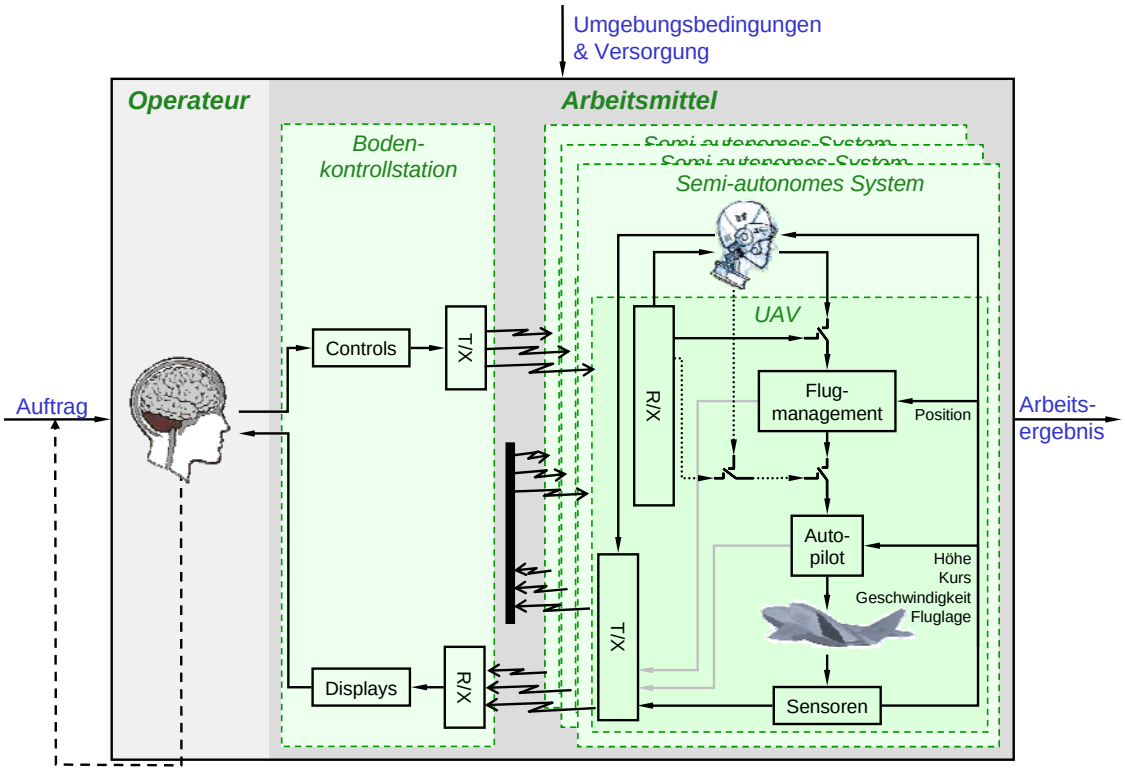


Abbildung 2-7: Führung mehrerer semi-autonomer UAVs

In dieser Konfiguration müssen den UAVs keine detaillierten Vorgaben mehr gemacht werden, wie sie eine gegebene Aufgabe bearbeiten sollen (z.B. Vorgabe eines Missionsplans), sondern die eingeführten *Supporting ACUs* sind in der Lage, Aufgabenstellungen auf dem Abstraktionsniveau eines (Teil-)Auftrags zu verstehen und unter Nutzung der ihnen zugeordneten Arbeitsmittel zu bearbeiten (z.B. Aufklärung eines Gebietes). Damit werden die UAVs unabhängiger von hochfrequenten, detaillierten Vorgaben des Operateurs, wodurch auch die Anforderungen an Datenfunkverbindungen und rechtzeitige Reaktionen des Operateurs auf zeitkritische Änderungen in der Umgebung sinken. Außerdem wird der Operateur von der Bedienung der UAV-Subsysteme entlastet, auch wenn er gegebenenfalls weiterhin auf diese zugreifen kann.

Sind die neu gebildeten semi-autonomen Systeme allerdings voneinander unabhängig (vgl. vereinfachte Darstellung in Abbildung 2-8), so muss der Operateur nach wie vor für eine adäquate Aufgabendisposition sorgen. Dies umfasst die Zuweisung von Aufgaben an die semi-autonomen Systeme, die Anpassung dieser Zuordnung im Falle relevanter Situationsänderungen und die Überwachung der Durchführung der Aufgaben durch die *Supporting ACUs*.

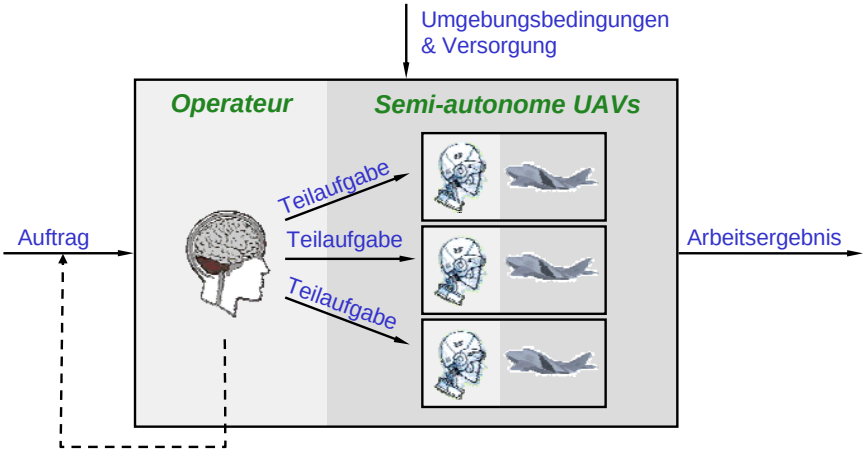


Abbildung 2-8: Unabhängige Supporting ACUs im Arbeitssystem „Führung mehrerer UAVs“

In einem nächsten Schritt können die semi-autonomen Systeme zusammengefasst werden, so dass sie ein disloziertes semi-autonomes System bilden, welches von einem Team aus *Supporting ACUs* geführt wird, wobei jeweils eine *Supporting ACU* einem UAV zugeordnet ist und sich an Bord dieses UAVs befindet (vgl. Abbildung 2-9).

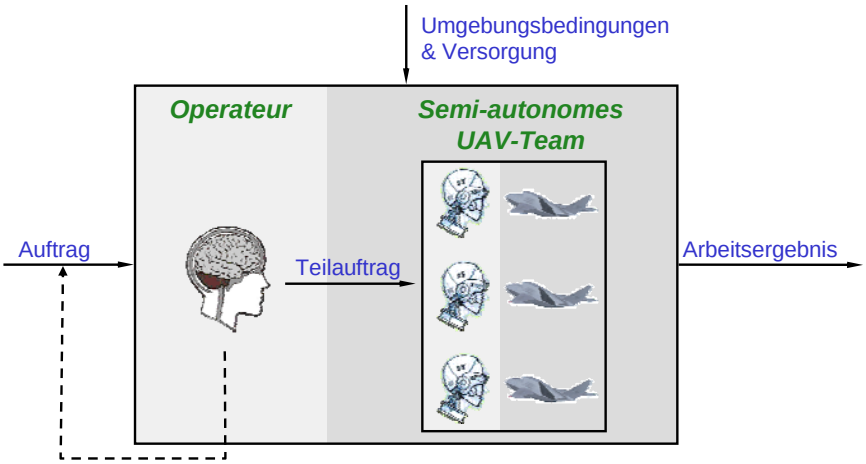


Abbildung 2-9: Team aus Supporting ACUs im Arbeitssystem „Führung mehrerer UAVs“

Diese Konfiguration ermöglicht, dass der Operateur eine Aufgabenstellung in Form eines Teilauftrags an ein Team von UAVs delegiert, welche selbständig notwendige Teilaufgaben ermitteln und diese einzelnen Teammitgliedern zuweisen. Dies schließt unter Umständen notwendige Rekonfigurationen auf Grund von Situationsänderungen ein. Die Entwicklung von *Supporting ACUs*, die über kooperative Fähigkeiten wie sie in einem solchen Arbeitssystem-Aufbau benötigt werden, ist Gegenstand der vorliegenden Arbeit und wird in den Kapiteln 3, 4 und 5 näher betrachtet. Während sich bisher bei Änderungen wie zum Beispiel dem Ausfall eines UAVs oder auch der Hinzunahme eines weiteren UAVs die Anzahl der Arbeitsmittel, die dem Operateur zur Verfügung standen, und sich damit die Struktur des Arbeitssystems änderte, steht dem Operateur nun ein Team als Arbeitsmittel zur Verfügung, das zwar in sich komplexer ist als ein semi-autonomes System mit einem UAV, ihm andererseits Aufgaben abnimmt und ebenfalls ein Verständnis der Zusammenarbeit mehrerer UAVs aufbaut. Spätestens mit diesem Schritt vollzieht sich der Übergang vom *operator* mehrerer dedizierter UAVs zum *capability manager* (vgl. [Frampton & Keirl, 2006]). Während sich im ersten Fall die Aufgabenlast des Operateurs mit der Anzahl der zu führenden UAVs erheblich ändern kann, liegt im zweiten Fall eine gänzlich andere Aufgabenstruktur vor, die nicht mehr prinzipiell von der Zahl der Vehikel abhängig ist. Im ersten Fall ist ein Hin- und

Herschalten des Operateurs zwischen mehreren nebenläufigen Führungsprozessen notwendig (*task switching*), für welches die Aufmerksamkeitsressourcen des Operateurs bei wachsender Anzahl von UAVs schnell erschöpft sind (vgl. [Cummings & Mitchell, 2005][Scott et al., 2006]). Im zweiten Fall kann die Aufmerksamkeit des Operateurs zumindest global einem Führungsprozess zugewendet werden.

Zwar ist ein semi-autonomes UAV-Team in der Lage, selbständig einen gegebenen Auftrag auszuführen, welcher vom Operateur in der Regel als Teilaspekt des ihm gegebenen Auftrags an die UAVs weitergegeben wird, jedoch können kooperative *Supporting ACUs* keine Initiative im Hinblick auf das Arbeitsziel des gesamten Arbeitssystems ergreifen, da sie dieses Arbeitsziel nicht kennen. Soll dies möglich sein, müssen die ACUs, die als Team aus *Supporting ACUs* im Bereich der Arbeitsmittel angeordnet waren, als *Operating ACUs* ausgeprägt werden, da sie in dieser Rolle das übergeordnete Arbeitsziel kennen und verfolgen können (vgl. Abbildung 2-10). Durch diesen Schritt ändert sich das Verhältnis zwischen Operateur und ACUs grundsätzlich weg von hierarchischer Überwachung hin zu kooperativer Zusammenarbeit im Hinblick auf das Arbeitsziel. Somit erfordert eine solche Konfiguration, dass die ACUs in der Lage sind, ein Team mit dem menschlichen Operateur zu bilden.

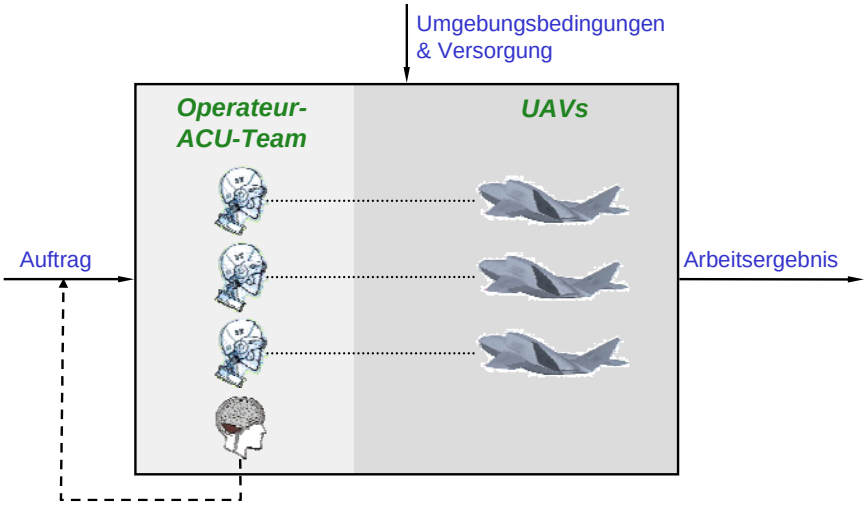


Abbildung 2-10: Team aus Operating ACUs und Mensch im Arbeitssystem „Führung mehrerer UAVs“

Dass hier wie auch schon in den beiden vorangegangenen Konfigurationen mehrere ACUs betrachtet werden, rührt daher, dass auch hier jedem UAV eine ACU zugeordnet und an Bord dieses UAVs angesiedelt sein soll, um möglichst unabhängig von Verfügbarkeit von Datenfunk und robust gegenüber dem Ausfall einzelner UAVs zu sein. Hier kommt also zum Tragen, dass Elemente der *Operating Force* ebenso wie *Operation Supporting Means* disloziert sein dürfen (vgl. [Onken & Schulte, in Vorb.]).

Aus Sicht des Arbeitssystem-Entwurfs orientiert sich der zuletzt vorgestellte Arbeitssystem-Aufbau an der Maxime, so viel Automation wie möglich auf Seiten des Bedieners ins Arbeitssystem einzuführen [Onken & Schulte, in Vorb.]. Diese Empfehlung basiert auf der Annahme, dass umso bessere Entscheidungen getroffen werden und damit die Leistung des Arbeitssystems als Ganzes umso besser wird, je größer der Anteil der eingesetzten Automation ist, die das übergeordnete Arbeitsziel versteht.

Der Unterschied zwischen einem Team aus *Supporting ACUs* und einem Team aus *Operating ACUs* soll abschließend an einem Beispiel erläutert werden, bei dem ein bemannter Hubschrauber über mehrere UAVs zur Routenaufklärung verfügen kann. Dieser bemannte Hubschrauber befindet sich auf dem Weg zu einem Ziel und benötigt

Aufklärungsdaten entlang der geplanten Route. Darüber hinaus gibt es mehrere Alternativrouten. Werden die UAVs nun von einem Team von *Supporting ACUs* geführt, so hat ein Operateur lediglich die Möglichkeit, den UAVs den Auftrag zu geben, eine bestimmte Route aufzuklären und muss bei Aufklärungsergebnissen, welche eine Routenänderung bedingen, selbst den Auftrag erteilen, dass eine andere Route aufgeklärt werden soll, da die UAVs nicht wissen, dass der Hubschrauber auf dem Weg zu einem bestimmten Zielpunkt ist. Werden die UAVs hingegen von einem Team von *Operating ACUs* geführt, so können diese bei entsprechenden Aufklärungsergebnissen selbständig die Aufklärung einer alternativen Route vorschlagen bzw. durchführen.

Im Sinne einer weiteren Optimierung des betrachteten Arbeitssystems „Führung mehrerer UAVs“ gemäß [Onken & Schulte, in Vorb.] sollte in allen vorgestellten Konfigurationen eine *Operating ACU* eingeführt werden (vgl. Abschnitt 2.1.2). Dies ist insbesondere für die Konfigurationen wichtig, in denen es noch keine *Operating ACUs* gibt, um den in Zusammenhang mit wachsender Komplexität von Automation zu erwartenden Problemen [Billings, 1997] entgegen zu wirken. Diese *Operating ACU* übernimmt die Funktion eines Assistenzsystems für den menschlichen Operateur, d.h. sie unterstützt ihn bei der Durchführung seiner spezifischen Aufgaben. Im Rahmen der Überwachung eines Teams aus *Supporting ACUs* (Abbildung 2-9) müsste ein solches Assistenzsystem beispielsweise ein Verständnis von „guter Teamarbeit“ haben, um erkennen zu können, wann die Zusammenarbeit der UAVs nicht mehr die zur Erfüllung der ihnen zugewiesenen Aufgaben nötige Qualität hat oder Optimierungspotential besteht. Um Empfehlungen für einen Eingriff seitens des Operateurs geben bzw. einen Eingriff selbst durchführen zu können, ist darüber hinaus Wissen über Handlungsoptionen nötig, die angewendet werden können, um Teamarbeit zu verbessern. Darüber hinaus ist ein Verständnis der Aufgaben und des aktuellen Beanspruchungszustands des Operateurs nötig, um situationsangepasst Unterstützung bieten zu können.

Gegenstand dieser Arbeit ist es jedoch, zunächst die Kooperation mehrerer *Supporting ACUs* auf Seiten der Arbeitsmittel im Sinne der in Abbildung 2-9 dargestellten Konfiguration zu betrachten, um im ersten Schritt kooperative Fähigkeiten losgelöst von den Schwierigkeiten, die die Integration eines menschlichen Teammitglieds mit sich bringt, realisieren zu können. Im Rahmen der Evaluierung dieser *Supporting ACUs* werden schließlich Anforderungen identifiziert, welche zusätzlich erfüllt sein müssen, um den Übergang von einem rein maschinellen Team aus *Supporting ACUs* zu einem Mensch-Maschine-Team aus *Operating ACUs* (Ausprägung: *substitution*) und Mensch im Sinne von Abbildung 2-10 durchführen zu können (siehe Abschnitt 6.4).

3 Konzept für wissensbasierte Kooperation künstlicher kognitiver Einheiten

Bisher waren die Verantwortlichkeit und die Kompetenz für die Verfolgung eines Arbeitsziels im Arbeitssystem dem Menschen als Bediener vorbehalten, während technische Systeme als Teil der Arbeitsmittel entsprechende Teilaufgaben bearbeiteten. Diese „klassische“ Zuordnung hat ihren Ursprung in den charakteristischen Stärken und Schwächen von Mensch und Maschine. So sind Menschen zum Beispiel in der Lage, zu improvisieren, Muster zu erkennen und zu generalisieren, während Maschinen beispielsweise schnell große Datenmengen verarbeiten und parallel mehrere Aufgaben bearbeiten können (siehe z.B. [Fitts, 1951]). Mit fortschreitenden Technologien zum Beispiel im Umfeld der künstlichen Intelligenz sind heutzutage allerdings viel versprechende Ansätze verfügbar, die Maschinen in die Lage versetzen, auch Aufgabenstellungen bearbeiten zu können, die dem Bereich menschlicher Kognition zuzuordnen sind. Derartige Fähigkeiten haben eine große Relevanz bei der Führung dynamischer Prozesse (vgl. [Hollnagel & Woods, 2005]) und sind Voraussetzung für die Entwicklung kognitiver Automation (siehe Abschnitt 2.1.2.3), die weitreichende Aufgabenstellungen im Arbeitssystem übernehmen soll.

In diesem Kapitel wird nun ein Konzept vorgestellt, wie Automation mit kognitiven und kooperativen Fähigkeiten ausgestattet werden kann, um ihrer Rolle als *Operating* bzw. *Supporting ACU* im Arbeitssystem gerecht werden zu können. Dazu muss eine ACU zunächst in der Lage sein, das Arbeitsziel bzw. den ihr zugewiesenen Teilauftrag zu verstehen und basierend darauf die Mission einschließlich aller Teilaufgaben durchzuführen. Hierzu muss verfügbare konventionelle Automation aufgabengerecht eingesetzt und es müssen übergeordnete Zielstellungen wie zum Beispiel Flugsicherheit berücksichtigt werden. Darüber hinaus ist die Fähigkeit zur Zusammenarbeit mit anderen Arbeitssystemkomponenten (vgl. Abschnitt 2.2) essentiell, die sich auf die Koordination der Zuweisung von Aufgaben an Teammitglieder ebenso erstreckt wie beispielsweise auf die Abstimmung der Nutzung von begrenzt verfügbaren Ressourcen.

Abschnitt 3.1 gibt zunächst einen Überblick über das Konzept maschineller Kognition und Kooperation, bevor die entsprechenden Bestandteile in den Abschnitten 3.2 („Kognition“) und 3.3 („Kooperation“) im Hinblick auf ihre Umsetzung im Rechner detailliert betrachtet werden. Dies umfasst jeweils eine Betrachtung verfügbarer Technologien und deren Anwendung in relevanten Applikationen sowie eine Vorstellung des in der vorliegenden Arbeit verwendeten Ansatzes.

3.1 Konzeptüberblick

Ziel der vorliegenden Arbeit ist die Entwicklung von *Supporting ACUs*, die als Team in die Arbeitsmittel eines Arbeitssystems zur Führung mehrerer UAVs eingebunden sind (vgl. Abschnitt 2.3.3, Abbildung 2-9). Die Kernfrage dieser Arbeit lautet daher:

Wie können kooperative Fähigkeiten von Automation auf Basis künstlicher Kognition im Rechner systematisch realisiert werden?

Um diese Frage beantworten zu können, werden zwei Teilaspekte betrachtet, nämlich zum einen wie Kognition im Rechner modelliert werden kann (vgl. Abschnitt 3.2) und

zum anderen was Kooperation ausmacht und wie kooperatives Verhalten erzeugt werden kann (vgl. Abschnitt 3.3).

Menschliche Kognition als „*allgemeine Bezeichnung für Prozesse und Produkte von Wahrnehmung, Erkennen, Denken, Schlussfolgern, Urteilen, Erinnern usw.*“ [Pschyrembel, 2002] wird von verschiedenen so genannten kognitiven Architekturen wie zum Beispiel ACT-R oder Soar im Rechner nachgebildet (vgl. Abschnitt 3.2.1). Hierbei steht die Modellierung des Menschen einschließlich seiner Stärken und Schwächen im Vordergrund, so dass solche Architekturen beispielsweise Fehlverhalten auf Basis von Ressourcenbeschränkungen, zeitverzögerte Reaktionen wegen Beschränkung der Verarbeitungsgeschwindigkeit von Information oder Effekte wie Vergessen aufweisen. Je nachdem wie stark die Modellierung solcher menschlicher Verarbeitungsengpässe ausgeprägt ist, sind kognitive Architekturen zunächst nur bedingt für die Entwicklung künstlicher kognitiver Einheiten geeignet, da diese im Allgemeinen zwar die Stärken menschlicher Kognition nutzen sollen, die Schwächen aber möglichst vermeiden wollen, um letztlich möglichst fehlerfreies, rationales und dennoch menschähnliches Verhalten hervorbringen zu können. Im Rahmen dieser Arbeit soll der so genannte *Kognitive Prozess* verwendet werden, wie er von [Putzer & Onken, 2003] vorgeschlagen wurde. Hierbei handelt es sich um ein Modell menschlicher Informationsverarbeitung, das sich an der zentralen Eigenschaft menschlicher Kognition, nämlich der Ausrichtung an Zielen [Anderson, 2001], orientiert und Verhalten auf Basis explizit repräsentierter Ziele erzeugt. Der Kognitive Prozess wird in Abschnitt 3.2.3.2 ausführlich erläutert.

Wegen der Berücksichtigung von Zielen ist der Kognitive Prozess insbesondere für die Modellierung so genannten wissensbasierten menschlichen Verhaltens im Sinne der von [Rasmussen, 1983] vorgeschlagenen Regulationsebenen menschlichen Verhaltens geeignet (siehe Abschnitt 3.2.2). Dieses zeichnet sich dadurch aus, dass Handlungsziele in unbekannten Situationen die Grundlage für die Planung des weiteren Vorgehens bilden. Diese Einbeziehung explizit repräsentierter Ziele grenzt wissensbasiertes von regelbasiertem Verhalten ab. Letzteres ist dadurch gekennzeichnet, dass bekannte Situationen mit stereotypen Handlungsweisen verknüpft werden, die beim Auftreten einer Situation ausgeführt werden sollen. Die Möglichkeit, auf wissensbasierter Verhaltensebene zu agieren, versetzt Menschen also in die Lage, auch in unbekannten Situationen, für die nicht feststeht, was zu tun ist, sinnvoll, d.h. im Hinblick auf Ziele, agieren zu können.

Für künstliche kognitive Einheiten ist diese Fähigkeit insbesondere dann relevant, wenn sie in komplexen Umgebungen agieren sollen. Hierbei können im Allgemeinen nicht alle Situationskonfigurationen, die potentiell auftreten können, vom Systementwickler spezifiziert und mit entsprechenden Aufgaben oder Aktionen attributiert werden. Dies ist zum Beispiel bei kooperativen Flugmissionen wie sie im Rahmen dieser Arbeit betrachtet werden der Fall, da mit der Beteiligung mehrerer Akteure die Anzahl möglicher Situationskonfigurationen stark ansteigt. Gibt es beispielsweise bei der Bekämpfung eines gegnerischen Ziels durch ein UAV nur eine Möglichkeit für das Teilproblem der Zuweisung der Bekämpfungsaufgabe an ein UAV, so sind es bei der Bekämpfung von drei Zielen durch drei UAVs – sofern davon ausgegangen wird, dass jedes UAV bis zu drei Ziele bekämpfen kann – infolge rein kombinatorischer Betrachtungen bereits 27 Möglichkeiten. Im Allgemeinen sind jedoch in einem relevanten Szenario noch deutlich mehr Akteure bzw. vielfach parametrisierbare Umgebungselemente und zusätzliche Aufgaben vorhanden, so dass die Anzahl der Möglichkeiten mit einem Ansatz, der die Situationskonfigurationen beschreiben will, schnell nicht mehr geschlossen handhabbar wird.

Daher werden in dieser Arbeit wie oben angedeutet Ziele von Kooperation zur übergeordneten Strukturierung definiert, die beschreiben, wodurch kooperative Zusammenarbeit gekennzeichnet ist (z.B. Vermeidung von unnötiger redundanter Aufgabenausführung durch mehrere Teammitglieder). Um die Relevanz der Ziele in der jeweiligen Situation bewerten zu können, muss dabei ein Verständnis der aktuellen Zusammenarbeit im Team aufrechterhalten werden (z.B. Zuweisung von Aufgaben an Teammitglieder). Dieses bildet auch die Grundlage für die Auswahl von Handlungsalternativen, die Möglichkeiten darstellen, wie Ziele in entsprechenden Situationen tatsächlich erreicht werden können (z.B. Abbruch der Durchführung einer Aufgabe oder Anfrage an ein Teammitglied, die Bearbeitung einer Aufgabe abzubrechen). Eine Zusammenstellung der im Rahmen dieser Arbeit für kooperative Zusammenarbeit notwendigen Konzepte findet sich in Abschnitt 4.3.

Die Definition solcher Konzepte basiert auf der Betrachtung von menschlicher Kooperation und Teamarbeit sowie der Analyse von Forderungen an effektive Zusammenarbeit von Mensch und Maschine und der Einbeziehung von Methoden aus dem Bereich der Multi-Agenten-Systeme (siehe Abschnitt 3.3). Kooperation bezeichnet dabei die Zusammenarbeit mehrerer Akteure im Team um ein gemeinsames Ziel zu erreichen (vgl. Abschnitt 3.3.2). Dazu ist es nötig, die Aktivitäten der einzelnen Teammitglieder zu koordinieren, d.h. Abhängigkeiten wie zum Beispiel dass die Bearbeitung einer Aufgabe abgeschlossen sein muss, bevor eine andere begonnen werden kann, aufzulösen (vgl. Abschnitt 3.3.3). Hierzu wird Kommunikation eingesetzt, d.h. Information verschiedenster Art zwischen den Beteiligten ausgetauscht. Die vorliegende Arbeit beschränkt sich dabei auf explizite Kommunikation, d.h. speziell den Austausch von Nachrichten, und verwendet Interaktionsprotokolle zur Strukturierung von Dialogen von Teammitgliedern (vgl. Abschnitt 3.3.4).

In den folgenden Abschnitten wird nun auf die einzelnen Teilkonzepte des Ansatzes näher eingegangen, wobei Abschnitt 3.2 die Umsetzung künstlicher Kognition im Rechner adressiert und Abschnitt 3.3 kooperatives Verhalten näher betrachtet. Kapitel 4 beschreibt anschließend eine Methode zur Definition von Wissensmodellen und geht detailliert auf diejenigen Wissensmodelle ein, die im Umfeld kognitiver Kooperation relevant sind.

3.2 Künstliche Kognition

Um künstliche Kognition realisieren zu können, ist es nahe liegend, als grundlegenden Ansatz Modelle menschlicher Kognition zu verwenden. Menschliche Kognition (lat. *cognoscere* – „erkennen“) umfasst sämtliche Wahrnehmungs-, Gedächtnis- und Denkprozesse, die menschlichem Verhalten zu Grunde liegen (vgl. [Solso, 2005]). Sie zeichnet sich dadurch aus, dass sie stets zielgerichtet ist, d.h. versucht, Ziele zu erreichen bzw. Gegebenheiten, die das Erreichen eines Ziels behindern, zu beseitigen [Anderson, 2001].

Kognition resultiert in menschlichem Verhalten, das sich nach [Rasmussen, 1983] in drei Kategorien klassifizieren lässt. Auch wenn es stets zielgerichtet ist, sind die verfolgten Ziele nur im Rahmen wissensbasierten Verhaltens explizit repräsentiert, wobei die Verarbeitung expliziter Ziele verhältnismäßig vieler Ressourcen bedarf. Dadurch dass Ziele auf regel- und fertigkeitsbasierter Verhaltensebene jedoch lediglich implizit berücksichtigt werden, wird insgesamt dennoch effizientes menschliches Verhalten möglich.

Im Folgenden werden nun zunächst im Rechner umgesetzte Modelle menschlicher Kognition vorgestellt, bevor die verschiedenen Ebenen menschlichen Verhaltens nach [Rasmussen, 1983] charakterisiert werden und auf Ansätze eingegangen wird, die insbesondere für die Modellierung wissensbasierten Verhaltens geeignet sind. Dieser Abschnitt beschäftigt sich also mit der Beantwortung der Teilfrage: *Wie kann wissensbasiertes kognitives Verhalten im Rechner realisiert werden?*

3.2.1 Modellierung menschlicher Kognition

Die kognitive Psychologie als wissenschaftliche Disziplin untersucht menschliches Denken und unterscheidet dabei zwei Ansätze zur Modellierung von Kognition. Das informationsverarbeitende Modell nimmt an, dass Information sequentiell verarbeitet wird, wobei insbesondere verarbeitende Teilfunktionalitäten und die Art der Wissensrepräsentation von Interesse sind. Das neurowissenschaftliche Modell hingegen nimmt die neuronale Struktur des zentralen Nervensystems des Menschen als Vorbild für die Verarbeitung von Information, die sich insbesondere durch Parallelität auszeichnet. [Solso, 2005]

In diesem Abschnitt werden zwei kognitive Architekturen vorgestellt, welche dem informationsverarbeitenden Modell zugeordnet werden können. Unter einer kognitiven Architektur versteht man eine (im Rechner implementierte) Theorie der Mechanismen und Strukturen, auf denen menschliche Kognition beruht [Lehman et al., 1998][Byrne, 2003]. Kognitives Verhalten wird in diesem Kontext erzeugt, indem eine solche Architektur mit Inhalt, d.h. Wissen, gefüllt wird, was [Lehman et al., 2006] durch folgende schematische Formel beschreiben:

Verhalten = Architektur + Inhalt
[Lehman et al., 2006]

Neben ACT-R und Soar als prominente kognitive Architekturen und *Unified Theories of Cognition* im Sinne von [Newell, 1990], die im Folgenden vorgestellt werden, gibt es eine Vielzahl weiterer Architekturen mit verschiedenen Schwerpunkten. Einen Überblick hierüber geben zum Beispiel [Ritter et al., 2002].

3.2.1.1 ACT-R

Bei ACT-R (*Adaptive Control of Thought – Rational*) [Anderson et al., 2004][ACT-R Homepage] handelt es sich um eine kognitive Architektur, deren Schwerpunkt auf der Modellierung menschlicher Kognition mit ihren Stärken und Schwächen liegt, so dass menschliches Verhalten mit eben diesen Stärken und Schwächen nachgebildet werden kann. Somit können Vorhersagen über menschliches Verhalten getroffen werden bzw. kann menschliches Verhalten, welches zum Beispiel in psychologischen Experimenten beobachtet wurde, nachgebildet und erklärt werden [Taatgen et al., 2006].

Die folgende Darstellung von ACT-R basiert im Wesentlichen auf [Anderson et al., 2004], [Taatgen et al., 2006] und [Ritter & Kim, 2007]. Abbildung 3-1 zeigt den modularen Aufbau von ACT-R, wobei verschiedene Elemente der Architektur mit Hirnregionen des Menschen in Verbindung gebracht werden und jedes Modul eine andere Art von Information verarbeitet. Das visuelle Modul dient hierbei zur Objektlokalisation und -erkennung, das manuelle Modul dem Einsatz der Motorik, d.h. zum Beispiel der Hände, das Absichtenmodul der Speicherung von Absichten sowie Zielen und Unterzielen und das deklarative Modul dem Vorhalten von deklarativem Wissen in Form von Fakten. Diese Module tauschen Informationseinheiten (*chunks*) mit dem zentralen Produktionensystem aus. Hierzu werden Zwischenspeicher (*buffer*) verwendet, deren Grö-

ße auf eine Informationseinheit beschränkt ist. Das Produktionensystem basiert auf prozeduralem Wissen, welches in Form von Regeln hinterlegt ist, und kann Muster in den Zwischenspeichern erkennen bzw. diese verändern, um Anfragen an die Module zu stellen. Während die einzelnen Module parallel zueinander arbeiten, verarbeitet das Produktionensystem Regeln seriell, d.h. eine Regel pro Zyklus.

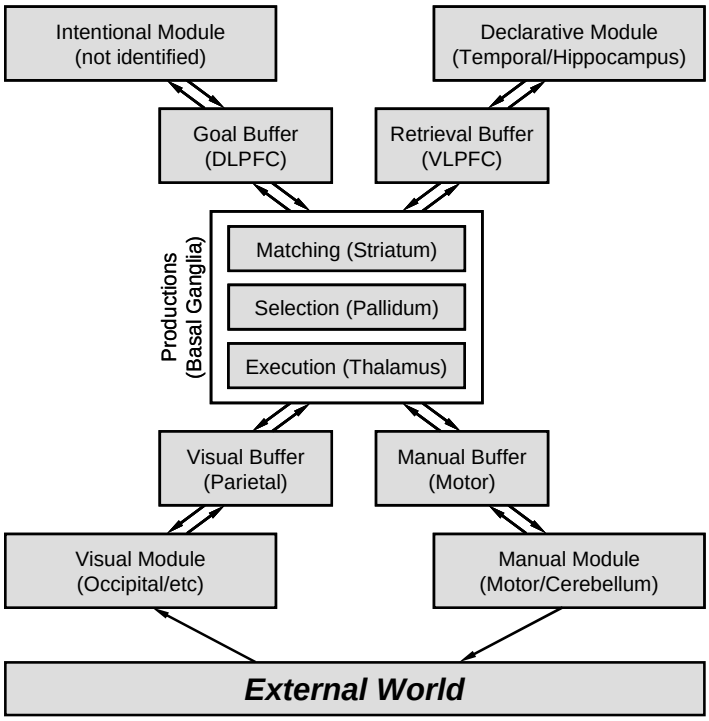


Abbildung 3-1: Aufbau von ACT-R; in Klammern: Hirnregionen, mit denen spezifische Funktionalitäten assoziiert werden (DLPFC: Dorsolateral Prefrontal Cortex; VLPFC: Ventrolateral Prefrontal Cortex) [Anderson et al., 2004]

Der symbolischen Repräsentation von Fakten und Prozeduren stellt ACT-R subsymbolische Informationen zur Seite, die die Verwendung des symbolisch repräsentierten Wissens bestimmen. Damit können Effekte wie Vergessen oder auch menschliches Fehlverhalten modelliert werden. Solches Wissen kommt zum Beispiel dann zum Tragen, wenn vom deklarativen Modul ein Faktum angefragt wird. In diesem Fall wird das Faktum mit der größten Aktivierung zur Verfügung gestellt, wobei letztere unter anderem vom Zeitpunkt des letzten Aufrufs und dem Übereinstimmungsgrad mit der Spezifikation der Information abhängt. Weiterhin wird subsymbolisch repräsentiertes Wissen genutzt, um für Produktionen den erwarteten Nutzen im Hinblick auf das momentan zu erreichende Ziel zu berechnen, so dass die Produktion mit den niedrigsten erwarteten Kosten (z.B. benötigte Zeit für die Durchführung) ausgewählt wird, wenn mehrere Regeln angewendet werden könnten. Parameter, die die Grundlage für die Berechnung des subsymbolisch repräsentierten Nutzens von symbolisch repräsentiertem Wissen darstellen, werden in ACT-R mit Hilfe von Lernmechanismen angepasst. Darüber hinaus wird auch die Integration von Mechanismen zum Lernen von Produktionen in ACT-R untersucht [Anderson et al., 2004].

Anwendung von ACT-R

Die Anwendung von ACT-R erstreckt sich hauptsächlich auf die kognitive Psychologie, wobei in verschiedenen Forschungszweigen wie zum Beispiel Wahrnehmung und Aufmerksamkeit oder Problemlösen und Entscheiden quantitative Daten, die in Experimenten mit Menschen erhoben wurden, mit denen verglichen werden, die von einem ACT-

R-Modell stammen ([ACT-R Homepage], vgl. auch [Jones, 1996]). Die Anwendung von ACT-R erstreckt sich über diese grundlagenorientierten Felder hinaus vereinzelt in Applikationsdomänen wie zum Beispiel die Luftfahrt, wo beispielsweise Pilotenverhalten einschließlich Pilotenfehlern beim Rollen auf Flughäfen modelliert wurde [Byrne & Kirlik, 2002][Byrne & Kirlik, 2003] und die Entwicklung eines ACT-R-Modells angedacht aber bisher nicht umgesetzt ist, welches ein Mitglied eines menschlichen Teams zur Führung des Predator-UAVs ersetzen soll [Gluck et al., 2005]. Jedoch setzt sich auch hier die skizzierte Schwerpunktsetzung fort, nämlich die exakte Nachbildung menschlichen Verhaltens auf Basis der Struktur des menschlichen Gehirns mit den genannten Eigenschaften wie zum Beispiel Ressourcenengpässen und Vergessen. Daher eignet sich ACT-R nur bedingt für die Entwicklung künstlicher kognitiver Einheiten zur UAV-Führung im Rahmen dieser Arbeit, da hier die Modellierung von rationalem und möglichst fehlerfreiem Verhalten im Vordergrund steht.

3.2.1.2 Soar

Neben ACT-R stellt Soar [Laird et al., 1987][Newell, 1990][Lehman et al., 1998] [Lehman et al., 2006] eine weitere prominente kognitive Architektur dar (für einen Vergleich siehe zum Beispiel [Jones, 1996] und [Johnson, 1997]). Obwohl Soar als allgemeingültige kognitive Architektur mit dem Ziel der Nachbildung menschlicher Kognition entworfen wurde, so erstreckt sich ihre Nutzung hauptsächlich auf die Modellierung rationalen Verhaltens [Jones, 1996]. Soar basiert dabei auf den folgenden Eigenschaften kognitiven Verhaltens, welches nach [Newell, 1990] (a) zielgerichtet ist, (b) eine reichhaltige, komplexe, detaillierte Umgebung widerspiegelt, (c) eine große Menge an Wissen benötigt, (d) Symbole und Abstraktionen nutzt, (e) flexibel und abhängig von der Umgebung ist und (f) aus der Umgebung und Erfahrung lernt [Lehman et al., 1998]. Je nach Verwendung kann Soar als Theorie von Kognition, als Menge von Prinzipien und Einschränkungen kognitiver Verarbeitung, als Implementierung dieser Prinzipien und Einschränkungen in einer Programmiersprache oder als Programmiersprache im Umfeld der Künstlichen Intelligenz angesehen werden [Ritter & Kim, 2006].

Probleme werden in Soar als Suche nach einem Zielzustand in einem sogenannten *Problemraum* beschrieben. Ein Problemraum enthält eine Menge von Operatoren, die auf einen aktuellen Zustand angewendet werden können, um in einen neuen Zustand zu gelangen (vgl. Abbildung 3-2). [Laird et al., 1987]

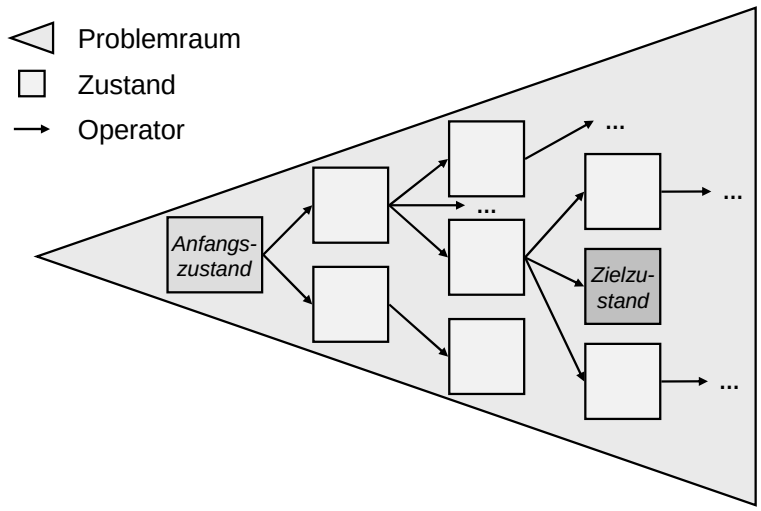


Abbildung 3-2: Problemraum als Rahmen für die Lösung eines Problems (vgl. [Lehman et al., 2006])

Die Auswahl und Anwendung von Operatoren ist daher in Soar und insbesondere in dessen Entscheidungszyklus (vgl. Abbildung 3-3, [Laird, 2006]) zentral. Auf die Erfassung von Daten aus der Umgebung (*input*) folgt der Einsatz allen Wissens, welcher in dieser Situation relevant ist, einschließlich des Vorschlags von Operatoren, die in der aktuellen Situation angewendet werden könnten (*propose*). Im nächsten Schritt wird einer der vorgeschlagenen Operatoren auf Basis vorhandenen Wissens ausgewählt (*select*). Der ausgewählte Operator wird anschließend angewendet (*apply*) und schließlich werden – falls vorhanden – Kommandos an die Umgebung gesendet (*output*).

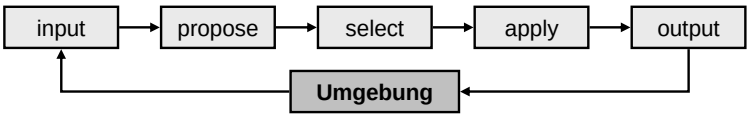


Abbildung 3-3: Soar Entscheidungszyklus (vgl. [Laird, 2006])

Soar geht davon aus, dass es genau vier mögliche (generische) Fälle gibt, die die Lösung eines Problems behindern können (*impasses*) und die in unterschiedlichen Phasen des Entscheidungszyklus relevant werden [Wray & Jones, 2006], nämlich

1. Im aktuellen Zustand werden keine Operatoren zur Ausführung vorgeschlagen (*state no-change*)
2. Die Anwendung eines ausgewählten Operators verändert den Zustand nicht bzw. es ist nicht bekannt, wie ein ausgewählter Operator angewendet werden kann (*operator no-change*).
3. Es sind mehrere Operatoren vorgeschlagen, aber es liegt kein Wissen vor, das die Auswahl eines Operators steuert (*operator tie*).
4. Es sind mehrere Operatoren vorgeschlagen, aber das Wissen zur Steuerung der Auswahl eines Operators ist widersprüchlich (*operator conflict*).

Tritt ein solcher Fall ein, erzeugt Soar automatisch einen neuen Problemraum, der dem aktuellen untergeordnet ist und dessen Ziel die Auflösung des erkannten *impasses* ist. Wird ein solcher Konflikt durch die Anwendung entsprechenden Wissens gelöst, so sorgt der in Soar implementierte Lernmechanismus (*Chunking*) dafür, dass Wissen über die Auflösung eines *impasses* gespeichert und beim nächsten Auftreten dieses Konflikts eingesetzt wird, so dass die Lösung nicht erneut vom System erarbeitet werden muss.

Im Kern basiert Soar wie auch ACT-R auf einem Produktionensystem, das aus einem Langzeitspeicher, in dem Applikationswissen in Form von Regeln hinterlegt ist, und einem Kurzzeitspeicher (in Soar: Arbeitsspeicher) besteht (vgl. Abbildung 3-4).

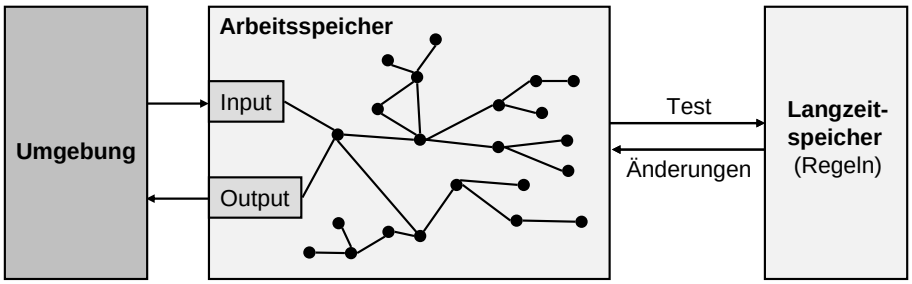


Abbildung 3-4: Prinzipieller Aufbau von Soar (vgl. [Putzer, 2004])

Teile des Arbeitsspeichers, in dem Wissen symbolisch in Form eines semantischen Netzes repräsentiert ist, sind dabei für die Interaktion mit der Umgebung reserviert. Trotz der Repräsentation von Wissen in Form von Regeln und deren Anwendung auf einen Kurzzeitspeicher unterscheidet sich Soar durch seine architekturellen Merkmale

wie zum Beispiel dem Vorhandensein von Problemräumen und deren automatische Erzeugung oder dem integrierten Lernmechanismus deutlich von klassischen Produktionensystemen (vgl. [SoarTech, 2002]).

Dass Soar für die Entwicklung interaktiver komplexer Systeme prinzipiell gut geeignet ist, zeigt die Vielzahl von Anwendungen, für die Soar eingesetzt wurde und wird, auch wenn es speziell im Hinblick auf die Entwicklung künstlicher kognitiver Einheiten einige Defizite aufweist (z.B. keine explizite Repräsentation von Applikationszielen, keine Unterstützung objektorientierter Paradigmen). Im Folgenden werden zwei dieser Anwendungen beschrieben, die sehr bekannt und im Umfeld kooperativer UAV-Führung von besonderer Bedeutung sind.

Anwendung: TacAir-Soar

Im Rahmen des von der DARPA geförderten Projekts TacAir-Soar, das an der University of Michigan bearbeitet wurde und aus dem die Firma SoarTech als Spin-Off Unternehmen hervorging, wurden für das Training von Kampfflugzeugpiloten sowohl eigene als auch gegnerische Kräfte, d.h. Piloten, mit Hilfe von über 7500 Regeln in Soar simuliert [SoarTech, 2005]. Deren Verhalten sollte dem von menschlichen Piloten auf taktischer Ebene möglichst nahe kommen [Jones et al., 1993][Rosenbloom et al., 1994] [Jones et al., 1999]. Hierbei wurde auf Basis von Expertenbefragungen ein Agent pro Flugzeug implementiert und in mehreren Simulationskampagnen getestet. Das Expertenwissen wurde in einer Operatorenhierarchie repräsentiert. Zusätzlich gibt es implizite Zielsetzungen wie zum Beispiel das Aufrechterhalten von Situationsbewusstsein und opportunistische Ziele wie zum Beispiel Überleben, welche unabhängig von der Operatorenhierarchie ausgewählt werden [Jones et al., 1999].

Ein Teil von TacAir-Soar befasst sich mit der Koordination mehrerer Agenten sowohl im Rahmen rein maschineller als auch gemischter Teams, d.h. Teams, welche aus Menschen und maschinellen Agenten bestehen [Laird et al., 1994][Laird et al., 1998]. Sie basiert auf der Nachbildung existierender Methoden und Praktiken militärischer Organisationen, so dass die entwickelten Agenten in solche Organisationen eingebunden werden können. Diese Strukturen stellen im Wesentlichen eine statische Hierarchie dar, in der Vorausplanung und Training dazu genutzt werden, den Kommunikationsaufwand und eine Neuorganisation der beteiligten Flugzeuge auf ein Minimum zu reduzieren. Am Beispiel einer Formation aus Leader und Wingman beschreiben [Laird et al., 1994] Verhaltensmuster im Bereich des Manövrierens, Sensierens, Waffeneinsatzes, Formationsflugs und der Zielzuweisung, welche die Grundlage für koordiniertes Verhalten bilden. Die Koordination der Flugzeuge findet dabei durch gemeinsame Verhaltensgrundsätze, Missionsbriefing, Beobachtung anderer (sehr eingeschränkt implementiert) und explizite Kommunikation durch Funk – ähnlich menschlichem Nachrichtenaustausch – statt [Laird et al., 1998]. Um die gestellten Zielsetzungen zu erreichen, nämlich möglichst schnell auf Änderungen der Umgebung reagieren zu können und existierende militärische Organisationsstrukturen nachzubilden, wurde Koordination hier auf regelbasierter Verhaltensebene (vgl. Abschnitt 3.2.2.2) implementiert und lässt sich wie folgt charakterisieren: *„although an agent knows how to coordinate, it may not know why its particular action or method of collaboration is the best. (It just knows it is the appropriate action to take.)" [Laird et al., 1998]* Dieser Ansatz ist für die spezifische Applikation und deren Zielsetzungen gut geeignet, schränkt allerdings die Flexibilität und Fähigkeit von Agenten, sich in unbekannten Situationen zurechtzufinden, gegenüber einer Vorgehensweise, die wissensbasiertes Verhalten mitberücksichtigt, ein (vgl. Abschnitt 3.2.2.1).

Hinsichtlich der Entwicklung solch umfangreicher regelbasierter Systeme zeigt sich am Beispiel von TacAir-Soar, dass dieses Software-System wegen der großen Anzahl von Regeln kaum wartbar und erweiterbar ist (vgl. [Pearson & Laird, 2003]). Bei der hier vorliegenden Systematik der Wissensrepräsentation, die im Wesentlichen dadurch gekennzeichnet ist, dass alle Regeln gleichwertig nebeneinander stehen, müsste nämlich jeder Entwickler eigentlich alle bereits implementierten Produktionen kennen und deren Verhalten im Detail verstehen, bevor eine neue Regel hinzugefügt werden kann (vgl. [Yakir & Kaminka, 2007]).

Anwendung: STEAM

STEAM (Shell for TEAMwork) [Tambe, 1997] stellt ein allgemeingültiges, in Soar implementiertes Modell von Teamarbeit dar, das auf einer Hierarchie von gemeinsamen Absichten basiert. Darüber hinaus überwachen Teammitglieder die Leistung des Teams und seiner Mitglieder und organisieren das Team bei Bedarf neu. Außerdem wird überflüssige Kommunikation soweit möglich vermieden.

In [Tambe, 1997] wird die Anwendung von STEAM in drei Domänen beschrieben: (1) die Durchführung einer Angriffsmission, bei der acht Hubschrauber nach einer festgelegten Vorgehensweise gegnerische Ziele bekämpfen, (2) die Durchführung einer Transportmission, bei der Transporthubschrauber – von Begleithubschraubern beschützt – Truppen an Land bringen und (3) RoboCup Soccer. Obwohl STEAM als applikationsunabhängiges Framework für Teamarbeit gedacht ist, waren dennoch umfangreiche Änderungen in der Regelbasis notwendig, als STEAM von der Anwendung im Rahmen von Hubschraubermissionen auf die Anwendung im Rahmen von RoboCup Soccer angepasst wurde.

Teamarbeit in STEAM basiert auf der expliziten Repräsentation von Zielen und Plänen eines Teams (nicht aber der Ziele von Teamarbeit) sowie der Verpflichtungen einzelner Teammitglieder als wesentliche Voraussetzung für Koordination und Kommunikation im Team. Missionsrelevante Zielsetzungen sind dabei in hierarchischen reaktiven Plänen hinterlegt, welche eine Hierarchie von Operatoren darstellen. Abbildung 3-5 zeigt eine solche Hierarchie für eine Angriffsmission, wobei Teamoperatoren durch eckige Klammern gekennzeichnet sind und ein Stern angibt, dass typischerweise Sub-Teams diese Aufgabe ausführen. Eine gemeinsame Absicht aller Teammitglieder kommt dann zustande, wenn alle Teammitglieder einen (Team-)Operator auswählen.

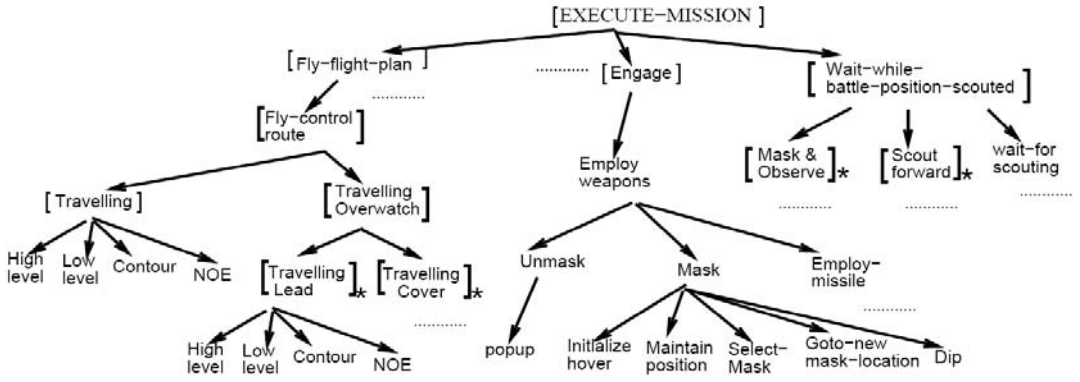


Abbildung 3-5: Team-Operatoren für eine Angriffsmission in STEAM (aus [Tambe, 1997])

Die Anwendung von Operatoren erfolgt je nach Art des Operators in einem Team-Zustand oder einem Zustand, der der Ausführung eigener Operatoren vorbehalten ist. Ein Team-Zustand enthält dabei Informationen über die gemeinsame Sichtweise des Teams

hinsichtlich der externen Welt. Üblicherweise wird er mit Informationen über das Team wie zum Beispiel den Teammitgliedern, möglichen Sub-Teams, verfügbaren Kommunikationskanälen und dem (vorbestimmten) Teamleiter initialisiert. Bevor ein Teamoperator angewendet werden kann, muss zunächst mit Hilfe des *establish commitment*-Protokolls, das durch den Teamleiter ausgeführt wird, eine gemeinsame Absicht geschaffen werden. Diese wird erzeugt, falls alle Teammitglieder dem Vorschlag zustimmen.

Die Überwachung von Teammitgliedern basiert auf Rollen, d.h. abstrakten Spezifikationen der Aktivitäten (Operatoren), die ein einzelner Agent oder ein Sub-Team ausführen kann, um die übergeordnete Aktivität zu erreichen. Basierend auf diesen Rollen leitet STEAM ab, ob ein Teamoperator unerreichbar geworden ist. Je nachdem, welche Ursache vorliegt, kann im Prinzip jedes Teammitglied entsprechend reagieren. Damit gibt es zwar eine Abhängigkeit von den a-priori festgelegten Team- und Sub-Teamleitern, ohne die keine gemeinsamen Absichten eingegangen werden können (s.o.), allerdings können sich alle Teammitglieder einbringen, wenn es um die konkrete Durchführung der Absichten geht, so dass diese dezentral koordiniert wird.

Wie auch bei TacAir-Soar findet Koordination in STEAM auf regelbasierter Verhaltensebene statt, ohne die übergeordneten Ziele von Teamarbeit (nicht zu verwechseln mit den Zielen der Zusammenarbeit wie zum Beispiel das Erfüllen einer Mission) explizit zu repräsentieren. Entgegen TacAir-Soar-Agenten ist es STEAM-Agenten nicht möglich, im Rahmen eines Teams, das sich aus Menschen und Maschinen zusammensetzt, zu agieren, da weder menschliche Koordinationsstrukturen nachgeahmt werden noch Verhalten im Team auf Basis einer Repräsentation expliziter Ziele erklärt werden kann. Diese Eigenschaften ändern sich auch mit der vor kurzem erfolgten Re-Implementierung von STEAM in Java und gleichzeitiger Umbenennung in Machinetta [Schurr et al., 2005] nicht.

3.2.2 Qualitative Modellierung menschlichen Verhaltens

Menschliche Kognition wie sie im vorhergehenden Abschnitt im Hinblick auf ihre Modellierung im Rechner charakterisiert wurde, resultiert in Verhalten. Dieses kann mit Hilfe des Modells von [Rasmussen, 1983] charakterisiert werden, welches zwischen menschlichem Verhalten auf fertigkeitsbasierter, regelbasierter und wissensbasierter Ebene unterscheidet (siehe Abbildung 3-6).

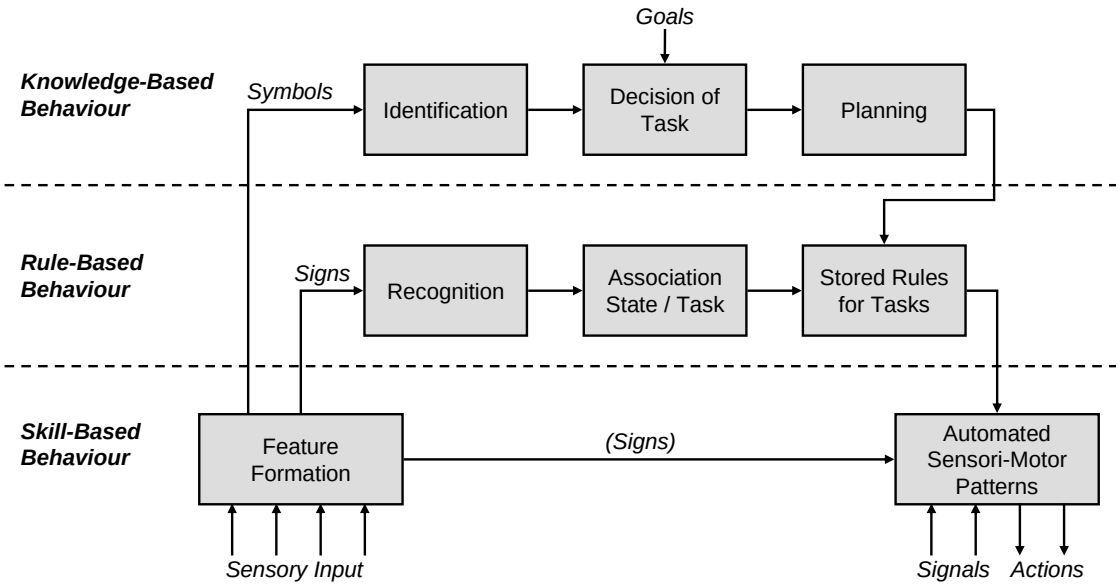


Abbildung 3-6: Modell menschlichen Verhaltens [Rasmussen, 1983]

Menschliches Verhalten entsteht grundsätzlich durch das Zusammenwirken aller drei Verhaltensebenen. Dabei ist fertigkeitsbasiertes Verhalten durch die unbewusste Ausführung automatisierter und hoch integrierter sensomotorischer Muster gekennzeichnet. Regelbasiertes Verhalten ist im Gegensatz dazu bewusstseinspflichtig und kommt immer dann zum Einsatz, wenn in einer bestimmten Situation bekannt ist, was zu tun ist. Wissensbasiertes Verhalten schließlich orientiert sich in unbekannten Situationen, für die keine Verhaltensregeln verfügbar sind, bei der Entscheidung, was als nächstes getan werden soll, an explizit repräsentierten Handlungszielen. [Rasmussen, 1983]

Im Folgenden werden die drei genannten Verhaltensebenen detaillierter betrachtet, wobei der Schwerpunkt auf der Betrachtung wissensbasierten Verhaltens und der dazu nötigen Konzepte (z.B. Ziele) und Mechanismen (z.B. Mittel-Ziel-Analyse) liegt, da im weiteren Verlauf der Arbeit Kooperation von *Supporting ACUs* auf dieser Verhaltensebene modelliert werden soll. In diesem Zusammenhang wird auch auf eine interpretierte Version des Rasmussen-Modells von [Onken & Schulte, in Vorb.] zurückgegriffen, welches eine einheitliche, im Hinblick auf eine Rechnermodellierung stringentere Darstellungsform aufweist. Anschließend werden in Abschnitt 3.2.3 zwei Theorien (*BDI-Modell* und *Kognitiver Prozess*) sowie zugehörige Architekturen und relevante Applikationen vorgestellt, deren Fokus gerade die Umsetzung dieses wissensbasierten Verhaltens im Rechner ist.

3.2.2.1 Merkmale der wissensbasierten Verhaltensebene

Menschen gehen hauptsächlich dann wissensbasiert vor, wenn sie sich in Situationen befinden, in denen regel- bzw. fertigkeitsbasierte Verhaltensmuster nicht zur Verfügung stehen oder Fehlverhalten auf einer dieser beiden Ebenen auftritt. Dabei zeichnet sich wissensbasiertes Verhalten durch die Orientierung an explizit formulierten und repräsentierten Zielen aus [Rasmussen, 1983]. Daneben kann auf wissensbasierter Verhaltensebene über parallel ablaufendes regel- und fertigkeitsbasiertes Verhalten reflektiert werden, um auf diese Weise zu einer Kontrolle der eigenen Handlungen zu gelangen.

Die von [Rasmussen, 1983] genannten Funktionsblöcke *identification*, *decision of task* und *planning* (siehe Abbildung 3-6) werden von [Onken & Schulte, in Vorb.] als *Identifikation*, *Zielebestimmung* und *Problemlösung* bezeichnet und stellen die drei wesentlichen kognitiven Teilfunktionalitäten dar, welche zu wissensbasiertem Verhalten beitragen.

Die Teilfunktionalität *Identifikation* verknüpft Situationsmerkmale, die als Symbole vorliegen, mit Konzepten und stellt diejenigen Konzepte bereit, die in der aktuellen Situation zutreffen und damit relevant sind (siehe Abbildung 3-7).

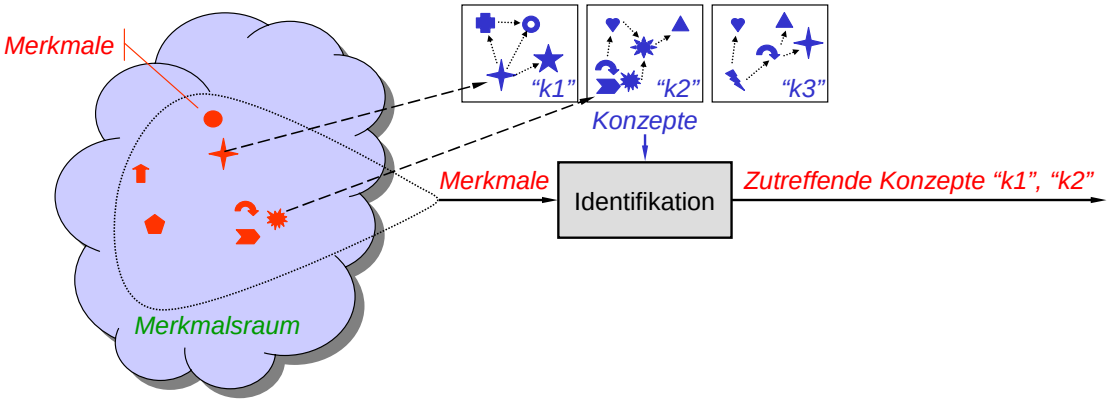


Abbildung 3-7: Teilfunktionalität "Identifikation" auf wissensbasierter Verhaltensebene (vgl. [Onken & Schulte, in Vorb.])

Konzepte stellen dabei deklaratives Wissen, d.h. Tatsachen und Episoden („Wissen, was“) [Solso, 2005], dar und setzen Merkmale zueinander in Beziehung bzw. assoziieren vorhandenes Wissen mit Merkmalen. Als Beispiel hierfür soll eine Situation dienen, in der ein Fahranfänger mit den Merkmalen „Fahren auf der Autobahn auf dem Weg in den Urlaub“, „Rote Lampe blinkt im Instrumentenpanel“ und „Ölkannensymbol leuchtet auf“ konfrontiert ist und auf regelbasierter Verhaltensebene keine Assoziation dieser Situation mit einer konkreten Aufgabe besteht. Mit dem Merkmal „Rote Lampe“ könnte ein Konzept „Gefahr“ assoziiert werden, das „Ölkannensymbol“ mit „Motoröl“, „Motor“ und „Mangel an Motoröl“ mit „Gefahr des Defekts für Motor“. Beim Menschen impliziert jedes dieser hier im Text symbolisch bezeichneten Konzepte wiederum eine meist sehr umfangreiche Bedeutungswelt (Semantik), die sich im Allgemeinen nicht genau spezifizieren lässt.

Die Teilfunktionalität *Zielebestimmung* ermittelt aus der Menge aller potentiell zu verfolgenden Ziele einer Person diejenigen, die in der gegebenen Situation relevant sind und die somit erreicht werden sollen. Basis hierfür ist eine Analyse der Umgebung, die in Form der Konzepte, die als zutreffend klassifiziert wurden, vorliegt. Ziele können nach [Eisenführ & Weber, 2003] in Fundamental- und Instrumentalziele unterschieden werden. Fundamentalziele werden dabei um ihrer selbst Willen verfolgt werden, d.h. benötigen keiner weiteren Begründung (z.B. Wahrung der Flugsicherheit), sind aber jeweils nur in einem bestimmten Kontext fundamental, d.h. können in einem anderen Betrachtungskontext auch instrumental sein. Instrumentalziele begünstigen das Erreichen eines übergeordneten, fundamentaleren Ziels (z.B. Vermeidung der Kollision mit Gelände). Die wichtigsten Punkte beim Umgang mit Zielen in Entscheidungssituationen werden wie folgt zusammengefasst:

> - *Fundamentalziele sind von Instrumentalzielen zu unterscheiden. Für die Bewertung der Alternativen sollten nur solche Ziele herangezogen werden, die innerhalb des jeweiligen Entscheidungskontexts fundamental sind.*
> - *Das System von (Fundamental-)Zielen, die in einer bestimmten Entscheidungssituation von Bedeutung sind, sollte bestimmte Anforderungen erfüllen: Vollständigkeit, Redundanzfreiheit, Messbarkeit, Unabhängigkeit und Einfachheit.*
> - *Eine hierarchische Strukturierung der (Fundamental-)Ziele dient dazu, die [im vorigen Punkt] […] genannten Anforderungen an das Zielsystem zu erfüllen. Für jedes (Fundamental-)Ziel, das zur Bewertung der Alternativen herangezogen werden soll, ist eine Zielvariable (=Attribut) zu bestimmen, die die Zielerreichung möglichst treffend und eindeutig wiedergibt.*
>
> *[Eisenführ & Weber, 2003]*

[Walsdorf, 2002] baut im Rahmen der Entwicklung eines Zielemodells für ein wissensbasiertes Pilotenassistenzsystem auf den Arbeiten von [Eisenführ & Weber, 2003] auf und benennt die folgenden Eigenschaften, mit denen einzelne Ziele, die Teil eines Systems von Zielen sind, beschrieben werden können:

- *Vernetztheit* – Eine Zielsituation ist im Allgemeinen durch mehrere Ziele beschrieben, die untereinander vernetzt sind. Diese Verknüpfungen können entweder positiv oder negativ sein, wobei negative Verknüpfungen Zielkonflikte darstellen, die aufgelöst werden müssen.

- *Relevanz* – Die Relevanz eines Ziels gibt an, ob dieses im gegebenen Situationskontext von Bedeutung ist.
- *Polarität* – Ein Ziel ist entweder positiv, d.h. hier werden Situationen beschrieben, die angestrebt werden und deren Erfülltheitskriterien genau definiert sind, oder negativ, d.h. hier wird angestrebt, dass eine gegebenen Situation nicht erwünscht ist, aber nicht, wie sie verändert werden soll. Positive Ziele können nach [Cohen & Levesque, 1990] weiter in sog. „*achievement goals*“ und „*maintenance goal*s“ unterteilt werden, welche sich dadurch unterscheiden, dass die beschriebene Zielsituation noch nicht erreicht ist bzw. dass sie erreicht ist und beibehalten werden soll.
- *Grad der Erfülltheit* – Diese Eigenschaft gibt ein Maß dafür an, inwieweit die Zielsituation der aktuell vorherrschenden Situation entspricht.
- *Gewichtung* – Die Gewichtung von Zielen stellt ein Hilfsmittel zur Handhabung von Zielkonflikten dar.

Im obigen Beispiel könnten auf Basis der zutreffenden Konzepte mehrere Ziele als relevant klassifiziert werden, so zum Beispiel das Fundamentalziel „Am Zielort ankommen“ und das untergeordnete Instrumentalziel „Schaden am Auto vermeiden“.

Die durch die Teilfunktion Zielebestimmung als relevant identifizierten Ziele stellen Eingangsgrößen für die letzte Teilfunktion der wissensbasierten Verhaltensebene dar – die *Problemlösung*, die oft synonym als *Planung* bezeichnet wird. Hier soll eine Vorgehensweise gefunden werden, die geeignet ist, die Ziele zu erreichen. Grundlage hierfür sind verfügbare Handlungsalternativen wie zum Beispiel „auf dem Seitenstreifen anhalten“, „die nächste Werkstatt anfahren“, „telefonisch Rat einholen“ oder „weiterfahren“. Newell und Simon nennen als menschliche Problemlösungsstrategie in diesem Zusammenhang die Mittel-Ziel-Analyse (*means-ends analysis*) [Newell & Simon, 1972]. Hierbei werden zunächst Unterschiede zwischen dem angestrebten und dem tatsächlichen aktuellen Zustand identifiziert, und es wird versucht, diese Unterschiede nach und nach durch die Anwendung von Operatoren (Handlungsoptionen) zu eliminieren. Gibt es mehrere Operatoren, die als geeignet angesehen werden, den aktuell betrachteten Unterschied zu reduzieren, muss einer davon ausgewählt werden. Anschließend wird überprüft, ob dieser Operator angewendet werden kann, d.h. ob Unterschiede zwischen dem aktuellen Zustand und den Voraussetzungen für die Anwendung des Operators bestehen. Unter Umständen werden hierbei wieder Unterschiede identifiziert, die eliminiert werden müssen, bevor der Operator angewendet werden kann (vgl. [Rich & Knight, 1991][Anderson, 2001]). Somit muss eine Rückwärtssuche ausgehend vom angestrebten Zielzustand durchgeführt werden, um relevante Operatoren finden und nützliche Unterziele bestimmen zu können [Langley & Rogers, 2005]. Sofern es mehrere mögliche Vorgehensweisen gibt, werden deren Auswirkungen im Hinblick auf den zu erreichenden Zielzustand überprüft – entweder durch „Trial and Error“ oder durch die hypothetische Anwendung auf Grundlage des Verständnisses der funktionalen Zusammenhänge in der Umgebung und Voraussage der Effekte des Plans [Rasmussen, 1983]. Ergebnis der Problemlösung ist eine Abfolge von Aufgaben, die von nachgeschalteten Funktionalitäten ausgeführt werden können.

Wissensbasiertes Verhalten erfordert eine große Menge an explizit repräsentiertem Wissen, dessen Verarbeitung lange dauert und viele mentale Ressourcen benötigt [Rasmussen, 1983]. Dennoch ist die Fähigkeit zu wissensbasiertem Verhalten für den Menschen außerordentlich wichtig, da er hiermit in die Lage versetzt wird, sich auch in Situationen, deren Konfiguration und Zeitpunkt des Auftretens nicht bekannt ist und denen insbesondere keine Aufgabe zugeordnet ist, zielgerichtet und damit im Rahmen

seines verfügbaren Wissens bestmöglich zu verhalten. Darüber hinaus kommt die wissensbasierte Verhaltensebene permanent einer Überwachungsfunktion von Routinehandlungen angesichts der übergeordneten Ziele nach. Im Umfeld der Agententechnologie nennt [Maes, 1990] darüber hinaus folgende Gründe, weshalb explizit repräsentierte Ziele wichtig sind:

1. Neue oder geänderte Ziele können einem Agenten zur Laufzeit mitgeteilt werden, ohne den Agenten neu programmieren zu müssen.
2. Verhalten und Handlungsauswahl sind stets von internen Zielen abhängig.
3. Die Auswahl von Handlungen wird einfacher, da mögliche Alternativen leichter eingeschränkt werden können.
4. Da komplexe Agenten komplexe Ziele haben ist es wichtig, dass sie zwischen diesen abwägen und Konflikte und Abhängigkeiten handhaben können, um das Erreichen der Ziele zu optimieren.
5. Ziele sind insbesondere für Lernen und Optimierung des Verhaltens wichtig.

Prinzipiell könnten Menschen jede Situation als „unbekannt“ klassifizieren und wissensbasiert zu einer Verhaltensentscheidung kommen. Sehr viel effizienteres Verhalten im Sinne von Reaktionszeiten und Verarbeitungsressourcen findet jedoch auf regel- und noch deutlich ausgeprägter auf fertigkeitsbasierter Ebene statt. Um einmal erarbeitete Lösungen auf wissensbasierter Ebene nicht immer wieder ableiten zu müssen und seine Effizienz im Laufe der Zeit kontinuierlich weiter zu steigern, kann der Mensch solche Lösungen lernen und beim erneuten Auftreten einer dann bekannten Situation regel- bzw. sogar fertigkeitsbasiert agieren (vgl. [Johannsen, 1993][Anderson, 2001], „Erlernen von Fertigkeiten“).

3.2.2.2 Merkmale der regelbasierten Verhaltensebene

Regelbasiertes Verhalten kommt in bekannten Situationen zum Einsatz und kann zum Beispiel bei der stereotypen Durchführung von Tätigkeiten in vielen Aufgabensituationen beobachtet werden [Johannsen, 1993]. Wesentliches Merkmal dieser Verhaltensebene ist die direkte Zuordnung von Merkmalen zu aktuellen Aufgaben auf Basis von bekannten Aufgabensituationen. [Rasmussen, 1983] nennt hierfür die Einzelschritte „Erkennen“ und „Assoziation Zustand/Aufgabe“, die von [Onken & Schulte, in Vorb.] zur Teilfunktionalität *Aufgabenerkennung* (vgl. Abbildung 3-8) zusammengefasst werden.

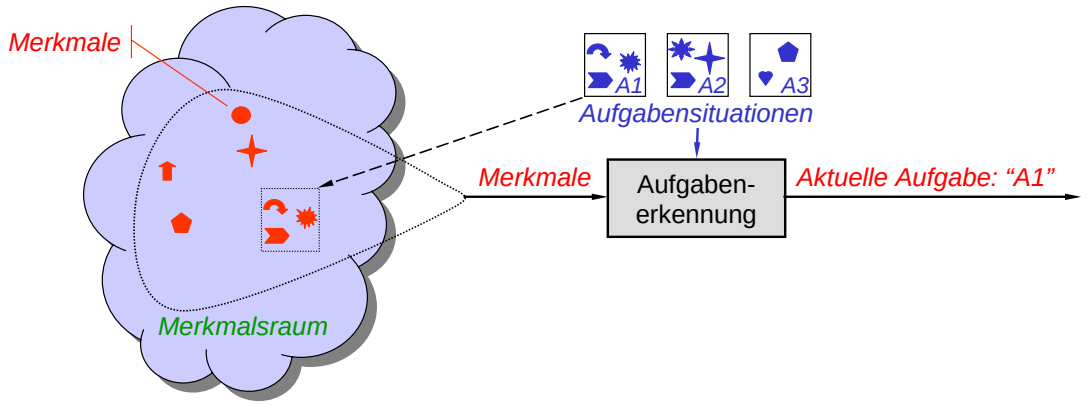


Abbildung 3-8: Teilfunktionalität "Aufgabenerkennung" auf regelbasierter Verhaltensebene (vgl. [Onken & Schulte, in Vorb.])

Als Beispiel für regelbasiertes Verhalten sei die Anwendung von Straßenverkehrsregeln angeführt. So wird zum Beispiel die Aufgabe „Anhalten“ in einer Situation erkannt, die durch die Merkmale „Eigenes Auto bewegt sich auf Ampel zu“ und „Ampel schaltet auf rot“ gekennzeichnet ist.

Im nächsten Schritt wird mit Hilfe von Verhaltensregeln die aktuell zu bearbeitende Aufgabe in Anweisungen zur Ansteuerung und Auswahl sensomotorischer Muster umgesetzt. Verhaltensregeln stellen dabei im Allgemeinen rezeptartige Abläufe dar, die Schritt für Schritt wiedergeben, was getan werden muss, um eine Aufgabe zu bearbeiten.

Da einer (bekannten) Situation direkt Aufgaben zugeordnet werden, sind das Verhalten und insbesondere die Verhaltensentscheidung auf regelbasierter Ebene deutlich effizienter als auf wissensbasierter Ebene. Ziele werden hier nur implizit über die Zuordnung von Aufgaben zu Situationen, nicht aber explizit repräsentiert und berücksichtigt. Allerdings kann regelbasiertes Verhalten nur dann zum Einsatz kommen, wenn die entsprechende Aufgabensituation bekannt ist. Hierbei besteht zusätzlich die Gefahr, dass relevante Merkmale, die nicht Teil des Musters zur Erkennung einer Aufgabe sind, unberücksichtigt bleiben und damit zu Fehlern führen (vgl. [Reason, 1991]).

3.2.2.3 Merkmale der fertigkeitsbasierten Verhaltensebene

Fertigkeitsbasiertes Verhalten ist gekennzeichnet durch die unbewusste Ausführung automatisierter und hoch integrierter sensomotorischer Muster, wofür in der Regel keine Aufmerksamkeitsressourcen bereitgestellt werden müssen [Rasmussen, 1983]. Beispiele hierfür sind das Greifen einer Kaffeetasse, das Halten der Spur beim Autofahren oder die Stabilisierung eines Hubschraubers durch einen erfahrenen Piloten.

Wesentliche Funktionalität auf dieser Ebene ist die Ansteuerung der zur Verfügung stehenden Motorik (Ausführung von Handlungen) auf Basis von Sinnesempfindungen (Sensorik) im Sinne einer kontinuierlichen Regelung, die an das aktuell relevante sensomotorische Muster angepasst wird. Effizientes Handeln auf fertigkeitsbasierter Ebene kommt allerdings erst durch das Zusammenspiel von Regelung mit einer Rückführung von Ist-Zuständen und einer Vorsteuerung, mit der schnelle, koordinierte Bewegungen durchgeführt werden können, zustande. So stellt die Bewegung einer Hand in Richtung einer Kaffeetasse eine Vorsteuerung dar, während das letztendliche, feinmotorische Greifen des Henkels optimiert wird, indem Zustandsdaten zurückgeführt werden (Tasten des Henkels und Vergleich mit der Position der Finger).

Sensomotorische Muster können von übergeordneten Verhaltensebenen durch Anweisungen ausgelöst und bewusst angepasst werden, wobei Anpassung hierbei als Auswahl eines ähnlichen sensomotorischen Musters verstanden werden kann. Ein Beispiel für die Anpassung eines sensomotorischen Musters „Laufen auf ebener Straße“ ist die Verarbeitung einer Anweisung „Achtung, die Straße ist rutschig“. [Rasmussen, 1983]

Generell kann das beobachtbare menschliche Verhalten als Aneinanderreihung der Ausführung sensomotorischer Muster angesehen werden [Rasmussen, 1983], wobei in den meisten Fällen die fertigkeitsbasierte Verhaltensebene nicht verlassen wird [Johannsen, 1993]. Da sie sich durch eine hohe Verarbeitungsgeschwindigkeit und Parallelität der Verarbeitung sowie niedrigen Ressourcenbedarf, aber auch niedrige Verbalisierungsmöglichkeit der Vorgänge auszeichnet, wird damit effizientes menschliches Verhalten möglich.

3.2.3 Modellierung wissensbasierten Verhaltens

Theorien, die wissensbasiertes Verhalten abbilden wollen, müssen in der Lage sein, Ziele explizit zu repräsentieren, da das wesentliche Kennzeichen wissensbasierten Verhaltens die explizite Repräsentation von Zielen ist. Dieser Abschnitt stellt *BDI-Agenten* (*Belief-Desire-Intention*) und den *Kognitiven Prozess* als zwei Theorien vor, welche diese Forderung erfüllen. Während sich das BDI-Modell auf die Definition der mentalen Konzepte *Belief* (Überzeugung), *Desire* (Wunsch) und *Intention* (Absicht) beschränkt, detailliert der Kognitive Prozess Verarbeitungschritte, a-priori vorhandene mentale Konzepte und zur Laufzeit im System vorhandenes Wissen im Sinne eines wissensbasierten Implementierungsansatzes.

An dieser Stelle sei darauf hingewiesen, dass die Begriffe „regelbasiert“ und „wissensbasiert“ verschiedene, voneinander unabhängige Bedeutungen haben. So sind Modelle bzw. Architekturen, deren Wissensrepräsentation regelbasiert ist, d.h. deren Wissen in Form von Regeln (Produktionen) hinterlegt ist, nicht auf regelbasiertes Verhalten im Sinne von Rasmussen beschränkt. Ferner bezeichnet die Informatik Systeme als „wissensbasiert“, die Wissen und Verarbeitung dieses Wissens trennen. Dieses architekturelle Merkmal sagt allerdings nichts darüber aus, ob ein solches System in der Lage ist, wissensbasiertes Verhalten zu zeigen. Wie später erläutert werden wird, ist der mit dieser Arbeit verfolgte Ansatz ein wissensbasiert implementiertes System, welches vor allem Regeln als Form der Wissensrepräsentation nutzt und welches verhaltensseitig wissensbasiertes und in geringerem Umfang auch regelbasiertes Verhalten zeigt.

3.2.3.1 BDI-Agenten

Im Bereich der Agententechnologie stellt die BDI-Agententheorie (*Belief-Desire-Intention*) [Rao & Georgeff, 1995] die bekannteste Theorie dar, auf deren Basis rationales Verhalten von Agenten erzeugt werden kann. Hier gibt es drei zentrale mentale Zustände: *beliefs* (Überzeugungen), die den Zustand der Umgebung repräsentieren, *desires* (Wünsche oder Motivationen), die die Menge von Handlungszielen darstellen, die für das System aktuell relevant sind, und *intentions* (Absichten), die angeben, welches Ziel gegenwärtig verfolgt wird. Letztere sind oft in Form einer Planbibliothek abgelegt, aus der derjenige Plan ausgewählt wird, welcher am besten geeignet scheint, das aktuelle Ziel zu erreichen und dessen Ausführung dann die aktuelle Absicht darstellt. Eine Absicht stellt also das Ergebnis des letzten Entscheidungsprozesses dar, und ermöglicht das Finden einer Balance zwischen Reaktivität auf externe Ereignisse und Ausrichtung des Verhaltens an Zielen in dem Sinn, dass auf Änderungen der Situation angemessen, d.h. nicht immer sofort aber rechtzeitig, reagiert wird.

Für die Implementierung von Agenten entsprechend der BDI-Theorie steht eine Vielzahl von Architekturen zur Verfügung, die von den früh entwickelten Systemen PRS (*Procedural Reasoning System*) und dMARS (*distributed Multi-Agent Reasoning System*) [d’Inverno et al., 1998] bis hin zu kommerziellen Entwicklungsumgebungen wie JACK^{TM} der Firma Agent Oriented Software Group reichen. JACK erweitert Java zur JACK Agent Language, indem agentenorientierte Konstrukte wie *agent*, *capability*, *beliefset* oder *plan* eingeführt werden. Nach dem Kompilieren des hiermit erzeugten Codes mit dem JACK Agent Compiler kann ein JACK Agent dann unter Verwendung des JACK Agent Kernels ausgeführt werden. In einer Erweiterung wurden darüber hinaus teamorientierte Konzepte in JACK eingeführt. [Weiß & Jakob, 2005]

Anwendung: Surrogate UAV

Auf Basis der BDI-Agententheorie wurde eine große Anzahl von Anwendungen entwickelt, wozu auch die Nachbildung eines Teams aus bemannten Kampfflugzeugen [Tidhar et al., 1998] und die Führung mehrerer UAVs durch einen Operateur zählen. Letztere Aktivität wurde im Auftrag des britischen Verteidigungsministeriums von der Firma QinetiQ durchgeführt, wobei die Führung der UAVs durch den Piloten eines einsitzigen Kampfflugzeugs erfolgte und JACK als BDI-Architektur eingesetzt wurde [Walker, 2006][Howitt & Richards, 2006][QinetiQ, 2006][QinetiQ, 2007].

Als Aufgabenstellung wurde eine zeitkritische Angriffsmission gegen ein hochwertiges, mobiles Ziel betrachtet, bei der das Kampfflugzeug außerhalb des Bedrohungsgebietes agiert und die ihm zur Verfügung stehenden UAVs führt, welche mit Sensoren und Waffen ausgerüstet sind. Wesentliche Missionselemente hierbei sind „Suche“ und „Angriff“ [Baxter & Horn, 2003][Baxter & Horn, 2005].

Im Zentrum des in [Baxter & Horn, 2003] und [Baxter & Horn, 2005] vorgestellten Gesamtsystems stehen die auf Basis von JACK implementierten Agenten, welche ein sich selbst organisierendes System für die Führung der UAVs darstellen. Hierbei gibt es vier verschiedene Arten von Agenten (vgl. Abbildung 3-9).

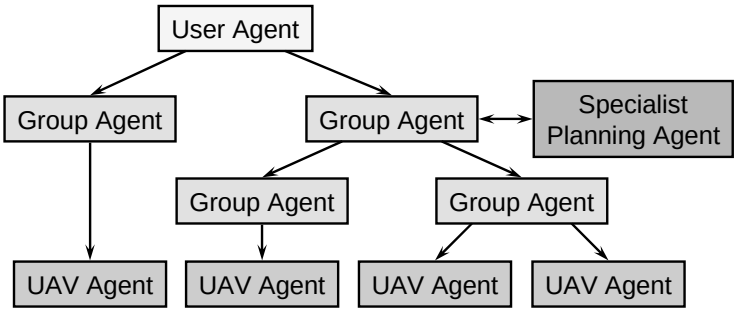


Abbildung 3-9: Agentenhierarchie [Baxter & Horn, 2005]

Der *User Agent*, der im System prinzipiell nur einmal existiert, verwaltet Anfragen vom und Informationen für den Operateur. Für diesen menschlichen Operateur überwacht er die Durchführung mehrerer paralleler Aufgaben und nutzt dafür *Group Agents*, welche er erzeugt und zerstört, und denen er UAVs zuweist. *Group Agents* sind also verantwortlich für die Planung, Koordination und Überwachung der Ausführung der ihnen zugewiesenen Aufgaben mit Hilfe der ihnen zugewiesenen UAVs. Dazu nutzen diese andere *Group Agents* oder *UAV Agents*. Typische Aufgaben auf Gruppenebene sind die Suche eines Ziels, der Angriff eines Ziels (Waffeneinsatz und Schadensfeststellung), Suche und Angriff in Kombination, das Fliegen einer Route in Formation und die Suche aus der Entfernung. *Specialised Planning Agents* kapseln Planungsfunktionalitäten für Suche und Angriff. *UAV Agents* schließlich interagieren direkt mit den UAV-Plattformen. Sie senden Kommandos an die Sensoren, Waffensysteme und Autopiloten, überwachen den Zustand der Plattform und schicken Sensor- und Zustandsinformationen an andere Agenten. Sie können Aufgaben für einzelne UAVs wie das Erfassen von Bildern eines spezifizierten Fahrzeugs am Boden planen und ausführen. Typische Verhaltensmuster sind der Flug einer spezifizierten Route, die Aufnahme von Bildern eines Fahrzeugs am Boden, Kreisen, um ein Fahrzeug am Boden zu überwachen, die Suche in einem bestimmten Gebiet und der Einsatz von Waffen. Während die *UAV Agents* direkt mit Plattformen verknüpft sind, ist aus der öffentlich verfügbaren Literatur unklar, welchen physikalischen Entitäten die anderen Agenten zuzuordnen sind.

Für die Koordination einer Teamaufgabe auf Gruppenebene nutzen *Group Agents* so genannte Koordinationsmatrizen (*Coordination Matrices*) für die Ausführung von Gruppenaufgaben, in denen hinterlegt ist, wie viele UAVs für die Aufgabe benötigt werden, welche Fähigkeiten diese UAVs mitbringen müssen, welche Aufgaben es auf Gruppenebene gibt und welche Aufgaben die einzelnen Gruppenmitglieder haben. Des Weiteren ist hinterlegt, wie die Aufgaben auf Gruppenebene mit denen auf Mitgliederebene verknüpft sind. Die Koordinationsmatrizen treten in drei Ausprägungen auf: der Planungsmatrix, der Überwachungsmatrix (*Monitor Matrix*), die der *Group Agent* verwendet, um den Fortschritt der Aufgabe zu überwachen, und der Auftragsmatrix (*Instructed Matrix*), die die Teilaufgaben für jedes Gruppenmitglied enthält und an die einzelnen *UAV Agents* geschickt wird.

Abbildung 3-10 zeigt ein Beispiel für die Koordinationsmatrix einer „Troop Bound"-Aufgabe. Dabei gibt es die vier Gruppenzustände „Preparing", „LeadMove", „Rear Move" und „Bounded" sowie die zwei Unteraufgaben „Lead" und „Rear". Der Zustand „Preparing" ist hierbei beispielsweise abgearbeitet, wenn die „Rear"-Untergruppe ihre „Establish Overwatch"-Aufgabe fertiggestellt hat.

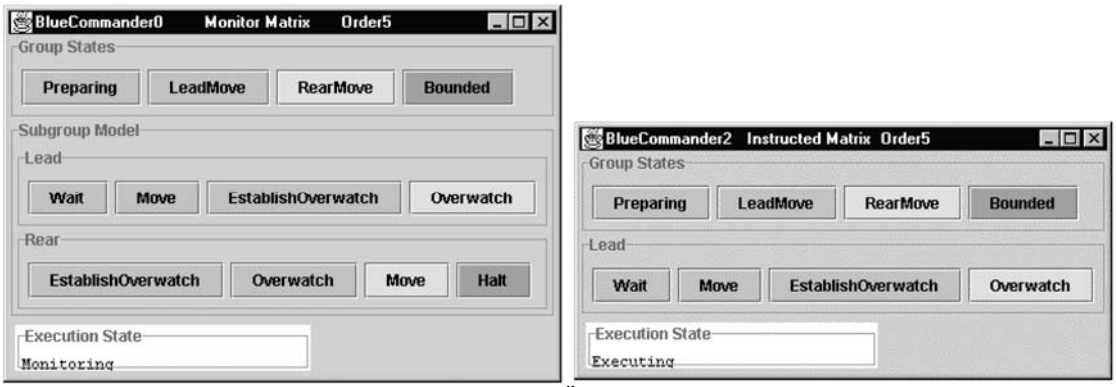


Abbildung 3-10: Koordinationsmatrizen: Überwachungsmatrix und Auftragsmatrix für eine „Troop Bound"-Aufgabe [Baxter & Horn, 2003]

Die Nutzung von Koordinationsmatrizen deutet hierbei darauf hin, dass die Zusammenarbeit im Team wie auch bei den bisher beschriebenen Anwendungen auf regelbasierter Verhaltensebene mit den bekannten Vor- und Nachteilen stattfindet. Die Nutzung von *group agents* stellt einen zentralisierten Ansatz zur Koordination dar, bei dem ein Ausfall dieser *group agents* den Erfolg der Mission gefährden und abhängig von der Zuordnung zu physikalischen Entitäten möglicherweise ein unnötig hohes Kommunikationsaufkommen bedingen kann (vgl. auch Abschnitt 3.3.3). Im Hinblick auf eine mögliche Integration menschlicher Teammitglieder in das UAV-Team ist die Nutzung von *group agents* problematisch, da diese an zentraler Stelle die Zusammenarbeit im Team koordinieren (vgl. [Laird et al., 1998]) und es fraglich ist, ob ein Mensch einem technischen Agenten unterstellt werden soll.

3.2.3.2 Der Kognitive Prozess

Der *Kognitive Prozess (KP)* [Putzer & Onken, 2003][Putzer, 2004][Frey, 2005] [Meitinger & Schulte, 2006a] stellt ein Modell menschlicher Informationsverarbeitung dar, welches als Grundlage für die Umsetzung rationalen Verhaltens im Rechner genutzt werden kann. Es basiert auf dem von [Rasmussen, 1983] vorgeschlagenen Modell menschlichen Verhaltens, wobei insbesondere die Berücksichtigung der wissensbasierten Verhaltensebene mit einer expliziten Repräsentation von Zielen im Vordergrund steht. Darüber hinaus werden weitere Forderungen an eine solche Theorie mit einbezo-

gen, wie zum Beispiel eine zentrale, objektorientierte Situationsrepräsentation (vgl. [Onken & Schulte, in Vorb.]).

Der KP beschreibt, wie rationales, zielgerichtetes Verhalten im Kontext einer sich dynamisch verändernden Umgebung erzeugt werden kann. Dabei enthält die Umwelt sowohl die physische Komponente des Kognitiven Prozesses (z.B. UAV) als auch Umgebungselemente wie Hindernisse. Wesentliche Merkmale des KPs sind die Ausrichtung des Verhaltens an Zielen, der wissensbasierte Ansatz im Sinne der Informatik, d.h. Trennung von applikationsspezifischem Wissen und applikationsunabhängiger Verarbeitung des Wissens, und die Repräsentation und Verarbeitung des Wissens auf symbolischer Ebene, was Vorgänge bei der Entscheidungsfindung für Menschen einsehbar und verständlich macht.

Hiermit können alle Fähigkeiten abgebildet werden, über die künstliche kognitive Einheiten (*Artificial Cognitive Units*, *ACUs*) verfügen müssen, wenn sie gemäß den Prinzipien kognitiver Automation (vgl. Abschnitt 2.1.2.3) entwickelt werden. Im Einzelnen sind dies:

- ***Verständnis der Situation.*** ACUs sollen in der Lage sein, die vorherrschende Situation zu verstehen (vgl. auch [Endsley, 1995]), d.h. insbesondere ein Verständnis des Arbeitsziels des Arbeitssystems (*Operating ACU*) bzw. des semi-autonomen Systems (*Supporting ACU*) zu entwickeln. Teilaspekte davon sind, was erreicht werden soll, welche Teilaufgaben dazu bearbeitet werden müssen und in welchem Stadium sich die Bearbeitung befindet.
- ***Zielgerichtetes Handeln.*** Im Rahmen des Arbeitsprozesses sollen Handlungsziele berücksichtigt und verfolgt werden, welche die Erfüllung des Arbeitsziels, die Zusammenarbeit im Team und übergeordnete Zielsetzungen wie zum Beispiel Sicherheit betreffen.
- ***Rationales Entscheiden.*** Entscheidungen sollen im Hinblick auf die zu erreichenden Zielsetzungen unter Berücksichtigung des gesamten vorhandenen Wissens gefällt werden (vgl. [Newell, 1990]).
- ***Problemlösungsfähigkeit.*** ACUs sollen in der Lage sein, auf Basis ihres Verständnisses der aktuellen Situation und Wissen um mögliche Handlungsalternativen (z.B. Einsatz von Arbeitsmitteln) Vorgehensweisen zu entwickeln, die sie ihren Zielen näher bringen.

Abbildung 3-11 zeigt den Kognitiven Prozess, welcher aus dem *Rumpf* (*Wissen*; ovaler Bereich) und den *Transformatoren* (wissensverarbeitende Funktionalitäten, Pfeile um den Rumpf) besteht. Dabei lässt sich Wissen in *a-priori Wissen* und *Situationswissen* unterteilen. Das *a-priori Wissen* wird vom Entwickler eines kognitiven Systems modelliert und besteht aus Umweltmodellen, Wünschen, Handlungsalternativen und Anweisungsmodellen. Umweltmodelle stellen Objekte in der Umwelt (z.B. Flugzeug) ebenso wie Relationen zwischen solchen Objekten (z.B. Gefahr) und abstrakte Gebilde (z.B. Flugplan) dar. Wünsche geben all diejenigen Ziele eines Systems an, welche potentiell verfolgt werden können (z.B. Gefahr vermeiden, Mission erfüllen). Handlungsalternativen stellen Optionen dar, die zu einem Plan kombiniert werden können, mit dem eine Zielsituation erreicht werden kann (z.B. Ausweichen, Nachricht senden) und Anweisungsmodelle hinterlegen Wissen, wie Handlungsalternativen ausgeführt werden (z.B. Bedienoperationen). Schließt man Lernprozesse zunächst aus, so ist das a-priori Wissen statisch, d.h. während der Laufzeit des Systems wird kein zusätzliches, „neues“ a-priori Wissen hinzugefügt.

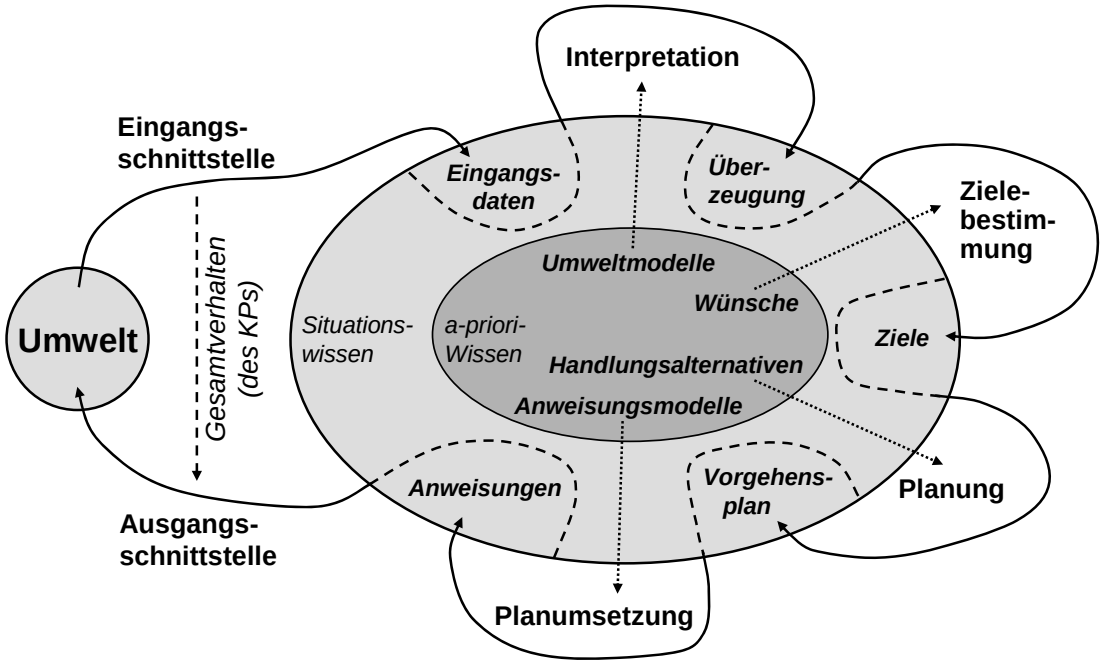


Abbildung 3-11: Der Kognitive Prozess

Das *Situationswissen* hingegen wird während der Laufzeit des Systems durch die Verarbeitung des a-priori Wissens durch die Transformatoren erzeugt. Es besteht aus Instanzen der im a-priori Wissen hinterlegten Modelle und wird in Eingangsdaten, Überzeugung, Ziele, Vorgehensplan und Anweisungen unterteilt. Eingangsdaten stellen dabei alle Informationen dar, welche durch Sensoren o.ä. aus der Umwelt zur Verfügung stehen. Üblicherweise erstreckt sich die Umwelt dabei nicht nur auf externe Objekte wie zum Beispiel Hindernisse sondern auch auf die physikalische Entität, der der KP zugeordnet ist wie zum Beispiel ein Flugzeug. Die Überzeugung enthält alle aus den Umweltmodellen gebildeten Instanzen und stellt das Verständnis des künstlichen kognitiven Systems bezüglich der aktuell vorherrschenden Situation dar. Die Ziele dagegen beschreiben im Gegensatz zur Ist-Situation eine hypothetische Soll-Situation, wie sie nach Meinung des Systems vorherrschen sollte. Das Erreichen der Soll- oder Zielsituation stellt dabei den Maßstab für die Auswahl zukünftiger Aktivitäten dar. Der Vorgehensplan setzt sich aus einzelnen Schritten zusammen und gibt an, wie das System versuchen wird, die aktuelle Situation in die Zielsituation zu überführen. Anweisungen schließlich sind diejenigen Aktionen, die das System tatsächlich ausführen soll, um den Vorgehensplan abzuarbeiten.

Transformatoren lesen Situationswissen, verwenden ihnen zugeordnete Modelle (gepunktete Pfeile in Abbildung 3-11) um dieses Wissen zu verarbeiten und erzeugen schließlich „neues“ Situationswissen. Sie arbeiten also entsprechend dem Schema „Eingabe-Verarbeitung-Ausgabe“. Dabei haben alle Transformatoren prinzipiell lesenden Zugriff auf das gesamte Situationswissen, lesen aber hauptsächlich in einem Bereich (gestrichelt umfasste Bereiche in Abbildung 3-11). Das erzeugte Wissen darf allerdings nur in einem dedizierten Bereich abgelegt werden. Ausnahmen stellen die Transformatoren „Eingangsschnittstelle“ und „Ausgangsschnittstelle“ dar, welche die Verbindung zur Umwelt herstellen. Im Einzelnen decken die Transformatoren folgende Funktionalitäten ab:

- Die *Eingangsschnittstelle* sammelt alle in der Umwelt zur Verfügung stehenden Informationen (z.B. von Sensoren, Nachrichtenempfängern) und stellt diese dem künstlichen kognitiven System in den Eingangsdaten zur Verfügung.

- Die *Interpretation* nutzt diese Eingangsdaten, um eine Überzeugung bezüglich der aktuell vorherrschenden Situation zu erzeugen. Diese Überzeugung besteht dabei aus Instanzen der in den Umweltmodellen hinterlegten Modelle.
- Die *Zielebestimmung* wertet die Überzeugung aus und bestimmt daraufhin, welche der Wünsche aktuell verfolgt, d.h. als Ziele instanziiert werden sollen.
- Die *Planung* bestimmt aus den Handlungsalternativen diejenigen, welche geeignet sind, die Zielsituation – beschrieben durch die (aktiven) Ziele – ausgehend von der durch die Überzeugung repräsentierten aktuellen Situation zu erreichen, so dass sich ein Vorgehensplan ergibt.
- Die *Planumsetzung* setzt die einzelnen Schritte des Vorgehensplans in Anweisungen um, wobei entsprechende Anweisungsmodelle zugrunde liegen.
- Die *Ausgangsschnittstelle* schließlich sendet die Anweisungen an die entsprechenden Effektoren in der Umwelt. Daneben ist es auch möglich, dass Anweisungen internes Verhalten steuern; sie werden dann von den entsprechenden Transformatoren berücksichtigt.

Das von außen wahrnehmbare Verhalten des KPs ergibt sich durch die Betrachtung der Eingangs- und Ausgangsschnittstelle und wird als Makroverhalten bezeichnet. Es ergibt sich aus dem Zusammenspiel des Verhaltens der einzelnen Modelle des a-priori Wissens (Mikroverhalten). Auch wenn zur Vereinfachung zunächst angenommen werden darf, dass die Transformatoren sequenziell arbeiten, so erfolgt die Verarbeitung tatsächlich parallel und ereignisgesteuert. In jeder Situation kommt also sämtliches Wissen zum Einsatz, das im Kontext dieser Situation relevant ist.

Im Hinblick auf die Entwicklung komplexer Systeme kann der Kognitive Prozess als Paradigma für die Entwicklung verschiedener Systemfähigkeiten verwendet werden (*Pakete*, horizontale Sichtweise in Abbildung 3-12), welche alle entsprechend des gleichen Schemas aufgebaut sind (*Transformatoren*, vertikale Sichtweise in Abbildung 3-12). Die einzelnen Pakete sind dabei durch entsprechende Verweise im a-priori Wissen und durch das gemeinsame Situationswissen zur Laufzeit mehr oder weniger stark miteinander verknüpft.

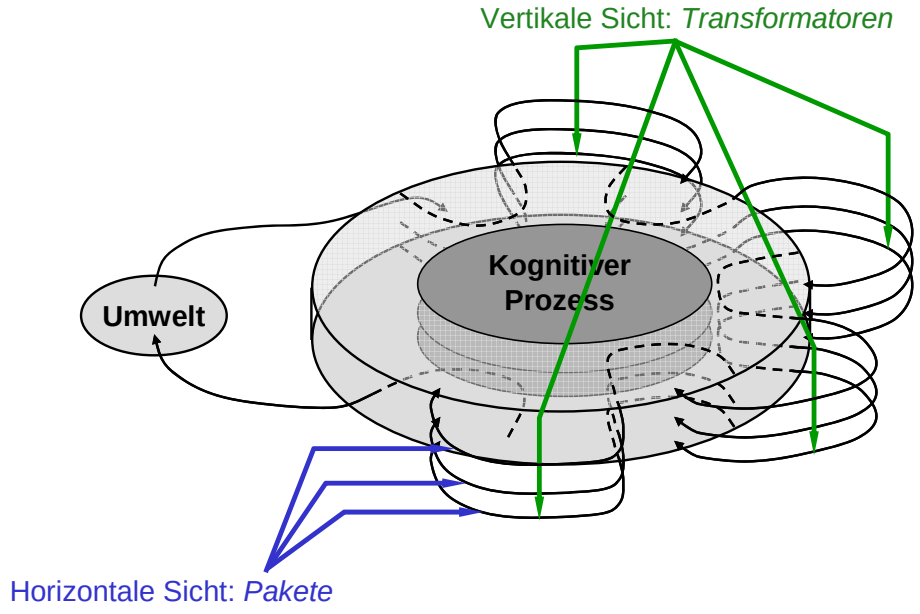


Abbildung 3-12: Sichtweisen auf den Kognitiven Prozess

Durch den wissensbasierten Ansatz ist es möglich, eine Architektur zu implementieren, die die applikationsunabhängigen Anteile des Kognitiven Prozesses, d.h. insbesondere die Transformatoren, abbildet, so dass die Entwicklung eines kognitiven Systems auf Basis des KPs auf die Modellierung des a-priori Wissens beschränkt wird. COSA (Cognitive System Architecture) stellt eine solche Architektur dar, welche im Rahmen dieser Arbeit eingesetzt wurde und in Abschnitt 5.1 näher erläutert wird. Sie basiert auf Soar als wissensverarbeitendem Prozessor, welcher um die Theorie des Kognitiven Prozesses erweitert wird, und stellt insbesondere die Programmiersprache CPL (Cognitive Programming Language) zur Verfügung, die objektorientierte Ansätze und die Möglichkeit zur Programmierung in mentalen Begriffen auf Basis der Soar-Syntax bereitstellt (siehe Abschnitt 5.1.2).

Während die Wissensrepräsentation im Soar-Arbeitsspeicher letztlich in Form eines semantischen Netzes bzw. als Produktionen im Soar-Langzeitspeicher erfolgt, werden Modelle des a-priori Wissens des KPs als frame-artige [Reimer, 1991] Klassen in CPL repräsentiert. Diese Frames enthalten dabei nicht nur statische Attribute, sondern sind um eine dynamische Komponente – das Verhalten eines Modells – erweitert, welches in Form von Produktionen repräsentiert wird. Wegen der Strukturierung des Soar-Arbeitsspeichers durch COSA kann die Wissensrepräsentation dort als semantisches Netz, dessen Knoten im Wesentlichen Frames sind, aufgefasst werden.

Anwendung: COSY^{flight}

Die Anwendung von COSA erstreckt sich bisher auf Applikationen im Bereich der Pilotenassistenz [Groth et al., 2006] und der UAV-Flugführung, wobei COSY^{flight} als Vertreter der letztgenannten Kategorie hier erläutert wird. Bei COSY^{flight} (Cognitive System for the flight domain) [Putzer, 2004][Frey, 2005] handelt es sich um ein Führungssystem für ein semi-autonomes UAV, das auf Basis des Kognitiven Prozesses und COSA implementiert wurde. Die Mission des UAVs umfasst die Aufklärung eines feindlichen Gebietes, wobei Wegpunkte spezifiziert werden können, die entweder überflogen oder an denen zusätzlich Aufklärungsfotos erstellt werden sollen. Das Verhalten des Systems basiert dabei auf der Implementierung der in Abbildung 3-13 dargestellten Wünsche.

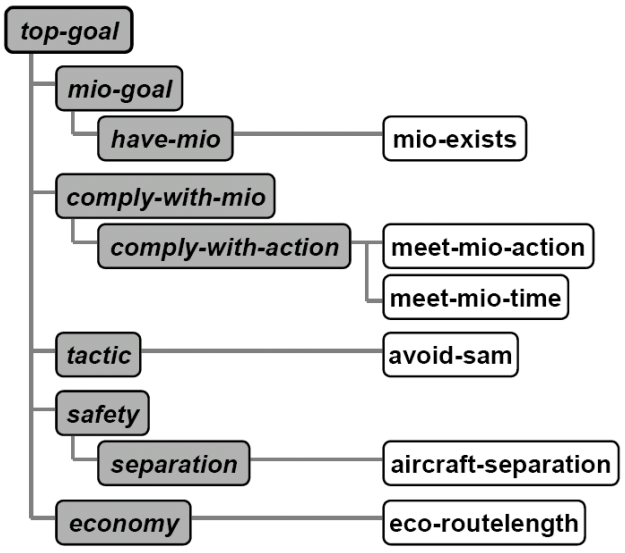


Abbildung 3-13: Hierarchische Struktur der Wünsche von COSY^{flight} [Putzer, 2004]

Das Ziel „mio-goal“ sorgt dafür, dass das System nach dem Start auf einen Auftrag (*mission order – mio*) wartet, „comply-with-mio“ stellt sicher, dass alle Elemente des Auftrags berücksichtigt werden, „tactic“ berücksichtigt taktische Elemente der Umge-

bung wie zum Beispiel SAM-Stellungen, die je nach Gewichtung aktiver Ziele vermieden („avoid-sam“) oder ignoriert werden, „safety“ sorgt für die Einhaltung von Sicherheitsgrenzen und ist hier durch die Separation von anderen Flugzeugen repräsentiert und „economy“ bewirkt schließlich die Auswahl der kürzesten Flugroute.

Während die Funktionalität von COSY^{flight} überschaubar ist, so ist es doch das erste System, welches auf Basis von COSA entwickelt wurde und damit einen prinzipiellen Funktionsnachweis erbrachte, dass der Kognitive Prozess in Verbindung mit COSA eine tragfähige Basis für die Entwicklung künstlicher kognitiver Systeme darstellt.

3.3 Kooperation

Nachdem im vorhergehenden Abschnitt die Fragestellung adressiert wurde, wie künstliche Kognition bzw. wissensbasiertes Verhalten technischer Systeme rechnergestützt realisiert werden kann, beschäftigt sich dieser Abschnitt nun damit, was Kooperation ausmacht und wie künstliche kognitive Einheiten auf Basis von explizit repräsentierten Zielen miteinander kooperieren können.

3.3.1 Gründe für Kooperation

Wird eine Aufgabenstellung durch mehrere Menschen und/oder ACUs zusammen bearbeitet, so bedeutet dies im Allgemeinen zunächst einen erhöhten Aufwand gegenüber der Bearbeitung durch einen Einzelnen [Piepenburg, 1991], da die Beteiligten zum Beispiel ihre Aktivitäten miteinander absprechen und sich gegenseitig relevante Informationen zur Verfügung stellen müssen (vgl. [Biggers & Ioerger, 2001]). Dennoch gibt es eine Reihe von Gründen, die für die Zusammenarbeit mehrerer Akteure sprechen, sofern dies grundsätzlich möglich ist (vgl. [Jennings, 1996][Wooldridge, 2002][Schulte et al., 2008]):

- Kein Einzelner hat die Fähigkeiten, Ressourcen oder Informationen, die nötig sind, um die gesamte Aufgabenstellung lösen zu können. So wird zum Beispiel im Rahmen der Durchführung einer Aufklärungsmission durch mehrere UAVs sowohl die Fähigkeit, Videobilder aufzunehmen als auch die Fähigkeit, über eine Relaisstation eine Funkverbindung zur Kontrollstation herzustellen, benötigt.
- Es gibt Abhängigkeiten zwischen den Aktivitäten Einzelner im Hinblick auf die Verwendung gemeinsam genutzter Ressourcen oder die Ergebnisse vorhergehender Aktivitäten. Zum Beispiel kann ein Transporthubschrauber erst dann an einem Landeplatz in feindlichem Gebiet landen, wenn dieser zuvor von anderen Kräften aufgeklärt wurde.
- Die Effektivität des Arbeitsprozesses oder die Qualität des Arbeitsergebnisses kann durch Zusammenarbeit gesteigert werden, indem zum Beispiel unnötige Redundanz bei der Bearbeitung von Teilaufgaben vermieden wird oder Teammitglieder über relevante Situationsänderungen informiert werden. Beispielsweise ist es unter Umständen ausreichend, wenn nur eines statt mehrerer UAVs mit gleichen Sensoren ein bestimmtes Gebiet aufklärt und die Aufklärungsergebnisse verschiedenen Interessenten zur Verfügung stellt.

Ergibt sich aus einem oder mehreren dieser Gründe eine Zusammenarbeit mehrerer Menschen und/oder ACUs im *Team*, so ergibt sich aus der *Kooperation* die Notwendigkeit nach *Koordination* der einzelnen Teammitglieder hinsichtlich ihrer Aktivitäten, wofür *Kommunikation* eingesetzt wird. Diese Begriffe werden in den folgenden Abschnitten detailliert und der Zusammenhang zwischen ihnen hergestellt.

3.3.2 Kooperation und Teamarbeit

Arbeiten mehrere Menschen und/oder ACUs zusammen, um ein gemeinsames Ziel zu erreichen, so kann diese Form der Zusammenarbeit als *Kooperation* bezeichnet werden [Doran et al., 1997][Ferber, 1999][Borghoff & Schlichter, 2000]. Diese zeichnet sich insbesondere dadurch aus, dass sich kooperierende Agenten gegenseitig unterstützen und füreinander verantwortlich sind ([Bratman, 1992] zitiert nach [Jennings, 1996]). Dies führt dazu, dass einzelne Agenten unter Umständen Aktionen nur ausführen und damit Kosten für sich selbst verursachen, um einen anderen Agenten zu unterstützen und damit dem gemeinsamen Ziel näher zu kommen [Jennings, 1994]. Dieser Sachverhalt unterscheidet kooperative Zusammenarbeit von koordinierter Zusammenarbeit einzelner Agenten, die zwar die aufzubringenden Aufwände Einzelner durch Koordination minimieren, aber in keinem Fall zusätzliche eigene Aufwände in Kauf nehmen, um einen anderen Agenten zu unterstützen.

Die Kooperation im Hinblick auf ein gemeinsames Ziel ist wesentliches Kennzeichen der Zusammenarbeit in einem *Team*, wie es auch im Rahmen dieser Arbeit aus mehreren *Supporting ACUs* aufgebaut wird. Charakteristische Merkmale eines Teams wurden aus einer Definition von [Salas et al., 1992] abgeleitet und lauten wie folgt:

> - *es handelt sich um **zwei oder mehr Personen** [und/oder Agenten];*
> - *eine Zusammenkunft erfolgt zwecks Erreichung eines **gemeinsamen Ziels**;*
> - *die Zusammenkunft ist meist **zeitlich begrenzt**;*
> - *das **Ziel ist festgelegt** und für alle Mitglieder gleich;*
> - *die **Rollen** der Mitglieder sind **definiert**;*
> - *die **Aufgaben** der einzelnen Mitglieder **können sich unterscheiden**;*
> - *die Mitglieder sind in der Erlangung eines Resultats **voneinander abhängig**;*
> - *die Mitglieder **interagieren abhängig** voneinander.*
>
> *[Stelling, 1999]*

Von diesen Kriterien werden insbesondere zwei häufig in anderen Definitionen genannt, nämlich

- das Erreichen eines gemeinsamen Ziels
  (z.B. [Fleishman & Zaccaro, 1992][Jennings, 1994][Malone & Crowston, 1994] [Tambe, 1997][Stone & Veloso, 1999][Tidhar & Sonenberg, 2000][Niermeyer, 2001][Yen et al., 2001][von Rosenstiel, 2003]) und
- die Abhängigkeit der Teammitglieder voneinander
  (z.B. [Salas et al., 1992][Fleishman & Zaccaro, 1992][Fan & Yen, 2004]).

Auch im Kontext des Arbeitssystems spielt das Erreichen eines gemeinsamen Ziels durch kooperierende Teammitglieder eine entscheidende Rolle. So ist es das gemeinsame Ziel eines Teams der *Operating Force*, das von außen gegebene Arbeitsziel zu erreichen bzw. einen gegebenen Auftrag zu erfüllen. Das gemeinsame Ziel eines Teams im Rahmen der Arbeitsmittel wie es im weiteren Verlauf der Arbeit betrachtet wird besteht in der erfolgreichen Bearbeitung der von der *Operating Force* zugewiesenen Aufgaben bzw. Teilaufträge.

Um nun das gemeinsame Ziel erreichen zu können, muss die Zusammenarbeit im Team verschiedenen Anforderungen genügen. Dabei wird die Annahme zugrunde gelegt, dass alle an der Teamarbeit Beteiligten grundsätzlich versuchen, als Team die bestmögliche Leistung zu erbringen. Letztere hängt von den Koordinations- und Kommunikationsmechanismen ab, die ein menschliches Team mit der Zeit entwickelt bzw. die einem maschinellen Team zur Verfügung gestellt werden. Diese Prozesse werden von organisatorischen und situativen Gegebenheiten beeinflusst, wozu auch die Aufgabe und Arbeitsumgebung sowie die Eigenschaften des Einzelnen und des Teams als Ganzes zählen [Salas et al., 1992] (vgl. auch [Fleishman & Zaccaro, 1992]). Ein umfassender Katalog an sich zum Teil überschneidenden Forderungen an Teamarbeit wurde von [Swezey & Salas, 1992] auf Basis einer Literaturrecherche zusammengestellt, die Richtlinien für die Gestaltung von Lehrgängen für Teamarbeit im Umfeld der Marine darstellen. Sie charakterisieren sowohl Fähigkeiten, die einzelne Teammitglieder benötigen, als auch die Zusammenarbeit im Team an sich. Welche dieser Forderungen im Einzelnen für Teams im Arbeitssystem umgesetzt werden können und sollen bzw. ob zusätzliche Forderungen berücksichtigt werden müssen, hängt dabei stark von der jeweiligen Anwendung und deren spezifischen Anforderungen ab. Im Rahmen dieser Arbeit werden Forderungen an die Zusammenarbeit von Mensch und Maschine zugrunde gelegt, die von [Billings, 1997] im Kontext der bemannten Luftfahrt aufgestellt wurden und von [Meitinger & Schulte, 2004] im Hinblick auf Teams, die aus künstlichen kognitiven Einheiten und ggf. Menschen bestehen, verallgemeinert wurden:

- Alle Teammitglieder müssen aktiv eingebunden sein.
- Alle Teammitglieder müssen angemessen informiert sein.
- Alle Teammitglieder müssen sich gegenseitig überwachen.
- Alle Teammitglieder müssen daher vorhersehbar handeln.
- Alle Teammitglieder müssen die Absicht(en) der anderen Teammitglieder kennen.

Die Umsetzung dieser Forderungen erfolgt durch die Repräsentation entsprechender Ziele, welche in Abschnitt 4.3.1 beschrieben werden.

Während zwar alle Teammitglieder in gleicher Weise dazu fähig sein müssen, in einem Team zusammenzuarbeiten, so unterscheiden sie sich im Allgemeinen doch hinsichtlich der Rollen, die sie im Team einnehmen. Die Rollendifferenzierung kann dabei durch die *Teamstruktur* beschrieben werden, welche die statische Komponente eines Teams darstellt. Sie wird durch *Teamprozesse* als dynamische Komponente ergänzt, welche alle Aktivitäten innerhalb eines Teams beschreiben, die Teamstrukturen entstehen lassen oder verändern bzw. die Leistung eines Teams bewirken oder beeinflussen. [von Rosenstiel, 2003][Swezey & Salas, 1992]

Im Umfeld von Multi-Agenten-Systemen unterscheiden [Biggers & Ioerger, 2001] hierarchische und verteilte Teamstrukturen. Hierarchisch geführte Teams basieren dabei auf einer strikten Befehlskette, in der eine zentrale Entscheidungsinstanz Aufgaben an Teammitglieder zuweist. Diese sind nicht verantwortlich für das Erreichen des gemeinsamen Teamziels, sondern müssen lediglich Entscheidungen im Verantwortungsbereich der ihnen zugewiesenen Aufgaben treffen [Biggers & Ioerger, 2001]. Somit sind Teammitglieder hierarchisch organisierter Teams auch nicht autorisiert, selbst die Initiative im Hinblick auf das Teamziel zu ergreifen und zum Beispiel eigenständig Aufgaben zu übernehmen, die das Erreichen des Teamziels begünstigen. Diese Art der Zusammenarbeit ist ähnlich der Überwachung von Arbeitsmitteln durch den Bediener im Arbeitssystem, der diese im Hinblick auf das Arbeitsziel einsetzt. Somit ist strukturell nur ein

einziges derartiges Team im Arbeitssystem möglich, bei der die *Operating Force* als Ganzes das Team leitet und jedes Arbeitsmittel ein Teammitglied im übertragenen Sinne darstellt. Eine solche Betrachtungsweise bietet sich an, sofern mehrere unabhängige semi-autonome Systeme als Arbeitsmittel zur Verfügung stehen.

In verteilten Teamstrukturen haben Teammitglieder keinen dezidierten Leiter, sondern können frei miteinander interagieren und selbst entscheiden, welches Teammitglied welche Aufgabe zu welchem Zeitpunkt bearbeiten soll [Biggers & Ioerger, 2001]. In ihren Verantwortungsbereich fällt auch, sicherzustellen, dass alle Aufgaben, die bearbeitet werden müssen, um das gemeinsame Ziel zu erreichen, einem Teammitglied zugewiesen sind. Der Koordinationsaufwand in einer derartigen Struktur ist gegenüber dem hierarchischen Ansatz deutlich größer, da die Teammitglieder sehr viel freier agieren können. Dem gegenüber steht eine erhöhte Flexibilität auf Situationsänderungen zu reagieren und die Unabhängigkeit von einer dedizierten Entscheidungsinstanz. Grundsätzlich ist hierbei die Eigeninitiative der Teammitglieder unbedingt notwendig [Biggers & Ioerger, 2001]. Im Kontext des Arbeitssystems trifft man solche Teamstrukturen typischerweise sowohl in Teams innerhalb der *Operating Force* als auch innerhalb der Arbeitsmittel an. Während alle Elemente der *Operating Force* grundsätzlich ein einziges Team bilden, können auf Seiten der Arbeitsmittel prinzipiell mehrere Teams als verschiedene Arbeitsmittel zur Verfügung stehen.

Auch wenn die beiden vorgestellten Teamstrukturen ihre Entsprechung in der Theorie des Arbeitssystems haben, so wird man in realen Systemen insbesondere im Hinblick auf verteilte Strukturen selten diese Reinformen antreffen. Wahrscheinlicher sind vorstrukturierte, verteilte Teamstrukturen, die einen Kompromiss zwischen Flexibilität und Koordinationsaufwand darstellen (vgl. auch Teams von Mitarbeitern in Unternehmen, z.B. [Gellert & Nowak, 2002]).

Abbildung 3-14 stellt die Teamstruktur dar, die die vorliegende Arbeit bei der Entwicklung eines Teams aus *Supporting ACUs* zugrunde legt. Sie geht davon aus, dass jedes Teammitglied über eine oder mehrere Fähigkeiten verfügt (in Abbildung 3-14 stellen die Farben „rot“ und „blau“ Platzhalter für Fähigkeiten dar) und im Rahmen der Teamarbeit zu bearbeitende Aufgaben bestimmter Fähigkeiten bedürfen.

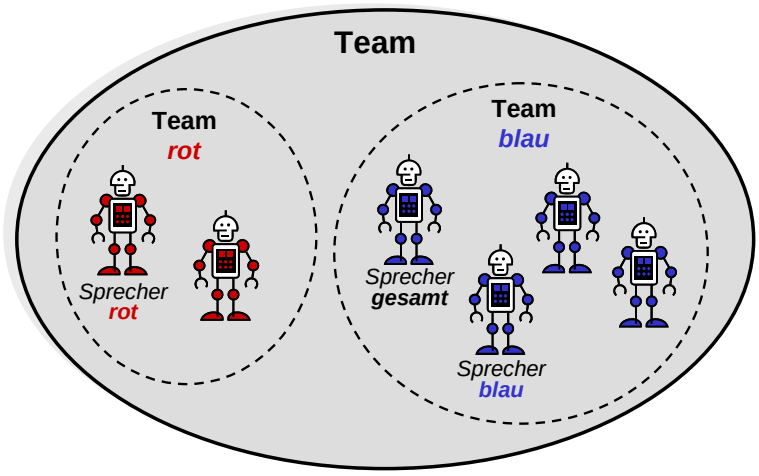


Abbildung 3-14: Teamstruktur mit Sub-Teams, die entsprechend der Fähigkeiten der Teammitglieder gebildet werden (Farbe symbolisiert Fähigkeit)

Grundsätzlich wird von einer verteilten Teamstruktur ausgegangen, in der alle Teammitglieder gleichberechtigt agieren und Verantwortung für das Erreichen des Teamziels tragen. Um die Nachteile einer solchen verteilten Teamstruktur möglichst zu vermeiden

und gleichzeitig ihre Vorteile nutzen zu können, werden hier zwei strukturierende Elemente eingeführt: Zum einen wird für jede im Team vorhandene Fähigkeit ein Sub-Team gebildet, dem alle Teammitglieder, die über die entsprechende Fähigkeit verfügen, zugeordnet sind. Somit kann Koordinationsaufwand verringert werden, indem zum Beispiel Anfragen, die eine bestimmte Fähigkeit voraussetzen, gezielt an ein solches Sub-Team gestellt werden können (vgl. auch Koordination durch Organisation im nächsten Abschnitt). Zum anderen wird in jedem Team bzw. Sub-Team die Rolle eines Teamsprechers besetzt. Ähnlich dem Teamleiter eines hierarchischen Teams stellt er eine zentrale Anlaufstelle dar, spricht im Namen des Teams mit Agenten außerhalb des Teams und trifft Entscheidungen für das Team. Hierunter fallen allerdings lediglich Entscheidungen, die das Team als Ganzes und nicht nur einzelne Teammitglieder betreffen. Außerdem werden diese Entscheidungen transparent für das Team gefällt, so dass im Gegensatz zu hierarchischen Teams jedes Teammitglied in der Lage ist, die Rolle des Teamsprechers zu übernehmen, sofern dieser ausfällt.

3.3.3 Koordination

Zwischen den Aktivitäten der Mitglieder eines Teams bestehen im Allgemeinen Abhängigkeiten, mit denen in geeigneter Art und Weise umgegangen werden muss. Diesen Prozess bezeichnen [Malone & Crowston, 1994] als Koordination – eine Auffassung, die sich gegenüber einer Vielzahl weiterer Definitionen weitgehend durchgesetzt hat. Generell wird mit Koordination eine Erhöhung der Performanz bzw. eine Reduktion von Konflikten bei der Zusammenarbeit im Team angestrebt [Ferber, 1999]. Es soll also sichergestellt werden, dass ein Team in sich stimmig arbeitet, d.h. zum Beispiel dass laut [Durfee et al, 1992][Jennings, 1996]

- nicht eine Handlung eines Teammitglieds die vorangegangene eines anderen rückgängig oder eine geplante unmöglich macht,
- das Team zielgerichtet arbeitet,
- alle Teile der übergeordneten Aufgabenstellung von mindestens einem Teammitglied bearbeitet werden und
- die Teammitglieder so interagieren, dass ihre Aktivitäten so zusammenpassen, dass die übergeordnete Aufgabenstellung erfolgreich bearbeitet werden kann.

Im Kontext des Arbeitssystems tritt Koordination in unterschiedlichen Ausprägungen und an verschiedenen Stellen auf, konkret im Rahmen aller vorkommenden Teams wie sie im vorhergehenden Abschnitt charakterisiert wurden. Werden *Operating Force* und verschiedene Arbeitsmittel als hierarchisches Team mit der *Operating Force* als zentraler Entscheidungsinstanz aufgefasst, so fällt es in deren Zuständigkeitsbereich, Abhängigkeiten zwischen den Aktivitäten einzelner Arbeitsmittel („Teammitglieder“) aufzulösen. Mitglieder von Teams mit verteilter Teamstruktur, wie sie innerhalb der *Operating Force* bzw. der Arbeitsmittel vorkommen können, koordinieren sich meist eigenständig innerhalb des Teams.

Hierbei lassen sich nach [Ferber, 2001] verschiedene kooperative Interaktionssituationen unterscheiden (siehe Abbildung 3-15), welche dadurch gekennzeichnet sind, dass die Ziele der beteiligten Teammitglieder in allen Fällen kompatibel sind. Eine Beschränkung auf die Betrachtung dieser Fälle ergibt sich aus der Zusammenarbeit im Team, bei der alle Teammitglieder am gemeinsamen Ziel arbeiten und daher im Prinzip über kompatible Ziele verfügen. Inkompatible Ziele bedeuten im Gegensatz dazu, dass das Erreichen eines Zieles das Erreichen eines anderen unmöglich macht [Ferber, 2001].

Situationstyp	*Ziele*	*Ressourcen*	*Fähigkeiten*
Einfache Zusammenarbeit	kompatibel	ausreichend	unzureichend
Blockade	kompatibel	unzureichend	ausreichend
Koordinierte Zusammenarbeit	kompatibel	unzureichend	unzureichend

Abbildung 3-15: Kooperative Interaktionssituationen (vgl. [Ferber, 2001])

Die dargestellte Reihenfolge – *Einfache Zusammenarbeit, Blockade, Koordinierte Zusammenarbeit* – reflektiert hierbei den Grad der Abhängigkeit zwischen den an der Kooperation beteiligten Teammitgliedern, der von oben nach unten zunimmt.

Einfache Zusammenarbeit zeichnet sich dadurch aus, dass genügend Ressourcen aber nicht ausreichend Fähigkeiten bei jedem Teammitglied vorhanden sind, um ein gemeinsames Ziel zu erreichen. Diese Art der Zusammenarbeit ist also typisch für ein Team aus Spezialisten, in dem jeder entsprechend seiner Fähigkeiten bestimmte Teilaufgaben übernehmen kann, für die er genügend Ressourcen zur Verfügung hat und die er weitgehend unabhängig bearbeiten kann [Ferber, 2001]. Als Beispiel hierfür sei ein Team aus zwei UAVs angeführt, von denen das eine gegnerische Bedrohungen erkennen und unterdrücken kann, während das andere über entsprechende Bewaffnung verfügt, um ein dezidiertes Ziel bekämpfen zu können. Durch diese klare Trennung von Fähigkeiten gibt es im Allgemeinen nur eine bzw. wenige Möglichkeit(en), Aufgaben an Teammitglieder zuzuweisen, so dass der Koordinationsbedarf in dieser Konstellation sehr gering ist [Ferber, 2001]. Im Rahmen der im vorhergehenden Abschnitt vorgestellten Teamstruktur findet diese Art der Zusammenarbeit zwischen spezialisierten Sub-Teams statt, deren Organisation entsprechend den Fähigkeiten der Teammitglieder die Zuweisung bestimmter Aufgaben an eine Teilmenge aller beteiligten Agenten vorbestimmt.

Im Fall der *Blockade* sind mehrere Teammitglieder von der gleichen Ressource abhängig. Dies kann beispielsweise eine Maschine sein, die zwei Arbeiter in einer Fabrikhalle gleichzeitig benötigen, aber auch ein Ort, an dem sich zwei Teammitglieder zur gleichen Zeit aufhalten müssen oder wollen (z.B. Start-/Landebahn eines UAV-Stützpunktes). Daher besteht die Hauptaufgabe der Teammitglieder im Hinblick auf Kooperation darin, sich so zu koordinieren, d.h. die zur Verfügung stehenden Ressourcen so aufzuteilen, dass das gemeinsame Ziel möglichst gut erreicht werden kann [Ferber, 2001]. Je nachdem, welche Ressource in begrenzter Menge verfügbar ist, findet sich diese Interaktionssituation im Rahmen der in Abbildung 3-14 dargestellten Teamstruktur sowohl im Gesamtteam (z.B. Nutzung von Korridoren im Luftraum durch alle Teammitglieder) als auch innerhalb der Sub-Teams. In diesem Zusammenhang ist insbesondere auch die Ressource „Arbeitskraft der Teammitglieder“ von Bedeutung, die den zu bearbeitenden Aufgaben zugewiesen werden muss (vgl. [Malone & Crowston, 1994]).

Koordinierte Zusammenarbeit schließlich stellt eine Mischform der beiden vorhergehenden Fälle dar und ist daher auch die Form der Zusammenarbeit, die im weiteren Verlauf der Arbeit betrachtet wird. Hierbei hat kein einzelner Agent alle nötigen Fähigkeiten, um die Gesamtaufgabe zu bearbeiten und ist bei der Bearbeitung von Teilaufgaben zusätzlich von Ressourcen abhängig, die auch andere Teammitglieder benötigen. Dabei kann die Qualität des Arbeitsergebnisses von der Verteilung von Teilaufgaben auf Teammitglieder und der spezifischen Aufteilung von Ressourcen abhängen. Somit ist effektive Koordination erforderlich. [Ferber, 2001]

Koordination umfasst sowohl das Auflösen negativer als auch die Betrachtung positiver Abhängigkeiten zwischen Aktivitäten von Agenten (vgl. Abbildung 3-16). Negative Abhängigkeiten verhindern, dass Aktivitäten wie beabsichtigt ausgeführt werden können, weil ein Ressourcenkonflikt vorliegt oder die Zielsetzungen der Aktivitäten unvereinbar sind. Durch eine Kombination von Aktivitäten, die positiv voneinander abhängig sind, kann hingegen ein Nutzen für einen oder mehrere Agenten gezogen werden, so dass die Vorgehensweise bzw. Effizienz insgesamt besser wird. [von Martial, 1992]

Im Rahmen der Arbeit werden eine negative und zwei positive Abhängigkeiten betrachtet, nämlich die gemeinsame Nutzung von sich nicht verbrauchenden Ressourcen (*non-consumable resource*) sowie die Gleichheitsbeziehung (*equality*) und die Begünstigungsbeziehung (*favor*). Die Gleichheitsbeziehung bezeichnet dabei, dass eine Aktivität von einem beliebigen Agenten aus einer Menge prinzipiell geeigneter Agenten durchgeführt werden kann, während die Begünstigungsbeziehung beschreibt, dass eine (geplante) Aktivität eines Agenten die Durchführung einer Aktivität eines anderen Agenten fördert [von Martial, 1992].

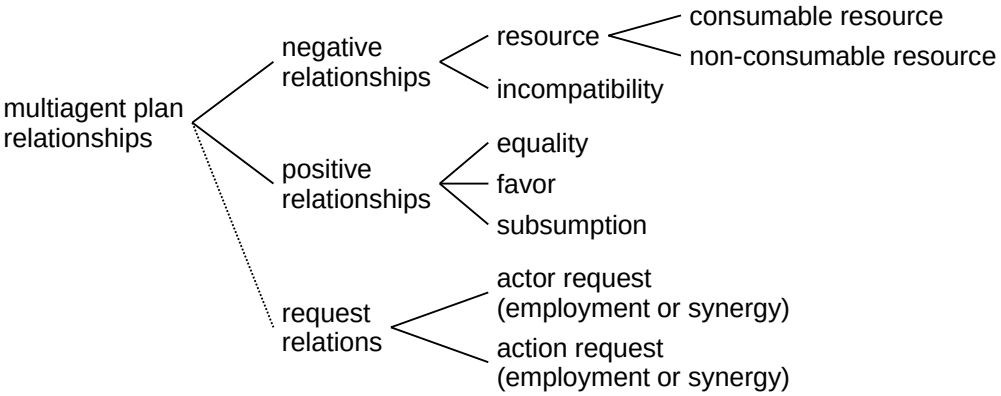


Abbildung 3-16: Klassifikation von Abhängigkeiten zwischen Aktivitäten [von Martial, 1992]

Daneben sind Teammitglieder im entwickelten Team aus *Supporting ACUs* auch durch Anfragen voneinander abhängig, da ein Teammitglied bei einem anderen anfragen kann, ob dieses eine bestimmte Aufgabe durchführt. Diese Art der Abhängigkeit ist eng verknüpft mit dem Vorgang, zu bearbeitende Aufgaben an Teammitglieder zuzuweisen, was nach [Malone & Crowston, 1994] als die Zuweisung der Ressource „Arbeitskraft" angesehen wird. Weiterhin ist in der vorliegenden Arbeit die Erzeuger/Verbraucher-Abhängigkeit im Sinne von [Malone & Crowston, 1994] relevant. Hierbei erzeugt eine Aktivität etwas, das von einer anderen benötigt wird. Dies ist zum Beispiel der Fall, wenn ein UAV seinen Flug erst dann fortsetzen kann, wenn eine Bedrohung von einem anderen UAV unterdrückt wurde. Schließlich sind speziell Teammitglieder dadurch voneinander abhängig, dass mehrere Teilaktivitäten zur Erreichung eines übergeordneten Ziels beitragen [Malone & Crowston, 1994].

Um derartige Abhängigkeiten auflösen zu können, werden Koordinationsmechanismen benötigt, welche sich nach [Jennings, 1996] in drei Kategorien einteilen lassen. Demnach kann Koordination im Rahmen einer *Organisationsstruktur*, auf Basis von *Informationsaustausch* oder durch *Planung* stattfinden (vgl. auch [Bond & Gasser, 1988] [Ferber, 2001]). Diese werden im Folgenden anhand des Beispiels „Aufgabenzuweisung im Team" näher erläutert.

In einer Organisationsstruktur werden den einzelnen Teammitgliedern ihre jeweiligen Zuständigkeiten fest zugewiesen, die zum Beispiel funktionaler oder räumlicher Art sein können (z.B. Teammitglied A unterdrückt alle SAM-Stellungen in Gebiet 1, Team-

mitglied B unterdrückt alle SAM-Stellungen in Gebiet 2). Diese Art der Koordination minimiert die dafür benötigte Kommunikationsbandbreite, da sich die Zuweisung der Zuständigkeiten in der Regel nicht ändert. Andererseits kann sie sich nicht an Änderungen der Umgebung (z.B. Ausfall eines Teammitglieds) anpassen, da diese zwar beeinflussen können, wie ein Teammitglied seine Aufgaben durchführt, die Zuweisung von Aufgaben an Teammitglieder aber starr ist (vgl. Abbildung 3-17).

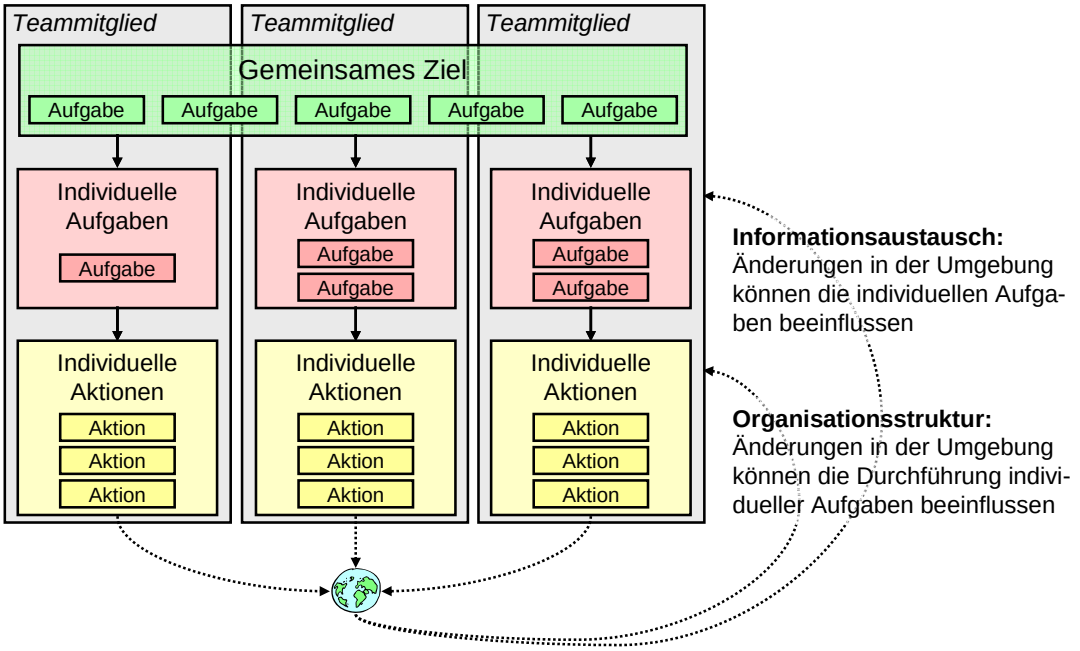


Abbildung 3-17: Koordination mittels Organisationsstruktur und Informationsaustausch

Koordinieren sich Agenten mit Hilfe von Informationsaustausch über zum Beispiel ihre aktuellen Absichten, so nimmt die Menge an ausgetauschter Information zu. Dafür steigt die Flexibilität, auf Situationsänderungen reagieren bzw. diese antizipieren zu können, da es den Teammitgliedern erlaubt ist, auch die Zuweisung von Aufgaben dynamisch an die Gegebenheiten der Situation anzupassen (vgl. Abbildung 3-17).

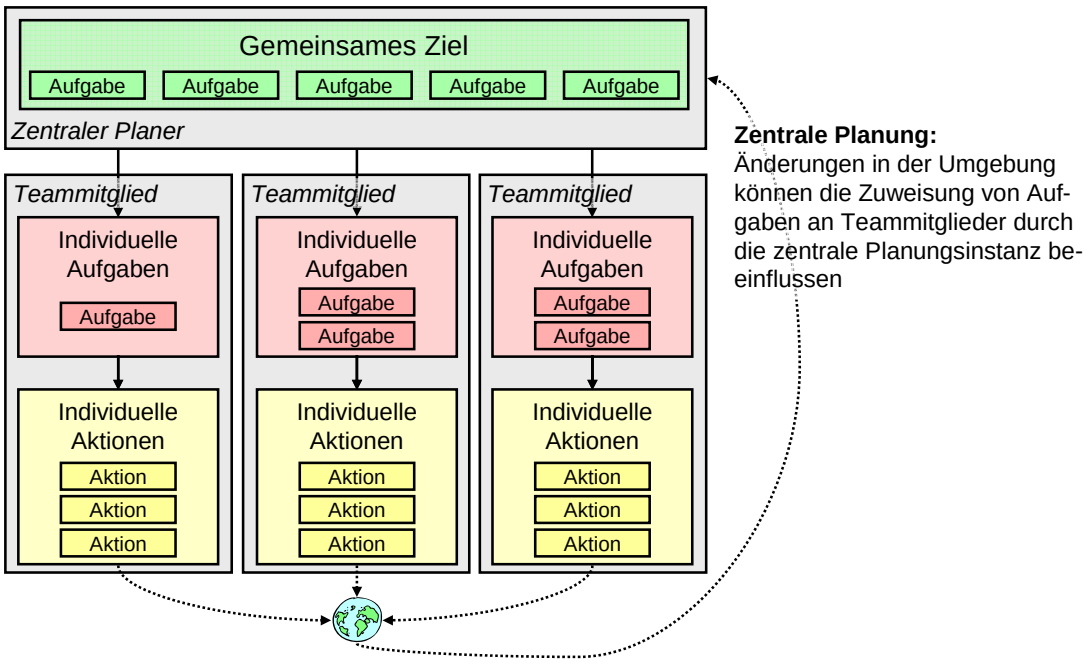


Abbildung 3-18: Koordination mittels zentraler Planung

Wird als grundlegender Ansatz zentrale Planung (im Gegensatz zu verteilter Planung) gewählt (vgl. Abbildung 3-18), so nimmt das Kommunikationsaufkommen weiter zu, da sämtliche zur Koordination nötigen Informationen von Agenten an die zentrale Planungseinheit und umgekehrt übermittelt werden müssen.

Werden von der zentralen Planung nicht nur Aufgaben zugewiesen, sondern sogar Aktionen einzelner Teammitglieder geplant, nimmt die Menge an Information noch weiter zu. Dafür kann die Koordination global optimiert werden. Nachteil eines solchen Ansatzes ist die Abhängigkeit von der Verfügbarkeit von Kommunikationslinien und einer zentralen Planungseinheit. Um die Vorteile verschiedener Ansätze nutzen zu können, ist es je nach Teamstruktur und Fähigkeiten der Teammitglieder sinnvoll, diese zu kombinieren [Meitinger & Schulte, 2005]. Ein solcher hybrider Ansatz wird in der vorliegenden Arbeit insbesondere für die Zuweisung von Aufgaben eingesetzt. Vor dem Hintergrund der in Abbildung 3-14 dargestellten Teamstruktur wird eine Organisationsstruktur (Sub-Teams) verwendet, um Zuständigkeiten für Aufgabentypen, die spezifische Fähigkeiten benötigen, an ein Sub-Team, dessen Mitglieder über eben diese Fähigkeiten verfügen zuzuweisen. Dies beeinträchtigt die Flexibilität eines Teams nicht, da andere Teammitglieder ohnehin nicht in der Lage sind, derartige Aufgaben zu übernehmen. Innerhalb der Sub-Teams können Aufgaben dann flexibel auf Basis von Informationsaustausch zugewiesen werden (vgl. Abbildung 3-19).

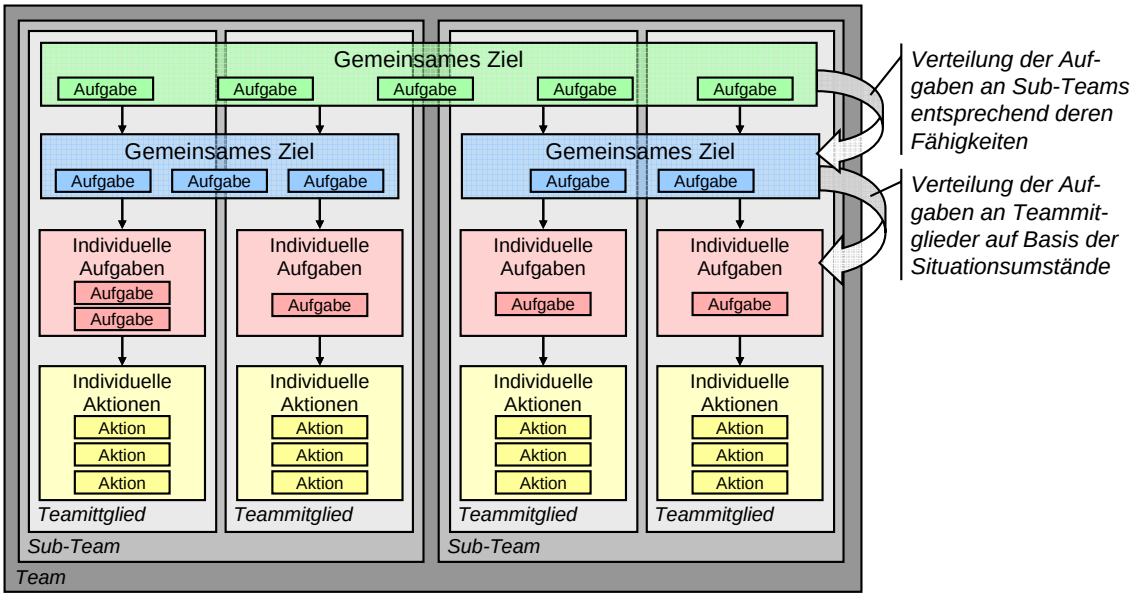


Abbildung 3-19: Hybrider Koordinationsansatz

Grundlage für die Zusammenarbeit im Team und insbesondere die Zuweisung von Aufgaben stellen dabei Verpflichtungen (*commitments*) einzelner Teammitglieder, Verpflichtungen eines Teams (*joint commitments*) und Konventionen (*conventions*) dar (vgl. [Jennings, 1996]). Verpflichtungen im Bereich der Agententechnologie wurden zuerst von [Cohen & Levesque, 1990] eingeführt und stellen Zusicherungen einzelner Agenten dar, bestimmte Aufgaben zu bearbeiten bzw. eine bestimmte Vorgehensweise zu verfolgen [Jennings, 1996]. Die gemeinsame Verpflichtung mehrerer Teammitglieder als *joint commitment* drückt sich dadurch aus, dass alle ein bestimmtes, gemeinsames Ziel erreichen wollen [Cohen & Levesque, 1991][Levesque et al., 1990]. Konventionen [Jennings, 1994][Jennings, 1995] beschreiben Umstände unter denen ein einzelner Agent seine Verpflichtungen überdenken soll [Jennings, 1996], während soziale Konventionen andere Agenten mit in Betracht ziehen und das Verhalten eines Teammit-

glieds gegenüber den anderen Teammitgliedern für den Fall spezifizieren, dass sich seine Verpflichtungen ändern [Jennings, 1996].

Im Rahmen dieser Arbeit gehen sowohl einzelne Teammitglieder als auch Teams Verpflichtungen ein, bestimmte Aufgaben einschließlich zugehöriger Teilaufgaben bzw. unterstützender Aufgaben zu bearbeiten. Dabei werden das Eingehen, das Durchführen und auch das Aufgeben von Verpflichtungen im Sinne von Konventionen und sozialer Konventionen auf Ziele abgebildet, die das Verhalten der *Supporting ACUs* bestimmen (vgl. Abschnitt 4.3). Dies gilt sowohl für Verpflichtungen einzelner Teammitglieder als auch für die Verpflichtungen eines Teams, wobei hier dem Teamsprecher eine besondere Rolle zukommt. Dieser ist nämlich dafür verantwortlich, Verpflichtungen für das Team einzugehen und aufzugeben und die Mitglieder des Teams ebenso wie Interessenten außerhalb des Teams (z.B. Operateur) über die Verpflichtungen zu informieren.

Ziele in diesem Zusammenhang orientieren sich im Wesentlichen an den in [Jennings, 1994] vorgestellten (sozialen) Konventionen, die aussagen, dass eine Verpflichtung eines einzelnen Agenten aufgegeben werden soll, sofern diese erfüllt oder unerreichbar geworden ist oder die Motivation für die Verpflichtung nicht mehr vorhanden ist (vgl. auch [Cohen & Levesque, 1990]). Gleiches gilt für gemeinsame Verpflichtungen eines Teams, wobei hier zusätzlich gefordert wird, dass Teammitglieder sich gegenseitig informieren, wenn sich ihre individuellen Verpflichtungen ändern und dies für eine gemeinsame Verpflichtung von Bedeutung ist. Darüber hinaus soll das Fortbestehen einer gemeinsamen Verpflichtung überprüft werden, wenn sich hierfür relevante Verpflichtungen einzelner Teammitglieder ändern.

Neben der Nutzung gemeinsamer Verpflichtungen gibt es eine Vielzahl weiterer Methoden im Umfeld der Koordination mehrerer Agenten. Verbreitet ist die Verwendung regulierender Normen wie zum Beispiel Straßenverkehrsregeln [Ferber, 2001] [Wooldridge, 2002]. Dieser Ansatz wird im Rahmen der vorliegenden Arbeit für die Zuweisung einer gemeinsam genutzten Ressource genutzt. Einen Überblick über weitere Methoden wie zum Beispiel die gegenseitige Modellierung von Teammitgliedern geben [Wooldridge, 2002] und [Ferber, 2001].

3.3.4 Kommunikation

Kommunikation, d.h. „*intensiver Informationsaustausch“ [Schneider, 1996]*, stellt die Grundlage für Kooperation und Koordination im Team dar [Borghoff & Schlichter, 2000][Fan & Yen, 2004][Ferber, 1999][Grosz & Kraus, 1996][Jennings, 1994][von Rosenstiel, 2003][Schneider, 1996]. Hierbei kann zwischen expliziter und impliziter Kommunikation unterschieden werden. Explizite Kommunikation bezeichnet intentionalen Austausch von Information auf verbale oder nicht-verbale Art und Weise (vgl. [Russell & Norvig, 2003]), zum Beispiel durch das Senden von Nachrichten oder die Nutzung von Handzeichen. Implizite Kommunikation hingegen leitet Informationen auf Basis der Beobachtung des Verhaltens anderer ab.

Im Arbeitssystem findet dabei sowohl explizite als auch implizite Kommunikation statt. Sofern eine ACU mit einer anderen ACU kommuniziert treten dabei üblicherweise wenig Probleme auf, da sich technische Systeme stets an vorgegebene Syntax und Semantik halten und Verhaltensmodelle von technischen Systemen, welche für das Ableiten von Informationen im Rahmen von impliziter Kommunikation notwendig sind, im Allgemeinen gut erstellt werden können. Anders verhält es sich bei der Interaktion zwischen Mensch und ACU. Um Probleme hierbei zu vermeiden, müsste der menschliche Interaktionspartner entweder gezwungen werden, sich an vorgegebene Protokolle und

Syntaxen zu halten oder die ACU in die Lage versetzt werden, im Rahmen expliziter Kommunikation mit spezifisch menschlichem und damit mitunter imperfektem Verhalten umzugehen. Hierbei lässt der letztgenannte Ansatz nicht nur eine größere Akzeptanz beim Menschen vermuten, sondern von ihm wird auch eine größere Performanz erwartet. Im Rahmen einer Realisierung könnten hierzu Verhaltensmodelle von Menschen eingesetzt werden, wofür das qualitative Mehrfachressourcen-Modell von [Wickens, 1984] einen ersten Ansatzpunkt bietet. In [Gerlach, 1996] werden weitere, hierfür relevante technische Ansätze im Rahmen eines Dialogmanagers für ein Pilotenassistenzsystem vorgeschlagen. Ähnliche Betrachtungen sind auch für implizite Kommunikation notwendig und wurden bereits zum Beispiel im Kontext der Erkennung von Pilotenabsichten durch Beobachtung von Pilotenverhalten angestellt [Strohal & Onken, 1998].

Um (explizit) miteinander kommunizieren zu können, sind nach [Shannon & Weaver, 1948] grundsätzlich mehrere Komponenten nötig: ein Sender und Empfänger, eine Sprache oder Kodierung der Nachricht, ein Kommunikationskanal oder -medium und ein Kontext oder eine Situation, in der sich die Akteure befinden [Biggers & Ioerger, 2001]. [Hollnagel & Woods, 2005] erweitern das Kommunikationsmodell im Hinblick auf die Analyse von Mensch-Maschine-Systemen so, dass ein geschlossener Kreis bei der Kommunikation zweier Systeme entsteht (vgl. Abbildung 3-20). Hierbei wird insbesondere betont, dass nach Eingang einer Information ein interner Verarbeitungsschritt nötig ist, bevor eine Antwort gesendet werden kann.

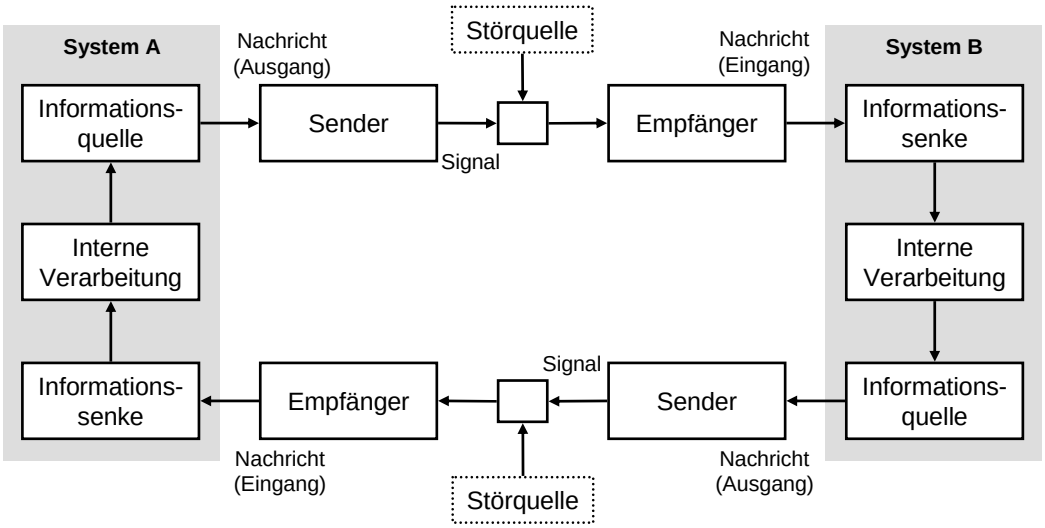


Abbildung 3-20: Erweitertes Kommunikationsmodell [Hollnagel & Woods, 2005]

Alle genannten Komponenten sind auch bei der Kommunikation mehrerer *(Supporting) ACUs* nötig. Sie stellen jeweils ein System A bzw. B im Kommunikationsmodell dar und verfügen in ihren Arbeitsmitteln über die nötigen technischen Komponenten zum Nachrichtenaustausch wie zum Beispiel Sende- und Empfangsanlage. Im weiteren Verlauf der Arbeit werden lediglich die internen Schritte der Informationsverarbeitung, d.h. das Interpretieren und Erzeugen von Nachrichten im Kontext der Gesamtsituation und insbesondere vor dem Hintergrund aktueller Handlungsziele unter Berücksichtigung von Syntax und Semantik betrachtet.

Mit solchen Nachrichten sollen im Umfeld von Kooperation und Koordination in der Regel andere Teammitglieder im Sinne der Ziele beeinflusst werden, da diese im Rahmen verteilter Teamstrukturen nicht gezwungen werden können, etwas zu tun (vgl. [Wooldridge, 2002]). Kommunikation wird hier also wie in vielen anderen Multiagen-

ten-Systemen vor dem Hintergrund der Sprechakttheorie betrachtet, die von Austin und Searle begründet wurde [Austin, 1962][Searle, 1969] und davon ausgeht, dass bestimmte natürlichsprachige Äußerungen – Sprechakte genannt – über die Eigenschaft verfügen, dass sie ebenso wie Handlungen den Zustand der Umgebung verändern [Wooldridge, 2002]. Nachrichten werden also als Aktionen aufgefasst, welche die an der Kommunikation beteiligten Agenten beeinflussen.

Sprechakte bestehen dabei nach Austin aus drei Komponenten: der *Lokution*, der *Illokution* und der *Perlokution*. Die Lokution stellt dabei die tatsächliche Äußerung, d.h. zum Beispiel eine Nachricht im Textformat, die Illokution die Absicht des Senders und die Perlokution die Wirkung des Sprechakts beim Empfänger dar [Ferber, 1999] [Wooldridge, 2002]. Illokutionen werden von Searle in die folgenden fünf Typen klassifiziert [Ferber, 1999][Wooldridge, 2002]:

- *Assertiva* informieren über den Zustand der Umgebung (z.B. „Das Ziel ist 20 Meilen entfernt.“)
- *Direktiva* fordern den Empfänger auf etwas zu tun (z.B. „Kläre das Zielgebiet auf!“)
- *Kommissiva* verpflichten den Sender zu einer Handlung (z.B. „Ich werde das Zielgebiet aufklären.“)
- *Expressiva* informieren über den Zustand des Senders (z.B. „Ich verfüge über einen bildgebenden Sensor.“)
- *Deklarativa* verändern die Umgebung durch die Äußerung entsprechend der Äußerung (z.B. „Ihr seid ab sofort ein Team.“)

Eine Illokution ist also durch einen Typ (Performativ) und einen zugehörigen Inhalt gekennzeichnet und kann daher durch die Form `<Performativ>(<Inhalt>)` repräsentiert werden [Ferber, 1999].

Da mit jeder Äußerung im Sinne der Sprechakttheorie eine bestimmte Absicht verfolgt wird, d.h. eine bestimmte Wirkung auf den Empfänger ausgeübt werden soll, können Sprechakte nicht als „richtig“ oder „falsch“ klassifiziert werden, sondern ob sie im Sinne der beabsichtigten Wirkung erfolgreich waren oder nicht. [Ferber, 1999] nennt in diesem Zusammenhang folgende Problemfelder, die das Scheitern eines Sprechakts zur Folge haben können:

- die Äußerung des Sprechakts, also dass eine Nachricht zum Beispiel nicht richtig übertragen wurde oder der Empfänger die Sprache nicht versteht,
- die Interpretation des Sprechakts, d.h. dass der Empfänger den Inhalt einer Nachricht nicht wie beabsichtigt versteht, und
- die Ausführung der Aktion, die der Sprechakt bewirkt hat, weil der Empfänger zum Beispiel nicht die nötigen Fähigkeiten oder Ressourcen hat.

Um insbesondere Probleme im Bereich der Interpretation von Sprechakten soweit möglich von vornherein zu vermeiden, wurden im Kontext von Multiagenten-Systemen diverse Agentenkommunikationssprachen entwickelt, von denen die beiden bekanntesten KQML (Knowledge Query and Manipulation Language, [Finin et al., 1994]) und die FIPA ACL (Foundation for Intelligent Physical Agents Agent Communication Language, [FIPA Homepage]) sind. Beide definieren ein Format für Nachrichten, sowie eine Menge von Performativen, die sich den oben genannten Typen von Illokutionen zuordnen lassen. Die FIPA ACL vermeidet dabei die Nachteile von KQML (unvollständige und unnötig große Menge von Performativen, keine genaue Definition der Bedeu-

tung der Performative) und spezifiziert neben dem Nachrichtenformat [FIPA Nachrichtenformat, 2002] und den (formal definierten) Performativen [FIPA Performative, 2002] auch diverse Sprachen für den Inhalt der Nachrichten und Interaktionsprotokolle.

Während die ersten Aspekte einzelne Nachrichten betrachten, beschreiben Interaktionsprotokolle mögliche Abfolgen von Nachrichten, wobei mit der Initiierung einer Interaktion stets eine bestimmte Wirkung erreicht werden soll. Dabei sind ein Initiator (*initiator*) und ein Teilnehmer (*participant*) an einem Dialog beteiligt. Im Rahmen dieser Arbeit wird der Ablauf von Dialogen mit Hilfe von Zustandsübergangsdiagrammen dargestellt (vgl. [Winograd & Flores, 1986]), wobei Nachrichten Zustandsübergänge bewirken. Eine solche Formalisierung bewirkt, dass ein Initiator eines Dialogs unabhängig von Entscheidungen des Teilnehmers stets eine Antwort auf gesendete Nachrichten erhält und diese als Grundlage für zukünftige Entscheidungen verwenden kann. Im Allgemeinen steht eine spezifische Abfolge von Nachrichten dabei für eine erfolgreiche Durchführung des Dialogs in dem Sinn, dass damit die beabsichtigte Wirkung erzielt wurde, während Abweichungen von diesem Weg auf Probleme hinweisen, so dass unter Umständen mit anderen Aktionen darauf reagiert werden kann. Im Folgenden werden die im Rahmen dieser Arbeit betrachteten Dialoge, welche sich an die genannten Spezifikationen der FIPA anlehnen, kurz vorgestellt und in diesem Zusammenhang die Nachrichtenfolge genannt, welche auf eine erfolgreiche Durchführung des Dialogs hinweist:

- ***Request*** – Der Initiator bittet den Teilnehmer, eine Aufgabe zu bearbeiten, was dieser annehmen oder ablehnen kann. Im Fall der Annahme wird der Initiator zusätzlich über den Ausgang der Aufgabendurchführung benachrichtigt (siehe Abbildung 3-21, vgl. [FIPA Request Interaktion, 2002]). Die Nachrichtenfolge *request – agree – inform:done* deutet auf eine erfolgreiche Durchführung des Dialogs hin.

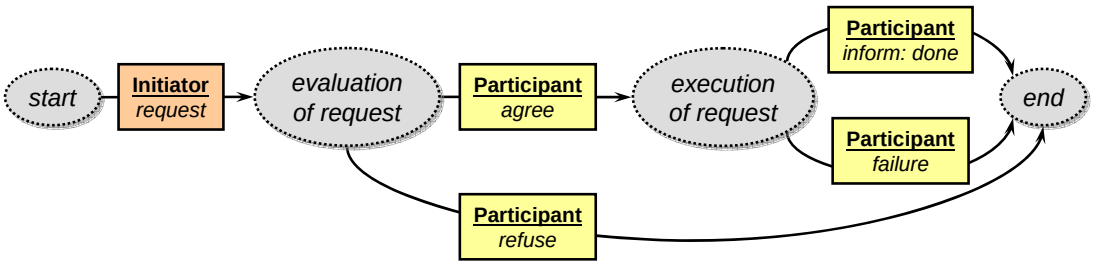


Abbildung 3-21: Request-Dialog (vgl. [FIPA Request Interaktion, 2002])

- ***Propose*** – Der Initiator schlägt dem Teilnehmer vor, selbst eine Aufgabe zu bearbeiten, was der Teilnehmer annehmen oder ablehnen kann. Im Fall der Annahme informiert der Initiator den Teilnehmer außerdem über das Ergebnis der Aufgabenbearbeitung, d.h. ob diese erfolgreich war oder nicht (siehe Abbildung 3-22, vgl. [FIPA Propose Interaktion, 2002]). Die Nachrichtenfolge *propose – accept-proposal – inform:done* deutet auf eine erfolgreiche Durchführung des Dialogs hin.

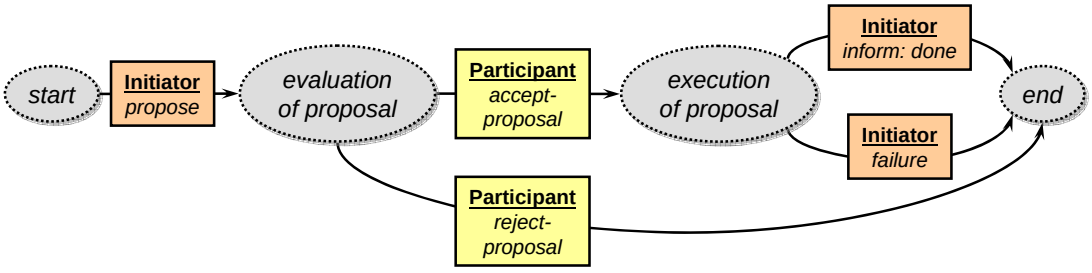


Abbildung 3-22: Propose-Dialog (vgl. [FIPA Propose Interaktion, 2002])

- ***Query*** – Entspricht einem *Request*-Dialog, wobei nicht die Bearbeitung einer Aufgabe, sondern eine Information angefragt wird (siehe Abbildung 3-23, vgl. [FIPA Query Interaktion, 2002]). Die Nachrichtenfolge *query – agree – inform* deutet auf eine erfolgreiche Durchführung des Dialogs hin.

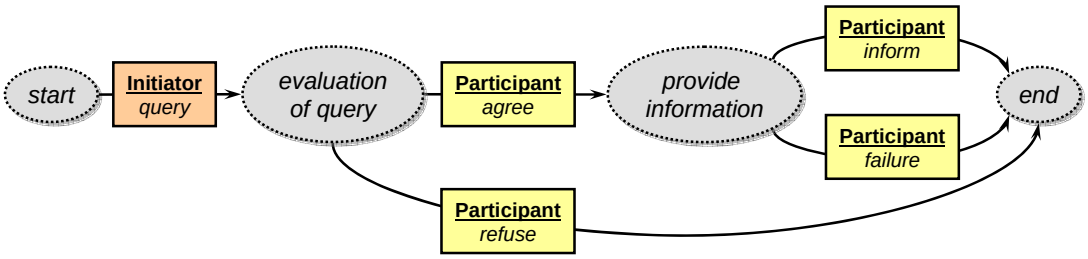


Abbildung 3-23: Query-Dialog (vgl. [FIPA Query Interaktion, 2002])

- ***Subscribe*** – Entspricht einem *Query*-Dialog, der nicht nach Bereitstellen einer Information beendet ist, sondern bei dem eine spezifizierte Information jedes Mal wieder bereitgestellt wird, wenn sich diese geändert hat (siehe Abbildung 3-24, vgl. [FIPA Subscribe Interaktion, 2002]). Die Nachrichtenfolge *query – agree – inform* deutet auf eine erfolgreiche Durchführung des Dialogs hin.

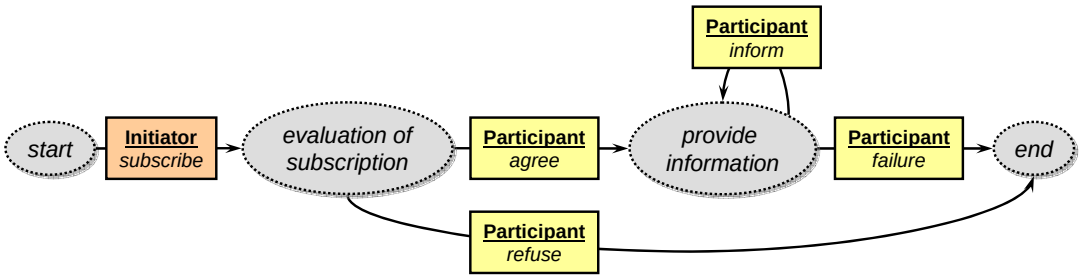


Abbildung 3-24: Subscribe-Dialog (vgl. [FIPA Subscribe Interaktion, 2002])

- ***Inform*** – Hier wird Information nicht wie bei einem *Query*- oder *Subscribe*-Dialog auf Anfrage zur Verfügung gestellt, sondern proaktiv vom Initiator des Dialogs (siehe Abbildung 3-25).

Abbildung 3-25: Inform-Dialog

- ***Cancel*** – Ein Dialog dieses Typs wird vom Initiator verwendet, um einen anderen Dialog, den er selbst initiiert hat, abzubrechen, was auch den Abbruch der wegen dieses Dialogs durchgeführten Aufgaben bzw. Bereitstellung von Information umfasst (siehe Abbildung 3-26, vgl. z.B. [FIPA Request Interaktion, 2002]). Die Nachrichtenfolge *cancel – inform:done* deutet auf eine erfolgreiche Durchführung des Dialogs hin.

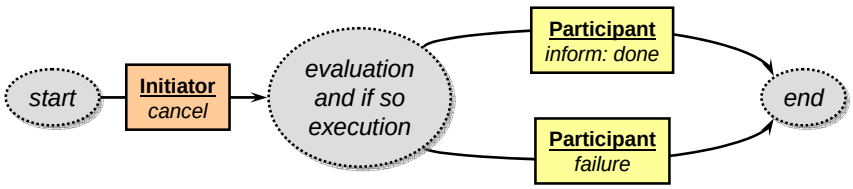


Abbildung 3-26: Cancel-Dialog (vgl. z.B. [FIPA Request Interaktion, 2002])

- ***Instruct*** – Diese Dialogart ist im Rahmen der FIPA ACL nicht vorgesehen, wird aber hier verwendet, um Anweisungen wie z.B. zu einer Position zu fliegen oder sich zur Durchführung einer Aufgabe zu verpflichten von der *Operating Force* an eine oder mehrere *Supporting ACUs* zu senden. Im Gegensatz zu Anfragen (*requests*) können Anweisungen nicht abgelehnt, sondern müssen ausgeführt werden. Anschließend wird eine Rückmeldung über den Erfolg gegeben (siehe Abbildung 3-27). Diese Art der Interaktion ermöglicht es der *Operating Force*, sofern nötig, das Verhalten einer *Supporting ACU* zu übersteuern, da diese Anweisungen stets höher priorisieren muss als von ihr selbst initiierte Handlungen. Die Nachrichtenfolge *instruct – inform:done* deutet auf eine erfolgreiche Durchführung des Dialogs hin.

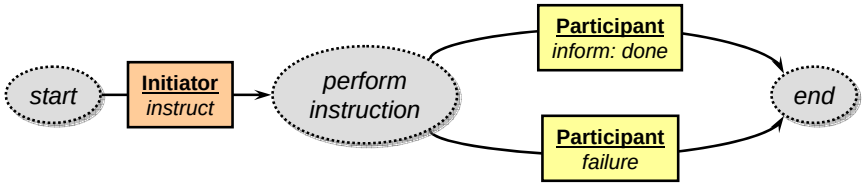


Abbildung 3-27: Instruct-Dialog

Solange einer der betrachteten Dialoge zwischen zwei einzelnen Agenten als Initiator und Teilnehmer stattfindet, werden Nachrichten jeweils an den anderen Agenten adressiert (vgl. Abbildung 3-28).

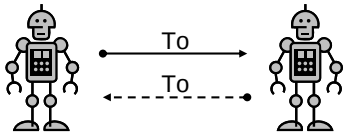


Abbildung 3-28: Nachrichtenfluss zwischen zwei Agenten

Betrachtet man jedoch nicht nur bilaterale Kommunikation, sondern Kommunikation mit und innerhalb Teams, so muss definiert werden, wer im Laufe einer Interaktion welche Nachrichten an wen sendet. Hier kann zunächst den Fall betrachtet werden, dass ein Agent außerhalb eines Teams mit einem Team kommuniziert, wie er zum Beispiel bei der Übermittlung eines Teilauftrags oder einer Aufgabe von der *Operating Force* an ein Team aus *Supporting ACUs* auftritt. Hier scheint eine Reduktion auf die obige Konfiguration nahe liegend, die herbeigeführt werden kann, indem das Team durch seinen Teamsprecher vertreten wird und dieser mit dem außenstehenden Agenten kommuniziert. Diese Vereinfachung bringt jedoch zwei Implikationen mit sich: Zum Einen muss der Teamsprecher Informationen, die er von außen erhält und die das gesamte Team betreffen, an die anderen Teammitglieder weiterleiten und zum Anderen bewirkt ein Ausfall des Teamsprechers den Abbruch sämtlicher laufenden Dialoge, ohne dass ein anderes Teammitglied in der Lage wäre, diese weiterzuführen. Diese Problematik wird entschärft, wenn der Nachrichtenfluss wie in Abbildung 3-29 dargestellt gestaltet wird. Hierbei sendet der Agent außerhalb des Teams seine Nachrichten grundsätzlich an alle Teammitglieder und erhält Antworten vom Teamsprecher, ohne dass für ihn transparent sein muss, wer der Teamsprecher ist. Der Teamsprecher wiederum schickt alle Nachrichten an den außenstehenden Agenten und zusätzlich in Kopie an seine Teamkollegen (vgl. Adresszeilen-Präfixe „To“ bzw. „Cc“ im E-Mail-Verkehr), so dass diese in der Lage sind, Dialoge mitzuverfolgen und unter Umständen ein neuer Teamsprecher diese weiterführen kann.

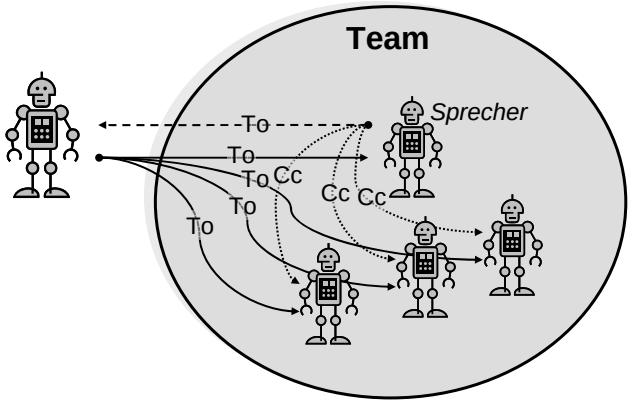


Abbildung 3-29: Nachrichtenfluss zwischen einem Agenten und einem Team

Neben Kommunikation mit einem Team gibt es auch Kommunikation innerhalb eines Teams. Diese tritt zum Beispiel auf, wenn ein Teammitglied anderen vorschlägt, die Rolle des Teamsprechers zu übernehmen. Hierbei wird der Nachrichtenfluss entsprechend Abbildung 3-30 gestaltet: Der Initiator eines Dialogs sendet Nachrichten an alle Teammitglieder, spricht also das Team als Ganzes an. Darauf reagiert jedes Teammitglied individuell, d.h. schickt seine Nachrichten an den Initiator und wählt dabei den Typ der Nachricht so, wie es ihm adäquat erscheint. So mag es für ein Teammitglied akzeptabel sein, dass der Initiator vorschlägt, die Rolle des Teamsprechers zu übernehmen, während ein anderes Teammitglied diesen Vorschlag ablehnt. Um es anderen Teammitgliedern zu ermöglichen, bereits gesendete Nachrichten in die Entscheidungsfindung miteinzubeziehen, werden die Nachrichten der Teammitglieder nicht nur an den Initiator, sondern zusätzlich in Kopie an die anderen Teammitglieder geschickt.

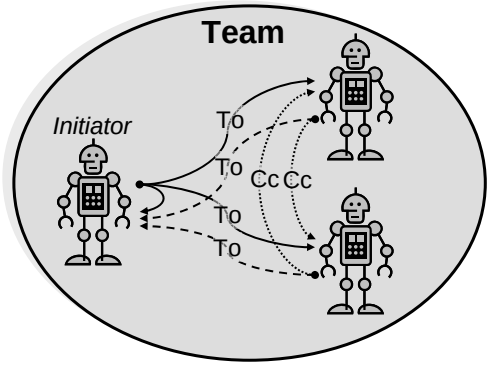


Abbildung 3-30: Nachrichtenfluss innerhalb eines Teams

Derartige Nachrichtenflüsse müssen im verwendeten Nachrichtenprotokoll realisiert werden können. Das im Rahmen dieser Arbeit verwendete ist in Tabelle 3-1 dargestellt und basiert auf dem in der FIPA ACL vorgeschlagenen Protokoll [FIPA Nachrichtenformat, 2002]. Zunächst werden hier die Teilnehmer am Dialog spezifiziert, wobei zwischen Sender und sendendem Agenten sowie zwischen Empfänger und empfangenden Agenten unterschieden wird. Für den Fall der Antwort eines Teams auf die Anfrage eines außenstehenden Agenten wäre zum Beispiel das Team der Sender, der außenstehende Agent der Empfänger, der Teamsprecher der sendende Agent und sämtliche Teammitglieder sowie der außenstehende Agent empfangende Agenten. Weiterhin besteht eine Nachricht aus dem Typ der Nachricht, der aus einer Menge möglicher Werte entsprechend dem aktuellen Zustand des Dialogs ausgewählt werden kann, einem Inhalt, wie zum Beispiel einer Aufgabe, und verschiedenen Parametern, die der Organisation des Ablaufs von Dialogen dienen.

Parameter	***Erläuterung***
Teilnehmer am Dialog	
`sender`	Identität des Senders der Nachricht (Agent oder Team)
`receiver`	Identität des Empfängers der Nachricht (Agent oder Team)
`sending-agent`	Identität des Agenten, der die Nachricht tatsächlich sendet
`receiving-agent`	Identität des/der Agenten, an den/die die Nachricht gesendet wird
Typ der Nachricht	
`performative`	Typ der Nachricht (*accept-proposal, agree, cancel, failure, inform, instruct, propose, query, refuse, reject-proposal, request, subscribe*)
Nachrichteninhalt	
`content`	Inhalt der Nachricht oder Objekt einer Aktion (Bedeutung wird vom Empfänger interpretiert)
Organisation des Ablaufs des Dialogs	
`protocol`	Verwendetes Interaktionsprotokoll / Dialog-Typ
`conversation-id`	Global eindeutige Kennung eines Dialogs, welche der Identifikation von Nachrichten dient, die zu diesem Dialog gehören
`reply-by`	Zeitpunkt, bis zu dem der Sender spätestens eine Nachricht vom Empfänger erwartet
`sent-at`	Zeitpunkt, zu dem die Nachricht gesendet wird

Tabelle 3-1: Nachrichtenformat (vgl. [FIPA Nachrichtenformat, 2002])

4 Modellierung des a-priori Wissens

Nachdem im vorhergehenden Kapitel künstliche Kognition und Kooperation als Hauptbestandteile der Zusammenarbeit mehrerer *Supporting ACUs* auf wissensbasierter Verhaltensebene detailliert betrachtet wurden, stellt sich nun die Frage, wie die charakterisierten kooperativen Fähigkeiten auf Grundlage der Theorie des Kognitiven Prozesses in ein lauffähiges Softwaresystem umgesetzt werden können.

Grundsätzlich bedarf die Entwicklung eines kognitiven Systems auf Basis des Kognitiven Prozesses ebenso wie die Entwicklung eines jeden technischen Systems einer strukturierten Vorgehensweise. Im Umfeld des Kognitiven Prozesses sind hierbei Entwurfsmethoden aus der Softwaretechnik, der Agententechnologie und dem *Knowledge Engineering* relevant [Putzer, 2004]. Da in all diesen Disziplinen im Prinzip entsprechend den Entwicklungsphasen in der klassischen Softwaretechnik vorgegangen wird, wird zunächst ein Überblick über diese Vorgehensweise gegeben. Da der Kognitive Prozess einen wissensbasierten Ansatz verfolgt, d.h. Wissen von dessen Verarbeitung trennt, werden außerdem einzelne Phasen im Hinblick auf die Entwicklung wissensbasierter Systeme weiter detailliert (Abschnitt 4.1).

Die Nutzung des Kognitiven Prozesses für die Entwicklung eines kognitiven Systems beeinflusst vor allem die in der Entwurfsphase verwendeten Methoden, wobei insbesondere die KP-Methode relevant ist (siehe Abschnitt 4.2). Sie unterstützt die Modellierung des a-priori Wissens im Kognitiven Prozess und wird in Abschnitt 4.3 auf die hier vorliegende Problemstellung angewendet, so dass schließlich die für kooperative Fähigkeiten von *Supporting ACUs* notwendigen Modelle des a-priori Wissens vorliegen.

4.1 Vorgehensweise bei der Softwareentwicklung

4.1.1 Softwaretechnik

Wie oben erwähnt wird Software im Allgemeinen in mehreren Phasen, die in der klassischen Softwaretechnik definiert werden, entwickelt, und zwar auch im Bereich der Agententechnologie und des *Knowledge Engineering* (vgl. [Buchanan et al., 1983] [Karbach & Linster, 1990][Puppe, 1991][Görz, 1993][Wooldridge & Jennings, 1998]). Diese Phasen sind in Abbildung 4-1 links entsprechend [Balzert, 2000] dargestellt.

Während in der (Projekt-)*Planungsphase* organisatorische und finanzielle Rahmenbedingungen abgesteckt werden, wird das zu entwickelnde System in der *Definitionsphase* inhaltlich beschrieben, indem Anforderungen an die Funktionalität definiert werden. Es folgt die Erstellung einer Softwarearchitektur in der *Entwurfsphase*. Je nach angewendeter Entwurfsmethode wird diese Architektur zum Beispiel durch eine Menge von Klassen (Objektorientierter Entwurf) oder funktionaler Module (modularer und strukturierter Entwurf) sowie Anforderungen an diese einzelnen Elemente beschrieben. Ergebnis der sich anschließenden *Implementierungsphase* ist ein im Rechner lauffähiges Programm, wobei hier jede beliebige Programmiersprache bzw. Architektur verwendet werden kann, die dem Problem angemessen ist. In der *Abnahme- und Einführungsphase* wird das implementierte und getestete System an den Nutzer übergeben. Den Abschluss der Systementwicklung bildet die *Wartungs- und Pflegephase*, welche der Stabilisie-

rung, Korrektur, Optimierung, Leistungsverbesserung, Anpassung, Änderung und Erweiterung des Systems dient. [Balzert, 2000]

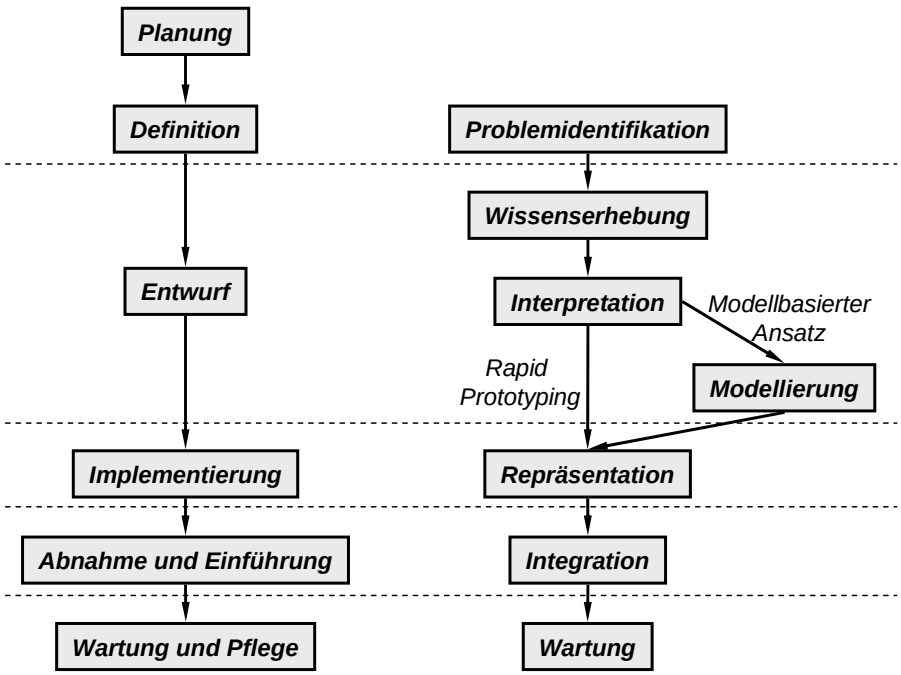


Abbildung 4-1: Entwicklungsphasen in der Softwaretechnik (links, [Balzert, 2000]) und im Knowledge Engineering (rechts, [Buchanan et al., 1983][Karbach & Linster, 1990][Görz, 1993])

Insbesondere wenn es um die Entwicklung hoch komplexer und integrierter Systeme wie beispielsweise Flugzeuge geht, reicht die Betrachtung des dargestellten Phasenmodells für Software nicht aus. Hier muss die Entwicklung eines aus Hard- und Software bestehenden Gesamtsystems berücksichtigt werden und zwar insbesondere im Hinblick auf die Anforderungen, die Zulassungsbehörden an die Systementwicklung stellen (z.B. Dokumentation). In diesem Umfeld gibt es elaborierte Vorgehensmodelle wie beispielsweise das V-Modell [V-Modell XT, 2008] oder speziell in der Luftfahrt die *Aerospace Recommended Practice* (ARP) 4754 [SAE, 1996].

Prinzipiell unterliegt die Entwicklung kognitiver Automation, die im Bereich der UAV Flugführung gegebenenfalls auch sicherheitskritische Funktionen umfassen kann, dabei den gleichen Randbedingungen wie alle anderen Systemfunktionalitäten. Daher kann zunächst davon ausgegangen werden, dass in zukünftigen industriellen Entwicklungs- und Zulassungsprozessen etablierte Verfahren eingesetzt werden, die jedoch unter Umständen an bisher nicht verwendete Methoden angepasst werden müssen. Da kognitive Automation zudem das Potenzial besitzt, Hintergrundwissen aus der Anforderungsdefinition zur Laufzeit zu berücksichtigen, ist die Weiterentwicklung von Systementwicklungsprozessen sinnvoll bzw. notwendig (vgl. [Jarasch & Schulte, 2008]).

4.1.2 Knowledge Engineering

Im Bereich des *Knowledge Engineering*, welches sich hauptsächlich mit der Entwicklung von wissensbasierten Expertensystemen (z.B. medizinische Diagnosesysteme) befasst, unterteilen [Karbach & Linster, 1990] die Entwurfsphase weiter in die Wissenserhebung, die Interpretation des erhobenen Wissens durch den Wissensingenieur und je nach gewähltem Ansatz die explizite Modellierung des Wissens (vgl. Abbildung 4-1, rechts).

4.1.2.1 Wissenserhebung

Bevor Wissen als Grundlage eines wissensbasierten Systems genutzt werden kann, muss es für den Systementwickler verfügbar sein. Während bei klassischen Expertensystemen das Wissen von Domänenexperten möglichst detailgetreu nachgebildet werden soll, sind künstliche kognitive Systeme nicht per se auf die Nachbildung von Expertenwissen beschränkt. In der Regel wird dennoch auf Wissen von Experten, die gleiche oder ähnliche Fragestellungen bearbeiten, zurückgegriffen, um zumindest einen Ausgangspunkt für die weitere Entwicklung zu haben.

Nach [Puppe, 1991] gibt es drei grundsätzlich verschiedene Herangehensweisen bei der Erhebung von Wissen, nämlich *indirekten Wissenserwerb* bei dem ein Wissensingenieur mit Hilfe verschiedener Methoden wie zum Beispiel Expertenbefragung oder Literaturstudium Wissen erhebt und anschließend in ein System umsetzt, *direkten Wissenserwerb*, bei dem ein Experte selbst mit Hilfe geeigneter Werkzeuge sein Wissen formalisiert und *automatischen Wissenserwerb*, bei dem ein technisches, wissensbasiertes System Wissen selbst aus Quellen wie zum Beispiel Datenbanken extrahiert.

Bei der Entwicklung komplexer wissensbasierter Systeme wird zumeist auf indirekten Wissenserwerb zurückgegriffen, d.h. es werden Interviewtechniken eingesetzt oder Experten bei der Durchführung von Aufgaben beobachtet (vgl. [Karbach & Linster, 1990] [Puppe, 1991][Schreiber et al., 2000]). Auch die Analyse von (Fach-)Literatur fällt in diese Kategorie. Der Einsatz von Werkzeugen für direkten und Methoden für automatischen Wissenserwerb (siehe z.B. [Mitchell, 1997]) ist hier allerdings sehr erstrebenswert und Gegenstand vielfältiger Forschungsarbeiten.

4.1.2.2 Wissensinterpretation

An die Phase der Wissenserhebung schließt sich die Interpretation des dort erhobenen Wissens durch den Wissensingenieur an, der versucht, dieses Wissen in ein mentales konzeptuelles Modell zu überführen. Während bei der Vorgehensweise des *Rapid Prototyping* dieses mentale Modell unmittelbar in eine Rechnerrepräsentation umgesetzt wird, wird beim modellbasierten Entwurf die Phase der Wissensmodellierung eingeschoben (siehe nächster Abschnitt), welche das mentale Modell des Wissens explizit macht. [Karbach & Linster, 1990]

4.1.2.3 Wissensmodellierung

Die wohl bekannteste Methode zur Modellierung von Wissen ist CommonKADS [Schreiber et al., 2000], welche eine Suite von Modellstrukturen zur Verfügung stellt, die die Entwicklung eines wissensbasierten Systems unterstützen (vgl. Abbildung 4-2).

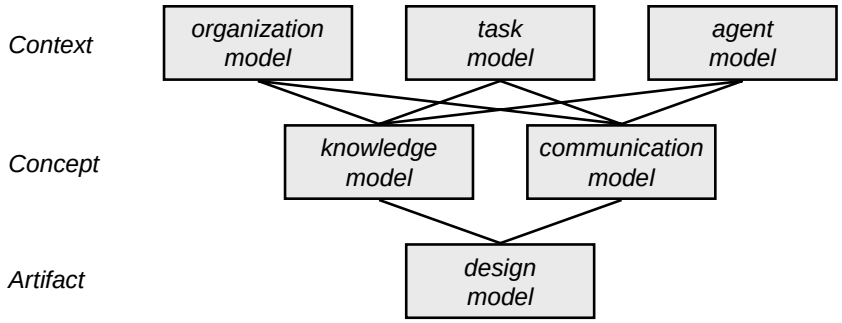


Abbildung 4-2: CommonKADS Modellstrukturen [Schreiber et al., 2000]

Die Organisations-, Aufgaben- und Agentenmodellstruktur unterstützen vor allem die Phase der Problemidentifikation, indem die Struktur der Organisation, in der ein System

zum Einsatz kommen soll, die relevanten Teilaufgaben des Geschäftsprozesses und die Bearbeiter der Aufgaben beschrieben werden. Die Wissens- und Kommunikationsmodellstruktur beschreiben die Art und Struktur des für die Durchführung von Aufgaben benötigten Wissens sowie die Struktur der entsprechenden Kommunikation und dienen damit der eigentlichen Wissensmodellierung. Die Entwurfsmodellstruktur schließlich zielt auf die Beschreibung der Implementierung ab [Schreiber et al., 2000]. Bei der Anwendung dieser Modellstrukturen wird dabei ein spiralförmiges Entwicklungsmodell zugrunde gelegt, welches ermöglicht, die Modelle iterativ zu entwickeln und aktuelle Erkenntnisse zu integrieren.

Im Hinblick auf die Umsetzung wissensbasierten Verhaltens werden die Anforderungen an hierfür geeignetes Wissen von CommonKADS nicht erfüllt. Insbesondere werden Handlungsziele nicht explizit modelliert, so dass sich bei Anwendung dieser Vorgehensweise Wissenslücken ergeben. Daher wird im Rahmen der vorliegenden Arbeit die KP-Methode zur Wissensmodellierung verwendet, welche die Struktur des Kognitiven Prozesses berücksichtigt. Sie wird in Abschnitt 4.2 beschrieben.

4.1.3 Einordnung

Die einzelnen Phasen der Systementwicklung wurden auch in dieser Arbeit durchlaufen. So stellt Kapitel 2 das Ergebnis der *Definitionsphase* dar, die hier in einer vereinfachten Arbeitssystemanalyse bestand. Üblicherweise sind dabei systematisch und detailliert die folgenden Punkte zu adressieren:

- **Identifikation der beteiligten Arbeitssysteme**
- **Ein-/Ausgänge des/der Arbeitsprozesse(s)**
  - Was ist das Arbeitsziel?
  - Unter welchen Rahmenbedingungen soll das Arbeitsziel erreicht werden?
  - Wie können das Arbeitsergebnis und der Arbeitsfortschritt beschrieben werden?
- **Verknüpfung der beteiligten Arbeitssysteme**
  - Welche Informationsflüsse finden zwischen den Arbeitssystemen statt?
- **Identifikation der Arbeitssystem-Komponenten**
  - Mensch(en)
  - Arbeitsmittel
  - *Operating ACUs*
  - *Supporting ACUs*
- **Beschreibung von Anforderungen an die Arbeitssystem-Komponenten**
  - Fähigkeiten und Aufgaben einzelner Arbeitssystem-Komponenten
  - Interaktionen zwischen den Arbeitssystem-Komponenten

Es folgt eine Darstellung des im Rahmen von Kooperation mehrerer *Supporting ACUs* relevanten Wissens als Ergebnis der *Wissenserhebung* und *Wissensinterpretation* in Kapitel 3. Abschnitt 4.3 dieses Kapitels wird die explizite *Modellierung* dieses Wissens auf Basis der KP-Methode beschreiben, bevor in Kapitel 5 die *Implementierung* der hier identifizierten Modelle und in Kapitel 6 die *Erprobung* des realisierten Prototyps vorgestellt wird.

4.2 Die KP-Methode

Die so genannte *KP-Methode* („Kognitive Prozess"-Methode) wird von [Putzer, 2004] für die Identifikation von Modellen des a-priori Wissens des Kognitiven Prozesses (siehe Abschnitt 3.2.3.2) vorgeschlagen. Sie unterstützt die Modellierung im Sinne eines objektorientierten Entwurfs (vgl. [Balzert, 2000]). Durch die Verwendung von mentalen Konzepten nähert sie das Verarbeitungsmodell der Programmierung an das Denkmodell des Menschen an und ermöglicht so einen unmittelbaren Übergang zwischen der Interpretation erhobenen Wissens und der Implementierung dieses Wissens. Unter Umständen kann bereits in der Phase der Wissenserhebung eine Vorstrukturierung des Wissens erreicht werden, indem Experten in Interviews durch geeignete Fragetechniken dabei unterstützt werden, ihr Wissen entsprechend des Denkmodells des Kognitiven Prozesses zu strukturieren und zu verbalisieren. Diese Vorgehensweise wurde zum Beispiel im Rahmen der Analyse einer Hubschrauber-Schwarmoperation angewendet [Schmidt, 2007].

Konkret wird bei der Anwendung der KP-Methode zunächst ein statisches Modell erstellt, d.h. die Modelle des a-priori Wissens werden benannt und charakterisiert, also mit Attributen versehen. Dies entspricht dem Aufbau einer Ontologie, wobei hier folgende Definition zugrunde gelegt wird:

> *An ontology is a formal explicit description of concepts in a domain of discourse (**classes** (sometimes called **concepts**)), properties of each concept describing various features and attributes of the concept (**slots** (sometimes called **roles** or **properties**)), and restrictions on slots (**facets** (sometimes called **role restrictions**)). [Noy & McGuinness, 2001]*

Anschließend wird das dynamische Modell generiert, d.h. das Verhalten der Modelle des a-priori Wissens beschrieben. Im Einzelnen werden dabei die folgenden Schritte bearbeitet [Putzer, 2004]:

1. *Modellierung der Wünsche* – Da Ziele die Grundlage für wissensbasiertes Verhalten darstellen, werden im ersten Modellierungsschritt Wünsche, d.h. potentielle Ziele eines Systems, modelliert. Diese Modellierung kann hierarchisch und unter Verwendung einer Basisklasse für alle Wünsche erfolgen.
2. *Modellierung der Handlungsalternativen* – In diesem Schritt müssen Handlungsalternativen so modelliert werden, dass sämtliche Wünsche erreicht werden können. Auch hier kann eine Basisklasse für alle Handlungsalternativen verwendet werden, deren Gestaltung wesentlich von der eingesetzten Strategie zur Planung und Problemlösung abhängt.
3. *Modellierung der Anweisungsmodelle* – Aufgabe des Transformators Ausführung ist die Umsetzung eines Plans in Anweisungen. Daher sind zunächst die Handlungsalternativen dahingehend zu analysieren, aus welchen elementaren Anweisungen sie sich zusammensetzen und diese in Form von Anweisungsmodellen zu modellieren. Dabei ist die Komplexität der Anweisungsmodelle stark vom Detaillierungsgrad der Handlungsalternativen und der Art der erlaubten Anweisungen abhängig.
4. *Modellierung der Umweltmodelle* – Alle Objekte, welche von Wünschen, Handlungsalternativen oder Anweisungsmodellen genutzt werden, müssen in Umweltmodelle abgebildet werden. Wenn all diese Modelle identifiziert sind, werden sie in Kompositions- und Vererbungsstrukturen angeordnet und schließlich durch die Angabe von Attributen vervollständigt.

5. *Erstellung des dynamischen Modells* – Nach der Erstellung des statischen Modells in den ersten vier Modellierungsschritten folgt das dynamische Modell. Dabei wird zum Beispiel auf der Basis von Anwendungsfällen für jedes Modell des a-priori Wissens Verhalten festgelegt, welches die Erzeugung und Zerstörung von Instanzen und die Belegung von Attributen mit Werten umfasst. Dieses Verhalten hängt im Allgemeinen von anderen Modellen ab, so dass in diesem Modellierungsschritt sämtliche Interaktionen zwischen verschiedenen Modellen des a-priori Wissens definiert werden.

Bei der Entwicklung des statischen Modells, d.h. der Deklaration der vorhandenen Modelle des a-priori Wissens, sind ähnliche Fragestellungen relevant wie sie im Umfeld der Entwicklung von Ontologien in [Noy & McGuinness, 2001] adressiert werden, so dass die dort gegebenen Empfehlungen für die Gestaltung von Klassenhierarchien, Einführung neuer Klassen, Parametrisierungen von Klassen und ähnlichem direkt angewendet werden können.

Als schwieriger erweist sich die Gestaltung der Grenzen zwischen expliziter und impliziter sowie regel- und wissensbasierter Modellierung. Der erste Aspekt bezieht sich dabei vor allem auf die Modellierungstiefe und basiert auf der Annahme, dass die Integrität eines wissensbasierten Systems mit der Menge an explizit repräsentiertem Wissen zunimmt. Zur Verdeutlichung soll folgendes Beispiel dienen: Ein Umweltmodell, welches ein Wettergebiet im Luftraum beschreibt, könnte als Attribut „gefährlich“ (Wert: ja/nein) haben, welches alle weiteren Implikationen von Gefahr durch Wetter subsumiert (implizite Modellierung). Andererseits könnte Gefährdung als eigenes Konzept, beschrieben durch Attribute wie zum Beispiel „Art der Gefahr“ mit Werten wie „sicherheitsgefährdend“, „komforteinschränkend“ etc. und „Relevanz für das eigene Luftfahrzeug“ hinterlegt werden (explizite Modellierung), was ein tiefer gehendes Verständnis von „Gefahr“ bedeuten würde. Da diese Argumentationslinie beliebig fortgesetzt werden kann, muss die Trennlinie im Einzelfall auf Basis konkreter Anforderungen an das System gezogen werden und ist oft nicht zuletzt von den verfügbaren zeitlichen und finanziellen Mitteln für die Entwicklung abhängig.

Ähnlich verhält es sich mit der Abwägung, ob ein Aspekt auf wissens-, regel- oder sogar fertigkeitsbasierter Verhaltensebene modelliert werden soll. Hier kollidiert die Forderung nach Erklärungsfähigkeit, Flexibilität und Integrität eines Systems auch in unbekannten Situationen mit Anforderungen an Schnelligkeit und Performanz. So könnte zum Beispiel das Ausweichen vor einem Fremdflugzeug auf allen drei Verhaltensebenen modelliert werden. Die fertigkeitsbasierte Variante würde hierbei ein Anweisungsmodell vorsehen, das bei Vorliegen von Eingangsdaten, die auf eine Kollision hinweisen, umgehend Anweisungen zur Ausführung erstellen würde, die von den entsprechenden Effektoren auch unmittelbar ausgeführt würden. Bei einer regelbasierten Modellierung würde bei Vorliegen einer bekannten Situationskonfiguration in den Eingangsdaten eine Handlungsalternative instanziiert, die Ausweichen als eine Aufgabe repräsentieren würde, die von einem oder mehreren Anweisungsmodellen in Aktionen umgesetzt würde. Auf wissensbasierter Verhaltensebene würde zunächst ein Verständnis der Situation erzeugt und bei Vorliegen einer gefährlichen Konfiguration ein Ziel aktiviert, das das Vermeiden der Gefahr darstellt. Außerdem wären entsprechende Handlungsalternativen und Anweisungsmodelle nötig, um dieses Ziel erreichen zu können. Hierbei ist wiederum im Kontext der konkreten Anforderungen an die Applikation im Einzelfall abzuwägen, welche Forderung (Flexibilität und Erklärungsfähigkeit vs. Schnelligkeit) wichtiger ist. Dabei spielen auch wirtschaftliche Überlegungen eine Rolle, da der Entwicklungsaufwand für die Umsetzung der verschiedenen Verhaltensebenen unter-

schiedlich groß ist. Je nach Anforderungen an das zu entwickelnde System ist es gegebenenfalls notwendig und sinnvoll, das gewünschte Verhalten auf mehreren Verhaltensebenen parallel zu implementieren, um die Vorteile aller adressierten Ebenen nutzen zu können, auch wenn dies natürlich einen erhöhten Implementierungsaufwand bedeutet. Eine gewisse Redundanz kann insbesondere auch dadurch geschaffen werden, dass Verhalten auf verschiedenen Ebenen auf Basis unterschiedlicher Automationsparadigmen hinterlegt wird (z.B. konventionell und kognitiv, vgl. Abschnitt 2.1). In diesem Zusammenhang stellt zum Beispiel die wissensbasierte Reflexion über regelbasiertes Verhalten auf einer Art Meta-Ebene (vgl. [Frey, 2005]) einen viel versprechenden Ansatz dar, um die Vorteile beider Verhaltensebenen zu vereinen (vgl. Abschnitt 3.2.2).

Wie eingangs erwähnt nähert die KP-Methode durch die Einführung mentaler Begriffe das Programmiermodell an das Denkmodell des Menschen an. Dies gilt insbesondere für den Fall, wenn für die Implementierung des a-priori Wissens CPL (*Cognitive Programming Language*) verwendet wird, da hiermit das mit der KP-Methode erstellte Wissensmodell ohne weiteren Transformationsschritt direkt in eine Implementierung mündet. CPL wurde auch für die Implementierung im Rahmen dieser Arbeit verwendet und wird in Abschnitt 5.1.2 erläutert.

4.3 Modelle des a-priori Wissens

Dieser Abschnitt identifiziert nun mit Hilfe der KP-Methode auf Basis des in Kapitel 3 beschriebenen Wissens die Wissensmodelle, die nötig sind, um Kooperation auf wissensbasierter Verhaltensebene und speziell gemäß der Theorie des Kognitiven Prozesses im Rechner zu realisieren. Die hierdurch gewonnenen Modelle bilden dabei die Grundlage für die Implementierung eines Teams aus *Supporting ACUs*, die in Kapitel 5 vorgestellt wird. Neben den betrachteten kooperativen Fähigkeiten benötigen ACUs auch domänenspezifische Fähigkeiten, um Aufgabenstellungen verstehen und tatsächlich bearbeiten zu können. Dieser Aspekt wird im Weiteren soweit nötig adressiert, bildet jedoch nicht den Schwerpunkt der vorliegenden Arbeit.

Der Übersichtlichkeit halber werden im Folgenden zunächst die Wünsche identifiziert und dann die Schritte der KP-Methode ausgehend von jedem Wunsch durchlaufen, d.h. es werden Handlungsalternativen benannt, die geeignet sind, ein Ziel zu erreichen und Umweltmodelle definiert, die für die Instanziierung nötig sind. Anschließend werden Relevanz- und Erfülltheitskriterien für die Wünsche als dynamische Komponente des Modells erläutert. Danach werden die identifizierten Handlungsalternativen in ähnlicher Weise nacheinander detailliert, d.h. für die Durchführung notwendige Anweisungsmodelle und nötige Umweltmodelle benannt und das Verhalten der Handlungsalternativen charakterisiert. Das Verhalten der Umwelt- und Anweisungsmodelle, die bis dahin identifiziert wurden, ergibt sich direkt aus dem Kontext, so dass darauf nicht im Detail eingegangen wird.

4.3.1 Wünsche

Die für Kooperation wesentlichen Fundamentalziele sind in Abbildung 4-3 dargestellt. Sie lassen sich zur besseren Strukturierung und Übersichtlichkeit für den Designer auf Konzeptebene in übergeordnete Kategorien einordnen, welche einen fundamentaleren Kontext darstellen und es ermöglichen, Anforderungen an Zielsysteme zu erfüllen (vgl. [Eisenführ & Weber, 2003], Abschnitt 3.2.2.1). Die sich so ergebende Hierarchie wird allerdings bei dem in dieser Arbeit gewählten Implementierungsansatz nicht weiter berücksichtigt, sondern es werden lediglich die Ziele weiter betrachtet, denen keine weiteren Ziele untergeordnet sind. Um diese identifizierten Fundamentalziele erreichen zu

können, ist je nach Ziel eine Reihe weiterer Instrumentalziele notwendig, die allerdings zum Großteil applikationsspezifisch sind, so dass an dieser Stelle nicht weiter darauf eingegangen wird. So bedarf zum Beispiel die Durchführung einer konkreten Verpflichtung „Unterdrücke eine gegnerische Bedrohung“ Ziele wie zum Beispiel „Sei an einer Angriffsposition“.

Die in Abbildung 4-3 dargestellten Kategorien orientieren sich an der Einteilung von Zielen für ein Pilotenassistenzsystem nach [Walsdorf, 2002], der als Teilbereiche *(1) Wahrung der Sicherheit*, *(2) Erfüllung des Missionsauftrages* und *(3) Unterstützung der Besatzung bei deren Absichten* nennt.

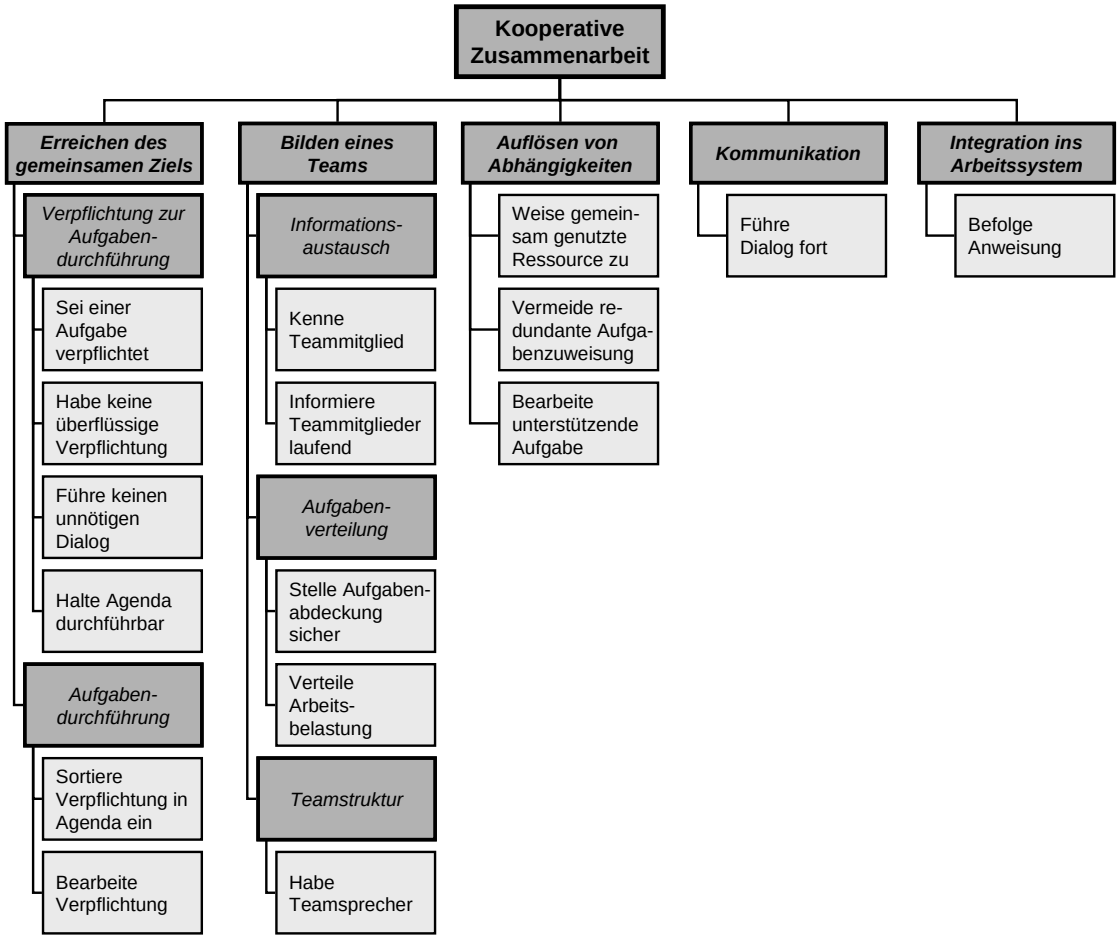


Abbildung 4-3: Hierarchie von Fundamentalzielen im Hinblick auf Kooperation in einem Team aus Supporting ACUs

Diese werden im Hinblick auf die Kooperation mehrerer *Supporting ACUs* und später gegebenenfalls *Operating ACUs* und Menschen erweitert. Da Ziele, die der Kategorie „Wahrung der Sicherheit“ zugeordnet werden können, überwiegend anwendungsspezifisch sind, werden diese hier nicht weiter detailliert, sind aber im ausgeführten System implementiert. Die Erfüllung des Missionsauftrags erstreckt sich hier auf das Erreichen des gemeinsamen Missionsauftrags, d.h. der gemeinsamen Zielsetzung im Team, während die Kategorie „Unterstützung der Besatzung bei ihren Absichten“ im Hinblick auf verschiedene Aspekte von Kooperation, die über die Unterstützung eines Piloten durch ein Assistenzsystem hinausgehen, ausgewietet wird, nämlich das Bilden eines Teams, das Auflösen von Abhängigkeiten zwischen Teammitgliedern und die Kommunikation im Team. Schließlich wird auch der Integration des zu entwickelnden Teams aus *Supporting ACUs* als Arbeitsmittel in ein Arbeitssystem Rechnung getragen, indem ein Wunsch formuliert wird, der die hierarchische Anordnung von *Operating Force* und

Operation-Supporting Means widerspiegelt und demnach Anweisungen vom Operateur befolgt werden müssen.

Gemeinsame Ziele eines Teams können durch adäquates Eingehen und Aufheben von Verpflichtungen sowie die tatsächliche Durchführung der Aufgaben erreicht werden, zu denen Teammitglieder verpflichtet sind. Hierzu muss es überhaupt Verpflichtungen geben, wenn zum Beispiel ein Team von einem Operateur dazu angewiesen wird, einen Auftrag durchzuführen. Ferner sollen Verpflichtungen sowohl einzelner Agenten als auch von Teams im Sinne der in Abschnitt 3.3.3 vorgestellten Konventionen nicht beliebig lange aufrechterhalten, sondern aufgegeben werden, sofern diese erfüllt, unerreichbar oder nicht mehr relevant sind. Hat ein Teammitglied die Bearbeitung einer Aufgabe von einem anderen angefragt, so führt dies im Allgemeinen dazu, dass der andere Agent eine Verpflichtung eingeht. Ist nun der Grund für diese Anfrage nicht mehr gegeben, soll auch diese Verpflichtung wegen fehlender Relevanz aufgegeben werden, was dem Abbruch des entsprechenden Dialogs gleich kommt. Da in der Regel nicht alle Verpflichtungen eines Agenten gleichzeitig bearbeitet werden können, wird hier der Begriff der Agenda eingeführt, die eine Reihenfolge repräsentiert, in der die Verpflichtungen bearbeitet werden sollen. Im Sinne der oben genannten Konventionen sollen nicht nur einzelne Verpflichtungen, sondern auch die Agenda als ganze durchführbar und erreichbar sein. Im Hinblick auf die tatsächliche Durchführung der Aufgaben, zu denen ein Teammitglied verpflichtet ist, muss jede Verpflichtung in die Agenda eingeordnet und zum entsprechenden Zeitpunkt auch tatsächlich bearbeitet werden.

Die Zusammenarbeit als Team lässt sich im Hinblick auf Informationsaustausch, Aufgabenverteilung und die Abbildung einer entsprechenden Teamstruktur detaillieren. So sollen sich die Mitglieder eines Teams in Anlehnung an die von Billings abgeleitete Forderung „Alle Teammitglieder müssen die Absichten der anderen Teammitglieder kennen“ (siehe Abschnitt 3.3.2) gegenseitig kennen, wobei hier insbesondere Ressourcen, Fähigkeiten und Verpflichtungen einzelner Teammitglieder betrachtet werden. Ebenso sollen sich die Teammitglieder entsprechend der Forderung „Alle Teammitglieder müssen angemessen informiert sein“ (siehe Abschnitt 3.3.2) und der in Abschnitt 3.3.3 skizzierten sozialen Konventionen gegenseitig mit Informationen versorgen. Hierunter fallen Ressourcen, Fähigkeiten und Verpflichtungen einzelner Teammitglieder ebenso wie gemeinsame Verpflichtungen von Teams. Die Aufgabenverteilung im Team soll so gestaltet werden, dass alle relevanten Aufgaben der gemeinsamen Zielsetzung bearbeitet werden (vgl. Abschnitt 3.3.3) und die Arbeitsbelastung im Team verteilt wird (vgl. Forderung „Alle Teammitglieder müssen aktiv eingebunden sein.“ in Abschnitt 3.3.2). Schließlich muss auch die angestrebte Teamstruktur, die hier einen Teamsprecher vorsieht (vgl. Abschnitt 3.3.2), umgesetzt werden, so dass ein entsprechender Wunsch formuliert wird.

Der Aspekt der Koordination mehrerer Teammitglieder wird durch Wünsche konkretisiert, welche das Auflösen der verschiedenen Abhängigkeiten, die betrachtet werden, adressieren (siehe Abschnitt 3.3.3). So wird die Abhängigkeit der gemeinsamen Nutzung einer sich nicht verbrauchenden Ressource durch den Wunsch repräsentiert, dass eine solche Ressource jeweils einem Teammitglied, das sie benötigt, zugewiesen ist. Die Gleichheitsbeziehung schlägt sich in dem Wunsch nieder, redundante Aufgabenzuweisung zu vermeiden und die Begünstigungsbeziehung wird aufgelöst, indem Aufgaben bearbeitet werden sollen, die andere Teammitglieder unterstützen.

Die Kommunikation im Team findet wie in Abschnitt 3.3.4 beschrieben auf Basis von Dialogen statt, welche in den entsprechenden Zuständen durch einen der beteiligten

Agenten fortgeführt werden müssen. Dies wird durch den Wunsch „Führe Dialog fort“ repräsentiert.

Jeder der hier identifizierten Wünsche muss im Sinne der KP-Methode weiter detailliert werden, und zwar durch die Beschreibung der Attribute, die dem Wunsch zugeordnet sind, sowie die Spezifikation seines Verhaltens. Dieses erstreckt sich hier auf die Charakterisierung der Relevanzkriterien, die zu einer Aktivierung eines Wunsches führen und der Erfülltheitskriterien, die eine Deaktivierung zur Folge haben. Da die Erfülltheitskriterien für Fundamentalziele im Allgemeinen eine Umkehrung der Relevanzkriterien darstellen, werden diese hier nicht explizit aufgeführt. Weiterhin werden die Umweltmodelle identifiziert, die für die Ausführung des Verhaltens nötig sind, und die Handlungsalternativen genannt, die geeignet sind, den jeweiligen Wunsch zu erreichen. Tabelle 4-1 gibt in diesem Sinn einen Überblick über die oben beschriebenen Wünsche.

Attribute	***Relevanz***	***Umweltmodelle***	***Handlungs-alternativen***
Sei einer Aufgabe verpflichtet			
Aufgabe, Verantwortlicher (Team, Akteur)	1. Team oder Akteur hat Auftrag angenommen, aber es besteht keine Verpflichtung des Teams zu den zugehörigen Aufgaben 2. Team oder Akteur hat Anfrage akzeptiert, eine Aufgabe durchzuführen, aber es besteht keine Verpflichtung des Teams zu dieser Aufgabe	Anfrage, Auftrag, Team, Akteur, Aufgabe, Verpflichtung	Gehe Verpflichtung ein
Habe keine überflüssige Verpflichtung			
Verpflichtung	1. Aufgabe, zu der Verpflichtung besteht, ist erfolgreich bearbeitet 2. Es besteht kein Grund für das Aufrechterhalten einer Verpflichtung (Anweisung, übergeordnete Verpflichtung des Teams, Dialog, Auftrag) 3. Verpflichtung kann nicht mehr erreicht werden 4. Eine Verpflichtung wird durch eine andere mit abgedeckt	Verpflichtung, Aufgabe, Akteur, Team, Anweisung, Dialog, Auftrag	Gib Verpflichtung auf
Führe keinen unnötigen Dialog			
Dialog	Dialog wurde begonnen, um ein Ziel zu erreichen, dieses ist allerdings nicht mehr relevant	Dialog	Brich Dialog ab (*cancel*)
Halte Agenda durchführbar			
--	Eine Verpflichtung, die Teil der Agenda ist, ist unerreichbar	Agenda, Verpflichtung	Gib Verpflichtung auf
Sortiere Verpflichtung in Agenda ein			
Verpflichtung	Eine Verpflichtung ist nicht in die Agenda eingeordnet	Agenda, Verpflichtung	Füge Verpflichtung ein
Bearbeite Verpflichtung			
Verpflichtung	Verpflichtung steht an erster Stelle in der Agenda	Agenda, Verpflichtung	*applikationsspezifisch je nach Art der Verpflichtung*
Kenne Teammitglied			
Teammitglied, Art der Information (Ressource, Fähigkeit, Verpflichtung)	Gewünschte Information über Teammitglied im selben Team wie der eigene Akteur ist nicht bekannt	Team, Akteur, Informationsspezifikation, Ressource, Fähigkeit, Verpflichtung	Frage Information einmalig an (*query*), Frage Information dauerhaft an (*subscribe*)

Tabelle 4-1: Detaillierung kooperationsrelevanter Wünsche

Informiere Teammitglieder laufend			
Art der Information (Ressource, Fähigkeit, Verpflichtung), Empfänger (Team, Operateur)	Aktueller Wert der referenzierten Information unterscheidet sich von der, die der Empfänger kennt	Akteur, Team, Teamsprecher, Informationsspezifikation, Ressource, Fähigkeit, Verpflichtung, Bekannte Information	Informiere (*inform*)
Stelle Aufgabenabdeckung sicher			
Aufgabe, Team	Team ist zu einer Aufgabe verpflichtet, aber kein einzelnes Teammitglied	Aufgabe, Team, Akteur, Verpflichtung	Gehe Verpflichtung ein, Frage Aufgabendurchführung an (*request*), Schlage Aufgabendurchführung vor (*propose*)
Verteile Arbeitsbelastung			
Akteur	Arbeitsbelastung eines Teammitglieds ist hoch, während die des eigenen Akteurs niedrig ist	Team, Akteur, Arbeitsbelastung	Schlage Aufgabenübernahme vor (*propose*)
Habe Teamsprecher			
Team	Team hat keinen Teamsprecher	Team, Teamsprecher	Schlage eigenen Akteur vor (*propose*)
Weise gemeinsam genutzte Ressource zu			
gemeinsam genutzte Ressource	Ressource wird von mehreren Akteuren benötigt	gemeinsam genutzte Ressource, Akteur	*Zuweisung erfolgt auf Basis gemeinsamer Normen*
Vermeide redundante Aufgabenbearbeitung			
Aufgabe, Akteur	Teammitglied und der eigene Akteur sind zur Durchführung einer Aufgabe verpflichtet, die nicht von mehreren bearbeitet werden muss/soll	Aufgabe, Akteur, Verpflichtung	Gib Verpflichtung auf, Schlage Aufgabenabbruch vor (*propose*), Frage Aufgabenabbruch an (*request*)
Bearbeite unterstützende Aufgabe			
Aufgabe, Akteur	Es gibt keine Verpflichtung zur Durchführung einer Aufgabe, die die Bearbeitung einer Aufgabe eines anderen Teammitglieds unterstützt	Team, Akteur, Aufgabe, Verpflichtung	Gehe Verpflichtung ein
Führe Dialog fort			
Dialog	1. Eigener Akteur ist verantwortlich für einen auf den aktuellen Zustand folgenden Zustandsübergang eines Dialogs 2. Team, dessen Teamsprecher der eigene Akteur ist, ist verantwortlich für einen auf den aktuellen Zustand folgenden Zustandsübergang eines Dialogs	Dialog, Akteur, Team, Teamsprecher, Zustandsübergang	Sende Nachricht
Befolge Anweisung			
Anweisung	Es ist eine Anweisung vorhanden, die noch nicht befolgt wurde und an ein Team gerichtet ist, dessen Teamsprecher der eigene Akteur ist	Anweisung, Team, Akteur, Teamsprecher	*applikationsspezifisch je nach Anweisung*

Tabelle 4-1 (fortgesetzt): Detaillierung kooperationsrelevanter Wünsche

4.3.2 Handlungsalternativen

Nach der Identifikation und Detaillierung der kooperationsrelevanten Wünsche im vorhergehenden Abschnitt werden nun die Handlungsalternativen erläutert, welche zur Verfügung stehen, um diese Wünsche zu erreichen. Dabei wird die Betrachtung auf diejenigen Handlungsalternativen beschränkt, die nicht applikationsspezifisch sind (siehe auch Tabelle 4-1, rechte Spalte). Diese erstrecken sich auf die Bereiche

- Eingehen, Aufgeben und Einordnen von Verpflichtungen (vgl. Abschnitt 3.3.3),
- Initiieren von Dialogen verschiedener Typen (vgl. Abschnitt 3.3.4), und
- Senden einer Nachricht.

Ein Teil dieser Handlungsalternativen bezieht sich auf verschiedene Möglichkeiten zur Verteilung von Aufgaben bzw. Verpflichtungen im Team (siehe Abbildung 4-4).

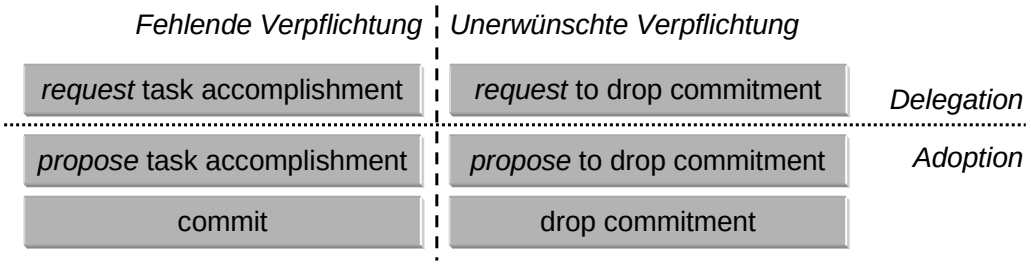


Abbildung 4-4: Handlungsalternativen im Hinblick auf Verteilung von Verpflichtungen im Team

Hierbei gibt es prinzipiell zwei Möglichkeiten, eine bestimmte Konfiguration zu erlangen, und zwar sowohl für den Fall, dass eine Verpflichtung bestehen soll, aber noch nicht besteht (Fehlende Verpflichtung), als auch, dass eine Verpflichtung nicht bestehen soll, aber dennoch besteht (Unerwünschte Verpflichtung). Zum einen kann der gewünschte Zustand durch Delegation erreicht werden, d.h. die Bearbeitung einer Aufgabe oder das Aufgeben einer Verpflichtung angefragt werden. Zum anderen kann ein Teammitglied seine eigenen Verpflichtungen ändern (Adoption). Hier besteht zudem die Möglichkeit, eine solche Änderung der Verpflichtung zunächst vorzuschlagen und nur bei positiver Rückmeldung tatsächlich durchzuführen, als auch ohne vorherige Interaktion mit anderen Teammitgliedern Verpflichtungen einzugehen bzw. aufzugeben. Welcher Weg dabei eingeschlagen wird, kann zum Beispiel durch die Zeitkritikalität der Aufgabe oder das zu erreichende Ziel beeinflusst werden. Arbeiten ACUs mit menschlichen Teammitgliedern zusammen, so sind in diesem Zusammenhang zusätzlich Informationen hinsichtlich der verfügbaren mentalen Ressourcen seitens des Menschen relevant.

Eine Zuordnung von Handlungsalternativen zu Wünschen und umgekehrt findet sich in Tabelle 4-1 und Tabelle 4-2. Verschiedene Möglichkeiten, ein Ziel zu erreichen, ergeben sich zum einen durch Instanziierung verschiedener Handlungsalternativen. Zum anderen kann eine Handlungsalternative mit unterschiedlichen Attributen mehrfach instanziiert werden. So kann zum Beispiel ein Dialog weitergeführt werden, indem eine Nachricht gesendet wird. Da allerdings unterschiedliche Arten von Nachrichten gesendet werden können, ergeben sich mehrere Optionen.

Tabelle 4-2 detailliert die kooperationsrelevanten Handlungsalternativen im Hinblick auf ihre Attribute, die Wünsche, die sie potentiell erreichen können, die Umweltmodelle, die zur Instanziierung benötigt werden und die Anweisungsmodelle, die zur Ausführung benötigt werden.

Attribute	***Relevanz: Wünsche***	***Umweltmodelle***	***Anweisungs-modell (Umweltmodell)***
Gehe Verpflichtung ein			
Aufgabe, Agent	Sei einer Aufgabe verpflichtet, Stelle Aufgabenabdeckung sicher, Bearbeite unterstützende Aufgabe, Befolge Anweisung	Team, Akteur, Teamsprecher, Aufgabe, Evaluierung einer Aufgabe, Anweisung	-- (Verpflichtung)
Gib Verpflichtung auf			
Verpflichtung	Habe keine überflüssige Verpflichtung, Halte Agenda durchführbar, Vermeide redundante Aufgabenzuweisung, Befolge Anweisung	Verpflichtung, Aufgabe, Akteur, Agenda, Anweisung	-- (Verpflichtung)
Füge Verpflichtung ein			
Verpflichtung, Position in Agenda	Sortiere Verpflichtung in Agenda ein	Aufgabe, Verpflichtung, Agenda	-- (Agenda)
Brich Dialog ab (cancel)			
Dialog	Führe keinen unnötigen Dialog	Dialog	-- (Dialog „cancel“)
Informiere (inform)			
Empfänger, Information	Informiere Teammitglieder laufend	Art der Information, Akteur, Team, konkrete Information (Fähigkeit, Ressource, Verpflichtung, gemeinsam genutzte Ressource, Teamsprecher, …)	-- (Dialog „inform“)
Schlage Aktion vor (propose)			
Empfänger, Vorschlag, Art des Vorschlags	Stelle Aufgabenabdeckung sicher, Verteile Arbeitsbelastung, Habe Teamsprecher, Vermeide redundante Aufgabenzuweisung	Akteur, Team, Aufgabe, Verpflichtung, Evaluierung einer Aufgabe, Teamsprecherwahl	-- (Dialog „propose“)
Frage Information einmalig an (query)			
Empfänger, Art der Information	Kenne Teammitglied	Akteur, Informationsspezifikation	-- (Dialog „query“)
Frage Aktion an (request)			
Empfänger, Anfrage, Art der Anfrage	Stelle Aufgabenabdeckung sicher, Vermeide redundante Aufgabenzuweisung	Aufgabe, Akteur, Team, Fähigkeit, benötigte Fähigkeit, Evaluierung einer Aufgabe	-- (Dialog „request“)
Frage Information dauerhaft an (subscribe)			
Empfänger, Art der Information	Kenne Teammitglied	Akteur, Informationsspezifikation	-- (Dialog „subscribe“)
Sende Nachricht			
Zustandsübergang, Inhalt der Nachricht	Führe Dialog fort	Dialog, Zustandsübergang, Akteur	Führe Senden einer Nachricht aus
Lösche Anweisung			
Anweisung	Befolge Anweisung	Anweisung	-- (Anweisung)

Tabelle 4-2: Detaillierung kooperationsrelevanter Handlungsalternativen

Fast alle Handlungsalternativen werden dabei durch Umweltmodelle ausgeführt und ändern somit zunächst den internen Zustand der ACU. So wird zum Beispiel ein Dialog intern initiiert und erst in einem nächsten Schritt eine zu diesem Dialog gehörige Nachricht gesendet, die dann die Umgebung beeinflusst. Da lediglich das Senden einer Nachricht im Sinne der Sprechakttheorie die Umgebung betrifft ist diese Handlungsalternative hier die einzige, zu der es ein klassiches Anweisungsmodell gibt. Neben den in Tabelle 4-2 aufgeführten Handlungsalternativen bedarf es zudem domänenspezifischer Handlungsalternativen und entsprechender Umwelt- und Anweisungsmodelle. Diese sind zum Beispiel im Umfeld des Ziels „Erfüllung einer Verpflichtung" relevant, da sich Verpflichtungen direkt auf die zu bearbeitenden Aufgaben im Kontext der Applikation beziehen. Hierbei stellt zum Beispiel die Handlungsalternative „Waffe einsetzen" eine Möglichkeit dar, die Aufgabe „Bedrohung bekämpfen" zu bearbeiten.

4.3.3 Anweisungsmodelle

Wie im vorhergehenden Abschnitt erläutert beschränken sich Anweisungsmodelle im Umfeld von Kooperation auf das Senden von Nachrichten, da alle andere Handlungsalternativen den internen Zustand ändern und nach und nach in Kommunikation münden, welche letztlich immer über das Senden einer Nachricht umgesetzt wird. Dieses Anweisungsmodell füllt die einzelnen Teile des in Abschnitt 3.3.4 vorgestellten Nachrichtenformats mit Werten. Ferner werden Nachrichten über eine entsprechende Schnittstelle tatsächlich gesendet.

Darüber hinaus existieren eine Reihe anwendungsspezifischer Anweisungsmodelle, welche einzelne Planelemente entweder direkt in eine Anweisung übersetzen (z.B. Planung eines Flugplans) oder in Form eines Skripts mehrere Anweisungen nacheinander abarbeiten (z.B. Abdrehen).

4.3.4 Umweltmodelle

Während Wünsche die Ursache für wissensbasiertes Verhalten einer ACU darstellen und Handlungsalternativen und Anweisungsmodelle gleichsam Mittel zum Erreichen von Zielen sind, stellen Umweltmodelle die hierfür nötige Basis dar. Sowohl die Existenz als auch die Attribute und das Verhalten eines Umweltmodells ergeben sich aus dem Kontext, in dem sie benötigt werden. So muss beispielsweise im Rahmen der Aktivierung des Wunsches „Vermeide redundante Aufgabenbearbeitung" festgestellt werden können, wenn zwei Akteure zur gleichen Aufgabe verpflichtet sind, obwohl dies überflüssig ist (vgl. Tabelle 4-1). Somit ist eine Repräsentation verschiedener Akteure, vorhandener Aufgaben und eingegangener Verpflichtung von Akteuren zu Aufgaben nötig. Ferner muss hinterlegt werden, wie viele Akteure sinnvoll gleichzeitig zu einer Aufgabe verpflichtet sind.

Da die Darstellung der Wünsche und Handlungsalternativen in Tabelle 4-1 und Tabelle 4-2 bereits einen Eindruck der für Kooperation notwendigen Umweltmodelle gibt und diese das Verhalten einer ACU nicht ursächlich bestimmen, werden sie hier nicht weiter detailliert.

5 Implementierung kooperativer *Supporting ACUs*

Basierend auf dem in Kapitel 3 beschriebenen Konzept und insbesondere den für Kooperation als relevant identifizierten Modellen des a-priori Wissens des Kognitiven Prozesses (vgl. Abschnitt 4.3) wurde auf Grundlage der kognitiven Systemarchitektur COSA ein Funktionsprototyp entwickelt, der in diesem Kapitel näher beschrieben wird. Hierbei handelt es sich konkret um ein Team aus *Supporting ACUs* zur Führung von UAVs in der in Abbildung 2-9 dargestellten Konfiguration. Diese erhalten von einem menschlichen Operateur Aufträge, die gemeinsam bearbeitet werden sollen. Zur Vereinfachung wird hier ein hoher Grad an Eigenständigkeit der *Supporting ACUs* angenommen, so dass auch Aktionen selbständig ausgeführt werden können, die zum Beispiel in realen militärischen Missionen nicht ohne Rückfrage bei einem menschlichen Operateur durchgeführt würden. Eine Berücksichtigung solcher Aspekte ist jedoch im Rahmen des vorliegenden Konzepts möglich, so dass diese Vereinfachung keine grundsätzliche Einschränkung darstellt.

Für eine gemeinsame Bearbeitung von Aufträgen duch ein Team aus *Supporting ACUs* sind sowohl individuelle Fähigkeiten wie beispielsweise die konkrete Durchführung von Teilaufgaben als auch kooperative Fähigkeiten wie die Aufgabenzuweisung im Team notwendig. Letztere sind nicht spezifisch im Hinblick auf eine konkrete Mission und werden in diesem Kapitel beschrieben. Aus ihnen ergeben sich jedoch Randbedingungen hinsichtlich der Struktur des missions- bzw. anwendungsspezifischen Wissens, auf die an den jeweiligen Stellen hingewiesen wird.

In Abschnitt 5.1 werden zunächst die auf dem Kognitiven Prozess basierende Architektur COSA (*Cognitive System Architecture*) und die Programmiersprache CPL (*Cognitive Programming Language*) vorgestellt, welche zur Modellierung von a-priori Wissen verwendet wurde. Die Abschnitte 5.2 und 5.3 erläutern im Anschluss die Implementierung grundlegender Fähigkeiten der ACUs, nämlich die Problemlösung auf Basis der Mittel-Ziel-Analyse und Kommunikation auf Grundlage von Interaktionsprotokollen. In Abschnitt 5.4 wird schließlich die Umsetzung konkreter kooperativer Fähigkeiten dargestellt, welche sich direkt aus den im vorigen Kapitel identifizierten Zielen kooperativer Zusammenarbeit ergeben.

5.1 Architektur

5.1.1 COSA (Cognitive System Architecture)

Für die Entwicklung kognitiver Systeme basierend auf der Theorie des Kognitiven Prozesses (siehe Abschnitt 3.2.3.2) steht die Architektur *COSA* (*Cognitive System Architecture*) [Putzer & Onken, 2003][Putzer, 2004] als Framework zur Verfügung, die im Rahmen der vorliegenden Arbeit für die Entwicklung künstlicher kognitiver Einheiten genutzt wurde. COSA implementiert die applikationsunabhängigen Aspekte des Kognitiven Prozesses, d.h. die Transformatoren und die formale Organisation des Wissens.

Abbildung 5-1 stellt den generischen Aufbau eines auf COSA basierenden Systems dar. Zentral ist hierbei der **Kernel**, welcher Soar (vgl. Abschnitt 3.2.1.2) als Prozessor nutzt und um eigene Konzepte erweitert. So strukturiert das in der KP-Bibliothek enthaltene Wissen den Arbeitsspeicher (≈ Situationswissen des KPs) und steuert die Abarbeitungs-

reihenfolge der Transformatoren des KPs. Da die Soar-Inferenzmaschine Wissen nur in Form von nativen Soar-Produktionen verarbeiten kann, muss Applikationswissen (a-priori Wissen) in Form von Soar-Regeln bereitgestellt werden. Wird dieses Wissen mit Hilfe der im folgenden Abschnitt erläuterten Programmiersprache CPL modelliert, so muss es mittels eines **Compilers** von CPL nach Soar übersetzt werden.

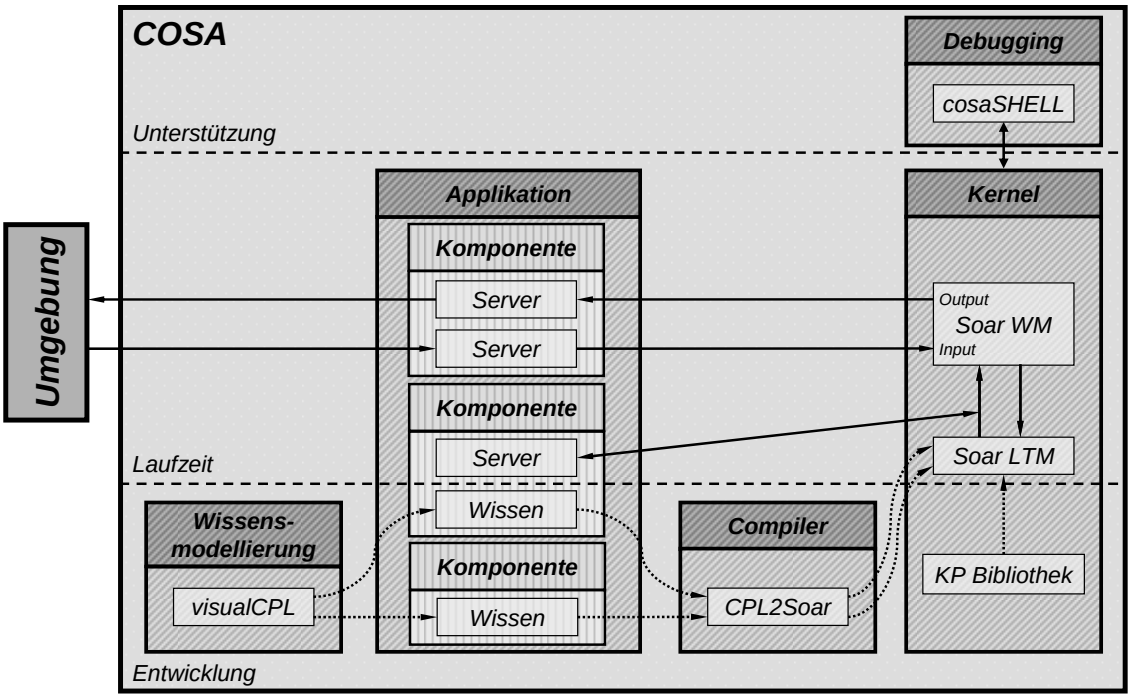


Abbildung 5-1: Cognitive System Architecture COSA
(WM: Working Memory, LTM: Long Term Memory)

Die eigentliche **Applikation** besteht aus mehreren Komponenten (≈ Pakete im Sinne der horizontalen Sichtweise auf den KP), welche aus a-priori Wissen und/oder Servern bestehen, die zum Beispiel zur Interaktion mit der Umgebung oder für numerische Berechnungen genutzt werden können. Teil von COSA ist darüber hinaus *visualCPL* als Oberfläche zur graphischen **Modellierung** in CPL und die *cosaSHELL* zur **Fehlersuche und Steuerung** des Kernels. Wird COSA für die Entwicklung eines künstlichen kognitiven Systems verwendet, so beschränkt sich die Entwicklung auf die Applikationskomponenten, da alle anderen Teile anwendungsunabhängig sind.

5.1.2 CPL (Cognitive Programming Language)

Die *Cognitive Programming Language (CPL)* [Putzer, 2004] stellt eine objektorientierte Erweiterung von Soar dar, um die Modellierung von Wissen auf der Abstraktionsebene mentaler Begriffe zu ermöglichen wie sie beispielsweise im Kognitiven Prozess (siehe Abbildung 3-11) durch die Struktur des a-priori Wissens definiert werden. Daher ist sie für die Realisierung von ACUs besonders gut geeignet. In Verbindung mit der KP-Bibliothek wird die Implementierung von Paketen (*package*) und Modellen des a-priori Wissens als Klassen (*class*) unterstützt. Jede Klasse ist dabei Teil eines Namensraums, der sie einer Art von Modell des a-priori Wissens des KPs und damit einem Transformator zuordnet (*belief* → Umweltmodell, *goal* → Wunsch, *plan* → Handlungsalternative, *schedule* → Anweisungsmodell). Klassen bestehen zum einen aus Attributen (*attributes*), die sie charakterisieren, und zum anderen aus Verhalten (*behavior*), welches beschreibt, wann Instanzen einer Klasse angelegt bzw. zerstört werden und mit welchen Werten Attribute belegt werden sollen. In diesem Zusammenhang wird in der Regel mit

anderen Klassen beziehungsweise Instanzen davon interagiert. Attribute können mit konkreten Werten verschiedenen Typs (z.B. Ganzzahl, String) versehen werden oder auf Instanzen beliebiger Klassen im Situationswissen verweisen. So ist zum Beispiel das Umweltmodell *gefahr* in Abbildung 5-2 durch einen Verweis auf das Flugzeug, von dem die Gefahr ausgeht, und die Größe der Gefahr charakterisiert.

```
package flugsicherheit
{
    ...
    class <belief> gefahr
    {
        attributes:
            link    flugzeug;
            string  groesse;
        behavior:
            sp {anlegen
                (instance[belief::flugzeug::*] <eigen> ^eigenes wahr)
                (instance[belief::flugzeug::*] <anderes> -^eigenes wahr)
                (instance[belief::entfernung::*] <entf> ^zwischen <eigen> ^zwischen <anderes>
                                                        ^entfernung klein)
             -->
                (elaborate <i> ^flugzeug <anderer> ^groesse hoch)
            }
    }; // ende class ‚gefahr‘
    ...
    class <goal> vermeide-gefaehrdung
    {
        attributes:
            link    gefaehrdung;
        behavior:
            sp {aktivieren
               (instance[belief::gefahr::*] <gefahr> ^flugzeug <fz> ^groesse << mittel hoch >>)
            -->
               (elaborate <i> ^gefaehrdung <fz>)
            }
    }; // ende class ‚vermeide-gefaehrdung‘
}; // ende package ‚flugsicherheit‘
```

Abbildung 5-2: CPL-Codebeispiel

Das Verhalten der in Abbildung 5-2 dargestellten Klasse *gefahr* beschreibt die Umstände, unter denen eine Instanz angelegt wird, und ist zu lesen als:

„Wenn (solange) es eine Instanz der Klasse Flugzeug gibt, die das eigene Flugzeug repräsentiert (es gibt ein Attribut „eigenes“, das mit dem Wert „wahr“ belegt ist)
und es eine Instanz der Klasse Flugzeug gibt, die nicht das eigene Flugzeug repräsentiert (es gibt kein Attribut „eigenes“, das mit dem Wert „wahr“ belegt ist)
und es eine Instanz der Klasse Entfernung gibt, die die Entfernung zwischen dem eigenen und dem anderen Flugzeug repräsentiert (es gibt zwei Attribute „zwischen“, die auf die oben definierten Instanzen von Flugzeug verweisen) und diese Entfernung klein ist (es gibt ein Attribut „entfernung“ mit dem Wert „klein“),
dann erstelle eine Instanz der Klasse „gefahr“, die auf das andere Flugzeug verweist und die Gefahr als hoch angibt.“

In gleicher Weise wird der Wunsch *vermeide-gefaehrdung* durch ein Attribut beschrieben, das auf die Gefährdung verweist und das Verhalten implementiert die Aktivierung als Ziel wie folgt:

„Wenn (solange) es eine Instanz der Klasse Gefahr gibt, die auf ein Flugzeug verweist und deren Größe „mittel“ oder „hoch“ ist,
dann erstelle eine Instanz der Klasse „vermeide-gefaehrdung“, die mit dem Attribut „gefaehrdung“ auf das Flugzeug verweist.“

5.1.3 Darstellung der Implementierung

Im weiteren Verlauf dieses Kapitels wird das im Laufe der Systementwicklung implementierte Wissen anhand von Abbildungen erläutert, welche Ausschnitte aus dem Situationswissen in Zusammenhang mit relevanten Modellen des a-priori Wissens darstellen.

Daher wird hier an dem im vorigen Abschnitt eingeführten CPL-Code-Beispiel erklärt, wie diese Abbildungen gelesen werden können, die sich aus den folgenden vier graphischen Elementen zusammensetzen:

- Kästen (umrandet mit durchgezogenen Linien, <namespace> Klasse) repräsentieren Modelle des a-priori Wissens. Die Zugehörigkeit zu einer Modellart (Umweltmodell, Wunsch, Handlungsalternative, Anweisungsmodell) ist durch den Namensraum (*belief, goal, plan, schedule*) sowie die farbliche Kodierung (gelb, blau, grün, rot) erkennbar.
- Kästen (umrandet mit gepunkteten Linien, Instanz) stellen Instanzen im Situationswissen dar. Die Zugehörigkeit zu einem Teilbereich des Situationswissens (Überzeugung, Ziel, Plan, Anweisung) ist durch die farbliche Kodierung (gelb, blau, grün, rot) sowie die zugeordnete Klasse erkennbar. Die fett gesetzte Beschriftung gibt dabei den Namen der Instanz an, während Attribute und deren Werte kursiv dargestellt sind (*Attributname = Wert*).
- Gepunkte Pfeile (⇢) geben an, zu welcher Klasse des a-priori Wissens eine Instanz im Situationswissen gehört („ist Instanz von“).
- Durchgezogene Pfeile (→) stellen Attribute vom Typ „Verweis“ dar, wobei der Attributname kursiv am Pfeil steht und der Wert des Verweises die Instanz ist, auf die der Pfeil zeigt.

Abbildung 5-3 stellt zwei Instanzen der Klasse **`flugzeug`** dar, wobei der Wert des Attributs *`eigenes`* der Instanz `flugzeug-0` *`wahr`* beträgt, während er bei `flugzeug-1` *`falsch`* ist. Außerdem ist eine Instanz der Klasse **`entfernung`** abgebildet, wobei der Wert des Attributs *`entfernung`* *`klein`* beträgt und zwei Attribute *`zwischen`* vorhanden sind, die auf die beiden Instanzen von **`flugzeug`** verweisen.

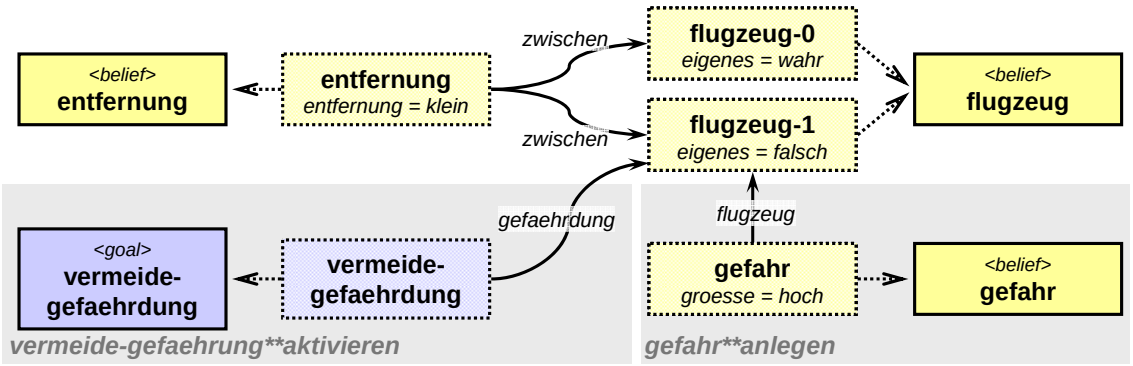


Abbildung 5-3:Beispiel für Ausschnitt aus dem Situationswissen

In dieser Konfiguration des Situationswissens feuert die Regel „anlegen“ der Klasse **`gefahr`** (siehe Abbildung 5-2), die bewirkt, dass eine Instanz der Klasse **`gefahr`** erzeugt wird, wobei das Attribut *`groesse`* mit dem Wert *`hoch`* belegt wird und das Attribut *`flugzeug`* auf `flugzeug-1` verweist. Nun sind die Vorbedingungen der Regel „aktivieren“ der Klasse **`vermeide-gefaehrdung`** (siehe Abbildung 5-2) erfüllt, woraufhin eine Instanz angelegt wird, deren Attribut *`gefaehrdung`* auf `flugzeug-1` verweist. Während die Instanzen `flugzeug-0`, `flugzeug-1`, `entfernung` und `gefahr` den Überzeugungen zuzuordnen sind, weil die entsprechenden Klassen Umweltmodelle darstellen, ist `vermeide-gefaehrdung` ein Ziel.

5.2 Paket „Problemlösung“

Da als Paradigma für die Planung des weiteren Vorgehens einer *Supporting ACU* die Mittel-Ziel-Analyse verwendet wird (vgl. Abschnitt 3.2.2.1), wird Wissen benötigt, welche Mittel (hier: Handlungsalternativen bzw. Instanzen davon) geeignet sind, welche Ziele (hier: Wünsche bzw. Instanzen davon) zu erreichen. Gibt es mehrere Alternativen, ein Ziel zu erreichen, muss entschieden werden, welche Option am geeignetsten ist. Darüber hinaus muss festgelegt werden, in welcher Reihenfolge die Elemente des Vorgehensplans ausgeführt werden. Für diese Fragestellungen relevantes, applikationsunabhängiges Wissen ist im Paket „Problemlösung“ hinterlegt. Dieses ist in den Klassen **end** und **means** implementiert, wobei die Klasse **end** sämtliches Wissen enthält, welches sich auf Ziele im Rahmen der Mittel-Ziel-Analyse bezieht und die Klasse **means** analog Wissen im Hinblick auf Mittel kapselt.

Aufgabe des Wissensingenieurs ist es, bei der Implementierung des Applikationswissens sämtliche Wünsche und Handlungsalternativen, die von der Mittel-Ziel-Analyse genutzt werden sollen, von den Klassen **end** und **means** abzuleiten und entsprechendes Wissen zu hinterlegen, mit denen Mittel-Ziel-Graphen aufgebaut werden können. Dies wird im Folgenden anhand des in Abbildung 5-4 dargestellten Beispiels erläutert.

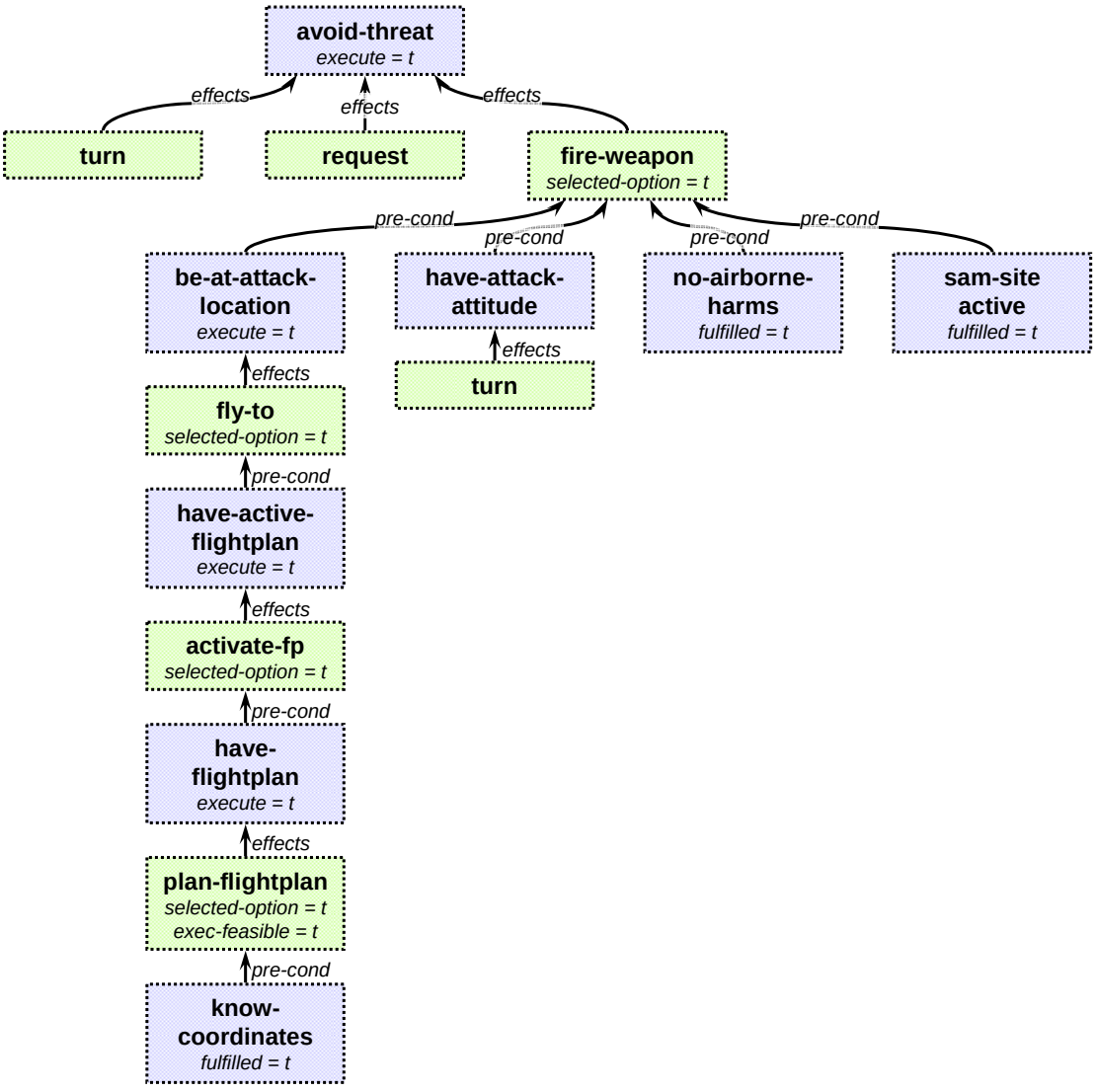


Abbildung 5-4: Beispiel für einen Mittel-Ziel-Graph

Für Wünsche muss Wissen über deren Relevanz und Erfülltheit hinterlegt werden. Die Relevanz für Fundamentalziele (in Abbildung 5-4 `avoid-threat`) ergibt sich dabei im Allgemeinen aus der Situation (`avoid-threat` ist z.B. relevant, wenn sich ein UAV im Erfassungsbereich einer SAM-Stellung befindet). Instrumentalziele stellen hingegen Vorbedingungen für die Ausführung eines Planelementes dar (z.B. kann ein Flugplan nur dann aktiviert werden, wenn es einen Flugplan gibt) und werden somit dann instanziiert, wenn eine entsprechende Instanz einer Handlungsalternative vorhanden ist. Für Handlungsalternativen muss Wissen implementiert werden, das angibt, welche Ziele (potentiell) durch Ausführung dieser Handlungsalternative erreicht werden können (z.B. kann `turn` sowohl das Erreichen von `avoid-threat` als auch `have-attitude` bewirken). Neben der korrekten Instanziierung von Zielen und Handlungsalternativen müssen für Ziele die Attribute *`pre-cond`* („ist Vorbedingung für die Ausführung von“) und *`fulfilled`* („ist erreicht“) bzw. für Handlungsalternativen das Attribut *`effects`* („ist geeignet für die Erreichung von“) belegt werden (vgl. Abbildung 5-4).

Das im Paket „Problemlösung“ hinterlegte Wissen wählt nun aus, welches (nicht erfüllte) Ziel als nächstes erreicht werden soll (Kennzeichnung durch das Attribut *`execute`* mit dem Wert *`t (true)`*) und welches Mittel dazu eingesetzt werden soll (Kennzeichnung durch das Attribut *`selected-option`* mit dem Wert *`t (true)`*). Hierzu wird der aufgebaute Mittel-Ziel-Graph von oben nach unten bearbeitet. Zunächst wird das oberste Ziel (hier: `avoid-threat`) als zu bearbeitend gekennzeichnet, sofern nicht explizit vermerkt ist, dass es (noch) nicht verfolgt werden soll. Anschließend wird ausgewählt, welche der vorgeschlagenen Handlungsalternativen (hier: `turn`, `request`, `fire-weapon`) eingesetzt werden soll, um dieses Ziel zu erreichen. Hinterlegt der Entwickler kein zusätzliches Wissen, erfolgt diese Auswahl zufällig. Für die ausgewählte Handlungsalternative wird nun wiederum bestimmt, welche der (noch) nicht erreichten Vorbedingungen (hier: `be-at-attack-location`, `have-attack-attitude`) zuerst bearbeitet werden soll. Auch hier erfolgt die Auswahl zufällig, sofern kein zusätzliches Wissen implementiert ist. Dieses Vorgehen wird so lange wiederholt, bis eine Handlungsalternative ausgewählt wird, für die keine unerfüllten Vorbedingungen mehr vorliegen und die daher mit dem Attribut *`execution-feasible`* gekennzeichnet werden kann (hier: `plan-flightplan`). Diese Handlungsalternative wird dann zur Ausführung vorgeschlagen. Im Normalfall führt die Ausführung zur Erreichung des entsprechenden Ziels (hier: `have-flightplan`), so dass nun die nächste Handlungsalternative (hier: `activate-flightplan`) keine unerfüllten Vorbedingungen mehr hat und ausgeführt werden kann.

5.3 Paket „Kommunikation“

Wie in Abschnitt 3.3.4 dargestellt basiert die Kommunikation der *Supporting ACUs* auf Dialogen, deren Ablauf mit Hilfe von Zustandsübergangsdiagrammen beschrieben wird. In diesem Abschnitt wird nun erläutert, wie dieser Ansatz im Rahmen des Paketes „Kommunikation“ umgesetzt wurde.

Dialoge werden mit Hilfe der Klassen **`dialog`**, **`dialog-state`** und **`dialog-transition`** repräsentiert, wobei zusätzlich jeder Dialogtyp in einer von **`dialog`** abgeleiteten Klasse (**`dialog-cancel`**, **`dialog-inform`**, **`dialog-instruct`**, **`dialog-propose`**, **`dialog-query`**, **`dialog-request`** und **`dialog-subscribe`**) abgebildet wird. Die Klasse **`dialog`** sorgt hier für die Erzeugung von Instanzen entsprechender Dialoge beim Eingang einer Nachricht, die Zuordnung von Nachrichten zu existierenden Dialogen, die Bestimmung des aktuellen Zustands eines Dialogs und die Zerstörung von beendeten Dialogen. Die abgeleiteten Klassen instanziieren die für sie spezifische Abfolge von Zu-

ständen und Übergängen und bestimmen, wann ein Übergang stattfinden kann. Außerdem wird hier situationsspezifisches Wissen für die Auswahl zwischen Handlungsalternativen, die sich auf das Ziel „Führe Dialog fort“ beziehen, hinterlegt. Schließlich werden auch Handlungsalternativen, die entsprechende Dialoge initiieren sollen, durch die für einzelne Typen von Dialogen spezifischen Klassen ausgeführt.

Abbildung 5-5 zeigt beispielhaft einen Dialog vom Typ *Request* wie er nach Eingang eines Auftrags vom Operateur an eine *Supporting ACU* in deren Situationswissen repräsentiert ist. Bei Eingang der entsprechenden Nachricht wird zunächst eine Instanz der Klasse **`dialog-request`** erzeugt, deren Attribute das verwendete Protokoll und die Kennung des Dialogs speichern und auf den Initiator (`actor-operator`) und Teilnehmer (`actor-self`) sowie den anwendungspezifischen Gesprächsgegenstand (hier: `mission-order`) verweisen. Des Weiteren werden Instanzen der Klassen **`dialog-state`** (`start`) und **`dialog-transition`** (`request`) erzeugt, die beide als dem Dialog zugehörig gekennzeichnet werden. Außerdem wird durch das Attribut *`transition`* von `start`, das auf `request` verweist, hinterlegt, dass der einzig mögliche Übergang auf einen nächsten Zustand durch eine Nachricht mit dem Performativ „request“ erfolgen kann und diese Nachricht vom Operateur gesendet werden muss (Attribut *`actor`*, das auf `actor-operator` verweist). Da diese Nachricht bereits gesendet wurde, wird dieser Übergang als erfolgt (*`done=t`*) gekennzeichnet und als Folgezustand `evaluate-request` erzeugt, der gleichzeitig der aktuelle Zustand des Dialogs ist. Auf ihn können die Übergänge `agree` und `refuse` folgen. Sobald eine Überprüfung der Machbarkeit des Auftrags erfolgt ist, wird dieser Zustand mit *`ready-for-action=t`* markiert, so dass nun folgende Zuständsübergänge durchgeführt werden können.

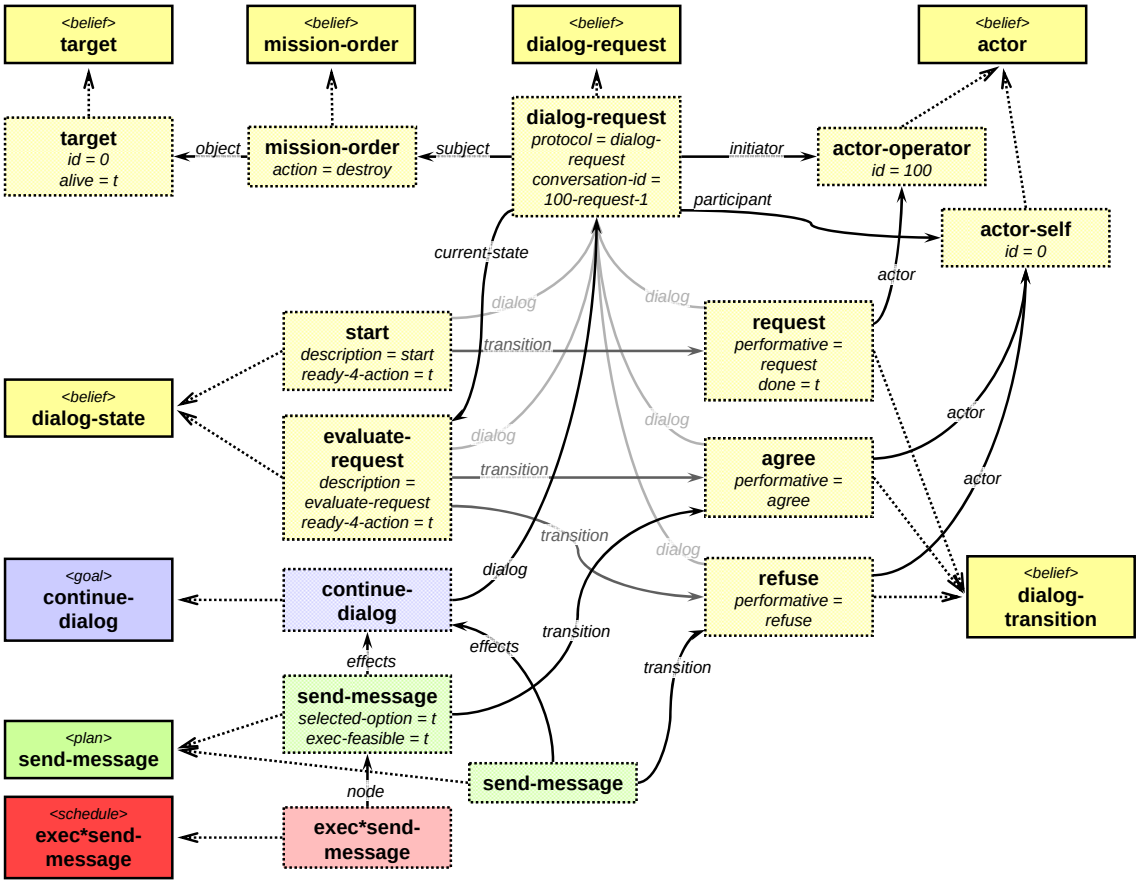


Abbildung 5-5: Beispiel für Dialog vom Typ "Request"

Da keine Nachricht eines entsprechenden Typs, die diesem Dialog zugeordnet werden kann, gesendet wurde und die ACU selbst (`actor-self`) die Zustandsübergänge `agree` bzw. `refuse` auslösen kann bzw. muss, wird eine Instanz des Wunsches **`continue-dialog`** angelegt, das mit Hilfe der Handlungsalternative **`send-message`** erreicht werden kann. Diese wird doppelt instanziiert, da es zwei mögliche Folgeübergänge gibt. Da es hier im Allgemeinen nicht sinnvoll ist, zufällig eine Option auszuwählen, sollte hier zusätzliches Wissen hinterlegt werden, welche Handlungsalternative in welcher Situation verwendet werden soll. Hier wird das Planelement `send-message` ausgewählt, welches den Übergang `agree` herbeiführen soll. Sobald diese Option ausgewählt ist (*`selected-option=t`*) und ausgeführt werden kann (*`exec-feasible=t`*) wird eine Instanz des Anweisungsmodells **`exec*send-message`** angelegt, die das verwendete Nachrichtenprotokoll mit den entsprechenden Werten belegt und die Nachricht schließlich sendet. Abbildung 5-6 zeigt den Dialog, nachdem dieser beendet wurde.

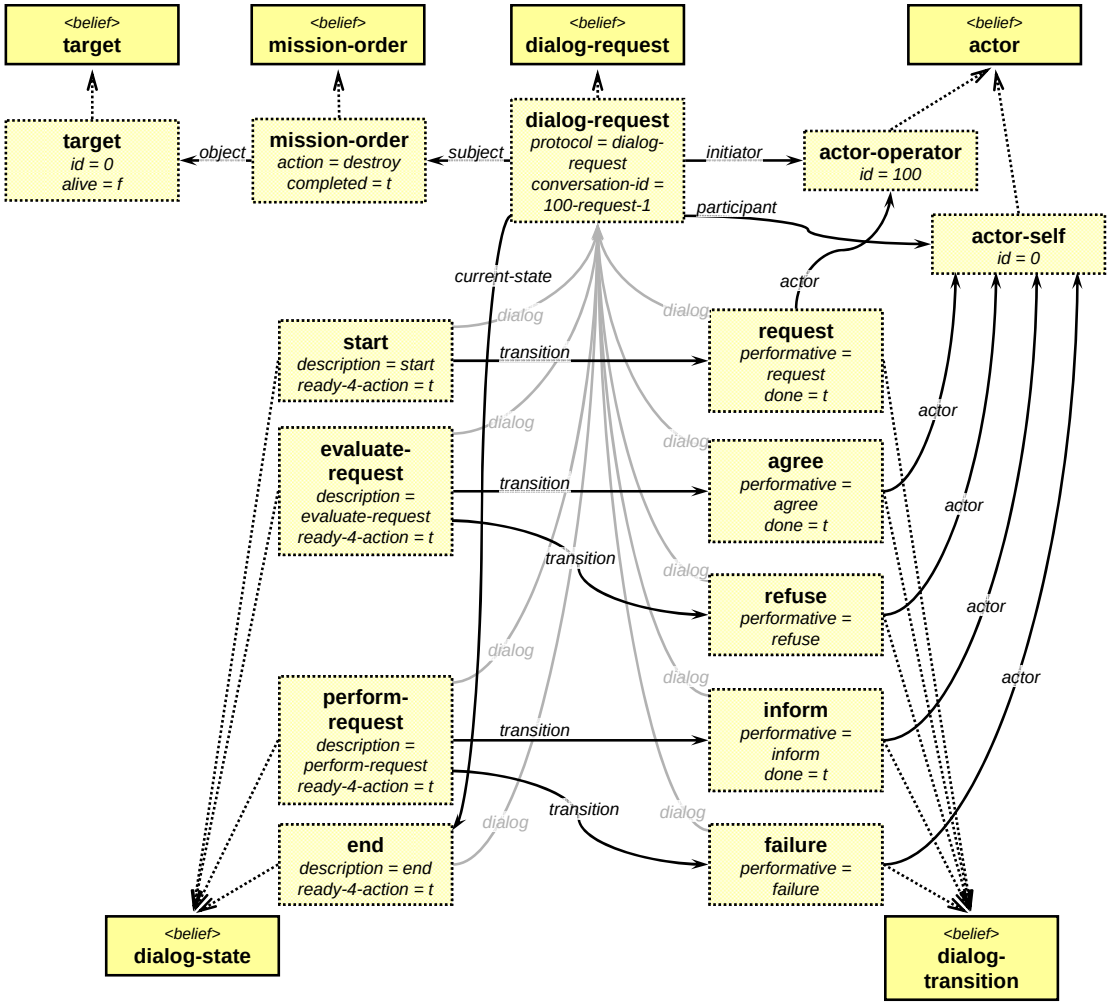


Abbildung 5-6: Beispiel für beendeten Dialog vom Typ "Request"

Interagieren nun nicht zwei Akteure wie zum Beispiel der Operateur und eine *Supporting ACU* miteinander, sondern ein Akteur mit einem Team, so ändert sich die Kommunikation im Prinzip nicht. Lediglich eines der Attribute *`initiator`* oder *`participant`* des Dialogs verweist nicht auf eine Instanz der Klasse **`actor`** sondern auf eine Instanz der Klasse **`team`**. Außerdem ist die Aktivierung des Ziels `continue-dialog` so gestaltet, dass sich die in Abschnitt 3.3.4 beschriebenen Kommunikationsflüsse ergeben, d.h. der Sprecher eines Teams Dialoge mit Akteuren außerhalb des Teams führt und bei teaminternen Dialogen alle Teammitglieder dem Initiator antworten.

5.4 Paket „Kooperation“

Nachdem im vorhergehenden Abschnitt die Implementierung von Wissen beschrieben wurde, welches an vielen verschiedenen Stellen im Gesamtsystem relevant ist, geht dieser Abschnitt auf das Zusammenspiel von Modellen ein, welche benötigt werden, um kooperative Fähigkeiten zu realisieren. Da das hier implementierte System auf wissensbasierter Verhaltensebene agiert, d.h. sämtliches Verhalten von entsprechenden Zielen ausgeht, orientiert sich die Beschreibung des implementierten Wissens an den in Abschnitt 4.3.1 identifizierten Wünschen, welche auch zur Strukturierung dieses Kapitels herangezogen werden:

- Abschnitt 5.4.1 beschäftigt sich mit den Wissensmodellen im Zusammenhang mit Zielen aus dem Bereich „Erreichen des gemeinsamen Ziels“.
- Abschnitt 5.4.2 beschreibt das Zusammenspiel von Wissensmodellen im Umfeld von Zielen, die das Bilden eines Teams betreffen.
- Abschnitt 5.4.3 erläutert Wissensmodelle, die mit Zielen aus dem Bereich „Auflösen von Abhängigkeiten“ zusammenhängen.
- Abschnitt 5.4.4 verweist auf eine Beschreibung der Wissensmodelle die für Kommunikation im Team relevant sind.
- Abschnitt 5.4.5 stellt Wissensmodelle im Zusammenhang mit Zielen aus dem Bereich „Integration ins Arbeitssystem“ dar.

Konkret werden Ziele dabei in den Kontext von Instanzen von Umweltmodellen und Handlungsalternativen gesetzt, welche letztlich in Verhalten resultieren. Für alle Wünsche und Handlungsalternativen gilt hierbei, dass sie von den Klassen **`end`** bzw. **`means`** abgeleitet wurden, um von dem in Abschnitt 5.2 beschriebenen Wissen für die Problemlösung berücksichtigt werden zu können.

5.4.1 Erreichen des gemeinsamen Ziels

5.4.1.1 Evaluierung der Durchführbarkeit einer Aufgabe

Bevor in den folgenden Abschnitten auf das Zusammenspiel von Wissensmodellen im Umfeld einzelner Wünsche bzw. Ziele eingegangen wird, soll hier erläutert werden, wie ein einzelnes Teammitglied bzw. ein Team beurteilen kann, ob eine Aufgabe bzw. ein Auftrag von ihm durchgeführt werden kann. Dies stellt im Folgenden immer wieder relevantes Wissen für den Vorschlag bzw. die Auswahl von Handlungsoptionen dar. Das hier beschriebene Wissen stellt einen sehr pauschalen und stark vereinfachten Ansatz dar, der aber im Rahmen der vorliegenden Arbeit ausreichend ist. Detaillierteres Wissen in diesem Zusammenhang ist stark anwendungsspezifisch und könnte bei Bedarf in beliebiger Komplexität in den hier vorgestellten Ansatz eingefügt werden, ohne dadurch die Struktur des Gesamtsystems zu ändern bzw. ändern zu müssen.

Die Bewertung der Durchführbarkeit einer konkreten Aufgabe durch ein einzelnes Teammitglied ist in Abbildung 5-7 dargestellt. Zentral ist hierbei die Instanz `evaluation`, die mit dem Attribut *`of`* auf die zu bewertende Aufgabe und mit dem Attribut *`by`* auf denjenigen, für den die Durchführbarkeit bestimmt werden soll, verweist. Das Verhalten der Klasse **`evaluation`** bestimmt nun, ob ein Teammitglied über die nötigen Ressourcen und Fähigkeiten zur Durchführung der Aufgabe verfügt. Dazu werden die benötigten Fähigkeiten für die Durchführung der Aufgabe (`req-capability`) mit den vorhandenen verglichen (`capability`) und entsprechend des Ergebnisses des Ver-

gleichs der Wert des Attributs *capabilities-ok* auf *t* (true) oder *f* (false) gesetzt. Gleiches gilt für die benötigten und verfügbaren Ressourcen (`req-resource` bzw. `resource`) und das Attribut *resources-ok*. Das Ergebnis der Evaluierung ist positiv (*feasible=t*), sofern der Akteur sowohl über die benötigten Fähigkeiten als auch Ressourcen verfügt. Die konkrete Ausprägung der Attribute der Klassen **`capability`** und **`resource`** sowie entsprechend **`required-capability`** und **`required-resouce`** sind dabei ebenso wie die konkret vorhandenen Aufgaben (hier: **`task-destroy`**, abgeleitet von **`task`**) anwendungsspezifisch.

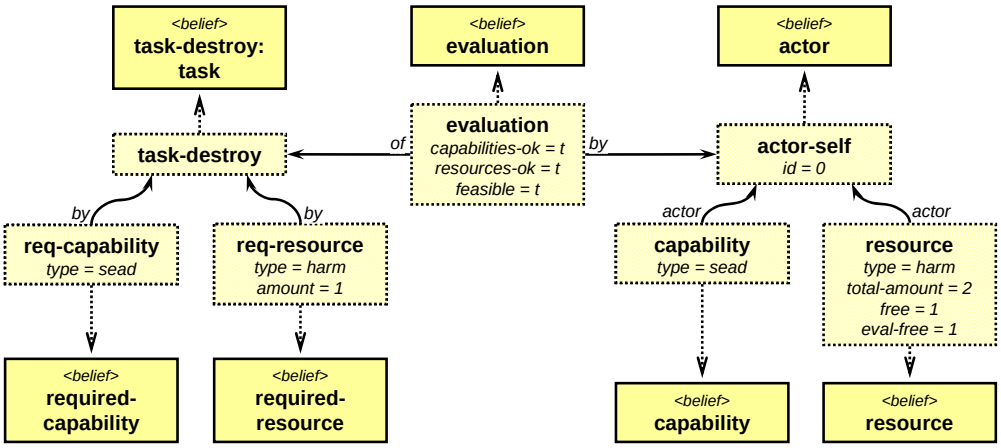


Abbildung 5-7: Evaluierung einer Aufgabe durch ein Teammitglied

Ähnlich verhält es sich, wenn die Durchführbarkeit einer Aufgabe durch ein Team evaluiert werden soll (siehe Abbildung 5-8). Allerdings werden hier die benötigten Ressourcen (`req-resource`) mit den über die Teammitglieder aufsummierten Ressourcen des entsprechenden Typs (`team-resource`) verglichen, während die Fähigkeiten als ausreichend bewertet werden, sofern mindestens ein Teammitglied über die benötigten Fähigkeiten verfügt.

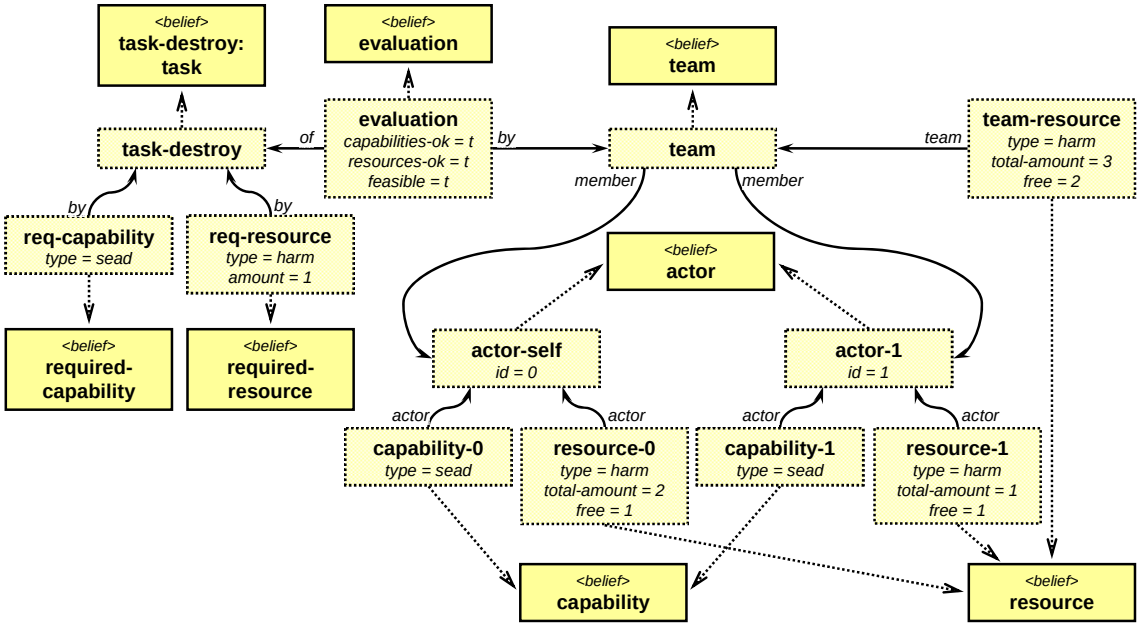


Abbildung 5-8: Evaluierung einer Aufgabe durch ein Team

Soll nun die Durchführbarkeit eines Auftrags beurteilt werden, dessen konkrete Ausprägung wiederum von der Applikation abhängt, aber welcher sich im Allgemeinen aus mehreren Teilaufgaben zusammensetzt, so müssen zunächst die einzelnen Teilaufgaben

durch das Team oder das Teammitglied bewertet werden. Ist jede Aufgabe für sich durchführbar, so wird angenommen, dass auch der gesamte Auftrag durchführbar ist (siehe Abbildung 5-9).

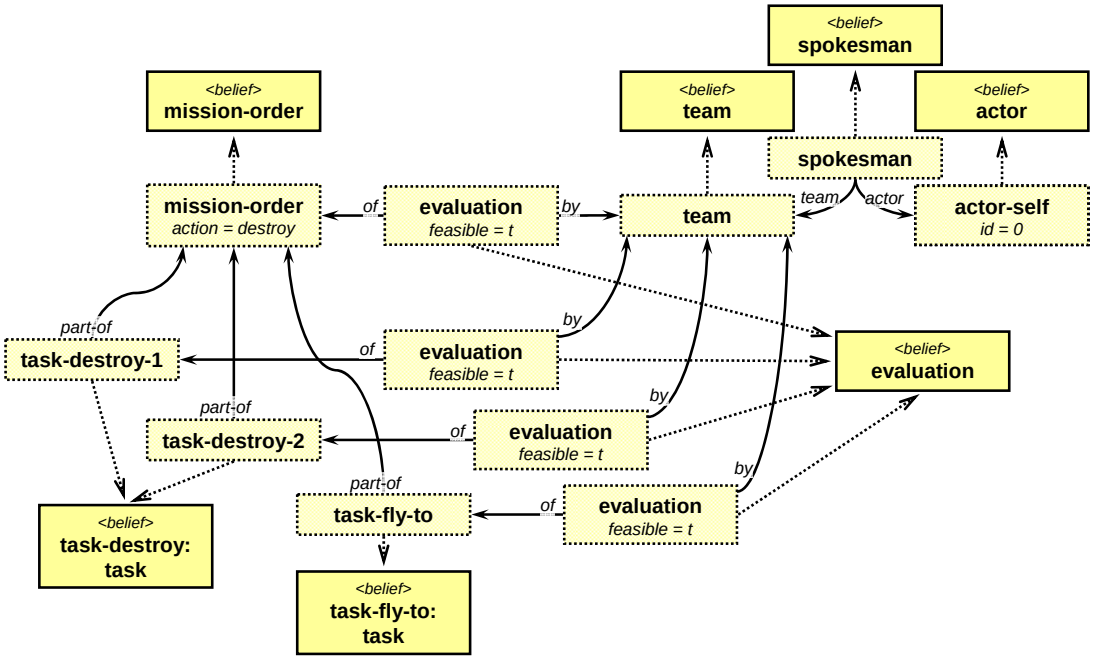


Abbildung 5-9: Evaluierung eines Auftrags

5.4.1.2 Sei einer Aufgabe verpflichtet

Hat ein Team einen Auftrag, der ihm zum Beispiel vom Operateur im Rahmen einer Anfrage übermittelt wurde, angenommen, so muss es Verpflichtungen zu allen Teilaufgaben dieses Auftrags eingehen, was auf Basis des Wunsches „Sei einer Aufgabe verpflichtet“ geschieht. In Abbildung 5-10 ist ein Auftrag (`mission-order`) dargestellt, welcher sich aus zwei Teilaufgaben (`task-destroy`, `task-fly-to`) zusammensetzt (vgl. Attribut *`part-of`*) und für den das Team `team` verantwortlich ist (Attribut *`respon-sible`*) und der vom Team angenommen wurde (*`accepted=t`*).

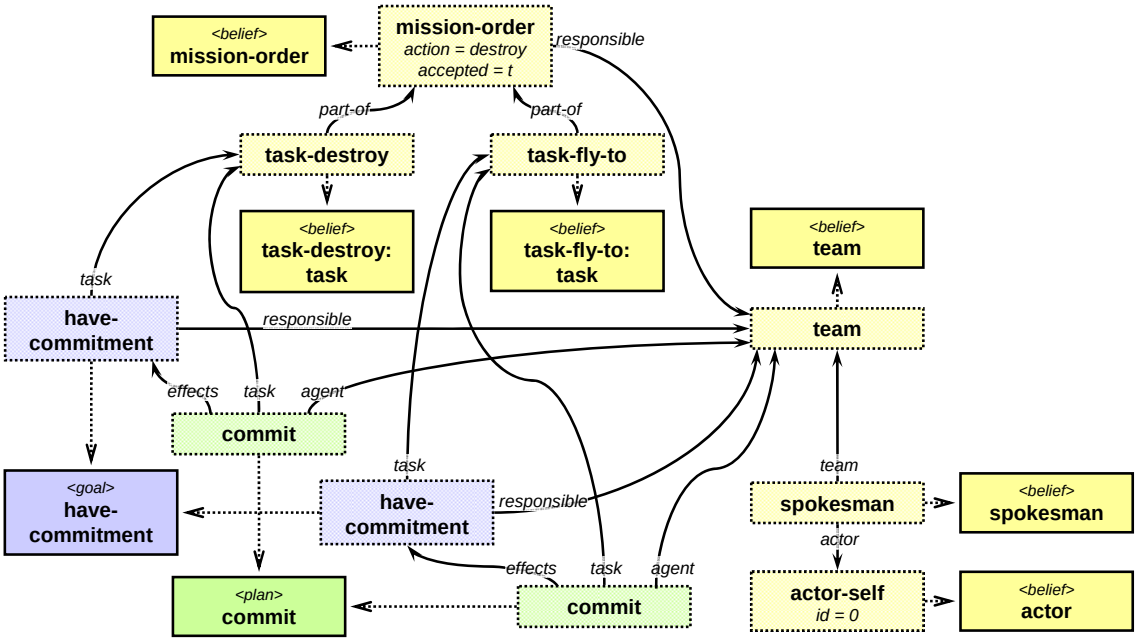


Abbildung 5-10: Eingehen von Verpflichtungen

Welche Aufträge eine ACU erhalten kann und aus welchen konkreten Teilaufgaben diese bestehen ist dabei abhängig von der Anwendung und muss daher in der Klasse **`mission-order`** und entsprechenden Aufgabenmodellen hinterlegt werden. Letztere müssen dabei wie hier **`task-destroy`** und **`task-fly-to`** von der übergeordneten Klasse **`task`** abgeleitet werden.

Da keine Verpflichtung des Teams zu den angesprochenen Aufgaben besteht, wird der Wunsch **`have-commitment`** für jede Aufgabe instanziiert. Sind die einzelnen Aufgaben für das Team durchführbar, so steht dem eigenen Akteur (`actor-self`) als Teamsprecher (`spokesman`) des Teams `team` die Handlungsalternative **`commit`** zur Verfügung, um ein solches Ziel zu erreichen, so dass diese instanziiert und ausgeführt wird. Hierdurch ändern sich die Verpflichtungen des Teams, was in der Folge Auswirkungen auf die Aktivitäten einzelner Teammitglieder hat (z.B. Eingehen individueller Verpflichtungen auf Basis des Ziels „Stelle Aufgabenabdeckung sicher“, siehe Abschnitt 5.4.2.3).

5.4.1.3 Habe keine überflüssige Verpflichtung

Ist ein Team zur Durchführung einer Aufgabe verpflichtet, welche bearbeitet, irrelevant oder unerreichtbar geworden ist, so soll diese Verpflichtung aufgegeben werden. Hierzu wurde der Wunsch „Habe keine überflüssige Verpflichtung“ implementiert. Dieser ist in Abbildung 5-11 als die Klasse **`do-not-have-commitment`** dargestellt und wird aktiviert, weil die Aufgabe `task-destroy`, zu der eine Verpflichtung (`commitment`) des Teams `team` besteht, erfolgreich bearbeitet wurde (*`completed=t`*). Eine Verpflichtung kann dabei auf beliebige Klassen verweisen, die Aufgaben darstellen, daher von **`task`** abgeleitet sein müssen und für die jeweilige Anwendung spezifisch sind.

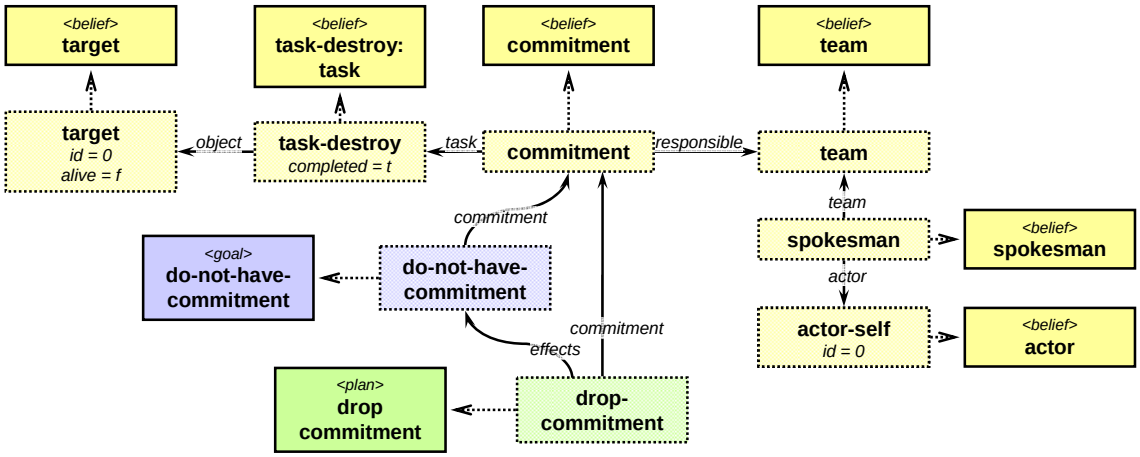


Abbildung 5-11: Aufgabe einer Verpflichtung

Die Aufgabe von Verpflichtungen eines Teams obliegt ebenso wie das Eingehen von Verpflichtungen dem Teamsprecher (`spokesman`), welcher das Planelement `drop-commitment` zur Erreichung des Ziels ausführt, woraufhin die Verpflichtung aufgegeben wird. Dadurch ändern sich auch hier die Verpflichtungen des Teams, so dass unter Umständen auch die Verpflichtungen einzelner Teammitglieder hiervon beeinflusst werden.

5.4.1.4 Führe keinen unnötigen Dialog

Nicht nur eigene Verpflichtungen, sondern auch die Verpflichtungen anderer Teammitglieder, die diese in Folge einer eigenen Anfrage eingegangen sind, können unnötig werden. Daher wird der Wunsch „Führe keinen unnötigen Dialog“ zum Ziel `abort-unnecessary-dialog` aktiviert, sobald es keinen Grund mehr für eine selbst initiierte An-

frage gibt. Abbildung 5-12 erläutert das Zusammenspiel des hierfür benötigten Wissens anhand eines Beispiels, bei dem ein UAV (`ucav`), welches dem eigenen Akteur (`actor-self`) zugeordnet ist, durch eine SAM-Stellung (`sam-site`) bedroht wird, was durch die Instanz `threat`, welche sowohl auf `ucav` als auch auf `sam-site` verweist, repräsentiert wird. Daraufhin wird ein Ziel `avoid-threat` aktiviert, welches im Beispiel erreicht werden soll, indem eine Anfrage (`request`) an das Team `team-sead` gestellt wird. Gegenstand dieser Anfrage ist die Durchführung der Aufgabe `task-suppress`, d.h. die Unterdrückung der gefährlichen SAM-Stellung. Aktuell befindet sich der hierzu initiierte Dialog (`dialog-request`) im Zustand `perform-request` (siehe Attribut `current-state`), was bedeutet, dass die angefragte Aufgabe in Bearbeitung ist.

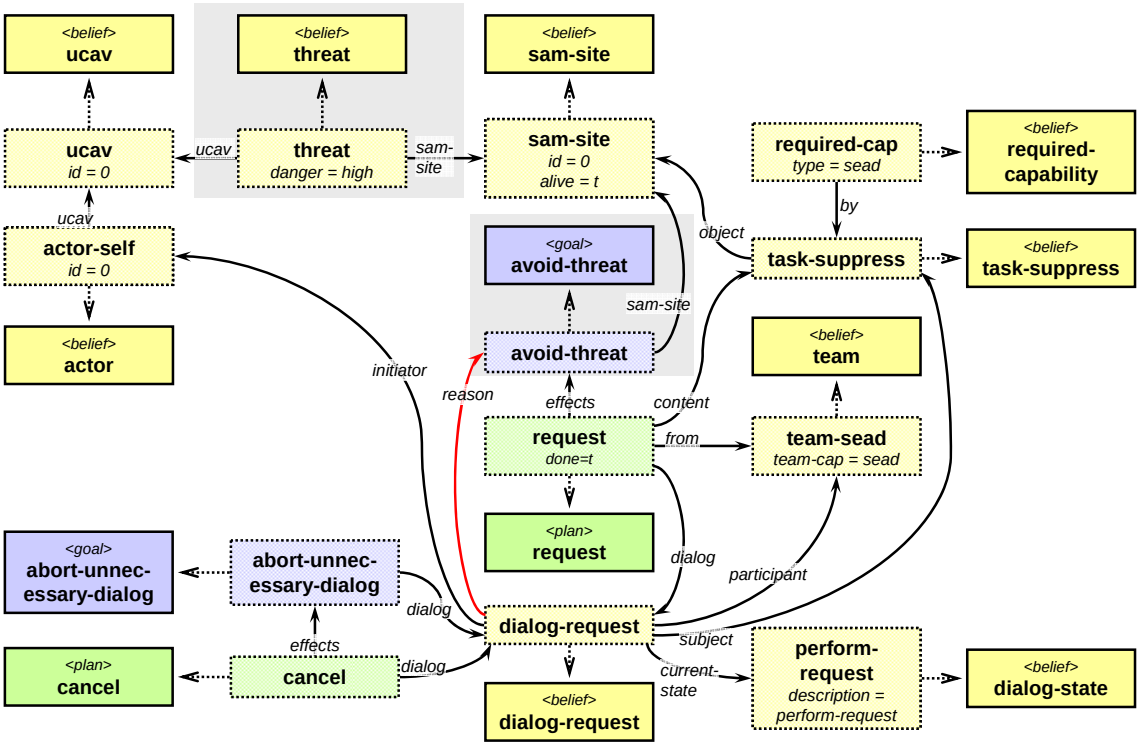


Abbildung 5-12: Abbruch eines unnötigen Dialogs

Fällt nun der Grund für diese Anfrage, d.h. das Ziel `avoid-threat`, weg, weil zum Beispiel keine Gefahr mehr besteht (in Abbildung 5-12 grau hinterlegt), so besteht aus Sicht des eigenen Akteurs keine Veranlassung mehr, dass die angefragte Aufgabe bearbeitet wird (Attribut *`reason`* von `dialog-request` verweist nun ins Leere). Daher wird eine Instanz des Wunsches **`abort-unnecessary-dialog`** angelegt, welche durch die Ausführung von `cancel` erreicht werden kann. Da dies hier die einzige Option darstellt, wird daraufhin ein Dialog vom Typ „Cancel“ begonnen, welcher im Falle erfolgreicher Durchführung einen Abbruch von `dialog-request` und damit der Bearbeitung der angefragten Aufgabe zur Folge hat.

5.4.1.5 Halte Agenda durchführbar

Während Verpflichtungen eines Teams nicht in einer Reihenfolge angeordnet werden müssen, da ein Team als Ganzes Verpflichtungen nicht tatsächlich bearbeitet, ist dies aus Sicht einer einzelnen *Supporting ACU* sehr wohl nötig, da hier die den Verpflichtungen zugeordneten Aufgaben tatsächlich bearbeitet werden. Hierzu wird der Begriff der Agenda eingeführt, in welcher die Verpflichtungen angeordnet werden. Konkret wird für jede Verpflichtung eines Akteurs eine Instanz der Klasse **`agenda-item`** ange-

legt (vgl. Abbildung 5-13), die auf die entprechende Verpflichtung (`commitment-1`, `commitment-2` bzw. `commitment-3`) verweist. Über Verknüpfungen mit dem Attribut *`next`* wird die Reihenfolge, in der die Verpflichtungen adressiert werden, festgelegt. Hier soll also zuerst `commitment-1`, dann `commitment-2` und schließlich `commitment-3` bearbeitet werden.

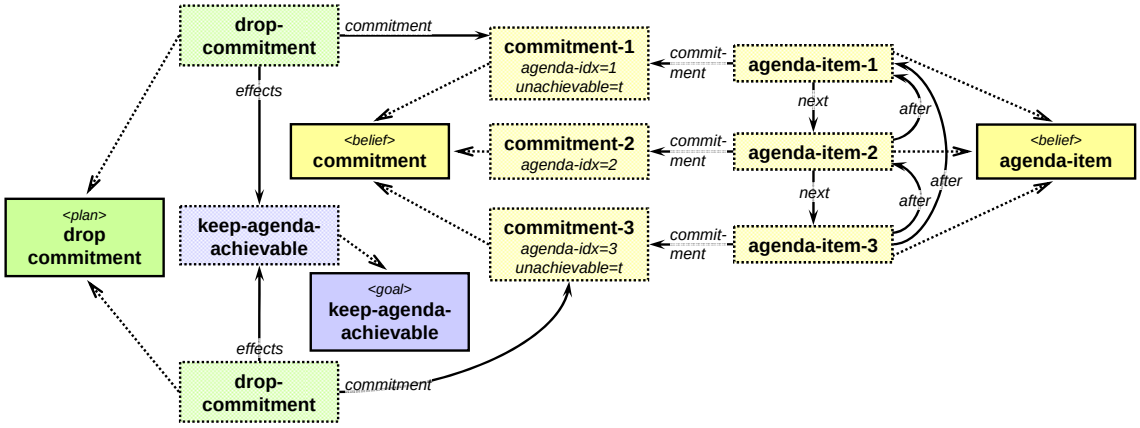


Abbildung 5-13: Gewährleistung der Durchführbarkeit der Agenda

Kommt eine ACU nun zu dem Schluss, dass einzelne Verpflichtungen für sie nicht erreichbar sind (siehe Attribut *`unachievable=t`* der Instanzen `commitment-1` und `commitment-3`), so ist für die Entscheidung, ob eine Verpflichtung aufgegeben werden soll, wesentliches Kriterium, dass die Agenda als ganzes bearbeitet werden kann. Daher wird der Wunsch „Halte Agenda durchführbar“ zum Ziel `keep-agenda-achievable` aktiviert, sobald mindestens eine Verpflichtung der Agenda als unerreichbar angesehen wird. Handlungsalternativen sind dann das Aufgeben der entsprechenden Verpflichtungen (`drop-commitment`). Können mehrere Verpflichtungen aufgegeben werden, so werden zunächst diejenigen aufgegeben, die am Ende der Agenda stehen (vgl. Attribut *`after`* von **`agenda-item`**).

5.4.1.6 Sortiere Verpflichtung in Agenda ein

Ist ein Akteur eine Verpflichtung eingegangen, so muss diese in seine Agenda eingeordnet werden. Hierzu wird der Wunsch „Sortiere Verpflichtung in Agenda ein“ zum Ziel `sort-commitment-into-agenda` aktiviert (vgl. Abbildung 5-14), welches auf die entsprechende Verpflichtung (hier: `commitment-2`) verweist.

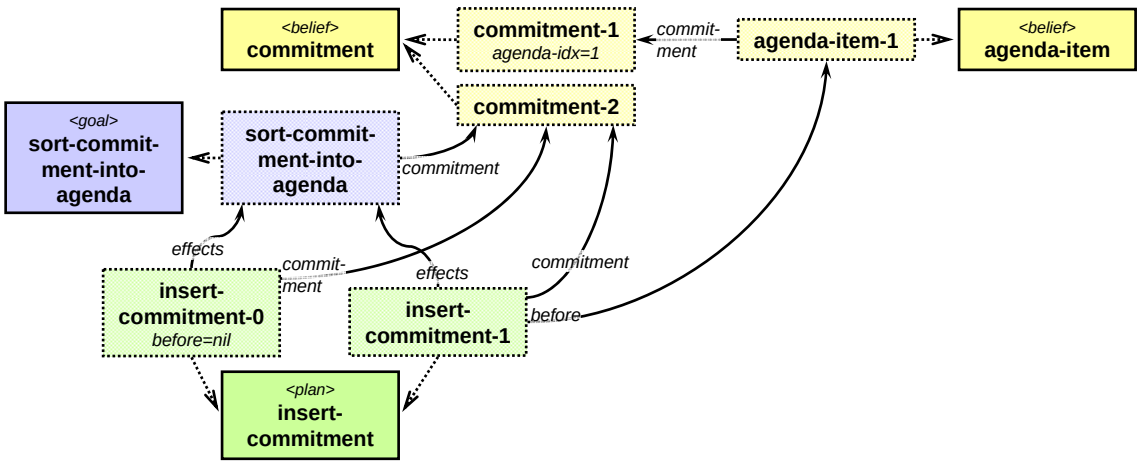


Abbildung 5-14: Einsortieren einer Verpflichtung in die Agenda

Die Handlungsalternativen **`insert-commitment`** unterscheiden sich durch die Position, an der die Verpflichtung in die Agenda eingeordnet werden kann. Das Planelement `insert-commitment-0` schlägt vor, die Verpflichtung an den Beginn der Agenda einzuordnen (*`before=nil`*), während `insert-commitment-1` die Einordnung nach `agenda-item-1` repräsentiert. Welche dieser Möglichkeiten ausgewählt wird, kann mit applikationsspezifischem Wissen gesteuert werden.

5.4.1.7 Bearbeite Verpflichtung

Die tatsächliche Bearbeitung von Aufgaben, zu denen eine Verpflichtung des eigenen Akteurs besteht, erfolgt wegen des Wunsches „Bearbeite Verpflichtung“. Dieser wird aktiviert, wenn eine Verpflichtung an erster Stelle in der Agenda steht (vgl. Abbildung 5-15, Ziel `comply-with-commitment`).

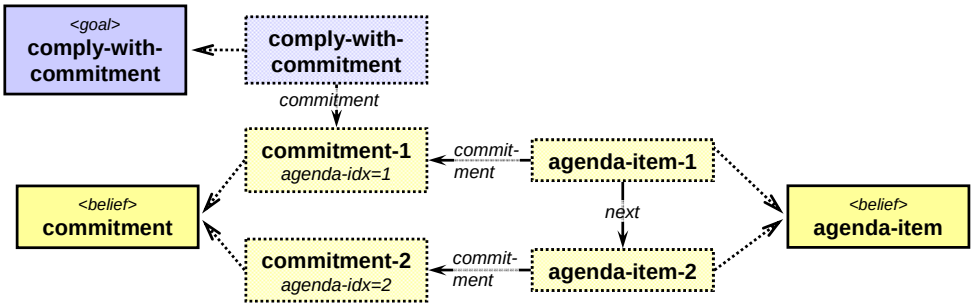


Abbildung 5-15: Bearbeitung einer Verpflichtung

Welche Handlungsalternativen zur Verfügung stehen, um dieses Ziel erreichen zu können, hängt von der spezifischen Anwendung ab, so dass an dieser Stelle nicht im Detail darauf eingegangen wird. Im Umfeld einer militärischen UAV-Mission, die unter anderem die Bearbeitung einer Aufgabe, eine SAM-Stellung zu bekämpfen, umfasst, können beispielsweise ähnliche Handlungsalternativen relevant sein, wie sie in Abschnitt 5.2 im Zusammenhang mit der Vermeidung einer Bedrohung dargestellt sind.

5.4.2 Bilden eines Teams

5.4.2.1 Kenne Teammitglied

Weiß ein Teammitglied nicht um Fähigkeiten, Ressourcen oder Verpflichtungen eines anderen Teammitglieds, so wird aus dem Wunsch „Kenne Teammitglied“ ein Ziel `know-team-member` aktiviert, um diese Art von Information zu gewinnen (siehe Abbildung 5-16).

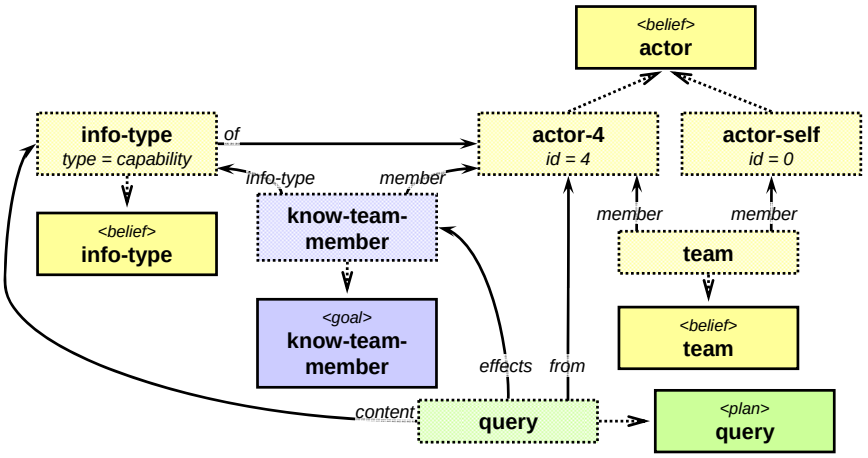


Abbildung 5-16: Kennenlernen eines Teammitglieds

Im dargestellten Fall kennt `actor-self` die Fähigkeiten von `actor-4` nicht (es ist keine Instanz der Klasse **`capability`** vorhanden, die die Fähigkeiten von `actor-4` repräsentiert), so dass `know-team-member` auf `actor-4` und auf eine Instanz der Klasse **`info-type`** verweist, die angibt, dass es sich hier um Information vom Typ „Fähigkeit" (*`type=capability`*) hinsichtlich `actor-4` (Attribut *`of`*) handelt.

Als Handlungsalternative steht hier die Anfrage der Information (`query`) direkt von `actor-4` (Attribut *`from`*) zur Verfügung, wobei mit dem Attribut *`content`* die gewünschte Information spezifiziert wird. In gleicher Weise kann Information anderer Art angefragt werden, wobei zusätzlich auch die Handlungsalternative **`subscribe`** zum Einsatz kommen kann, bei der Information nicht nur einmalig angefragt wird, sondern der Empfänger der Anfrage gebeten wird, immer dann nochmals eine Nachricht zu senden, wenn sich der Wert der angefragten Information geändert hat (vgl. Abschnitt 3.3.4).

5.4.2.2 Informiere Teammitglieder laufend

Teammitglieder können nicht nur Information von anderen anfragen, sondern sie stellen infolge des Wunsches „Informiere Teammitglieder laufend" auch anderen Teammitgliedern proaktiv Information zur Verfügung. Hierbei sind mehrere Aspekte von Bedeutung, nämlich wer von wem worüber informiert wird. In diesem Zusammenhang werden die folgenden Fälle abgedeckt:

- Jedes Teammitglied informiert die anderen Teammitglieder (d.h. das Team) über seine individuellen Ressourcen, Fähigkeiten, Verpflichtungen und den Bedarf an gemeinsam genutzten Ressourcen sowie relevante Informationen für die Bestimmung eines Teamsprechers.
- Jedes Teammitglied informiert den Operateur über seine individuellen Ressourcen, Fähigkeiten und Verpflichtungen.
- Der Sprecher eines Teams informiert sowohl die Mitglieder dieses Teams (d.h. das Team) als auch den Operateur darüber, dass er Teamsprecher ist und welche Verpflichtungen das Team hat.

Um das hierfür benötigte Wissen zu charakterisieren, werden beispielhaft zwei Konstellationen betrachtet, nämlich dass ein Teammitglied das Team über eine Änderung seiner vorhandenen Ressourcen informiert (siehe Abbildung 5-17) und dass ein Teamsprecher den Operateur über die Verpflichtungen des Teams informiert (siehe Abbildung 5-18).

In beiden Fällen wird die Art der bereitzustellenden Information mittels einer Instanz der Klasse **`info-type`** hinterlegt, wobei das Attribut *`type`* angibt, um welche Information es sich handelt (hier: *`resource`* bzw. *`commitment`*) und das Attribut *`of`* auf denjenigen verweist, auf den sich die Information bezieht (hier: `actor-self` bzw. `team`).

Da eine Informationsübermittlung nur dann stattfinden soll, wenn die entsprechende Information nicht bekannt ist oder sich geändert hat, muss eine ACU weiterhin wissen, welche Information dem Empfänger bekannt ist. Hierzu werden Instanzen der Klasse **`known-info`** angelegt. Sie verweisen auf die Art der Information (Attribut *`info-type`*) und denjenigen, der Kenntnis besitzt (Attribut *`known-by`*) und speichern in entsprechenden Attributen weiterhin den Wert der bekannten Information.

Wie bereits mehrmals angesprochen gilt auch hier, dass die Klassen **`info-type`** und **`known-info`** ebenso wie eine Repräsentation von Ressourcen (**`resource`**) und Fähigkeiten (**`capability`**) sowie gemeinsam genutzte Ressourcen stets vorhanden sein müssen, die konkrete Ausprägung von Attributen aber applikationsspezifisch ist.

In Abbildung 5-17 ist der Kenntnisstand des Teams `team` im Hinblick auf die relevante Information „Ressourcen des Akteurs mit der ID 0“ (vgl. `info-type`) aus Sicht der betrachteten ACU (ID 0) in der Instanz `known-info` hinterlegt. Diese stellt die Information bereit, dass das Team `team` (Attribut *`known-by`*) hinsichtlich der in `info-type` spezifizierten Information (Attribut *`info-type`*) meint, dass der Akteur mit der ID 0 zwei HARMs zur Verfügung hat (*`value-type=harm`*, *`value-total-amount=2`*), von denen eine noch nicht für die Nutzung im Rahmen der Bearbeitung bestehender Verpflichtungen eingeplant ist (*`value-free=1`*). Tatsächlich hat sich allerdings die Anzahl der zur Verfügung stehenden HARMs geändert (siehe Instanz `resource`) – es steht lediglich eine HARM zur Verfügung (*`total-amount=1`*), die außerdem schon für bestehende Verpflichtungen benötigt wird (*`free=0`*). Somit ist die dem Team bekannte Information überholt, so dass die Instanz `known-info` mit dem Attribut *`outdated`* gekennzeichnet und daraufhin das Ziel `keep-others-informed` aktiviert wird.

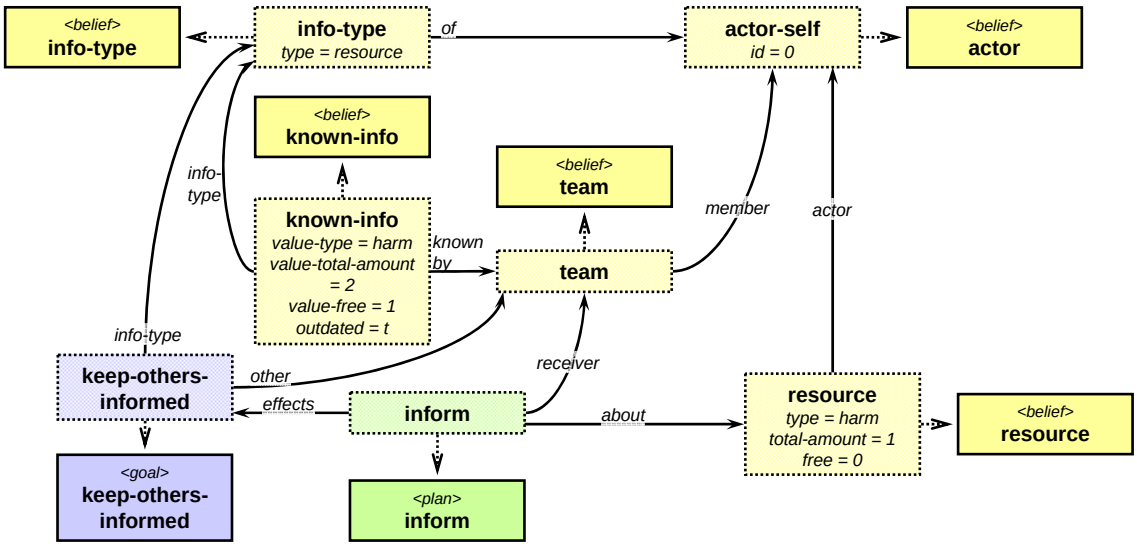


Abbildung 5-17: Information der Teammitglieder über geänderte Ressourcen eines Teammitglieds

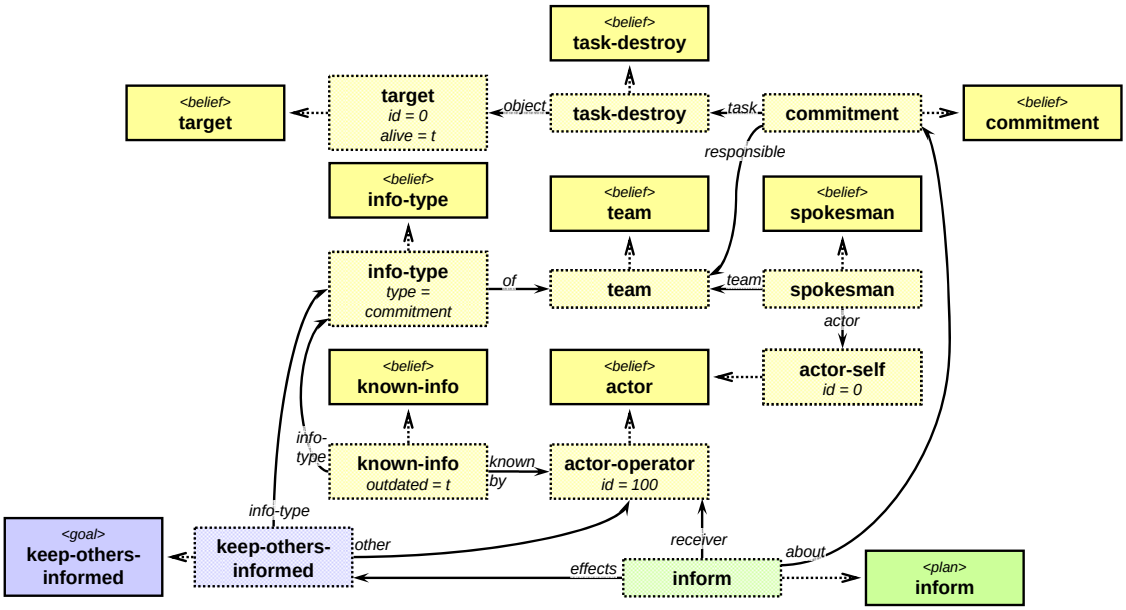


Abbildung 5-18: Information des Operateurs über Verpflichtungen eines Teams

Ähnlich verhält es sich bei dem in Abbildung 5-18 dargestellten Beispiel. Hier ist der Kenntnisstand des Operateurs (`actor-operator`) der, dass das Team keine Verpflich-

tungen hat, da in der Instanz *known-info* keine Attribute mit entsprechenden Werten belegt sind. Da es allerdings eine Verpflichtung des Teams gibt (vgl. Instanz commitment mit dem Attribut *responsible*, welches auf team verweist), ist auch diese bekannte Information veraltet und wird mit dem Attribut *outdated* gekennzeichnet. Auch hier wird deswegen das Ziel keep-others-informed aktiviert.

In beiden Fällen stellt die Handlungsalternative **inform** die einzige Möglichkeit dar, das Ziel zu erreichen und wird von dem im Paket „means-end“ hinterlegten Wissen (siehe Abschnitt 5.2) zur Ausführung vorgeschlagen. Daraufhin wird ein Dialog vom Typ „inform“ initiiert und entsprechend den in Abschnitt 5.3 erläuterten Mechanismen geführt, so dass dem Team bzw. Operateur der aktuelle Wert der für sie interessanten Information mitgeteilt wird. Sobald dieser Dialog erfolgreich abgeschlossen wurde, werden die Attribute der Instanzen von **known-info** aktualisiert. Da nun die aktuellen Werte bekannt sind, sind die Ziele keep-others-informed nicht mehr relevant und deaktivieren sich.

5.4.2.3 Stelle Aufgabenabdeckung sicher

Ist ein Team zur Durchführung von Aufgaben verpflichtet, so muss gewährleistet werden, dass all diese Aufgaben von einzelnen Teammitgliedern bearbeitet werden, was durch den Wunsch „Stelle Aufgabenabdeckung sicher“ erfolgt. Abbildung 5-19 stellt eine solche Situation dar, in der das Team team-all verpflichtet ist (commitment), die Aufgabe task-destroy durchzuführen, während kein Akteur (**actor**) eine individuelle Verpflichtung eingegangen ist.

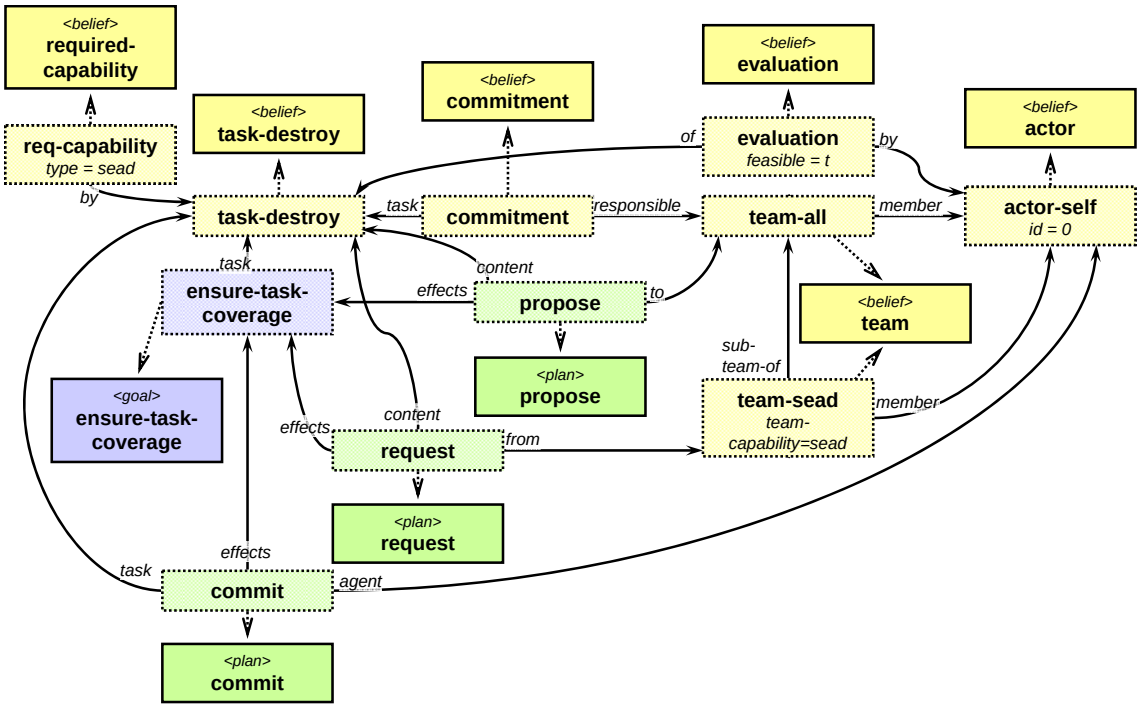


Abbildung 5-19: Sicherstellen der Aufgabenabdeckung

In dieser Situation wird der Wunsch **ensure-task-coverage** relevant und als Ziel aktiviert. Um dieses zu erreichen, stehen verschiedene Handlungsalternativen zur Verfügung. Da der Akteur selbst in der Lage ist, die Aufgabe durchzuführen (Instanz evaluation mit Attribut *feasible=t*), kann er dem Team team-all vorschlagen, die Aufgabe zu bearbeiten (propose) oder sich ohne vorherige Rückfrage zur Durchführung verpflichten (commit). Da es außerdem ein Team team-sead gibt, dessen Teammitglieder

alle über die Fähigkeit verfügen, die zur Durchführung der betreffenden Aufgabe benötigt wird (`req-capability` mit dem Attribut *`type=sead`* bzw. `team-sead` mit dem Attribut *`team-capability=sead`*), kann der Akteur außerdem eine Anfrage an dieses Team stellen, die Aufgabe zu bearbeiten (`request`).

Gibt es mehrere Instanzen von Handlungsalternativen, die geeignet sind, ein Ziel zu erreichen, so wird zunächst von dem im Paket „means-end“ implementierten Wissen eine zufällige Auswahl zwischen den Optionen getroffen (vgl. Abschnitt 5.2). Es kann allerdings zusätzliches Wissen hinterlegt werden, das die Selektion steuert. In dem im Rahmen dieser Arbeit entwickelten Prototyp ist beispielsweise hinterlegt, dass die Planelemente `propose` und `commit` einem `request` vorgezogen werden, während es keine Präferenz zwischen `propose` und `commit` gibt.

5.4.2.4 Verteile Arbeitsbelastung

Ist ein Teammitglied stark durch Verpflichtungen zu Aufgaben belastet, während ein anderes keine oder wenige Verpflichtungen hat, so wird der Wunsch „Verteile Arbeitsbelastung“ relevant. Abbildung 5-20 stellt eine Konfiguration dar, bei der das Teammitglied mit der ID 1 stark belastet ist (Instanz von **`workload`** mit dem Attribut *`is=high`*), während der eigene Akteur mit der ID 0 kaum belastet ist (`workload-0` mit dem Attribut *`is=low`*).

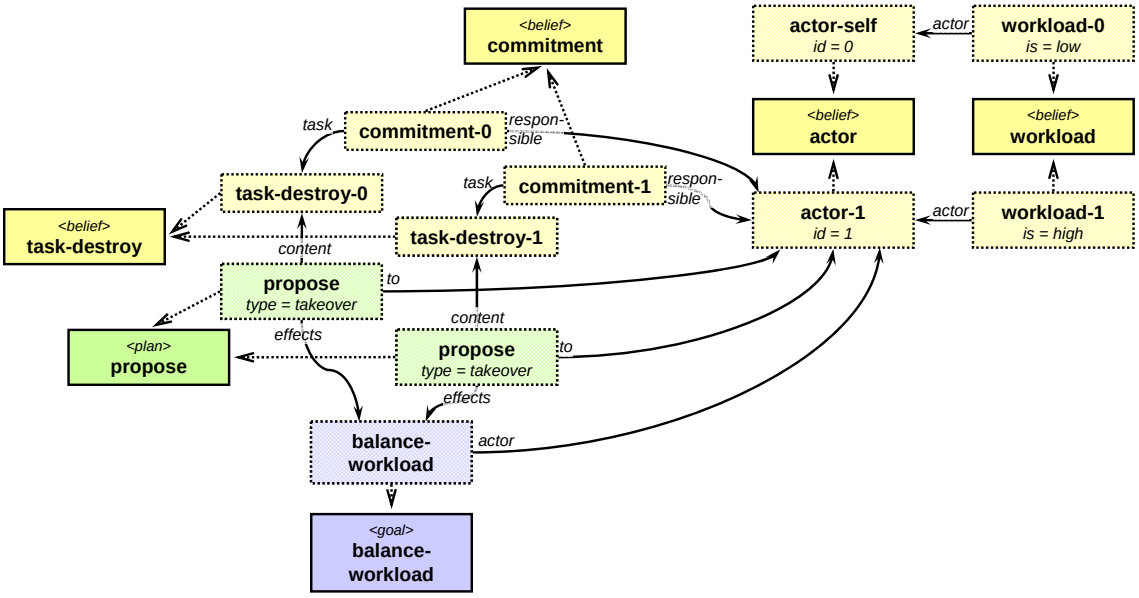


Abbildung 5-20: Verteilung der Aufgabenlast

Daraufhin wird der Wunsch **`balance-workload`** instanziiert, der auf `actor-1` als zu entlastendes Teammitglied verweist. Um dieses Ziel zu erreichen, kann der eigene Akteur dem Teamkollegen vorschlagen, eine der Aufgaben, zu der dieser verpflichtet ist (hier: `task-destroy-0`, `task-destroy-1`), zu übernehmen. Da hier angenommen wird, dass beide Aufgaben für den eigenen Akteur durchführbar sind, wird die Handlungsalternative **`propose`** zweimal instanziiert. Auch hier wurde (anwendungsspezifisches) Wissen zur Auswahl einer Option hinterlegt. So muss jede der hier dargestellten Aufgaben vom Typ „destroy“ an einem bestimmten Ort durchgeführt werden. Befindet sich nun der eigene Akteur näher an einem als an dem anderen Ort, so wird dieser Vorschlag bevorzugt.

Die Ausführung eines `propose`-Planelementes führt zur Initiierung eines Dialogs vom Typ „propose“. Im Gegensatz zu einem „üblichen“ Dialog vom Typ „propose“, bei dem

im Falle der Annahme des Vorschlags der Initiator eine Verpflichtung eingeht, werden hier durch die Kennzeichnung mit dem Typ „takeover“ bei der Annahme des Vorschlags zwei Aktionen ausgeführt: Der Initiator (hier: `actor-self`) geht eine Verpflichtung zur vorgeschlagenen Aufgabe ein, während gleichzeitig der Teilnehmer (hier: `actor-1`) seine Verpflichtung aufgibt.

5.4.2.5 Habe Teamsprecher

Wie in den Abschnitten 3.3.2 und 3.3.3 vorgestellt, werden im Team aus *Supporting ACUs* Sub-Teams entsprechend der Fähigkeiten der Teammitglieder gebildet, wobei in jedem (Sub-)Team die Rolle eines Teamsprechers besetzt werden muss, so dass der Wunsch „Habe Teamsprecher“ implementiert wurde. In Abbildung 5-21 sind drei Akteure (`actor-self`, `actor-1`, `actor-4`) dargestellt, wobei die ersten beiden hier über die Fähigkeit zur Unterdrückung gegnerischer Bedrohungen verfügen (SEAD), während letzterer die Fähigkeit „Attack“ besitzt. Wie bereits erwähnt, sind die betrachteten Fähigkeiten von der konkreten Anwendung abhängig, wenn auch stets Fähigkeiten als solche modelliert werden müssen. Alle drei Akteure sind Mitglieder des Teams `team-all` (vgl. Attribut *`member`*), in welchem bereits `actor-4` zum Teamsprecher bestimmt wurde (Instanz `spokesman` mit den Attributen *`team`* und *`actor`*).

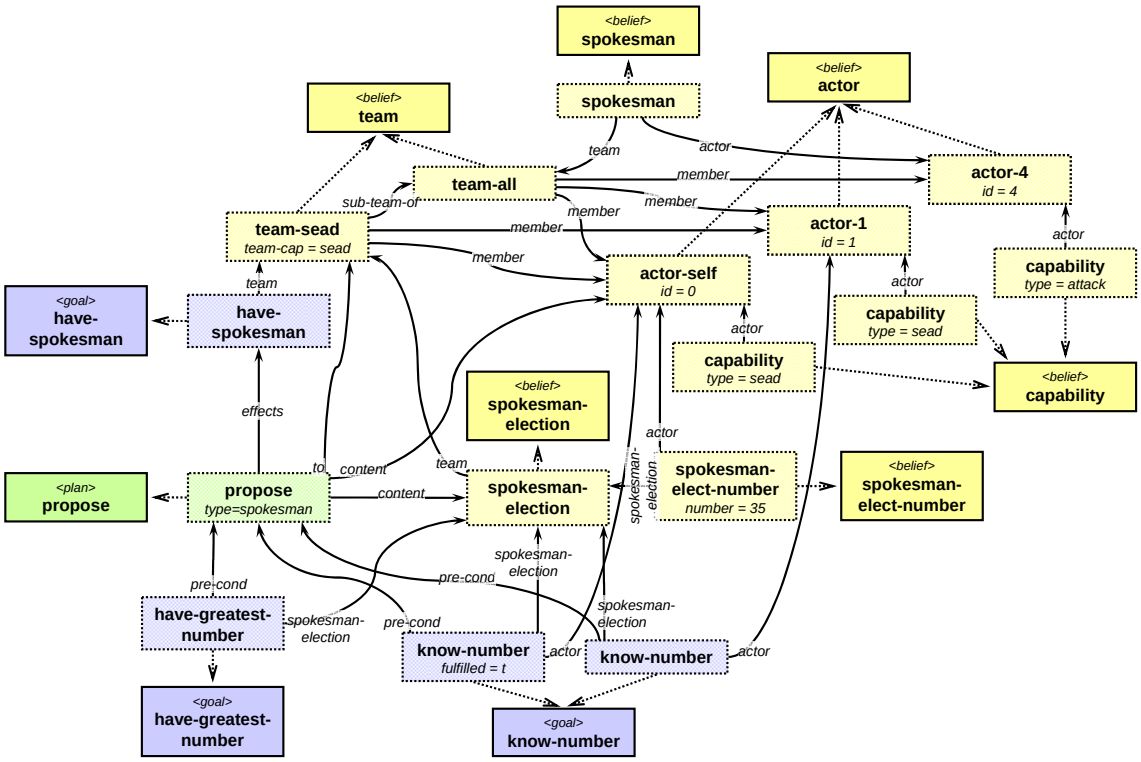


Abbildung 5-21: Bestimmung eines Teamsprechers

Da `actor-self` und `actor-1` über die gleiche Fähigkeit verfügen, sind sie Mitglieder des Sub-Teams `team-sead`, für das noch kein Teamsprecher bestimmt wurde. Daher wird das Ziel `have-spokesman` für dieses Team aktiviert und eine Instanz von **`spokesman-election`** für `team-sead` angelegt, auf die sich alle Informationen beziehen, die mit der Sprecherwahl zu tun haben. Das Ziel `have-spokesman` kann erreicht werden, indem ein Teammitglied vorschlägt, die Rolle des Teamsprechers zu übernehmen. Da hier eine Art Losverfahren implementiert wurde, bei dem jedes Teammitglied eine Zufallszahl bestimmt und derjenige mit der höchsten Zahl die Rolle übernehmen kann, müssen eine Reihe von Vorbedingungen erfüllt sein, bevor ein `propose` durchgeführt

werden kann. So muss eine Zahl für jedes Teammitglied bekannt sein (`know-number`) und der eigene Akteur die größte Zahl besitzen. Da `actor-self` bereits eine Zahl bestimmt hat (`spokesman-elect-number`), ist das Ziel `know-number`, welches mit dem Attribut *`actor`* auf `actor-self` verweist als erreicht gekennzeichnet (*`fulfilled=t`*). Sobald eine Zahl von `actor-1` bekannt ist, kann auch das Ziel `know-number`, welches mit dem Attribut *`actor`* auf `actor-1` verweist als erreicht gekennzeichnet werden. Ist die Zahl von `actor-self` größer als die von `actor-1`, so wird auch `have-greatest-number` mit *`fulfilled=t`* markiert und es kann ein Dialog vom Typ „propose“ initiiert werden. In diesem schlägt der Initiator dann vor, die Rolle des Teamsprechers zu übernehmen und tut dies, sofern die anderen Teammitglieder zustimmen.

5.4.3 Auflösen von Abhängigkeiten

5.4.3.1 Weise gemeinsam genutzte Ressource zu

Benötigen mehrere Akteure eine Ressource, die begrenzt verfügbar ist und von allen gemeinsam genutzt werden muss, so muss diese Abhängigkeit zunächst erkannt und anschließend aufgelöst werden, was auf Basis des Wunsches „Weise gemeinsam genutzte Ressource zu“ erfolgt. Das Vorgehen hierbei wird in Abbildung 5-22 anhand eines Beispiels dargestellt, bei dem mehrere UAVs (`ucav-0`, `ucav-1`), die von *Supporting ACUs* geführt werden (`actor-self`, `actor-1`), einen räumlich begrenzten Korridor (`corridor`) nutzen müssen, in den sie nicht gleichzeitig einfliegen können. Welche gemeinsam genutzten Ressourcen es gibt, und auf Basis welcher Kriterien deren Zuweisung erfolgen soll, hängt dabei von der betrachteten Applikation ab.

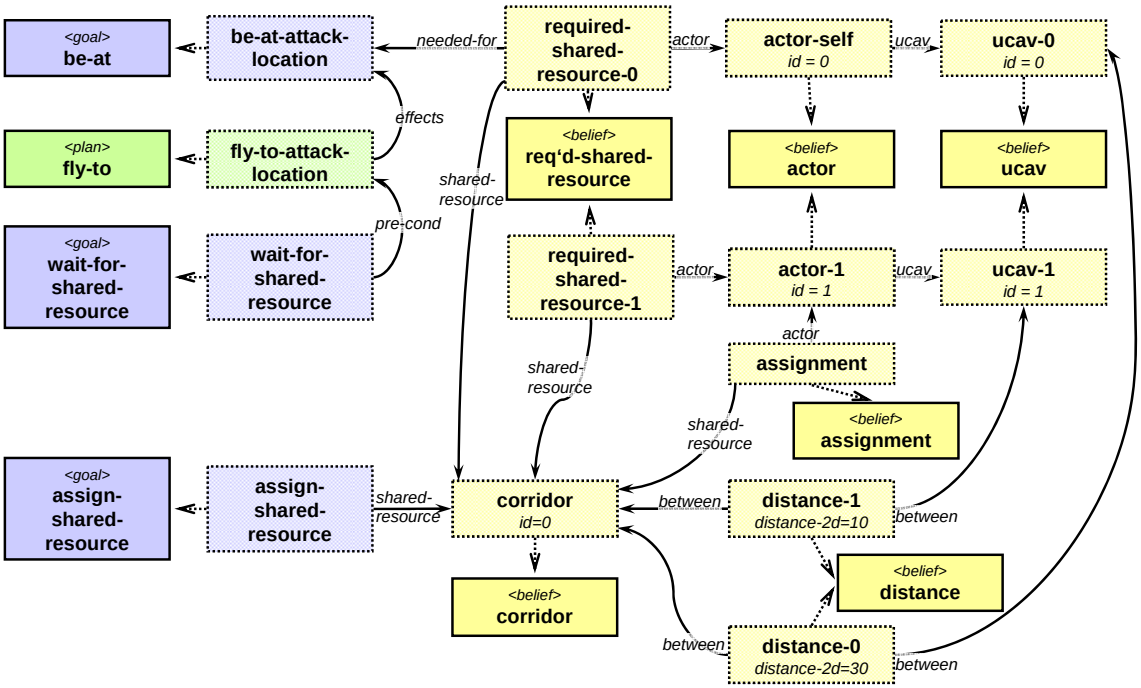


Abbildung 5-22: Zuweisung einer gemeinsam genutzten Ressource

Hier stellt jeder Akteur zunächst für sich fest, ob er aktuell eine gemeinsam genutzte Ressource benötigt – im dargestellten Beispiel ist dies der Fall, da der Korridor benötigt wird, um zu einer bestimmten Position fliegen zu können (vgl. Instanzen `be-at-attack-location` und `fly-to-attack-location`). Daher wird eine Instanz von **`required-share-resource`** angelegt, die auf die benötigte Ressource (Attribut *`shared-resource`*), den sie benötigenden Akteur (Attribut *`actor`*) und den Grund, warum sie

benötigt wird (Attribut *needed-for*) verweist. Auf Grund des in Abschnitt 5.4.2.2 beschriebenen Wissens wird dieser Bedarf allen Teammitgliedern mitgeteilt, die die Ressource ebenfalls benötigten könnten. Daher weiß der eigene Akteur, dass auch `actor-1` den Korridor benötigt (`required-shared-resource-1`). Dass nun mehrere Akteure die gleiche Ressource beanspruchen und deswegen diese Abhängigkeit aufgelöst werden muss, wird durch die Aktivierung des Ziels `assign-shared-resource` repräsentiert, welches auf die zuzuweisende Ressource verweist (Attribut *shared-resource*).

Die tatsächliche Zuweisung der Ressource erfolgt hier über einen impliziten Koordinationsmechanismus, bei dem – ähnlich wie im Straßenverkehr – allgemein bekannte Regeln das Vorgehen steuern. Im dargestellten Beispiel erfolgt die Zuweisung des Korridors nach dem Kriterium der Entfernung, d.h. der Korridor wird dem UAV, das sich am nächsten beim Korridor befindet, zugewiesen. Es werden also Instanzen der Klasse **`distance`** angelegt, welche jeweils die Entfernung zwischen einem UAV und dem Korridor repräsentieren (Attribute *between*) und die tatsächliche Distanz im Attribut *distance-2d* hinterlegen. Da nun die Entfernung zwischen `ucav-1` und `corridor` geringer ist als zwischen `ucav-0` und `corridor`, wird eine Instanz der Klasse **`assignment`** angelegt, die auf den Korridor als zugewiesene Ressource und `actor-1` als demjenigen, dem die Ressource aktuell zugewiesen ist, verweist.

Um nun zu verhindern, dass `actor-self` wie geplant seinen Flugweg fortsetzt, als ob er über die benötigte Ressource verfügen könnte, wird ein Ziel `wait-for-shared-resource` als Vorbedingung für die Ausführung von `fly-to-attack-location` instanziiert, welches die Ausführung dieses Planelementes so lange verhindert, bis die Ressource verfügbar ist.

5.4.3.2 Vermeide redundante Aufgabenzuweisung

Eine Abhängigkeit zwischen verschiedenen Akteuren besteht weiterhin, wenn die gleiche Aufgabe unnötigerweise von mehreren bearbeitet werden soll oder schon bearbeitet wird, d.h. eine Verpflichtung von mehreren Akteuren zu dieser Aufgabe besteht, so dass der Wunsch „Vermeide redundante Aufgabenzuweisung“ formuliert wurde.

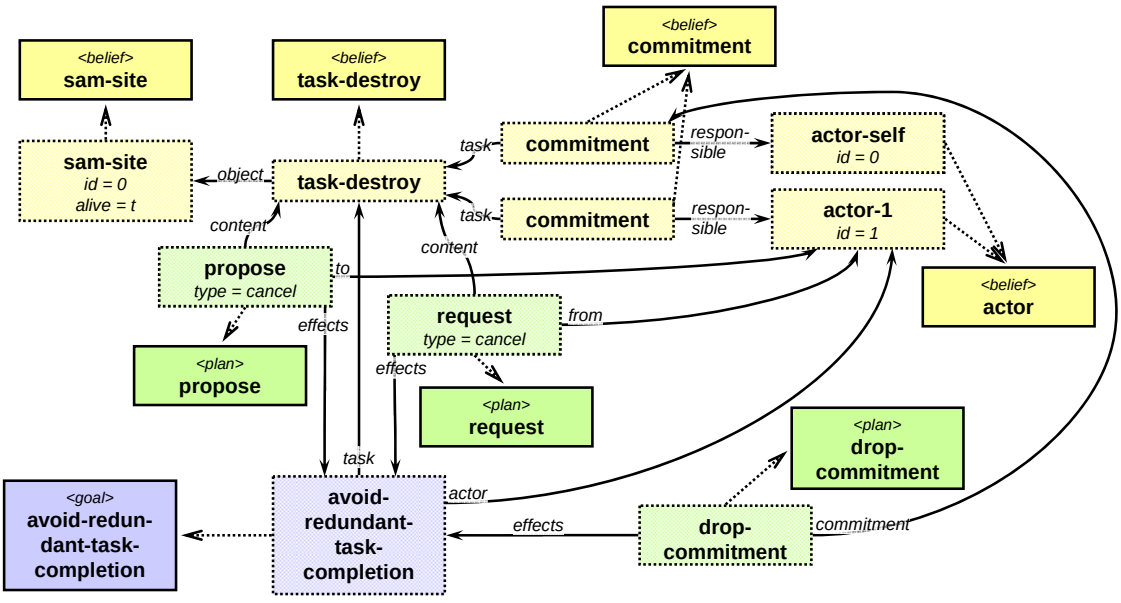


Abbildung 5-23: Vermeidung redundanter Aufgabenzuweisung

In Abbildung 5-23 stellt `task-destroy` diese Aufgabe stellvertretend für beliebige Aufgaben dar, wobei sowohl eine Verpflichtung (`commitment`) zu dieser Aufgabe von `actor-self` als auch von `actor-1` besteht. Daher wird das Ziel `avoid-redundant-task-completion` aktiviert, welches sowohl auf die Aufgabe, zu der mehrere Verpflichtungen bestehen, verweist (Attribut *`task`*) als auch auf den Akteur, der neben dem eigenen verpflichtet ist (Attribut *`actor`* mit Wert `actor-1`). Als Handlungsalternativen stehen **`propose`** und **`request`** zur Verfügung, die `actor-1` vorschlagen, dass der eigene Akteur die Verpflichtung aufgibt bzw. bei `actor-1` anfragen, ob dieser die Verpflichtung aufgibt. Dazu verweisen beide Instanzen auf die Aufgabe und `actor-1`. Daneben kann auch `drop-commitment` ausgewählt werden, infolge dessen die entsprechende Verpflichtung ohne Rückfrage bei einem anderen Akteur aufgegeben würde. Wie mehrfach angedeutet kann Wissen hinterlegt werden, das die Auswahl zwischen den Optionen steuert. Hier geschieht dies pauschal auf Basis der Entfernung zwischen dem UAV, welches von jedem Akteur geführt wird, und dem Ort, an dem die entsprechende Aufgabe durchzuführen ist.

5.4.3.3 Bearbeite unterstützende Aufgabe

Ist ein Mitglied des gleichen Teams wie der eigene Akteur zur Durchführung einer Aufgabe verpflichtet, so kann dieser unter Umständen dabei unterstützt werden, weshalb der Wunsch „Bearbeite unterstützende Aufgabe“ hinterlegt ist. In dem in Abbildung 5-24 dargestellten Beispiel besteht eine Verpflichtung (`commitment`) des Teammitglieds `actor-4` zur Aufgabe `task-destroy`, deren Bearbeitung durch Berücksichtigung der Aufgabe `task-suppress` unterstützt werden kann (Attribut *`supporting`*). Welche Aufgaben in der betrachteten Anwendung unter welchen Umständen die Durchführung einer anderen begünstigen und deswegen durch das Attribut *`supporting`* verknüpft werden müssen, muss dabei applikationsspezifisch modelliert werden. Hier unterstützt beispielsweise die Aufgabe `task-suppress`, die SAM-Stellung `sam-site` zu unterdrücken, deshalb die Durchführung der Aufgabe `task-destroy`, das Ziel `target` zu bekämpfen, weil die SAM-Stellung `sam-site` das Ziel `target` schützt (`covering`).

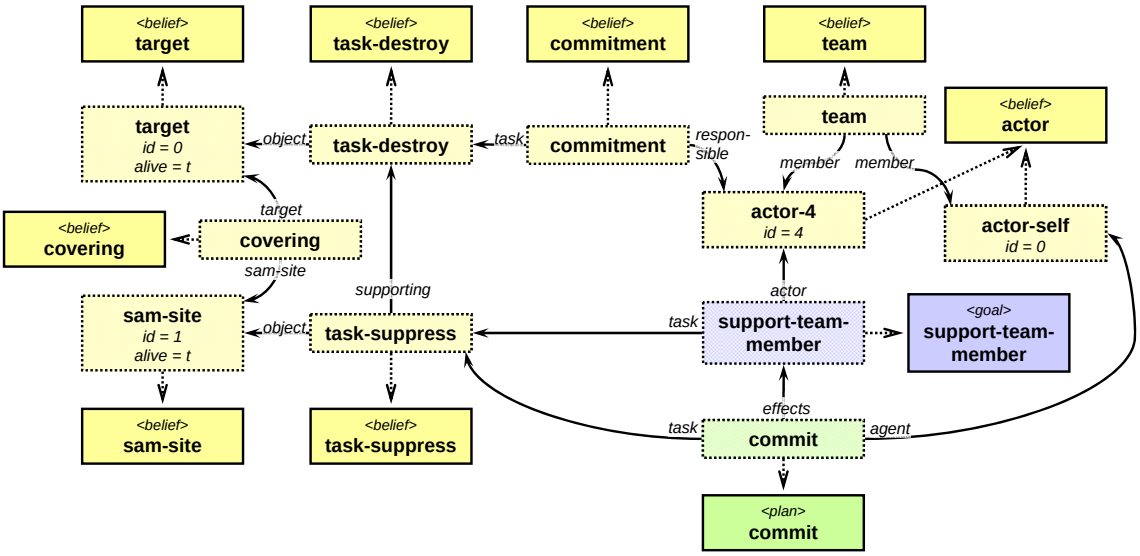


Abbildung 5-24: Unterstützung eines Teammitglieds

Da es noch keine Verpflichtung zu der unterstützenden Aufgabe (hier: `task-suppress`) gibt, wird das Ziel `support-team-member` aktiviert, welches zum einen auf das Teammitglied verweist, welches bei der Durchführung seiner Aufgaben unterstützt werden

kann (Attribut *actor*), und zum anderen auf die Aufgabe, mit deren Bearbeitung die Unterstützung konkret erfolgen kann (Attribut *task*). Sofern der eigene Akteur in der Lage ist, die betreffende Aufgabe zu übernehmen (vgl. Abschnitt 5.4.1.1), steht das Planelement `commit` zur Verfügung, das nach seiner Ausführung zu einer entsprechenden Verpflichtung führt.

5.4.4 Kommunikation

Die Implementierung des in dieser Arbeit verwendeten dialogbasierten Ansatzes zur Kommunikation zwischen *Supporting ACUs* (vgl. Abschnitt 3.3.3) wurde einschließlich des in diesem Zusammenhang relevanten Wunsches „Führe Dialog fort“ bereits in Abschnitt 5.3 als eine grundlegende Fähigkeit des in diesem Kapitel vorgestellten Funktionsprototyps erläutert, so dass hier auf diesen Abschnitt verwiesen werden kann.

5.4.5 Integration ins Arbeitssystem

5.4.5.1 Befolge Anweisung

Im Rahmen der hierarchischen Unterordnung der *Supporting ACUs* als Arbeitsmittel für einen Operateur sollen diese primär selbständig Aufträge vom Operateur bearbeiten können. Daneben soll der Operateur die Möglichkeit haben, die Art und Weise beeinflussen zu können, wie ein Auftrag durchgeführt wird. Hierzu übermittelt er so genannte Anweisungen an die Arbeitsmittel. Einem Team von *Supporting ACUs* können dabei drei verschiedene Anweisungen geschickt werden: Das Eingehen und Aufgeben einer Verpflichtung sowie das Aufheben einer vorhergehenden Anweisung. In allen drei Fällen verläuft die Zusammenarbeit der Wissensmodelle ähnlich (vgl. Abbildung 5-25, Abbildung 5-26 und Abbildung 5-27). Sobald in Folge eines Dialogs vom Typ „Instruct“ eine Anweisung vorhanden ist (`instruction`), die nicht bereits erfolgreich bearbeitet wurde und die an ein Team gerichtet ist (Attribut *consignee*), dessen Teamsprecher (`spokesman`) der eigene Akteur ist (`actor-self`), wird eine Instanz des Wunsches „Befolge Anweisung“ (`comply-with-instruction`) erzeugt. Je nach Art der Anweisung kann dieses Ziel unterschiedlich erreicht werden. Handelt es sich um eine Anweisung, sich zu einer je nach Anwendung unterschiedlich ausgeprägten Aufgabe zu verpflichten (Instanz `instruction` mit Attribut *action=commit*), so ist zum Beispiel die Handlungsalternative **`commit`** geeignet (siehe Abbildung 5-25).

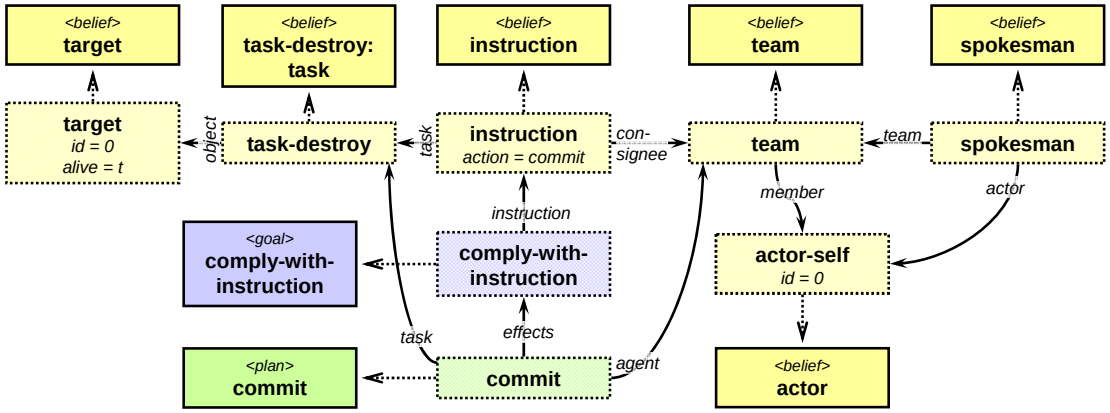


Abbildung 5-25: Durchführung der Anweisung "Eingehen einer Verpflichtung"

Soll eine vorhandene Verpflichtung (`commitment`) des Teams aufgegeben werden (Instanz `instruction` mit Attribut *action=drop-commitment*), wird die Handlungsalternative **`drop-commitment`** instanziiert (siehe Abbildung 5-26).

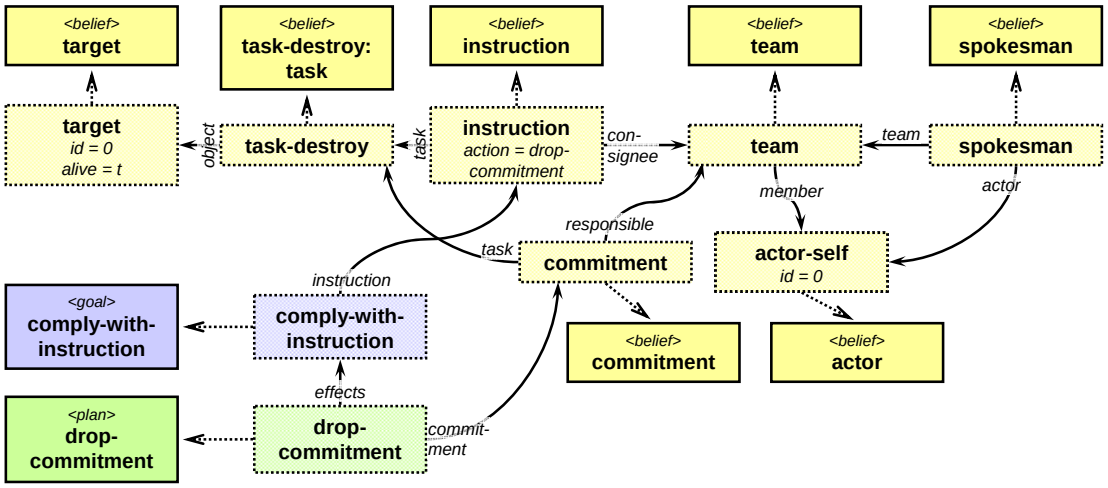


Abbildung 5-26: Durchführung der Anweisung "Aufgeben einer Verpflichtung"

In beiden Fällen verweist die Anweisung auf eine anwendungsspezifische Aufgabe, die hier von einer übergeordneten Klasse **`task`** abgeleitet wird, um als Aufgabe erkennbar zu sein und über Attribute zu verfügen, die für das hier beschriebene Wissen notwendig sind. Je nach Aufgabenstellung müssen vom Entwickler andere Aufgabentypen als Spezialisierungen dieser Klasse implementiert werden.

Besteht die Anweisung schließlich darin, eine vorhergehende Anweisung, welche mittels einer Kennung (*`instruction-id`*) gekennzeichnet wird, aufzuheben (Instanz `instruction` mit Attribut *`action=delete-instruction`*), so kommt schließlich die Handlungsalternative **`delete-instruction`** zum Tragen (siehe Abbildung 5-27).

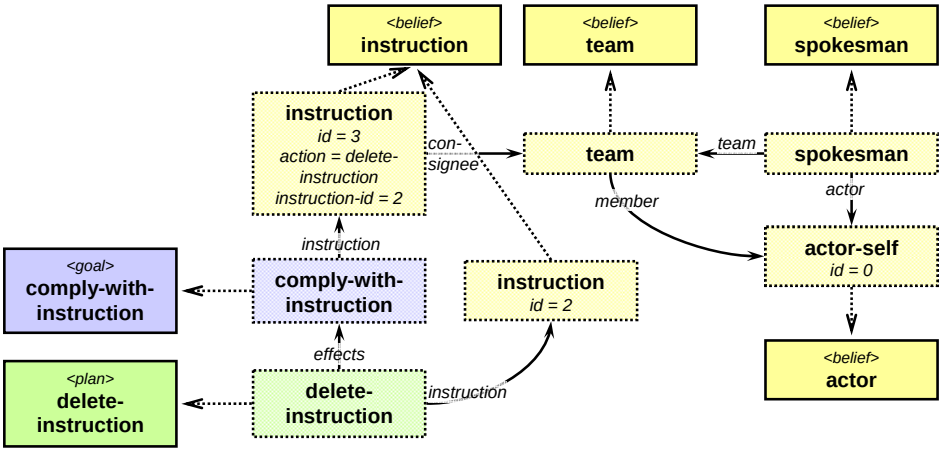


Abbildung 5-27: Durchführung der Anweisung "Aufheben einer Anweisung"

6 Evaluierung

Um künstliche kognitive Einheiten wie sie in Kapitel 2 als *Supporting ACUs* in ein Arbeitssystem zur Führung mehrerer UAVs eingeführt wurden in die Lage zu versetzen, auf wissensbasierter Verhaltensebene zu kooperieren, wurde in Kapitel 3 ein Konzept vorgestellt, welches die Aspekte künstlicher Kognition auf Basis des Kognitiven Prozesses und Kooperation zusammenführt. In Kapitel 4 wurde die Modellierung des für Kooperation relevanten a-priori Wissens im Sinne des Kognitiven Prozesses erläutert, dessen Umsetzung als einzelne, implementierte Wissensmodelle und deren Zusammenwirken in Kapitel 5 beschrieben wurde. Gegenstand dieses Kapitels ist nun die Evaluierung von Konzept und Implementierung unter verschiedenen Gesichtspunkten anhand des lauffähigen Systems.

Zunächst wird in Abschnitt 6.1 die Evaluierungsumgebung vorgestellt, die den beschriebenen Prototyp in eine Beispielanwendung integriert, wobei als Szenario eine vereinfachte, kooperative SEAD-/Attack-Mission gewählt wird. Der Nachweis, dass mit dem gewählten Ansatz die gewünschte Funktionalität erreicht werden kann, erfolgt anschließend sowohl im Hinblick auf Teilfunktionalitäten wie sie durch die Ziele von Kooperation definiert werden als auch deren Zusammenspiel im gesamten Szenario (Abschnitt 6.2). In einem zweiten Schritt wird wissensbasiertes Verhalten der *Supporting ACUs* nachgewiesen (Abschnitt 6.3). Schließlich wird in das Team aus *Supporting ACUs* ein menschliches Teammitglied integriert, um erste Erkenntnisse hinsichtlich der Eignung des Ansatzes für Mensch-Maschine-Kooperation im Hinblick auf die in Abbildung 2-10 dargestellte Arbeitssystemkonfiguration zu gewinnen (Abschnitt 6.4).

Das im Folgenden beschriebene Verhalten der ACUs und damit das der UAVs entspricht nicht immer üblichen Vorgehensweisen im realen militärischen Umfeld, wo beispielsweise stets vor Missionsbeginn eine Rollen- und Aufgabenzuweisung an die beteiligten Teammitglieder vorgenommen wird. Im Rahmen der vorliegenden Arbeit wurde hingegen bewusst auf eine Vorgabe von initialen Randbedingungen verzichtet, um die Anforderung nach maximaler Flexibilität aufzustellen und schließlich diese auch zu demonstrieren. Eine Einschränkung der Freiheitsgrade durch Missionsvorgaben oder andere Randbedingungen kann allerdings jederzeit in das Konzept und den Funktionsprototypen integriert werden. Im Übrigen orientieren sich die funktionalen Anforderungen an dem im Folgenden beschriebenen, bewusst sehr einfach gehaltenen Szenario, welches extern vorgegeben war.

6.1 Evaluierungsumgebung

6.1.1 Szenario

Die Erprobung der Fähigkeiten der entwickelten kooperativen *Supporting ACUs* findet anhand einer SEAD-/Attack-Mission statt. Diese basiert auf einem Szenario, das im Rahmen einer GARTEUR-Arbeitsgruppe (*Group for Aeronautical Research and Technology in Europe*) entwickelt wurde [Platts et al., 2004] und dieses um einige Aspekte erweitert (vgl. auch [Meitinger & Schulte, 2007]). Hierbei sollen fünf UCAVs (*Uninhabited Combat Aerial Vehicles*) zusammenarbeiten, um infolge eines Auftrags von einem Operateur ein gegnerisches militärisches Ziel zu bekämpfen (vgl. Abbildung

6-1). Dieses Ziel wird von SAM-Stellungen geschützt, die zum Teil vor Missionsbeginn bekannt sind, aber auch anderen, die im Verlauf der Mission unerwartet auftauchen können („Pop-Ups“). Vier der fünf UCAVs (im Folgenden *SEAD-UCAVs* genannt) sind in der Lage, SAM-Stellungen auszuschalten bzw. temporär zu unterdrücken, während das fünfte (im Folgenden *Attack-UCAV* genannt) über die notwendige Bewaffnung verfügt, um das gegnerische Ziel zu bekämpfen. Da sich das Attack-UCAV nicht vor einer Bedrohung durch SAM-Stellungen schützen kann, ist eine Zusammenarbeit zwischen den UCAVs zwingend notwendig, um eine erfolgreiche Durchführung der Mission zu ermöglichen. Ausgangs- und Endpunkt der Mission ist der heimische Stützpunkt, von dem aus die FLOT (*Forward Line of Own Troops*) innerhalb eines gegebenen Korridors überquert werden muss, um in das bedrohte Gebiet zu gelangen.

Die SEAD-UCAVs sind mit einem *Radar Warning Receiver* begrenzter Empfindlichkeit ausgestattet, der es zunächst ermöglicht, die Richtung, in der ein Radar strahlt, mit einer gewissen Genauigkeit zu bestimmen. Basierend auf den Vorgaben von [Platts et al., 2004] werden die Fähigkeiten dieses Sensors in der vorliegenden Implementierung des Szenarios insofern ausgeweitet, als den UCAVs nicht nur die Richtung, sondern auch die Position von Radarsendern, d.h. SAM-Stellungen, im Sinne eines *Emitter Locator Systems* zugänglich wird. Des Weiteren verfügen die SEAD-UCAVs über einen *Missile Approach Warner*, der angibt, ob eine SAM-Stellung und wenn ja welche das jeweilige UCAV angreift und in welcher Entfernung sich ein herannahender Flugkörper befindet. Im Gegensatz zu den SEAD-UCAVs verfügt das Attack-UCAV nicht über derartige Sensoren.

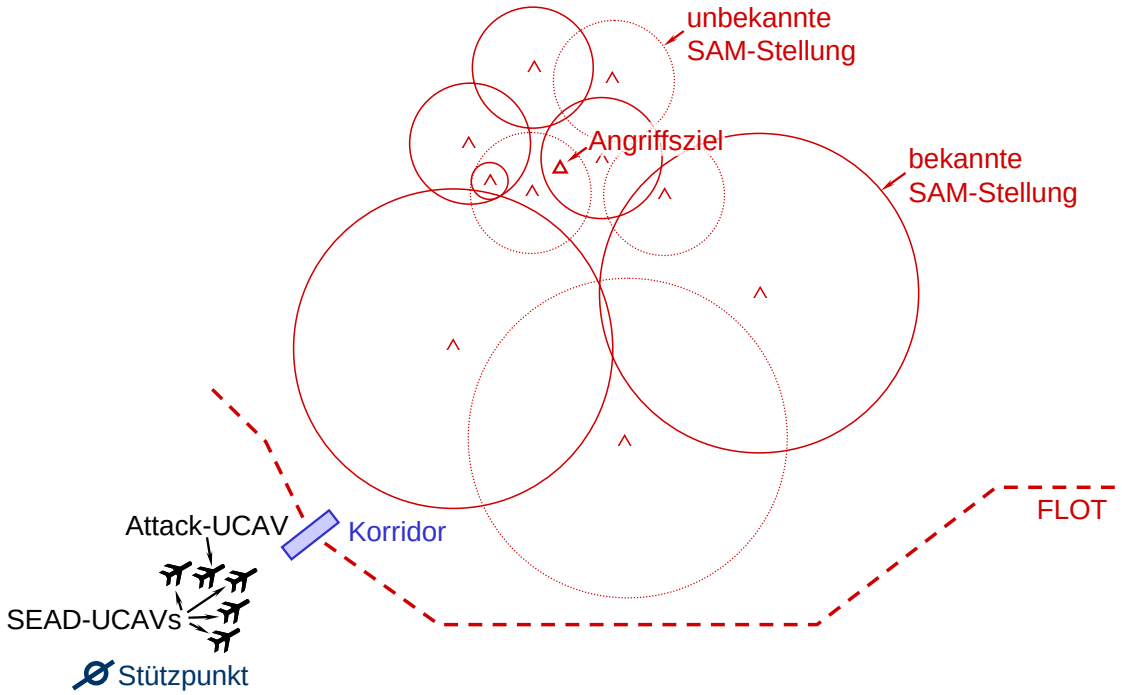


Abbildung 6-1: Szenarienelemente (FLOT: Forward Line of Own Troops, UCAV: Uninhabited Combat Aerial Vehicle, SEAD: Suppression of Enemy Air Defense, SAM: Surface to Air Missile)

Ausgehend von dem in [Platts et al., 2004] dargestellten Szenario und militärisch verfügbaren Technologien zum Datenaustausch (z.B. MIDS/Link 16) wird ideale Kommunikation zwischen den UCAVs angenommen. Zum einen bewirkt dies, dass gesendete Nachrichten immer den Empfänger erreichen. Zum anderen kennen alle UCAVs den Steuerkurs und die Position der anderen UCAVs sowie die Position und den Zustand aller neu entdeckten SAM-Stellungen.

Die SEAD-UCAVs verfügen über eine begrenzte Anzahl HARMs (*High-Speed Anti Radiation Missiles*), mit denen sie SAM-Stellungen bekämpfen können. Eine HARM kann auf eine SAM-Stellung abgefeuert werden, sofern deren Position bekannt ist und die Voraussetzungen für einen Waffeneinsatz erfüllt sind. Dies ist der Fall, wenn sich die SAM-Stellung innerhalb der Waffenreichweite befindet und das UCAV auf die SAM-Stellung hin ausgerichtet ist. Das Attack-UCAV verfügt über Bewaffnung zur Bekämpfung des Ziels, welche über dem Ziel eingesetzt werden muss.

Die zu entwickelnden ACUs haben Zugriff auf Zustandsdaten des eigenen UCAVs, d.h. Position, Kurs, Autopiloten- und Flugmanagementdaten, ebenso wie auf Sensordaten und verfügbare Ressourcen. Eine Steuerung der UCAVs erfolgt über das Einstellen von Autopilotenwerten und -betriebsarten für Geschwindigkeit, Höhe und Kurs oder über das Flugmanagementsystem. Letzteres kann Flugpläne erstellen, diese in verschiedenen Betriebsarten abfliegen und das UCAV an einer Stelle kreisen oder in Formation mit einem anderen UCAV fliegen lassen. Darüber hinaus besteht die Möglichkeit für die ACUs, die vorhandene Bewaffnung einzusetzen (vgl. [Meitinger & Schulte, 2007]).

6.1.2 Problemstrukturierung

Um die im vorherigen Abschnitt skizzierte Problemstellung lösen zu können, sind in Anlehnung an [Platts et al., 2007] Fähigkeiten in den folgenden Teilbereichen nötig:

- *Flugführung eines einzelnes Vehikels*, d.h. Bereitstellung und Ansteuerung von Automationsfunktionen wie zum Beispiel Flugmanagementsystem, Autopilot, Trajektorienerzeugung und -verfolgung
- *Flugführung mehrerer Vehikel*, d.h. Kollisionsvermeidung und kooperative Flugwegplanung
- *Verhaltenssteuerung eines einzelnen Vehikels*, d.h. Durchführung konkreter Aufgaben einschließlich taktischer Verfahren zum Beispiel zur Bedrohungsvermeidung
- *Verhaltenssteuerung mehrerer Vehikel*, d.h. Aufgabenzuweisung, strategische Zusammenarbeit

Dementsprechend müssen alle diese Punkte in einem ausgeführten System adressiert werden. Da der Schwerpunkt der vorliegenden Arbeit auf der Verhaltenssteuerung mehrerer Vehikel liegt, wurden die anderen Punkte, insbesondere die, welche die Flugführung betreffen, so vereinfacht und abstrahiert betrachtet, dass die für ein lauffähiges Gesamtsystem notwendige Minimalfunktionalität verfügbar war.

6.1.3 Simulationsumgebung

Die Evaluierungsumgebung setzt sich im Wesentlichen aus einer Simulation des beschriebenen Szenarios (rechter Teil in Abbildung 6-2) und der Implementierung der *Supporting ACUs* zusammen (linker Teil in Abbildung 6-2).

Die tatsächliche Simulation des Szenarios findet hierbei in der „Szenariensimulation“ statt, welche mittels eines Terminalprogramms („Termio“) gesteuert wird, über das auch Änderungen des Szenarios eingespielt werden können. Des Weiteren hat sie Zugriff auf eine Flugplanungsfunktion für die einzelnen UCAVs. Die zwischen den ACUs bzw. zwischen ACUs und Operateur ausgetauschten Nachrichten werden über den Prozess „Nachrichtenverwaltung“ vom Absender einer Nachricht an die entsprechenden Empfänger weitergeleitet. Der Operateur kann über eine Konsole Nachrichten an die ACUs senden bzw. von diesen empfangene einsehen und sich einen Überblick über das Szenario verschaffen, soweit es den UAVs bekannt ist.

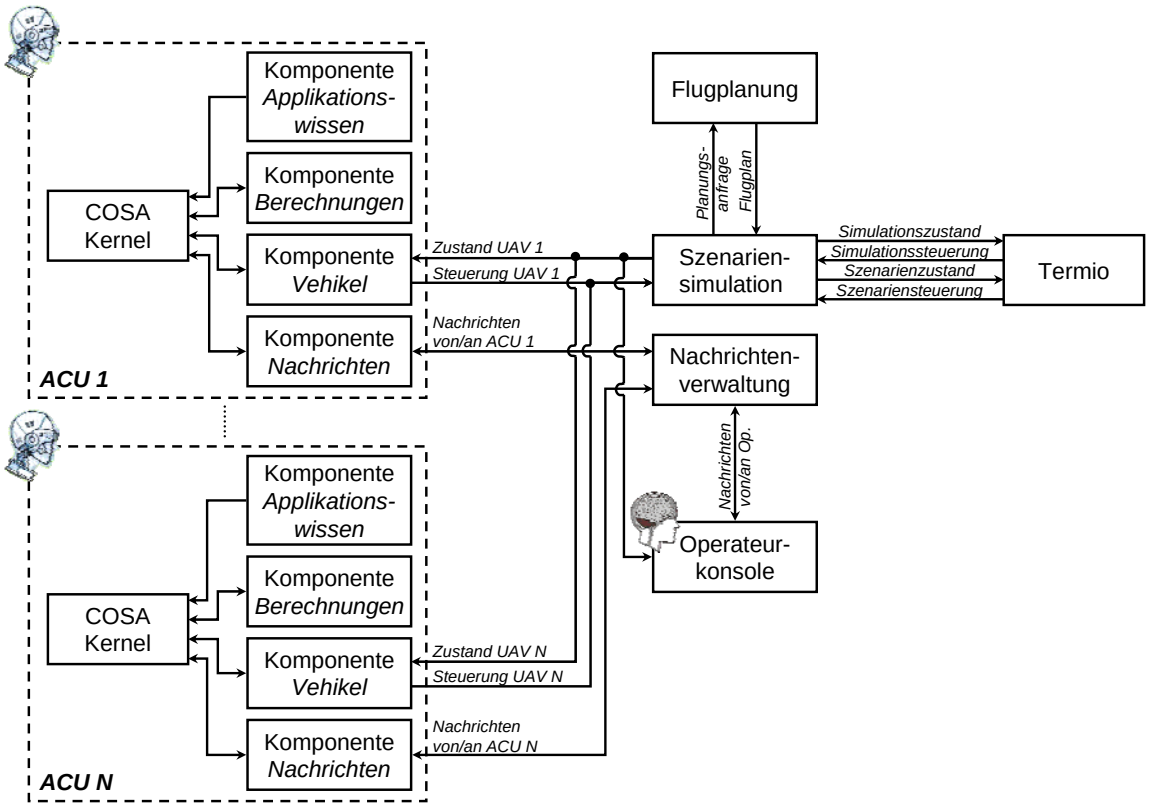


Abbildung 6-2: Evaluierungsumgebung

Die ACUs sind wie in Kapitel 5 erläutert als COSA-Applikation implementiert, wobei je eine ACU für die Führung eines UCAVs zuständig ist. Grundlage für das Verhalten der ACUs stellt das Applikationswissen dar, wobei eine Reihe weiterer Komponenten mit Servern für die Anbindung zur Außenwelt („Vehikel“ und „Nachrichten“) sowie für die Durchführung von numerischen Berechnungen zur Verfügung stehen. Neben dem in Kapitel 5 ausführlich vorgestellten Wissen hinsichtlich Problemlösung, Kommunikation und Kooperation wird im Sinne der im vorigen Abschnitt erläuterten Fähigkeiten weiteres, zumeist applikationsspezifisches Wissen benötigt, so dass sich der gesamte Funktionsprotoyp im Einzelnen aus den folgenden Paketen zusammensetzt:

- ***Kooperation*** – enthält Wissen, das zur koordinierten Zusammenarbeit im Team notwendig ist. Hierunter fällt Wissen um den Operateur, das Team und die Teammitglieder, deren Aufgaben und die Gestaltung der Zusammenarbeit.
- ***Kommunikation*** – enthält Wissen hinsichtlich der Initiierung und des Ablaufs von Dialogen verschiedenen Typs sowie der Bestimmung des aktuellen Zustands und der Fortführung von Dialogen.
- ***Problemlösung*** – enthält Wissen über zu treffende Entscheidungen und die Reihenfolge der Abarbeitung von Planelementen, die festgelegt werden muss, wenn mit Hilfe des Paradigmas der Mittel-Ziel-Analyse das weitere Vorgehen geplant wird.
- ***Mission*** – enthält Wissen, das zur konkreten Durchführung einzelner Aufgaben nötig ist.
- ***Umgebung*** – enthält Wissen über konkrete Objekte der Umgebung wie zum Beispiel UCAVs und SAM-Stellungen, abstrakte Konstrukte wie zum Beispiel Flugpläne und Relationen zwischen solchen Objekten und/oder Konstrukten wie zum Beispiel Entfernung, die vor allem für die Lagebeurteilung genutzt werden.

- ***Sicherheit*** – enthält Wissen hinsichtlich der Wahrung der Flugsicherheit und Vermeidung taktischer Bedrohung.
- ***Priorisierung*** – enthält Wissen über die Priorisierung von Zielen, welche nicht gleichzeitig verfolgt werden können.
- ***Überwachung*** – enthält Wissen hinsichtlich der Bearbeitung von Anweisungen, welche vom Operateur eingehen können.

Tabelle 6-1 gibt einen Überblick über den Umfang dieser Pakete im Hinblick auf die Anzahl der Klassen (aufgeschlüsselt nach Anzahl der Umweltmodelle, Wünsche, Handlungsalternativen und Anweisungsmodelle) sowie die Anzahl der implementierten CPL-Regeln, die das Verhalten der Klassen bilden. Zudem ist die Anzahl der Soar-Produktionen angegeben, die durch Kompilieren des CPL-Codes zustande kommen und letztlich im Soar-Prozessor verarbeitet werden.

Paket	***Anzahl Klassen***					***Anzahl CPL-Regeln***	***Anzahl Soar-Regeln***
	Umwelt-modelle	*Wünsche*	*Handlungs-alternativen*	*Anweisungs-modelle*	***gesamt***		
Kooperation	37	15	3	-	*55*	*344*	440
Kommunikation	12	2	7	1	*22*	*192*	216
Problemlösung	-	1	1	-	*2*	*17*	21
Mission	2	9	8	8	*27*	*124*	154
Umgebung	26	-	-	-	*26*	*158*	208
Sicherheit	-	5	1	1	*7*	*23*	32
Priorisierung	-	1	-	-	*1*	*28*	31
Überwachung	2	1	1	-	*4*	*26*	37
gesamt	**79**	**34**	**21**	**10**	***144***	***912***	**1139**

Tabelle 6-1: Überblick über den Umfang der Pakete des Funktionsprototyps

In dieser Übersicht wird deutlich, dass der Schwerpunkt der Implementierung auf der Bearbeitung von Fragestellungen der Kooperation und Kommunikation sowie dafür benötigten Umgebungsmodellen liegt. Auf diese Komplexe entfallen rund zwei Drittel aller implementierten Regeln, während das hinterlegte Wissen im Hinblick auf missionsspezifisches Verhalten eher minimalistisch ausgeprägt ist. Ferner fällt auf, dass mehr als die Hälfte der Klassen Umweltmodelle sind. Diese tragen im Sinne kognitiver Verarbeitung zum Aufbau des Situationsverständnisses bei, welches für wissensbasiertes Verhalten elementar ist. Eine Auflistung der in den einzelnen Paketen implementierten Modelle findet sich in Anhang A. Für eine detailliertere Beschreibung dieser Modelle wird hier auf [Meitinger & Schulte, 2007] verwiesen, worin eine Version der Implementierung zum Zeitpunkt der Projektberichtserstellung umfassend dokumentiert ist.

Die zur Evaluierung verwendete Rechnerumgebung ist in Abbildung 6-3 dargestellt. Hierbei handelt es sich um sechs Linux-PCs, welche über einen Switch mit einer Übertragungsgeschwindigkeit von 1 GBit vernetzt sind und mittels NFS (Network File System) auf zentral zur Verfügung gestellten Massenspeicher zugreifen. Weitere Details zu den verwendeten Betriebssystemen, Prozessoren und der Arbeitsspeichergröße können der Abbildung entnommen werden.

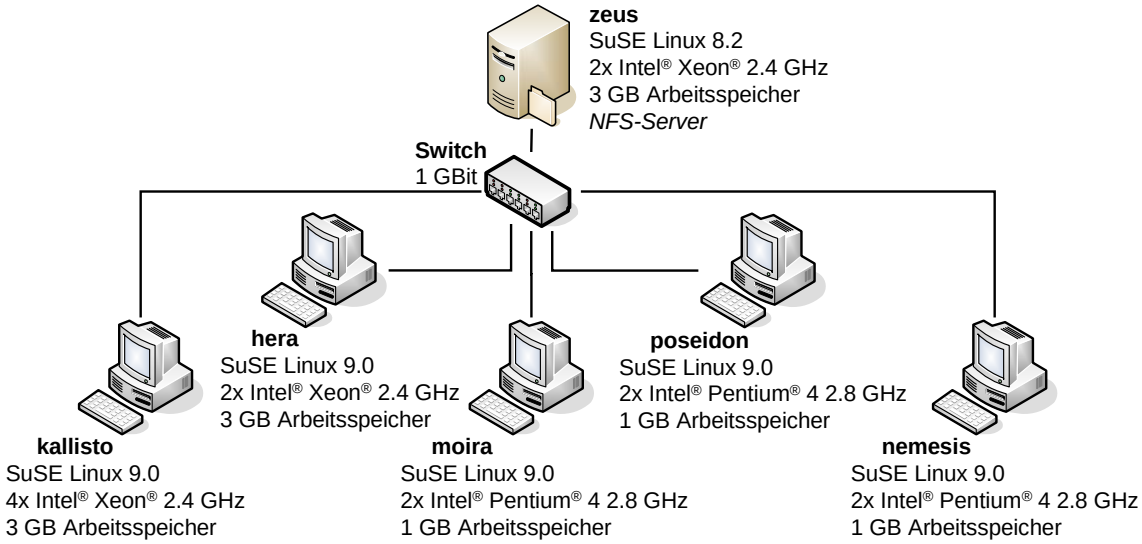


Abbildung 6-3: Rechnerkonfiguration

Die Aufteilung der in Abbildung 6-2 dargestellten Simulationsbestandteile auf Rechner gestaltete sich derart, dass die komplette Simulation des Szenarios (rechter Teil in Abbildung 6-2) sowie je eine ACU einem Rechner zugeordnet war. Je nach Anzahl der betrachteten ACUs musste dabei ein Rechner sowohl das Szenario simulieren als auch eine ACU darstellen.

6.2 Funktionalität

Um zu zeigen, dass mit dem gewählten Ansatz und implementierten Wissen das gewünschte kooperative Verhalten entstehen kann, wird in diesem Abschnitt anhand des im vorigen Abschnitt vorgestellten Beispielszenarios das Verhalten des Funktionsprototyps der ACU erläutert. Zunächst wird in Abschnitt 6.2.1 auf Einzelfähigkeiten der *Supporting ACUs* im Rahmen relevanter Teilprobleme eingegangen wird bevor in Abschnitt 6.2.2 das Zusammenspiel im Gesamtszenario dargestellt wird.

6.2.1 Einzelfähigkeiten

Gegenstand dieses Abschnitts ist es, anhand relevanter Teilprobleme des vorgestellten Anwendungsbeispiels das Verhalten der entwickelten *Supporting ACUs* aufzuzeigen, wie es aus den im Umfeld von Kooperation relevanten Zielen resultiert. Für die Bearbeitung dieser Teilprobleme wurde(n) die jeweils betrachteten ACU(s) mit dem kompletten Wissen, wie es im Hinblick auf die Gesamtapplikation entwickelt wurde, ausgestattet. Die Applikation wurde also nicht auf das jeweils betrachtete Teilproblem zugeschnitten. Vielmehr soll hier untersucht und dargestellt werden, wie sich das Gesamtsystem bei der Bearbeitung überschaubarer Teilaufgaben verhält, um dadurch das Verhalten der ACUs in umfangreicheren Szenarien begreifbarer zu machen. Tabelle 6-2 gibt einen Überblick darüber, welche kooperationsrelevanten Wünsche (vgl. Abschnitte 4.3.1 und 5.4) mit den einzelnen Teilproblemen adressiert werden, die durch die jeweils vorhandenen Szenarienelemente charakterisiert sind.

Verschiedene Aspekte einzelner Wünsche werden dabei in unterschiedlichen Teilproblemen demonstriert. Die Relevanz der Wünsche beschränkt sich jedoch nicht auf die Fälle, in denen sie primär beschrieben werden. In den folgenden Abschnitten werden nun die Effekte der einzelnen Wünsche im Kontext der Teilprobleme im Detail vorgestellt.

Teilproblem / ***Wunsch***	*1 SEAD-UAV* *1 SAM-Stellung*	*2 SEAD-UAVs* *1 SAM-Stellung*	*2 SEAD-UAVs* *2 SAM-Stell'gen* *FLOT Korridor*	*1 SEAD-UAV* *1 Attack-UAV* *1 SAM-Stellung* *1 Target*
Sei einer Aufgabe verpflichtet	Einzelne ACU		Team	
Habe keine überflüssige Verpflichtung	Einzelne ACU		Team	
Führe keinen unnötigen Dialog				X
Halte Agenda durchführbar	X			
Sortiere Verpflichtung in Agenda ein	X			
Bearbeite Verpflichtung	X			
Kenne Teammitglieder		X		
Informiere Teammitglieder laufend	Einzelne ACU-Operateur	Einzelne ACU-Team	Teamsprecher-Team/Operator	
Stelle Aufgabenabdeckung sicher			X	
Verteile Arbeitsbelastung			X	
Habe Teamsprecher			X	
Weise gemeinsam genutzte Ressource zu			X	
Vermeide redundante Aufgabenbearbeitung		X		
Bearbeite unterstützende Aufgabe				X
Führe Dialog fort	Einzelne ACU-Operateur		Team-Teammitglied/Operateur	
Befolge Anweisung		Einzelne ACU		Team

Tabelle 6-2: Überblick über Teilprobleme und damit adressierte, kooperationsrelevante Wünsche

Das Verhalten der ACUs wird dabei vor allem anhand ihrer Kommunikation untereinander beschrieben. Daher wird an dieser Stelle kurz darauf eingegangen, wie die Darstellung der Kommunikation in Form von Tabellen zu lesen ist (vgl. Abbildung 6-4).

Kennung	***Protokoll***				
Zeit	***Performativ***	***Sender***	***Empfänger***	***Inhalt***	
0-0	*dialog-inform*				Dialog
00:00	inform	0	100	(information(actor 0)(type capability)) (capability(actor 0)(type sead))	Nachricht
100-0	*dialog-request*				Dialog
00:06	request	100	0, 1	(mission-order(action destroy) (object sam-site)(sam-site(id 0)))	Nachricht
00:06	agree	0, 1	100		Nachricht
06:50	inform	0, 1	100		Nachricht

Abbildung 6-4: Darstellung von Kommunikation zwischen ACUs

Die Anordnung der einzelnen ausgetauschten Nachrichten erfolgt nach ihrer Zugehörigkeit zu Dialogen, welche chronologisch nach dem Zeitpunkt ihrer Initiierung aufgeführt sind. Die Darstellung eines jeden Dialogs beginnt dabei mit einer hellgrau hinterlegten Zeile, welche die global eindeutige Kennung eines Dialogs sowie das verwendete Protokoll enthält. Die Kennung eines Dialogs wird vom Initiator eines Dialogs aus seiner eigenen ID sowie einer fortlaufenden Nummer generiert. In Abbildung 6-4 sind also zwei Dialoge dargestellt, wobei der erste vom Typ „inform“ ist und mittels der Kennung 0-0 identifiziert werden kann und der zweite einen Dialog vom Typ „request“ mit der Kennung 100-0 darstellt.

Jeder Dialog setzt sich aus einer oder mehreren Nachrichten zusammen, welche sich durch fünf Parameter auszeichnen, nämlich dem Zeitpunkt, zu dem sie gesendet wurden, den Typ der Nachricht (Performativ), die ID des Senders, die ID des/der Empfänger(s) und den Inhalt der Nachricht. Dem Operateur ist hierbei stets die ID 100 und den ACUs sind IDs zwischen 0 und 4 zugewiesen. Ist in der Spalte, welche den Sender bzw. Empfänger darstellt, mehr als eine ID angegeben, so ist eine Nachricht vom Teamsprechers des Teams, welches sich aus den ACUs mit den IDs zusammensetzt, geschickt bzw. an ein Team aus ACUs gesendet worden.

Der Inhalt der Nachrichten ist weitgehend in der Form dargestellt, wie er tatsächlich zwischen den ACUs ausgetauscht wird und stellt eine String-basierte Repräsentation von Ausschnitten des Situationswissens der ACUs dar. Wird wie in der ersten in Abbildung 6-4 aufgeführten Nachricht Information ausgetauscht, so setzt sich der Inhalt dieser Nachricht aus einer Spezifikation der Art der Information (Fähigkeit, Ressource oder Verpflichtung eines Akteurs oder Teams) und dem Wert dieser Information zusammen. Fähigkeiten (*capability*) werden durch einen Akteur (*actor*) und die Art der Fähigkeit (*type*) näher charakterisiert, Ressourcen (*resource*) beziehen sich ebenfalls auf einen Akteur (*actor*) und werden mit Art der Ressource (*type*), vorhandener Menge (*total-amount*) und für den Akteur noch frei verfügbarer Menge (*free*) attributiert. Letztere ergibt sich als Differenz der vorhandenen Menge und der für eingegangene Verpflichtungen voraussichtlich nötigen Menge. Verpflichtungen (*commitment*) schließlich sind einem Akteur oder Team zugeordnet (*responsible*), beziehen sich auf eine Aufgabe (*task*) und werden – sofern es sich um individuelle Verpflichtungen handelt – an eine bestimmte Position in der Agenda des verantwortlichen Akteurs eingeordnet (*agenda-idx*).

6.2.1.1 Teilproblem 1: *1 SEAD-UCAV, 1 SAM-Stellung*

Die Ausgangssituation von Teilproblem 1, das sich aus einem SEAD-UCAV (ID 0) und einer SAM-Stellung (ID 0) zusammensetzt, ist in Abbildung 6-5 dargestellt. Aufgabe des SEAD-UCAVs ist es, infolge eines Auftrags vom Operateur die SAM-Stellung zu bekämpfen. Hierzu geht die ACU eine entsprechende Verpflichtung ein und versucht, dieser gerecht zu werden, d.h. die SAM-Stellung tatsächlich zu bekämpfen. In Lauf a gelingt ihr das nicht, da sich die SAM-Stellung bei jedem Angriff lediglich temporär abschaltet, so dass die ACU die Verpflichtung wieder aufgibt, da sie keine HARMs mehr zur Verfügung hat, und zum Stützpunkt zurückfliegt. In Lauf b kann die SAM-Stellung erfolgreich bekämpft werden, woraufhin auch hier das UCAV zum Stützpunkt zurückfliegt. An dieser Stelle sei angemerkt, dass die Reaktion einer SAM-Stellung auf einen Angriff (temporäre Abschaltung bzw. Zerstörung) auf Basis von Wahrscheinlichkeiten modelliert ist, so dass das UCAV keine Möglichkeit hat, über ein in irgendeiner Weise optimiertes Angriffsverhalten den Erfolg einer Bekämpfung zu beeinflussen. Neben der Durchführung des Auftrags wird im Folgenden weiterhin erläutert, mit welchen Nachrichten die ACU den Operateur über ihren aktuellen Zustand informiert.

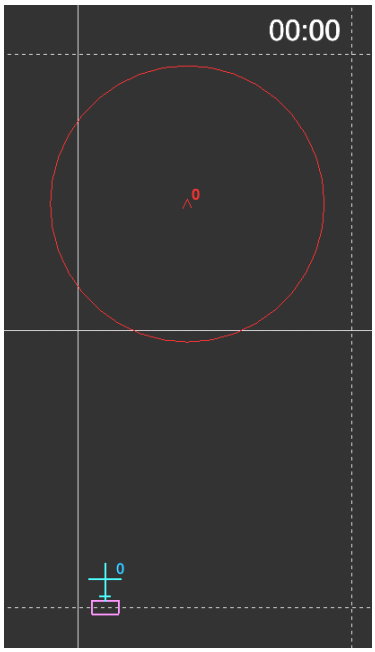


Abbildung 6-5: Ausgangssituation Teilproblem 1

Lauf a

Abbildung 6-6 stellt einen Ablauf des hier betrachteten Szenarios dar (für eine Erläuterung der Trajektorie vgl. Abschnitt 6.3.2.3), während in Tabelle 6-3 die zwischen Operateur und ACU geführten Dialoge aufgelistet sind. Zunächst informiert die ACU den Operateur infolge des Ziels *Informiere Teammitglieder laufend* über ihre Fähigkeiten, Ressourcen und Verpflichtungen (Dialog-Kennungen 0-0, 0-1 und 0-2). Hierzu wird jeweils ein Dialog vom Typ „inform“ initiiert, dessen einzige Nachricht als Inhalt sowohl die Art der übermittelten Information als auch den Wert dieser Information enthält. Die ACU verfügt zu diesem Zeitpunkt über die Fähigkeit zur Unterdrückung gegnerischer Luftabwehr (SEAD), zwei HARMs und keine Verpflichtungen.

Nach der Übermittlung des Auftrags vom Operateur, die gegenerische SAM-Stellung mit der ID 0 zu bekämpfen (Dialog-Kennung 100-0, Nachricht „request“) evaluiert die ACU die Machbarkeit dieses Auftrags und nimmt diesen an (Nachricht „agree“).

Da der Auftrag nun angenommen ist und aus zwei Aufgaben besteht, nämlich die SAM-Stellung zu bekämpfen und zum Stützpunkt zurückzukehren, wird für jede Aufgabe erst das Ziel aktiviert, eine Verpflichtung einzugehen und dann das Ziel, die jeweilige Verpflichtung in die Agenda einzuordnen. In diesem Zusammenhang wird der Operateur durch drei Dialoge über den aktuellen Stand der Verpflichtungen der ACU informiert (Dialog-Kennungen 0-3, 0-4, 0-5): Zunächst geht die ACU die Verpflichtung ein, zum Stützpunkt zurückzukehren (0-3). Anschließend ist sie zusätzlich die Verpflichtung zur Bekämpfung der SAM-Stellung eingegangen und hat die Rückkehr zum Stützpunkt bereits an erster Stelle in die Agenda eingeordnet (0-4). Schließlich ist die Bekämpfung der SAM-Stellung an erster und die Rückkehr zum Stützpunkt an zweiter Stelle in der Agenda angeordnet (0-5). Darüber hinaus wird dem Operateur mitgeteilt, dass die ACU nun zwar immer noch über zwei HARMs verfügen kann, aber eine davon für die eingegangenen Verpflichtungen reserviert ist, so dass die ACU nur noch über die andere frei verfügen kann (Dialog 0-6). Da die Verpflichtung, die SAM-Stellung zu bekämpfen, an erster Stelle in der Agenda steht, wird das Ziel aktiviert, diese Verpflichtung bzw. die damit assoziierte Aufgabe zu bearbeiten. Daher fliegt das UCAV an eine Angriffsposition und setzt dort eine HARM ein (Abbildung 6-6, 03:09).

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
0-0	*dialog-inform*			
00:00	inform	0	100	(information(actor 0)(type capability)) (capability(actor 0)(type sead))
0-1	*dialog-inform*			
00:01	inform	0	100	(information(actor 0)(type resource)) (resource(actor 0)(type harm)(total-amount 2.0)(free 2.0))
0-2	*dialog-inform*			
00:01	inform	0	100	(information(actor 0)(type commitment))
100-0	*dialog-request*			
00:06	request	100	0	(mission-order(action destroy)(object sam-site)(sam-site(id 0)))
00:06	agree	0	100	
06:50	failure	0	100	
0-3	*dialog-inform*			
00:07	inform	0	100	(information(actor 0)(type commitment)) (commitment(responsible(actor 0))(task(name task-fly-to)(object homebase)))
0-4	*dialog-inform*			
00:07	inform	0	100	(information(actor 0)(type commitment)) (commitment(responsible(actor 0))(task(name task-destroy)(object sam-site)(sam-site 0))) (commitment(agenda-idx 1)(responsible(actor 0))(task(name task-fly-to)(object homebase)))
0-5	*dialog-inform*			
00:09	inform	0	100	(information(actor 0)(type commitment)) (commitment(agenda-idx 1)(responsible(actor 0))(task(name task-destroy)(object sam-site)(sam-site 0))) (commitment(agenda-idx 2)(responsible(actor 0))(task(name task-fly-to)(object homebase)))
0-6	*dialog-inform*			
00:09	inform	0	100	(information(actor 0)(type resource)) (resource(actor 0)(type harm)(total-amount 2.0)(free 1.0))
0-7	*dialog-inform*			
03:01	inform	0	100	(information(actor 0)(type resource)) (resource(actor 0)(type harm)(total-amount 1.0)(free 0.0))
0-9	*dialog-inform*			
06:48	inform	0	100	(information(actor 0)(type resource)) (resource(actor 0)(type harm)(total-amount 0.0)(free 0.0))
0-10	*dialog-inform*			
06:49	inform	0	100	(information(actor 0)(type commitment)) (commitment(agenda-idx 1)(responsible(actor 0))(task(name task-fly-to)(object homebase)))
0-11	*dialog-inform*			
12:26	inform	0	100	(information(actor 0)(type commitment))

Tabelle 6-3: Teilproblem 1 – Lauf a – Dialoge
(K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-ID: 0, Operateur-ID: 100)

Danach informiert die ACU den Operateur, dass sie nun nur noch über eine HARM verfügt (Dialog 0-7). Da sich die SAM-Stellung temporär abschaltet, wartet das UCAV auf die erneute Aktivierung des Radars (Abbildung 6-6, 05:04). Nachdem auch der zweite Angriff lediglich die temporäre Abschaltung zur Folge hat (Abbildung 6-6, 07:21), hat das UCAV keine HARMs mehr (vgl. auch Dialog 0-9). Deswegen wird die Verpflichtung zur Bekämpfung der SAM-Stellung als unerreichbar gekennzeichnet und infolge des Ziels *Halte Agenda durchführbar* aufgegeben. Nun steht die Verpflichtung, zum Stützpunkt zurückzukehren an erster Stelle in der Agenda und wird bearbeitet (Abbildung 6-6, 07:21). Außerdem werden Nachrichten an den Operateur gesendet, die

ihm mitteilen, dass die Durchführung des Auftrags gescheitert ist (Nachricht vom Typ „failure“ im Rahmen des Dialogs 100-0) und sich die Verpflichtungen der ACU geändert haben (Dialog 0-10). Sobald das UCAV am Stützpunkt angekommen ist (Abbildung 6-6, 12:27) ist die Aufgabe, zum Stützpunkt zurückzukehren, erfolgreich bearbeitet und wird wegen des Ziels *Habe keine überflüssige Verpflichtung* aufgegeben. Auch diese Änderung der Verpflichtungen der ACU wird dem Operateur mitgeteilt (Dialog 0-11).

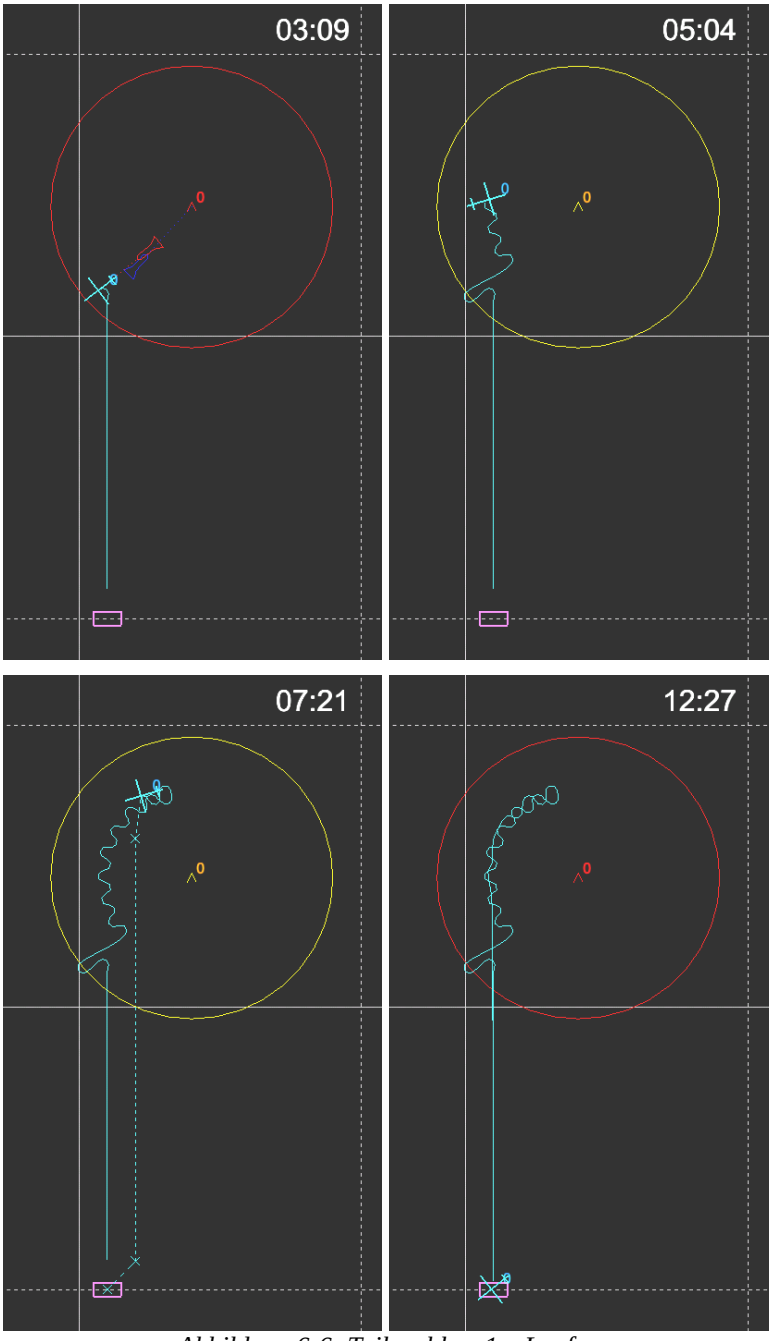


Abbildung 6-6: Teilproblem 1 – Lauf a

Lauf b

Ein anderer Verlauf der Mission ist in Abbildung 6-7 dargestellt. Hier ist zwar ebenfalls der erste Angriff auf die SAM-Stellung erfolglos (Abbildung 6-7, 05:01), der zweite Angriff (Abbildung 6-7, 05:48 & 10:36) erzielt jedoch das gewünschte Ergebnis. Somit ist die Aufgabe, die SAM-Stellung zu bekämpfen, erfolgreich bearbeitet und wird infol-

ge des Ziels *Habe keine überflüssige Verpflichtung* aufgegeben. Außerdem wird nun der Operateur mit einer Nachricht vom Typ „inform“ über den erfolgreichen Ausgang der Mission informiert (Dialog 100-0 in Tabelle 6-4).

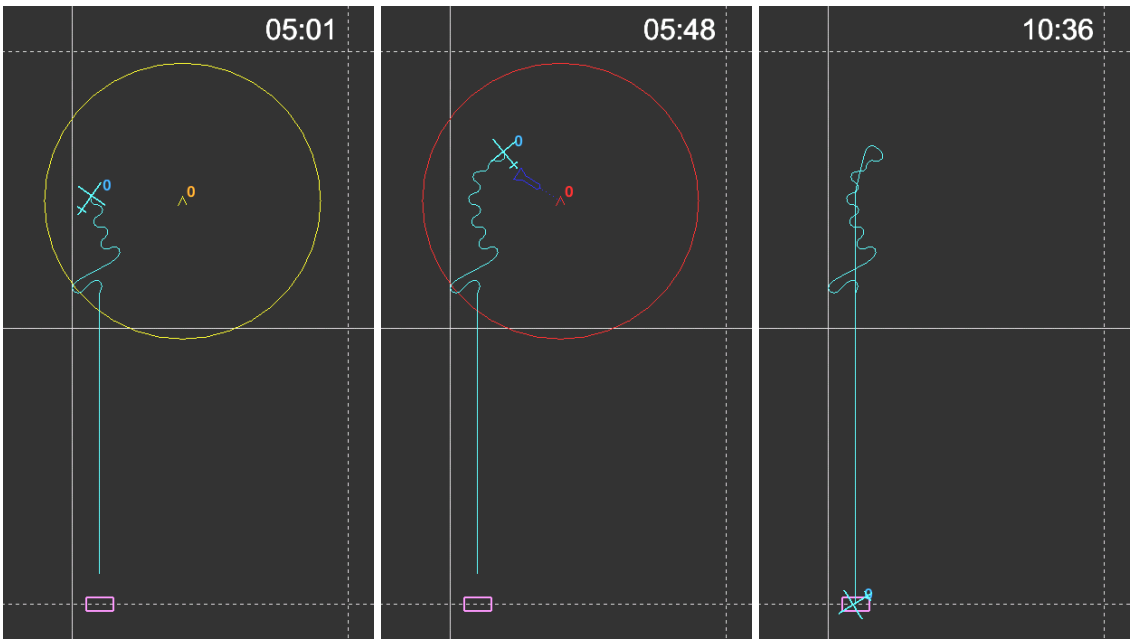


Abbildung 6-7: Teilproblem 1 – Lauf b

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
100-0	*dialog-request*			
00:06	request	100	0	(mission-order(action destroy)(object sam-site)(sam-site(id 0)))
00:07	agree	0	100	
05:50	inform	0	100	

Tabelle 6-4: Teilproblem 1 – Lauf b – Ausschnitt Dialoge
(K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-ID: 0, Operateur-ID: 100)

Zusammenfassung

Anhand dieses Teilproblems wurde gezeigt, wie sich eine einzelne ACU infolge eines angenommenen Auftrags zu Aufgaben verpflichtet (*Sei einer Aufgabe verpflichtet*), diese Verpflichtungen in ihre Agenda einsortiert (*Sortiere Verpflichtung in Agenda ein*) und anschließend bearbeitet (*Bearbeite Verpflichtung*). Außerdem wurde erläutert, wie die ACU darauf achtet, dass ihre Agenda durchführbar bleibt (*Halte Agenda durchführbar*) und Verpflichtungen aufgibt (*Habe keine überflüssige Verpflichtung*), wenn diese infolge ihrer verfügbaren Ressourcen nicht mehr erreichbar oder die entsprechenden Aufgaben erfolgreich bearbeitet sind. Schließlich wurde die Kommunikation der ACU mit dem Operateur hinsichtlich des Auftrags und der kontinuierlichen Informationsbereitstellung über den Zustand der ACU dargestellt (*Informiere Teammitglieder laufend, Führe Dialog fort*).

6.2.1.2 Teilproblem 2: *2 SEAD-UCAVs, 1 SAM-Stellung*

Abbildung 6-8 zeigt die Ausgangssituation von Teilproblem 2. Hier gibt es neben einer SAM-Stellung (ID 0) nun zwei SEAD-UCAVs (IDs 0 und 1), welche beide vom Operateur angewiesen werden, sich zur Bekämpfung der SAM-Stellung zu verpflichten. Infolge der gegenseitigen Information über die Verpflichtungen der einzelnen ACUs erkennen diese die Redundanz der Aufgabenzuweisung und stimmen sich auf Basis der Entfernung zur SAM-Stellung so ab, dass lediglich ein UCAV die SAM-Stellung bekämpft.

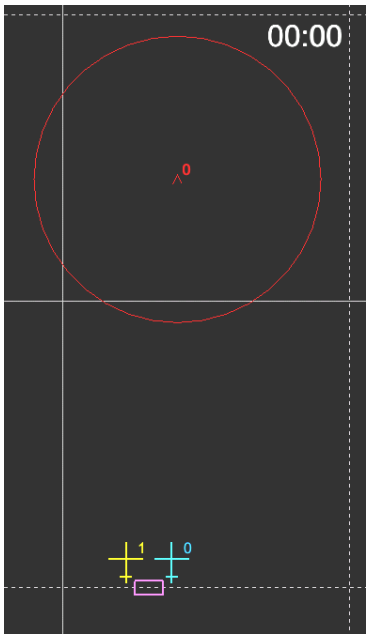


Abbildung 6-8: Ausgangssituation Teilproblem 2

Lauf a

Zu Beginn der Mission tauschen die beteiligten ACUs zunächst Informationen über ihre Fähigkeiten, Ressourcen und Verpflichtungen aus. In diesem Zusammenhang sind die Ziele *Kenne Teammitglied* und *Informiere Teammitglieder laufend* relevant, welche bewirken, dass einzelnen Teammitgliedern Informationen in Reaktion auf entsprechende Anfragen bereitgestellt werden (siehe Tabelle 6-5) bzw. dass das ganze Team über die gesamte Mission hinweg proaktiv informiert wird (siehe Tabelle 6-6).

In diesem Zusammenhang sei erwähnt, dass die infolge dieser Ziele initiierten Dialoge unter Umständen die Aktivierung des anderen Wunsches beeinflussen. So informiert ACU 0 das Team über seine Ressourcen (Dialog-Kennung 0-2) bevor ACU 1 Informationen über die Ressourcen von ACU 0 anfragt, so dass bei ACU 1 das Ziel *Kenne Teammitglied* bezüglich der Ressourcen von ACU 0 erfüllt ist, bevor ein Dialog vom Typ „Subscribe“ diesbezüglich begonnen wurde. Daher findet sich in Tabelle 6-5 kein solcher Dialog.

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
1-3	*dialog-subscribe*			
00:03	subscribe	1	0	(information(actor 0)(type commitment))
00:06	agree	0	1	
00:08	inform	0	1	
00:32	inform	0	1	(commitment(agenda-idx 1)(responsible(actor 0))(task(name task-destroy)(object sam-site)(sam-site 0)))
03:35	inform	0	1	
1-5	*dialog-query*			
00:11	query	1	0	(information(actor 0)(type capability))
00:12	agree	0	1	
00:12	inform	0	1	(capability(actor 0)(type sead))

Tabelle 6-5: Teilproblem 2 – Lauf a – Ausschnitt Dialoge infolge „Kenne Teammitglied“ von ACU 1 (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1, Operateur-ID: 100)

K	*Protokoll*			
Zeit	*Perf.*	*S*	*E*	*Inhalt*
0-2	*dialog-inform*			
00:04	inform	0	0, 1	(information(actor 0)(type resource)) (resource(actor 0)(type harm)(total-amount 2.0)(free 2.0))
1-7	*dialog-inform*			
00:14	inform	1	1, 0	(information(actor 1)(type commitment))
1-8	*dialog-inform*			
00:14	inform	1	1, 0	(information(actor 1)(type capability)) (capability(actor 1)(type sead))
1-1	*dialog-inform*			
00:15	inform	1	1, 0	(information(actor 1)(type resource)) (resource(actor 1)(type harm)(total-amount 3.0)(free 3.0))
0-18	*dialog-inform*			
00:37	inform	0	0, 1	(information(actor 0)(type resource)) (resource(actor 0)(type harm)(total-amount 2.0)(free 1.0))
1-10	*dialog-inform*			
00:40	inform	1	1, 0	(information(actor 1)(type commitment)) (commitment(responsible(actor 1))(task(name task-destroy)(object sam-site) (sam-site 0))
1-13	*dialog-inform*			
00:54	inform	1	1, 0	(information(actor 1)(type commitment))
0-26	*dialog-inform*			
03:37	inform	0	0, 1	(information(actor 0)(type resource)) (resource(actor 0)(type harm)(total-amount 1.0)(free 1.0))

Tabelle 6-6: Teilproblem 2 – Lauf a – Ausschnitt Dialoge infolge „Informiere Teammitglieder laufend“ (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1, Operateur-ID: 100)

Infolge einer Anweisung vom Operateur (Dialog-Kennung 100-0, Tabelle 6-7) und dem Ziel *Befolge Anweisung* geht ACU 0 eine Verpflichtung ein, SAM-Stellung 0 zu bekämpfen und meldet die erfolgreiche Durchführung der Anweisung an den Operateur (Nachricht „inform“), sowie die Änderung der Verpflichtungen an ACU 1 (Dialog 1-3, Tabelle 6-5). In gleicher Weise wird ACU 1 vom Operateur angewiesen, sich zur Bekämpfung von SAM-Stellung 0 zu verpflichten (Dialog-Kennung 100-1, Tabelle 6-7). Auch ACU 1 meldet die erfolgreiche Durchführung der Anweisung an den Operateur und die Änderung seiner Verpflichtungen an das Team (Dialog 1-10, Tabelle 6-6).

K	*Protokoll*			
Zeit	*Perf.*	*S*	*E*	*Inhalt*
100-0	*dialog-instruct*			
00:29	instruct	100	0	(instruction(id 0)(actor 0)(action commit)(task(name task-destroy) (object sam-site)(sam-site(id 0))))
00:31	inform	0	100	
100-1	*dialog-instruct*			
00:36	instruct	100	1	(instruction(id 1)(actor 1)(action commit)(task(name task-destroy) (object sam-site)(sam-site(id 0))))
00:49	inform	1	100	

Tabelle 6-7: Teilproblem 2 – Lauf a – Anweisungen vom Operateur (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1, Operateur-ID: 100)

Da nun beide ACUs zur Bekämpfung von SAM-Stellung 0 verpflichtet sind und beide um die Verpflichtung der anderen ACU zur gleichen Aufgabe wissen, aktivieren sowohl ACU 0 als auch ACU 1 das Ziel *Vermeide redundante Aufgabenbearbeitung*. Da sich UCAV 0 etwas näher an der SAM-Stellung befindet als UCAV 1, bittet ACU 0 mittels eines Dialogs vom Typ „request“ ACU 1, die Verpflichtung aufzugeben, während ACU 1 praktisch gleichzeitig vorschlägt, die Verpflichtung aufzugeben (siehe Tabelle 6-8). Da ACU 0 den Vorschlag von ACU 1 umgehend annimmt und ACU 1 die Ver-

pflichtung aufgibt, wird die Anfrage von ACU 0 zurückgewiesen, weil sie nun nicht mehr relevant ist.

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
0-20	*dialog-request*			
00:42	request	0	1	(type cancel) (task(name task-destroy)(object sam-site)(sam-site 0))
00:53	refuse	1	0	
1-12	*dialog-propose*			
00:44	propose	1	0	(type cancel) (task(name task-destroy)(object sam-site)(sam-site 0))
00:45	accept-proposal	0	1	
00:55	inform	1	0	

Tabelle 6-8: Teilproblem 2 – Lauf a –Dialoge infolge „Vermeide redundante Aufgabenbearbeitung“ (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1, Operateur-ID: 100)

Den weiteren Verlauf der Mission stellt Abbildung 6-9 dar. Während zunächst beide UCAVs in Richtung der SAM-Stellung geflogen sind, kehrt UCAV 1 nach Aufgabe der Verpflichtung zum Stützpunkt zurück, während UCAV 0 seiner Verpflichtung nachgeht (01:23) und nach erfolgreicher Bekämpfung der SAM-Stellung (03:29) ebenfalls zum Stützpunkt zurückfliegt (07:27).

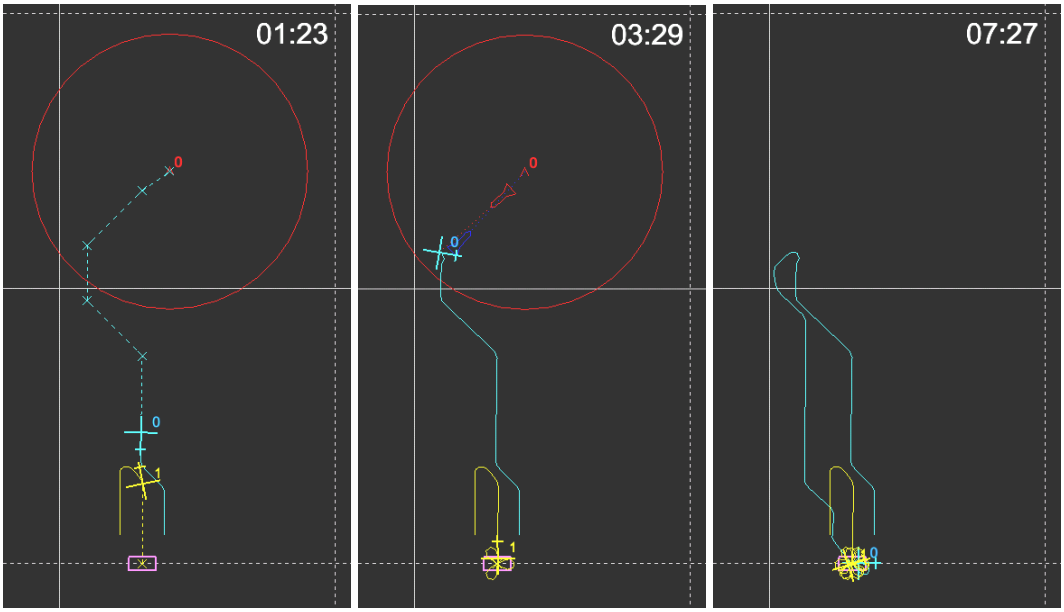


Abbildung 6-9: Teilproblem 2 – Lauf a

Lauf b

In einem anderen Verlauf des Szenarios stellt ACU 1 noch bevor es ACU 0 über seine neue Verpflichtung infolge der Anweisung des Operateurs informiert hat fest, dass redundante Aufgabenzuweisung im Team besteht. Da sie im konkreten Fall ohne Rückfrage bzw. Vorschlag bei ACU 0 diese Verpflichtung umgehend wieder aufgibt, entfallen somit die in Tabelle 6-8 für Lauf a dargestellten Dialoge.

Zusammenfassung

Anhand dieses Szenarios wurde gezeigt, dass einzelne ACUs auf Anweisungen des Operateurs mit entsprechenden Aktionen reagieren, d.h. hier konkret Verpflichtungen eingehen (*Befolge Anweisung*), dass ACUs innerhalb eines Teams auf unterschiedliche Arten Information hinsichtlich der Ressourcen, Fähigkeiten und Verpflichtungen der Teammitglieder austauschen (*Kenne Teammitglied*, *Informiere Teammitglieder laufend*)

und dass die ACUs Situationen auflösen können, in denen mehrere ACUs redundant zu einer Aufgabe verpflichtet sind (*Vermeide redundante Aufgabenbearbeitung*). Im Sinne einer zieleorientierten Vorgehensplanung auf wissensbasierter Verhaltensebene können hierbei verschiedene Vorgehensweisen Anwendung finden, um letztlich die gewünschte Zielsituation zu erreichen (vgl. auch Abschnitt 6.3.1).

6.2.1.3 Teilproblem 3: *2 SEAD-UCAVs, 2 SAM-Stellungen, FLOT, Korridor*

Teilproblem 3 setzt sich aus 2 SEAD-UCAVs (ID 0 und 1) und 2 SAM-Stellungen (ID 0 und 1) sowie einer FLOT und einem Korridor zusammen, über den die UCAVs in das bedrohte Gebiet gelangen sollen (siehe Abbildung 6-10). Infolge eines Auftrags vom Operateur soll das Team aus zwei UCAVs dort alle SAM-Stellungen bekämpfen. Zunächst wird im Team ein Teamsprecher bestimmt, welcher den Auftrag des Operateurs annimmt und das Team über dessen Verpflichtungen informiert. Im Anschluss werden die Aufgaben so im Team verteilt, dass jedem UCAV eine SAM-Stellung zugewiesen ist. Auf dem Flug zu den SAM-Stellungen müssen die UCAVs die FLOT überqueren und koordieren sich hinsichtlich der Nutzung des Korridors, den sie auch nach der Bekämpfung der SAM-Stellungen auf dem Rückflug zum Stützpunkt wieder passieren.

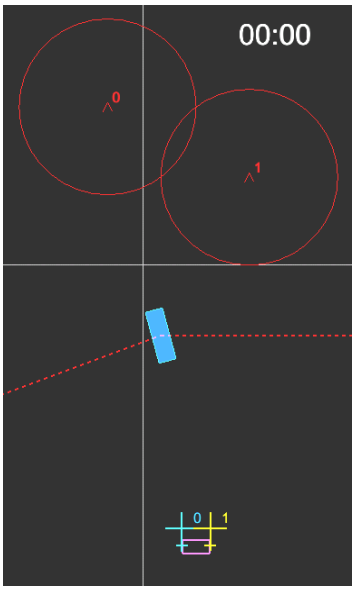


Abbildung 6-10: Ausgangssituation Teilproblem 3

Lauf a

Die zur Bestimmung eines Teamsprechers auf Basis des Ziels *Habe Teamsprecher* geführten Dialoge sind in Tabelle 6-9 zusammengestellt. Zunächst teilt jedes Teammitglied dem Team eine zufällig ausgewählte Zahl mit (Dialoge 0-7 und 1-6). Daraufhin sind bei ACU 1 die Vorbedingungen für den Vorschlag, die Rolle des Teamsprechers zu übernehmen, erfüllt, nämlich Zahlen von allen Teammitgliedern zu kennen und selbst die größte Zahl genannt zu haben. Somit schlägt ACU 1 dem Team vor, die Rolle des Teamsprechers zu übernehmen (Dialog 1-8), was von allen Teammitgliedern akzeptiert wird, so dass ACU 1 die Rolle annimmt und dies sowohl dem Team als auch dem Operateur mitteilt (Dialoge 1-10 und 1-12). Eine Aufgabe des Teamsprechers besteht darin, Dialoge im Namen des Teams zu führen, Verpflichtungen für das Team einzugehen und aufzugeben und sowohl das Team als auch den Operateur über die Verpflichtungen des Teams zu informieren (*Informiere Teammitglieder laufend*).

K	*Protokoll*			
Zeit	*Performativ*	*S*	*E*	*Inhalt*
0-7	*dialog-inform*			
00:02	inform	0	1, 0	(spokesman-election(team 1)(team 0)) (spokesman-election-number(actor 0)(number 70))
1-6	*dialog-inform*			
00:06	inform	1	0, 1	(spokesman-election(team 0)(team 1)) (spokesman-election-number(actor 1)(number 77))
1-8	*dialog-propose*			
00:08	propose	1	0, 1	(spokesman-election(team 0)(team 1))(actor 1)(type spokesman)
00:09	accept-proposal	0	1	
00:09	accept-proposal	1	0, 1	
1-10	*dialog-inform*			
00:11	inform	1	0, 1	(information(team 0)(team 1)(type spokesman)) (spokesman(actor 1)(team 0)(team 1))
1-12	*dialog-inform*			
00:12	inform	1	100	(information(team 0)(team 1)(type spokesman)) (spokesman(actor 1)(team 0)(team 1))

Tabelle 6-9: Teilproblem 3 – Lauf a – Ausschnitt Dialoge zur Teamsprecherwahl (K=Kennung, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1, Operateur-ID: 100)

K	*Protokoll*			
Zeit	*Perf.*	*S*	*E*	*Inhalt*
1-9	*dialog-inform*			
00:11	inform	1	0, 1	(information(team 0)(team 1)(type commitment))
100-0	*dialog-request*			
00:17	request	100	0, 1	(mission-order(action destroy-all)(object-type sam-site))
00:18	agree	0, 1	100	
12:27	inform	0, 1	100	
1-23	*dialog-inform*			
00:34	inform	1	0, 1	(information(team 0)(team 1)(type commitment)) (commitment(responsible(team 0)(team 1))(task(name task-destroy)(object sam-site)(sam-site 0))) (commitment(responsible(team 0)(team 1))(task(name task-fly-to)(object homebase))) (commitment(responsible(team 0)(team 1))(task(name task-destroy)(object sam-site)(sam-site 1)))
1-39	*dialog-inform*			
06:18	inform	1	0, 1	(information(team 0)(team 1)(type commitment)) (commitment(responsible(team 0)(team 1))(task(name task-destroy)(object sam-site)(sam-site 0))) (commitment(responsible(team 0)(team 1))(task(name task-fly-to)(object homebase)))
1-56	*dialog-inform*			
12:12	inform	1	0, 1	(information(team 0)(team 1)(type commitment)) (commitment(responsible(team 0)(team 1))(task(name task-fly-to)(object homebase)))
1-81	*dialog-inform*			
18:07	inform	1	0, 1	(information(team 0)(team 1)(type commitment))

Tabelle 6-10: Teilproblem 3 – Lauf a – Ausschnitt Dialoge zur Information bzgl. Team-Verpflichtungen (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1, Operateur-ID: 100)

In Tabelle 6-10 sind in diesem Zusammenhang geführte Dialoge dargestellt. Zu Beginn der Mission hat das Team keine Verpflichtungen (vgl. Dialog 1-9). Nach Eingang des Auftrags vom Operateur (Dialog 100-0) und dessen Annahme verpflichtet sich das Team durch den Teamsprecher, die mit dem Auftrag assoziierten Aufgaben – Bekämpfung von SAM-Stellung 0, Bekämpfung von SAM-Stellung 1 und Rückkehr zum Stützpunkt – zu bearbeiten (*Habe Verpflichtung*, Dialog 1-23). Nach erfolgreicher

Bekämpfung von SAM-Stellung 1 (vgl. auch Abbildung 6-11, 07:47) wird diese Verpflichtung des Teams aufgegeben (vgl. Dialog 1-39). Entsprechendes gilt wegen des Ziels *Habe keine überflüssige Verpflichtung* auch nach der erfolgreichen Bekämpfung von SAM-Stellung 0 (vgl. Abbildung 6-11, 12:27 und Dialog 1-56) und der Rückkehr zum Stützpunkt (vgl. Abbildung 6-11, 18:17 und Dialog 1-81).

Nach dem Eingehen entsprechender Verpflichtungen des Teams müssen die entsprechenden Aufgaben einzelnen Teammitgliedern zugewiesen werden (das Vorgehen hierbei wird später anhand von Lauf b und Tabelle 6-13 näher erläutert). Im vorliegenden Lauf a ergibt es sich, dass ACU 0 keine Verpflichtungen außer der Rückkehr zum Stützpunkt eingegangen ist, während ACU 1 sowohl SAM-Stellung 0 als auch SAM-Stellung 1 bekämpfen soll. Um die Aufgabenbelastung im Team zu verteilen (*Verteile Arbeitsbelastung*) schlägt ACU 0 daraufhin vor, die Bekämpfung von SAM-Stellung 0 zu übernehmen, was von ACU 1 akzeptiert wird (siehe Dialog 0-15 in Tabelle 6-11), so dass im Anschluss an diesen Dialog ACU 0 zur Bekämpfung von SAM-Stellung 0 und ACU 1 zur Bekämpfung von SAM-Stellung 1 verpflichtet ist.

K	***Protokoll***			
Zeit	***Performativ***	***S***	***E***	***Inhalt***
0-15	*dialog-propose*			
00:44	propose	0	1	(task(name task-destroy)(object sam-site)(sam-site 0))(type takeover)
00:46	accept-proposal	1	0	
00:51	inform	0	1	

Tabelle 6-11: Teilproblem 3 – Lauf a – Ausschnitt Dialog zur Aufgabenübernahme (K=Kennung, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1)

Um die entsprechenden Aufgaben nun bearbeiten zu können, müssen beide UCAVs über den ausgewiesenen Korridor in das bedrohte Gebiet fliegen. Zunächst teilt dabei jede ACU dem Team ihren Bedarf an der gemeinsam genutzten Ressource „Korridor" mit (Dialoge 1-22 und 0-21 in Tabelle 6-12). Da beide den Korridor benötigen, wird daraufhin das Ziel *Weise gemeinsam genutzte Ressource zu* relevant, woraufhin der Korridor ACU 1 zugewiesen wird, da sich UCAV 1 näher am Korridor befindet. Somit fliegt UCAV 1 als erstes in den Korridor ein, während UCAV 0 davor wartet (siehe Abbildung 6-11, 03:28).

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
1-22	*dialog-inform*			
00:37	inform	1	0, 1	(information(actor 1)(type shared-resource)) (shared-resource(actor 1)(corridor 0))
0-21	*dialog-inform*			
01:04	inform	0	1, 0	(information(actor 0)(type shared-resource)) (shared-resource(actor 0)(corridor 0))
1-35	*dialog-inform*			
03:33	inform	1	0, 1	(information(actor 1)(type shared-resource))
0-24	*dialog-inform*			
04:26	inform	0	1, 0	(information(actor 0)(type shared-resource))

Tabelle 6-12: Teilproblem 3 – Lauf a – Ausschnitt Dialoge zur Ressourcenzuweisung (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1, Operateur-ID: 100)

Nachdem ACU 1 dem Team mitgeteilt hat, dass keine gemeinsam genutzte Ressource mehr benötigt wird (Dialog 1-35), beginnt auch UCAV 0 mit dem Einflug in den Korridor (vgl. Abbildung 6-11, 04:23) und teilt dem Team mit, sobald die Ressource nicht mehr benötigt wird (Dialog 0-24). Nachdem beide UCAVs ihre Aufgaben erfolgreich bearbeitet haben, wird für den Rückflug wiederum der Korridor benötigt, so dass analo-

ge Dialoge ablaufen. Da UCAV 1 aber bereits vor UCAV 0 zurückfliegt, überschneidet sich der Bedarf zeitlich nicht, so dass die Ressource nicht einem der UCAVs zugewiesen werden muss, sondern beide über sie verfügen können, sobald sie benötigt wird (vgl. Abbildung 6-11, 06:04 und 07:47).

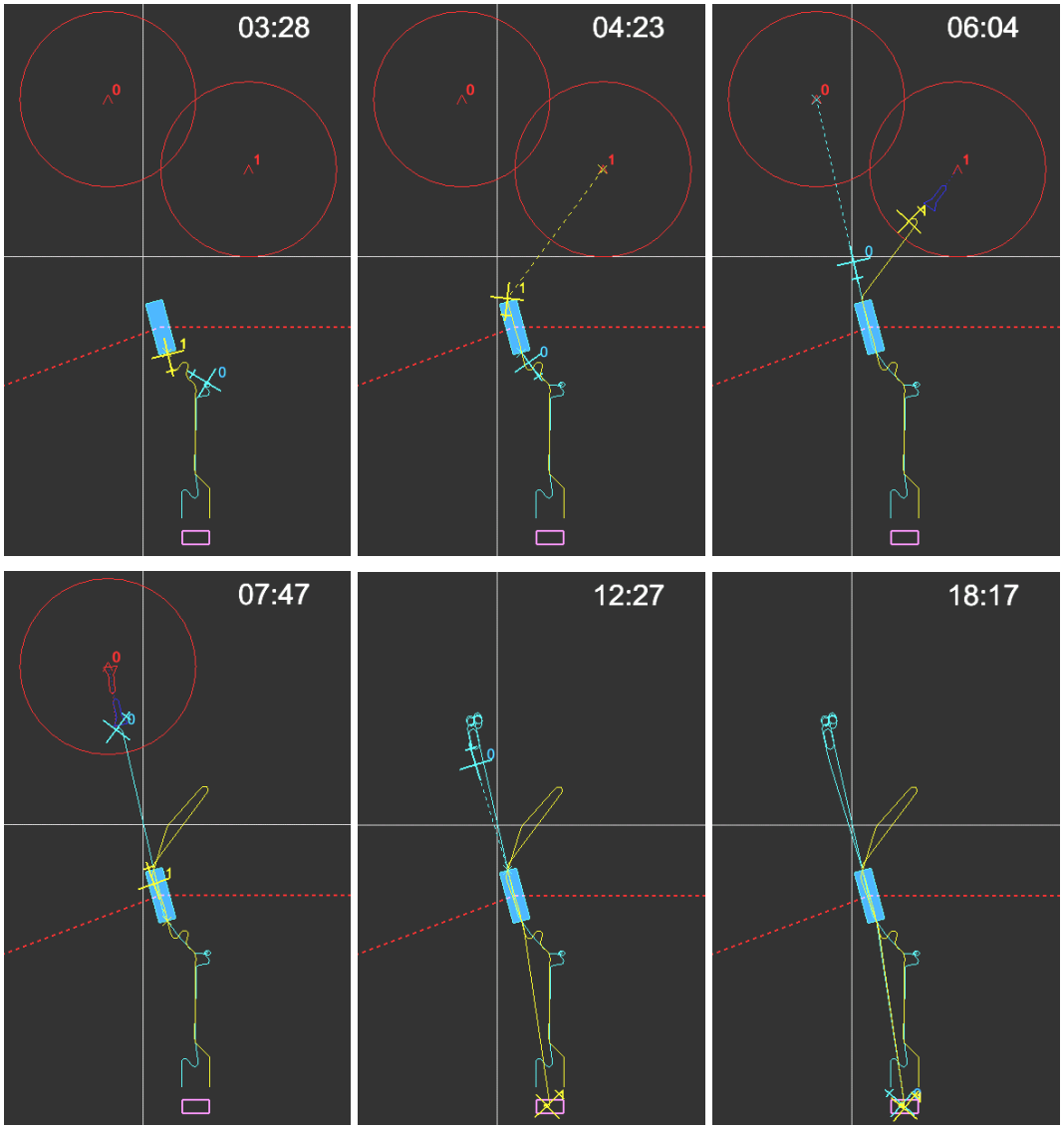


Abbildung 6-11: Teilproblem 3 – Lauf a

Lauf b

Anhand der im Rahmen eines anderen Laufs von Teilproblem 3 geführten Dialoge wird nun die Zuweisung von Aufgaben, zu denen sich das Team aus ACU 0 und 1 verpflichtet hat, erläutert. Hierbei wird sowohl die Aufgabenverteilung zu Beginn der Mission (siehe Tabelle 6-13) als auch eine Aufgabenneuverteilung während der Mission (siehe Tabelle 6-14) betrachtet.

Nachdem das Team zu Beginn der Mission Verpflichtungen eingegangen ist, die SAM-Stellungen 0 und 1 zu bekämpfen, schlagen sowohl ACU 1 als auch ACU 0 auf Basis des Ziels *Stelle Aufgabenabdeckung sicher* vor, die Aufgabe, SAM-Stellung 0 zu bekämpfen, zu übernehmen (vgl. Tabelle 6-13, Dialoge 1-15 und 0-8). Der Vorschlag von ACU 1 wird von beiden Teammitgliedern angenommen, so dass ACU 1 eine Verpflichtung eingeht und dies auch dem Team mitteilt (Dialog 1-17). Da nun bereits eine Verpflichtung zu dieser Aufgabe besteht, weist ACU 1 den Vorschlag von ACU 0 zurück,

so dass diese keine weitere Verpflichtung mehr eingeht. In ähnlicher Weise schlägt ACU 1 vor, die Bekämpfung von SAM-Stellung 1 zu übernehmen (Dialog 1-21). Da ACU 0 aber bereits ohne vorhergehenden Vorschlag eine solche Verpflichtung eingegangen ist, dies aber dem Team noch nicht mitgeteilt hat, weist sie diesen Vorschlag zurück. Nachdem ACU 1 durch eine entsprechende Mitteilung Kenntnis von der Verpflichtung von ACU 0 erlangt hat (Dialog 0-9), weist auch sie ihren eigenen Vorschlag zurück, so dass sich letztlich hier eine Aufgabenverteilung ergibt, bei der ACU 1 SAM-Stellung 0 bekämpft, während ACU 0 zur Bekämpfung von SAM-Stellung 1 verpflichtet ist.

Der Prozess der Aufgabenzuweisung basiert hierbei nicht auf einer prozeduralen Vorgehensweise, die a-priori festgelegt wurde, sondern ergibt sich zur Laufzeit aus den implementierten Zielen und Handlungsalternativen. Im vorliegenden Fall ist dabei nur sehr pauschales Wissen hinterlegt, welches den Planungsprozess steuert, so dass sich ein etwas umständliches Vorgehen ergibt. Eine Berücksichtigung von weiteren Faktoren wie beispielsweise der Ressourcensituation oder der Gefährdung einzelner Teammitglieder kann allerdings erreicht werden, indem entsprechendes zusätzliches a-priori Wissen implementiert wird.

K	***Protokoll***			
Zeit	***Performativ***	***S***	***E***	***Inhalt***
1-15	*dialog-propose*			
00:27	propose	1	0, 1	(task(name task-destroy)(object sam-site)(sam-site 0))
00:28	accept-proposal	0	1	
00:28	accept-proposal	1	0, 1	
14:36	inform	1	0, 1	
0-8	*dialog-propose*			
00:27	propose	0	0, 1	(task(name task-destroy)(object sam-site)(sam-site 0))
00:28	accept-proposal	0	0, 1	
00:30	reject-proposal	1	0	
1-17	*dialog-inform*			
00:29	inform	1	0, 1	(information(actor 1)(type commitment)) (commitment(responsible(actor 1))(task(name task-destroy) (object sam-site)(sam-site 0)))
1-21	*dialog-propose*			
00:33	propose	1	0, 1	(task(name task-destroy)(object sam-site)(sam-site 1))
00:34	reject-proposal	0	1	
00:42	reject-proposal	1	0, 1	
0-9	*dialog-inform*			
00:41	inform	0	0, 1	(information(actor 0)(type commitment)) (commitment(responsible(actor 0))(task(name task-destroy)(object sam-site)(sam-site 1)))

Tabelle 6-13: Teilproblem 3 – Lauf b – Ausschnitt Dialoge zur Aufgabenverteilung zu Missionsbeginn (K=Kennung, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1)

Im weiteren Verlauf der Mission gelingt es UCAV 0 nicht, SAM-Stellung 1 erfolgreich zu bekämpfen, so dass ACU 0 infolge nicht mehr vorhandener Bewaffnung in Verbindung mit dem Ziel *Habe keine überflüssige Verpflichtung* die entsprechende Verpflichtung aufgibt und diese Änderung ihrer Verpflichtungen dem Team mitteilt. Daraufhin schlägt ACU 1 auf Basis des Ziels *Stelle Aufgabenabdeckung sicher* vor, diese Aufgabe zu übernehmen (Dialog 1-37 in Tabelle 6-14), was von allen Teammitgliedern akzeptiert wird, so dass ACU 1 eine entsprechende Verpflichtung eingeht. Da es auch UCAV 1 nicht gelingt, SAM-Stellung 1 erfolgreich zu bekämpfen, zeigt ACU 1 schließlich mit einer Nachricht vom Typ „failure“ an, dass die Bearbeitung der vorgeschlagenen Aufgabe nicht erfolgreich war.

K	*Protokoll*			
Zeit	*Performativ*	*S*	*E*	*Inhalt*
1-37	*dialog-propose*			
09:28	propose	1	0, 1	(task(name task-destroy)(object sam-site)(sam-site 1))
09:33	accept-proposal	1	0, 1	
09:36	accept-proposal	0	1	
11:29	failure	1	0, 1	

Tabelle 6-14: Teilproblem 3 – Lauf b – Ausschnitt Dialog zur Aufgabenneuverteilung während der Mission (K=Kennung, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/1)

Zusammenfassung

Anhand dieser Problemkonfiguration wurde gezeigt, wie ein Team vorgeht, um seinen Teamsprecher zu bestimmen (*Habe Teamsprecher*), und wie dieser Teamsprecher für das Team infolge eines angenommenen Auftrags vom Operateur Verpflichtungen eingeht und unter Berücksichtigung des Bearbeitungszustands der entsprechenden Aufgaben aufgibt (*Sei einer Aufgabe verpflichtet*, *Habe keine überflüssige Verpflichtung*). Ferner wurde dargestellt, dass der Teamsprecher das Team über dessen Verpflichtungen auf dem Laufenden hält (*Informiere Teammitglieder laufend*, *Führe Dialog fort*). Weiterhin wurde erläutert, wie das ACU-Team Aufgaben im Team verteilt und bei entsprechendem Verlauf der Mission diese Verteilung überarbeitet (*Stelle Aufgabenabdeckung sicher*), wobei stets auf eine gleichmäßige Belastung der Teammitglieder geachtet wird (*Verteile Arbeitsbelastung*). Schließlich wurde auch auf die Fähigkeit des Teams, die gemeinsam genutzte Ressource „Korridor“ einzelnen Teammitgliedern zur Nutzung zuzuweisen, eingegangen (*Weise gemeinsam genutzte Ressource zu*).

6.2.1.4 Teilproblem 4: *1 Attack-UCAV, 1 SEAD-UCAV, 1 SAM-Stellung, 1 Target*

Die Ausgangssituation von Teilproblem 4 ist in Abbildung 6-12 dargestellt. Es setzt sich aus einem Attack-UCAV (ID 4), einem SEAD-UCAV (ID 0) und einem Angriffsziel zusammen, welches von einer bekannten SAM-Stellung (ID 0) geschützt wird.

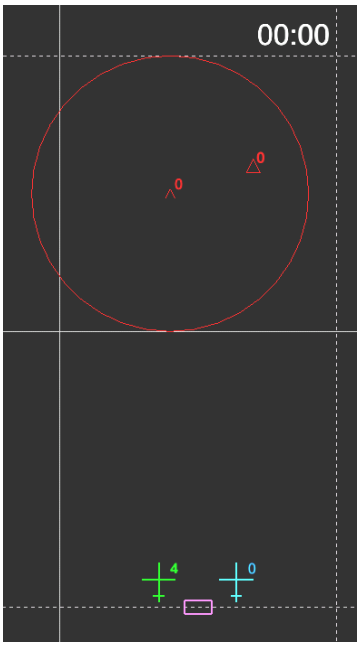


Abbildung 6-12: Ausgangssituation Teilproblem 4

Im ersten Durchlauf erhalten die ACUs vom Operateur den Auftrag, das Ziel zu bekämpfen, woraufhin das Attack-UCAV die Bekämpfung des Ziels übernimmt und vom SEAD-UCAV dabei unterstützt wird, indem es SAM-Stellung 0 unterdrückt. Im zweiten Durchlauf weist der Operateur das ACU-Team an, sich zur Bekämpfung des Angriffsziels zu verpflichten, woraufhin das Attack-UCAV eine entsprechende individuelle Verpflichtung eingeht und das SEAD-UCAV bittet, SAM-Stellung 0 zu unterdrücken. Nachdem der Operateur nun eine neue Anweisung an das Team schickt, die Verpflichtung zur Bekämpfung des Angriffsziels wieder aufzugeben, wird die Anfrage des Attack-UCAVs irrelevant und daher vorzeitig abgebrochen. Außerdem kehren die UCAVs unmittelbar nach Aufgabe der Team-Verpflichtung zum Stützpunkt zurück.

Lauf a

Tabelle 6-15 zeigt einen Teil der in Lauf a (siehe Abbildung 6-13) geführten Dialoge. Nach Eingang des Auftrags vom Operateur (Dialog 100-0), dessen Annahme durch den Teamsprecher sowie einer entsprechenden Verpflichtung des Teams verpflichtet sich ACU 4 zur Zerstörung des Ziels und teilt dies dem Team mit (Dialog 4-21).

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
100-0	*dialog-request*			
00:19	request	100	0, 4	(mission-order(action destroy)(object target)(target 0))
00:20	agree	4, 0	100	
05:24	inform	4, 0	100	
4-21	*dialog-inform*			
00:24	inform	4	4, 0	(information(actor 4)(type commitment)) (commitment(responsible(actor 4))(task(name task-destroy)(object target)(target 0)))
0-16	*dialog-inform*			
00:27	inform	0	4, 0	(information(actor 0)(type commitment)) (commitment (responsible(actor 0))(task(name task-suppress)(object sam-site)(sam-site 0)))

Tabelle 6-15: Teilproblem 4 – Lauf a – Ausschnitt Dialoge
(K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/4, Operateur-ID: 100)

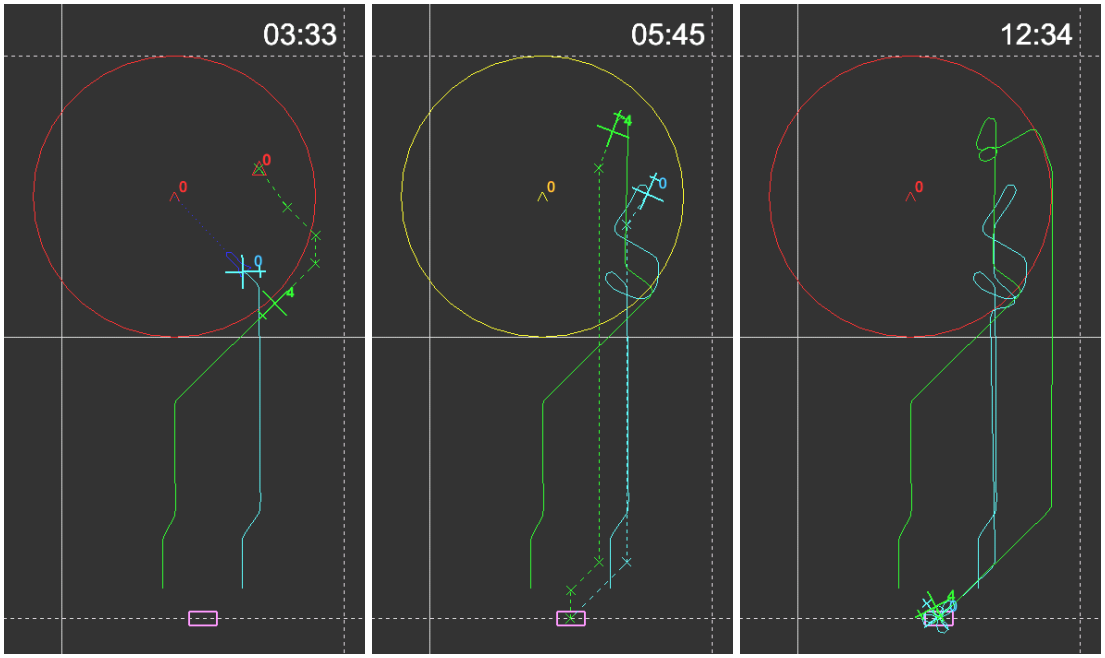


Abbildung 6-13: Teilproblem 4 – Lauf a

Da ACU 0 weiß, dass SAM-Stellung 0 das Ziel schützt und dessen Unterdrückung die Bekämpfung des Ziels unterstützt, verpflichtet sie sich auf Grund des Ziels *Bearbeite unterstützende Aufgabe* zur Unterdrückung der SAM-Stellung und informiert das Team darüber (Dialog 0-16). Im weiteren Verlauf gehen beide ACUs ihren Verpflichtungen nach – UCAV 4 fliegt zum Ziel, während UCAV 0 SAM-Stellung 0 angreift (Abbildung 6-13, 03:33). Nach der erfolgreichen Unterdrückung der SAM-Stellung passt UCAV 4 seinen Flugweg an die geänderte Bedrohungssituation an und kehrt ebenso wie UCAV 0 nach der erfolgreichen Bekämpfung des Angriffsziels zum Stützpunkt zurück (Abbildung 6-13, 05:45 und 12:34).

Lauf b

In einem weiteren Durchlauf des Szenarios erhalten die ACUs vom Operateur nicht einen Auftrag, das Ziel zu bekämpfen, welcher vom Team evaluiert, in der Regel angenommen und anschließend eine Verpflichtung zu dessen Durchführung eingegangen wird. Vielmehr wird das Team vom Operateur angewiesen, unmittelbar eine Verpflichtung zur Bekämpfung des Ziels einzugehen (siehe Tabelle 6-16, Dialog 100-0).

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
100-0	*dialog-instruct*			
00:37	instruct	100	0, 4	(instruction(id 0)(team 0)(team 4)(action commit)(task(name task-destroy)(object target)(target(id 0))))
00:39	inform	0, 4	100	
0-19	*dialog-inform*			
00:41	inform	0	0, 4	(information(team 0)(team 4)(type commitment))(commitment(responsible(team 0)(team 4))(task(name task-destroy)(object target)(target 0)))
4-15	*dialog-request*			
00:51	request	4	0	(task(name task-suppress)(object sam-site)(sam-site 0))
00:52	agree	0	4	
4-18	*dialog-inform*			
00:52	inform	4	0, 4	(information(actor 4)(type commitment))(commitment(responsible(actor 4))(task(name task-destroy)(object target)(target 0)))
0-22	*dialog-inform*			
00:55	inform	0	0, 4	(information(actor 0)(type commitment))(commitment(responsible(actor 0))(task(name task-suppress)(object sam-site)(sam-site 0)))

Tabelle 6-16: Teilproblem 4 – Lauf b – Ausschnitt Dialoge zu Missionsbeginn (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/4, Operateur-ID: 100)

ACU 0 als Sprecher des Teams geht für das Team die Verpflichtung ein, informiert den Operateur über die erfolgreiche Durchführung der Anweisung (100-0, „inform“) und teilt dem Team die Verpflichtung des Teams mit (Dialog 0-19). Auch hier geht ACU 4 eine Verpflichtung zur Zerstörung des Ziels ein (vgl. Dialog 4-18). Bevor sie dies jedoch dem Team mitteilt und ACU 0 so von sich aus eine Möglichkeit zur Unterstützung des Attack-UCAVs feststellen kann, fragt ACU 4 bei ACU 0 an, ob sie SAM-Stellung 0, welche das Ziel schützt, unterdrücken kann (Dialog 4-15). Diese Anfrage wird von ACU 0 angenommen und in der Folge eine entsprechende Verpflichtung eingegangen (vgl. Dialog 0-22). Wie im vorhergehenden Lauf gehen beide UCAVs im Folgenden ihren Verpflichtungen nach (Abbildung 6-14, 02:28).

Nun weist der Operateur das Team jedoch an, die Verpflichtung zur Zerstörung des Ziels wieder aufzugeben (Dialog 100-1 in Tabelle 6-17), welcher das Team infolge des Ziels *Befolge Anweisung* nachkommt, so dass es daraufhin keine Verpflichtungen mehr

hat (vgl. Dialog 0-29). Damit entfällt die Grundlage für die individuellen Verpflichtungen der Teammitglieder, so dass insbesondere ACU 4 die Verpflichtung zur Bekämpfung des Ziels aufgibt. Da nun kein Grund mehr besteht, dass das SEAD-UCAV die SAM-Stellung unterdrückt, wird ACU 0 mit einem Dialog vom Typ „cancel“ (Dialog 4-23) mitgeteilt, dass die Unterdrückung von SAM-Stellung 0, welche wegen der Anfrage 4-15 begonnen wurde, abgebrochen werden soll (*Führe keinen unnötigen Dialog*). Dieser Aufforderung kommt ACU 0 nach, so dass beide UCAVs zum Stützpunkt zurückkehren (Abbildung 6-14, 03:14 und 05:51).

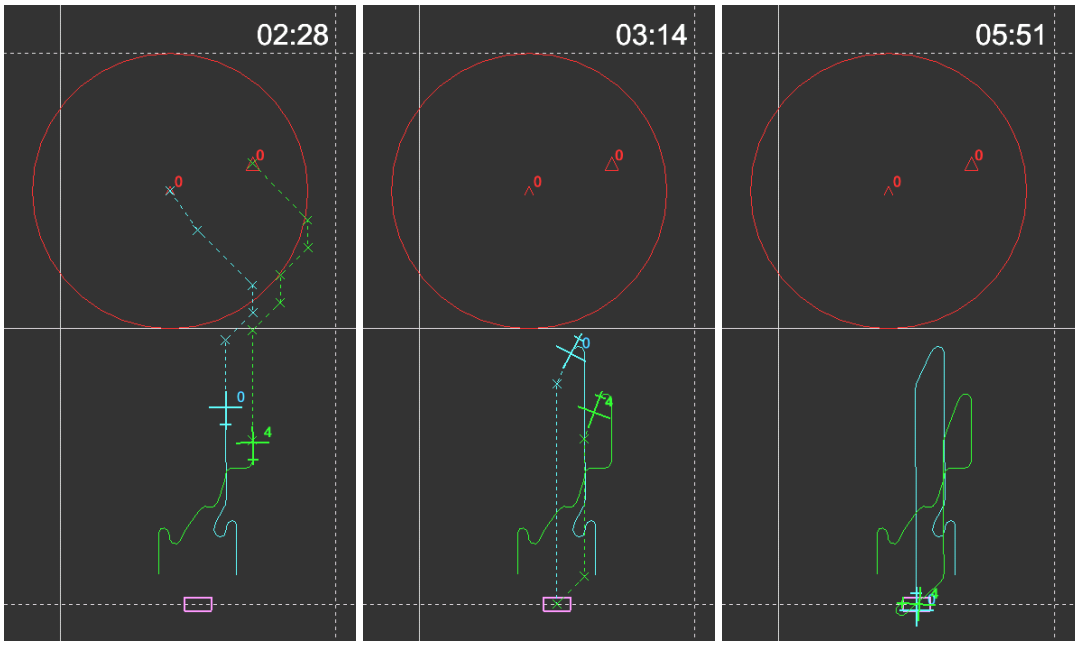


Abbildung 6-14: Teilproblem 4 – Lauf b

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
100-1	*dialog-instruct*			
02:33	instruct	100	0, 4	(instruction(id 1)(team 0)(team 4)(action drop-commitment)(task(name task-destroy)(object target)(target(id 0))))
02:48	inform	0, 4	100	
0-29	*dialog-inform*			
02:37	inform	0	100	(information(team 0)(team 4)(type commitment))
4-23	*dialog-cancel*			
02:49	cancel	4	0	(dialog 4-15)
02:57	inform	0	4	

Tabelle 6-17: Teilproblem 4 – Lauf b – Ausschnitt Dialoge bei Missionsabbruch
(K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/4, Operateur-ID: 100)

Zusammenfassung

Im Rahmen dieses Teilproblems wurde gezeigt, dass eine Unterstützung von Teammitgliedern sowohl proaktiv auf Basis der eingegangenen Verpflichtungen und Informationen über die Gesamtsituation (*Bearbeite unterstützende Aufgabe*) als auch auf Basis von Anfragen erfolgen kann. Diese Anfragen werden vor dem Hintergrund des Missionsverlaufs und der aktuellen Situation überprüft und abgebrochen, sofern kein Grund mehr für die Bearbeitung der angefragten Aufgabe besteht (*Führe keinen unnötigen Dialog*). Außerdem wurde dargestellt, dass auch ein Team Anweisungen vom Operateur, Verpflichtungen einzugehen bzw. aufzugeben, befolgt (*Befolge Anweisung*) und die Teammitglieder daraufhin entsprechende Aktivitäten durchführen.

6.2.2 Gesamtszenario

Nachdem in den vorhergehenden Abschnitten beispielhaft der Nachweis relevanter Einzelfähigkeiten für kooperative Missionserfüllung anhand einfacher Teilprobleme erbracht wurde, wird nun in diesem Abschnitt das Verhalten der ACUs und insbesondere deren Zusammenwirken im Kontext eines gesamten Szenarios erläutert (vgl. Abschnitt 6.1.1). An dieser Stelle sei nochmals darauf hingewiesen, dass keine Änderungen an der Wissensbasis vorgenommen wurden, sondern die ACUs ebenso wie im vorherigen Abschnitt auch hier über das in Kapitel 5 beschriebene a-priori Wissen verfügen.

Im zugrunde gelegten Szenario (siehe Abbildung 6-15) erhält ein Team aus einem Attack-UCAV (ID 4) und vier SEAD-UCAVs (IDs 0-3) vom Operateur den Auftrag, ein gegnerisches Ziel zu bekämpfen. Im gegnerischen Gebiet, welches durch eine FLOT definiert und über einen Korridor zu erreichen ist, befinden sich dabei sechs bekannte und vier unbekannte SAM-Stellungen unterschiedlicher Größe, für deren Bekämpfung jedem SEAD-UCAV zwei HARMs zur Verfügung stehen.

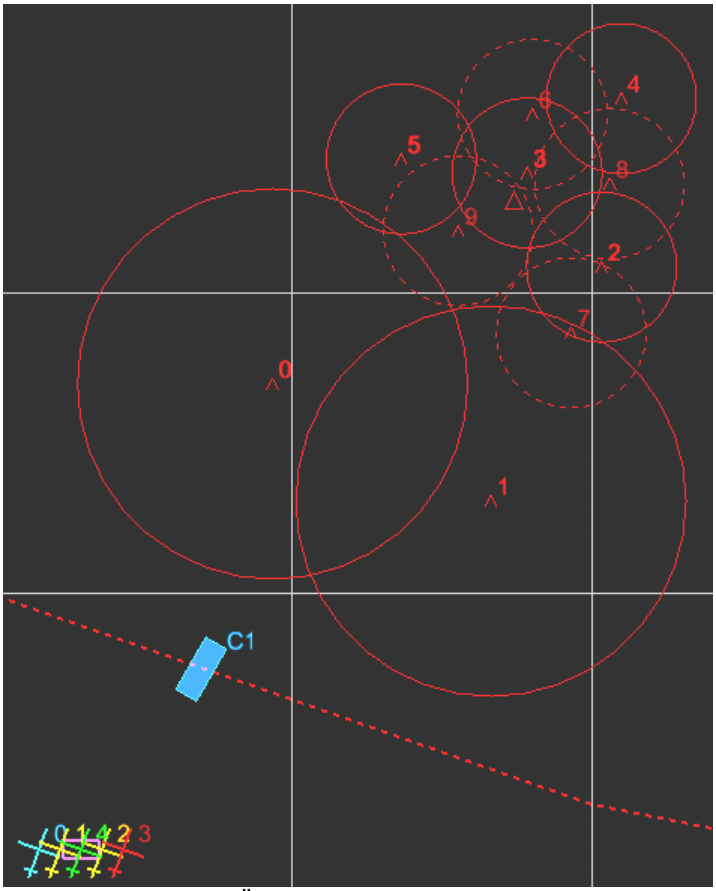


Abbildung 6-15: Überblick Szenario – Ausgangssituation

Zu Beginn der Mission wird sowohl für das gesamte Team als auch für das SEAD-Subteam infolge von Instanziierungen des Wunsches *Habe Teamsprecher* ein Teamsprecher bestimmt, welcher in beiden Teams ACU 1 ist. Diese nimmt den Auftrag des Operateurs an, woraufhin missionsrelevante Aufgaben im Team verteilt werden. Der zeitliche Ablauf dieses Vorgangs ist in Abbildung 6-16 und Abbildung 6-17 visualisiert, wobei die Zeit von oben nach unten fortschreitet, aber keine lineare Skalierung vorliegt, sondern Ereignisse nacheinander angeordnet werden. In umrandeten Kästen sind hierbei die Verpflichtungen der Teams bzw. der einzelnen ACUs dargestellt, welche dem gesamten Team mitgeteilt wurden und somit allen bekannt sind. Dialoge sind jeweils durch einen

Pfeil vom Initiator zum Teilnehmer symbolisiert, wobei ein grüner Haken die erfolgreiche Durchführung des Dialogs in dem Sinn kennzeichnet, dass die vom Initiator gewünschte Wirkung erreicht wird, während ein rotes Kreuz einen nicht beabsichtigten Verlauf und ein blauer Kreis einen vorzeitigen Abbruch des Dialogs durch den Initiator darstellt. Weiterhin stehen Pfeile, die mit „commit“ bzw. „drop commitment“ beschriftet sind, für Aktionen, bei denen eine ACU ohne Rückfrage bei einer anderen ACU eine Verpflichtung eingeht bzw. aufgibt.

Nach Bekanntwerden der Teamverpflichtung „Destroy Target“, die vom Teamsprecher auf Grund der Anfrage vom Operateur und einem Ziel *Habe Verpflichtung* eingegangen wurde, fragen die ACUs 0-3, welche allesamt nicht über die nötige Fähigkeit zur Bekämpfung des Ziels verfügen, bei ACU 4, welche dem Attack-UCAV zugeordnet ist, an, ob diese die Bekämpfung des Ziels übernehmen kann, um damit dem Ziel *Stelle Aufgabenabdeckung* im Team gerecht zu werden. ACU 4 verpflichtet sich daraufhin zur Bekämpfung des Ziels.

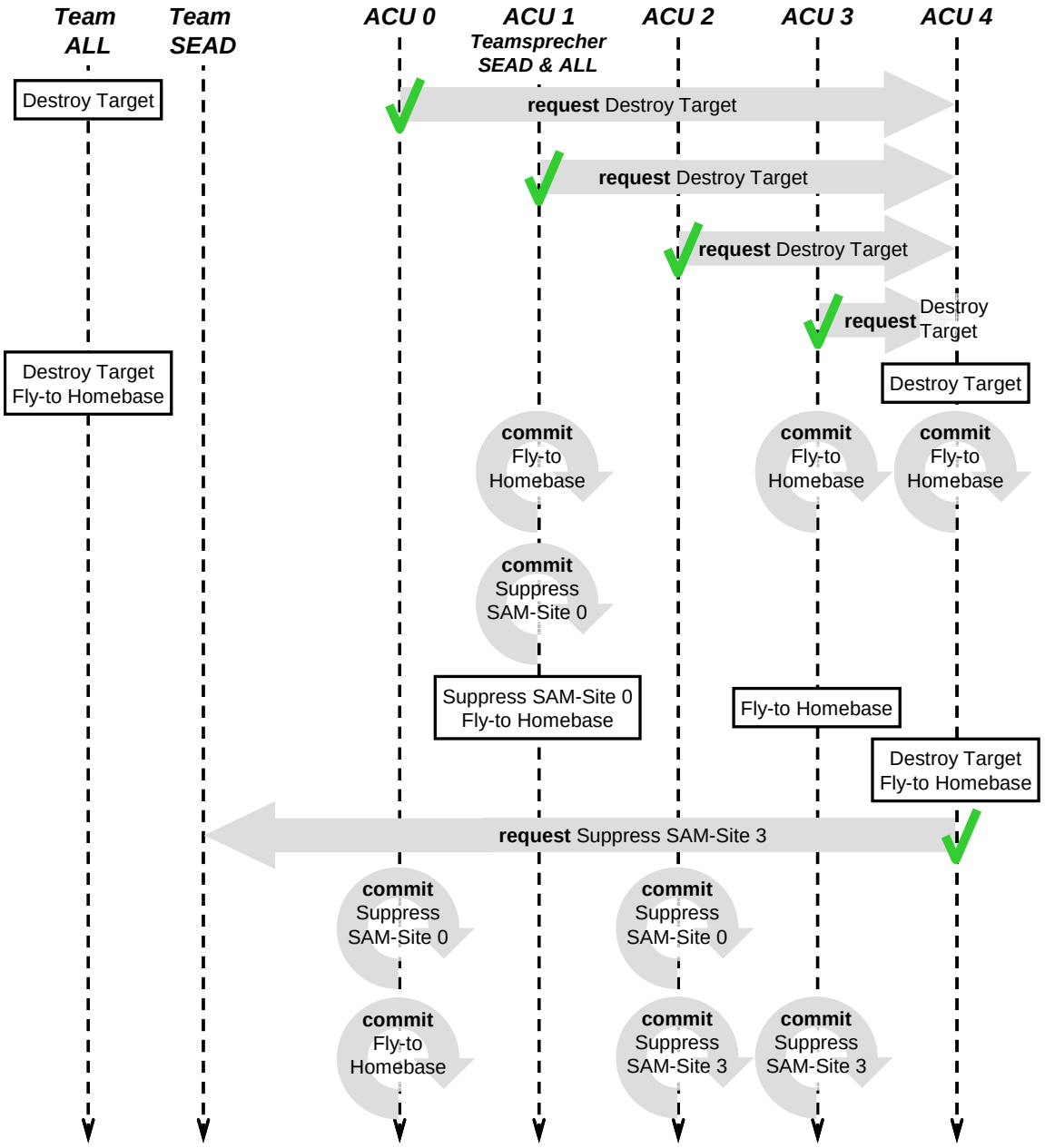


Abbildung 6-16: Aufgabenverteilung im Team zu Beginn der Mission

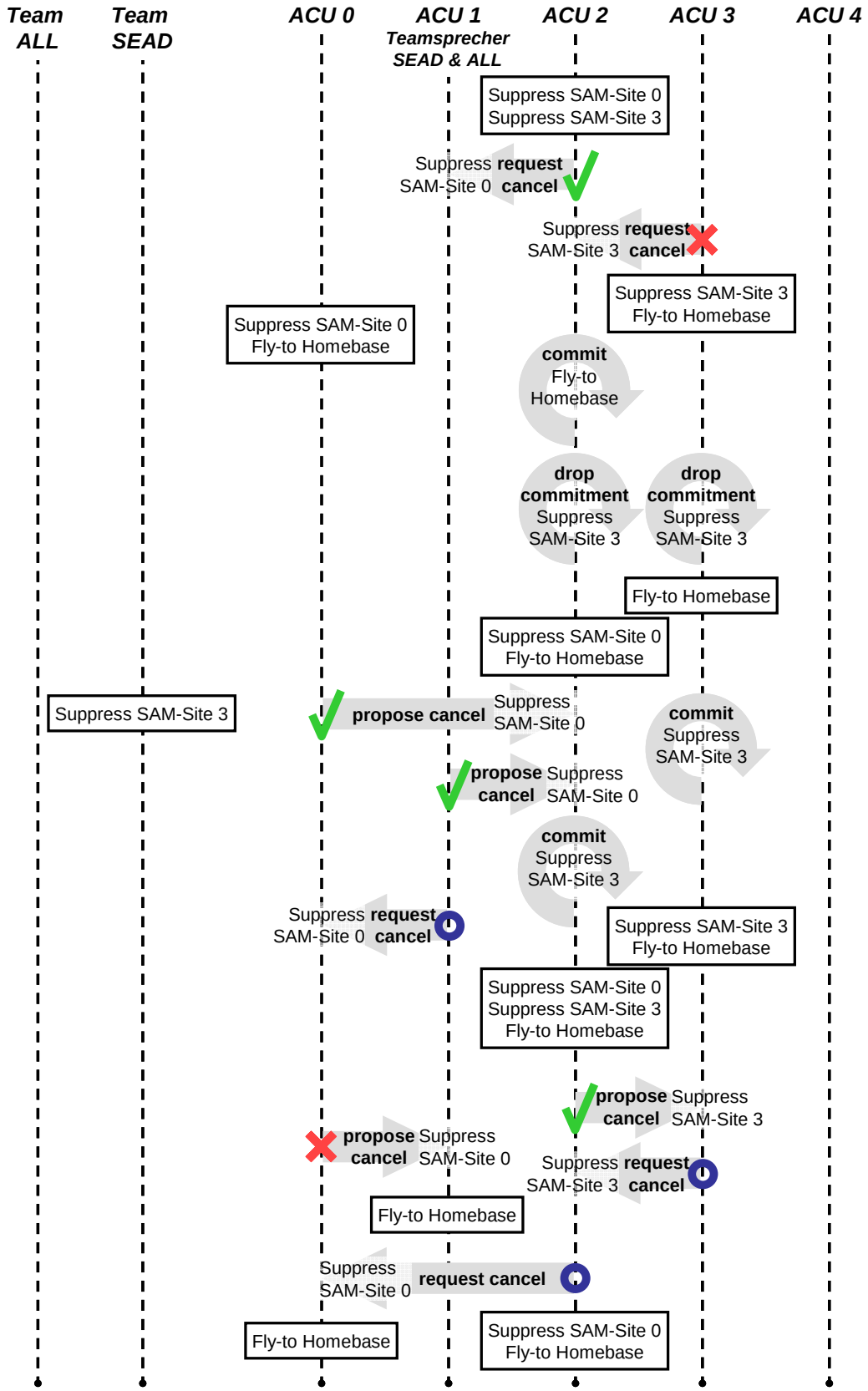


Abbildung 6-17: Aufgabenverteilung im Team zu Beginn der Mission (fortgesetzt)

In Folge der bekannt gewordenen Teamverpflichtung „Fly-to Homebase“ gehen anschließend alle ACUs zu unterschiedlichen Zeitpunkten eine entsprechende individuelle Verpflichtung ein.

Da SAM-Stellung 0 auf dem Weg zum Ziel liegt und SAM-Stellung 3 das Ziel schützt, stellt die Unterdrückung dieser beiden SAM-Stellungen jeweils die Unterstützung eines Teammitglieds, nämlich des Attack-UCAVs 4, dar. Somit verpflichtet sich ACU 1 gemäß dem Ziel *Bearbeite unterstützende Aufgabe* zur Unterdrückung von SAM-Stellung 0 während ACU 4 infolge des applikationsspezifischen Ziels *Vermeide Bedrohung* die Unterdrückung von SAM-Stellung 3 vom SEAD-Team anfragt. Noch bevor eine Teamverpflichtung infolge dieser Anfrage allen SEAD-Teammitgliedern bekannt ist, gehen ACUs 2 und 3 auf Grund des Ziels *Bearbeite unterstützende Aufgabe* Verpflichtungen zur Unterdrückung von SAM-Stellung 3 ein. In gleicher Weise verpflichten sich neben ACU 1 die ACUs 0 und 2 zur Unterdrückung von SAM-Stellung 0.

Im weiteren Verlauf wird im Team auf Basis der Ziele *Vermeide redundante Aufgabenbearbeitung* sowie *Stelle Aufgabenabdeckung sicher* durch eine Reihe von Anfragen und Vorschlägen, sowie dem Eingehen und Aufgeben von Verpflichtungen eine überschneidungsfreie Verteilung von Aufgaben hergestellt, nämlich:

- ***ACU 0:*** Protect UCAV 4, Fly-to Homebase
- ***ACU 1:*** Protect UCAV 4, Fly-to Homebase
- ***ACU 2:*** Suppress SAM-Site 0, Protect UCAV 4, Fly-to Homebase
- ***ACU 3:*** Suppress SAM-Site 3, Protect UCAV 4, Fly-to Homebase
- ***ACU 4:*** Destroy Target, Fly-to Homebase

Schon während der Zuweisung von Aufgaben an Teammitglieder beginnen die UCAVs ihren Flug ins Zielgebiet, wobei der Korridor für die Überquerung der FLOT als gemeinsam genutzte Ressource den einzelnen Teammitgliedern auf Basis des Ziels *Weise gemeinsam genutzte Ressource zu* nacheinander zugewiesen wird. Sobald die einzelnen UCAVs im Zielgebiet angekommen sind, beginnen sie dort mit der Bearbeitung ihrer Aufgaben (*Bearbeite Verpflichtung*), die auf Grund des Ziels *Sortiere Verpflichtung in Agenda ein* in der oben dargestellten Reihenfolge abgearbeitet werden sollen. So fliegt UCAV 2 auf SAM-Stellung 0 zu, um diese zu unterdrücken, während UCAV 3 sich in Richtung von SAM-Stellung 3 begibt (siehe Abbildung 6-18).

Beim Flug durch SAM-Stellung 0 wird UCAV 3 von dieser angegriffen und reagiert mit einem Gegenangriff (vgl. Abbildung 6-19), wobei sowohl SAM-Stellung 0 als auch UCAV 3 getroffen werden (vgl. Abbildung 6-21). Da nun die Verpflichtung von ACU 2 zur Unterdrückung von SAM-Stellung 0 irrelevant ist, gibt sie diese wegen des Ziels *Habe keine überflüssige Verpflichtung* auf.

Durch den Ausfall von ACU 3, die SAM-Stellung 3 unterdrücken sollte, wird es ferner nötig, diese Aufgabe neu zuzuweisen, was auf Basis des Ziels *Stelle Aufgabenabdeckung sicher* erfolgt. Dieses bewirkt, dass sich ACU 0 und 2 zur Unterdrückung von SAM-Stellung 3 verpflichten, jedoch die Redundanz der Zuweisung feststellen und somit das Ziel *Vermeide redundante Aufgabenzuweisung* relevant wird. Daraufhin gibt ACU 0 die Verpflichtung wieder auf und gleichzeitig fragt ACU 2 die Aufgabe der Verpflichtung an, so dass letztlich ACU 2 zur Unterdrückung von SAM-Stellung 3 verpflichtet bleibt (vgl. Abbildung 6-20) und deswegen auf diese SAM-Stellung zufliegt (vgl. Abbildung 6-21).

Nach der erfolgreichen Unterdrückung von SAM-Stellung 3 gibt ACU 2 die entsprechende Verpflichtung wegen des Ziels *Habe keine überflüssige Verpflichtung* auf. Das Attack-UCAV 4 fliegt daraufhin in den Bedrohungsbereich von SAM-Stellung 3 ein, bekämpft erfolgreich das Angriffsziel und gibt die entsprechende Verpflichtung ebenfalls auf Grund des Ziels *Habe keine überflüssige Verpflichtung* auf. Bei allen UCAVs steht nun die Rückkehr zum Stützpunkt an erster Stelle in der Agenda und wird wegen des Ziels *Bearbeite Verpflichtung* bearbeitet, indem zunächst ein Flugplan zum Korridor geplant und anschließend aktiviert wird (vgl. Abbildung 6-22).

Auf dem Rückweg aktiviert sich eine bis dahin unbekannte SAM-Stellung, welche UCAV 2 angreift, im Gegenzug von UCAV 2 angegriffen wird und deren Unterdrückung durch das SEAD-Team von UCAV 4 angefragt wird (vgl. Abbildung 6-23).

Nachdem die von UCAV 2 eingesetzte HARM die SAM-Stellung zerstört hat, werden die vom SEAD-Team und von ACU 2 eingegangenen Verpflichtungen zur Unterdrückung dieser SAM-Stellung wegen des Ziels *Habe keine überflüssige Verpflichtung* wieder aufgegeben und der Rückflug zum Stützpunkt wieder aufgenommen (vgl. Abbildung 6-24). Dort kommen vier der fünf ursprünglich vorhandenen UCAVs an (vgl. Abbildung 6-26), nachdem die Nutzung des Korridors wie zu Beginn der Mission auf Basis des Ziels *Weise gemeinsam genutzte Ressource zu* erfolgte (vgl.Abbildung 6-25).

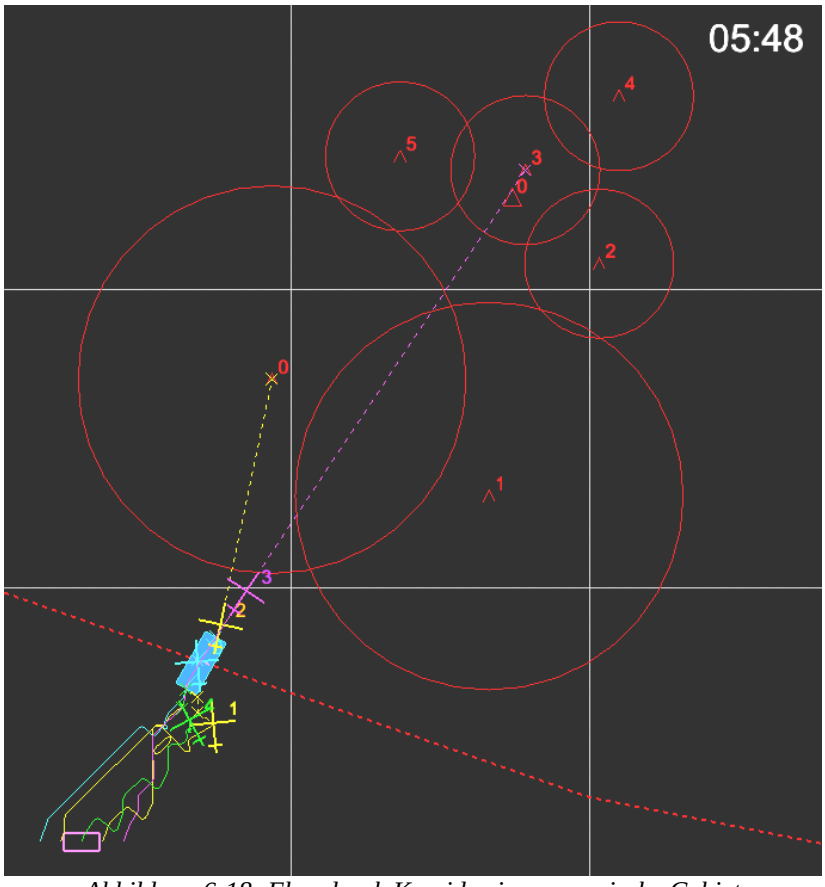


Abbildung 6-18: Flug durch Korridor ins gegnerische Gebiet

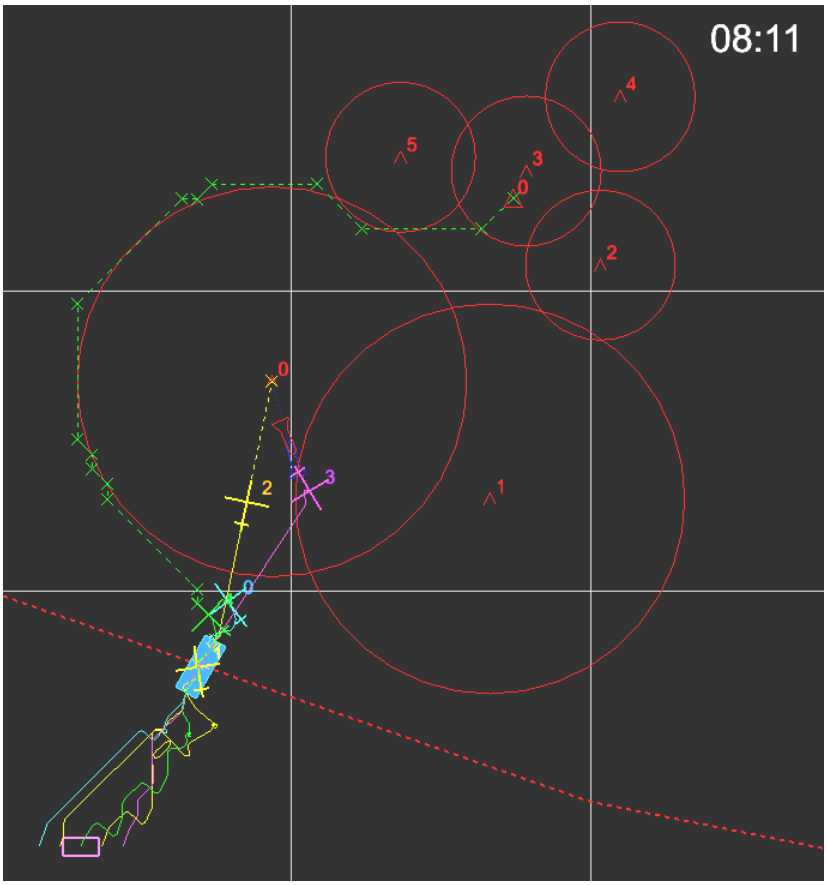


Abbildung 6-19: Angriff von SAM-Stellung 0 durch UCAV 3 sowie Gegenangriff der SAM-Stellung

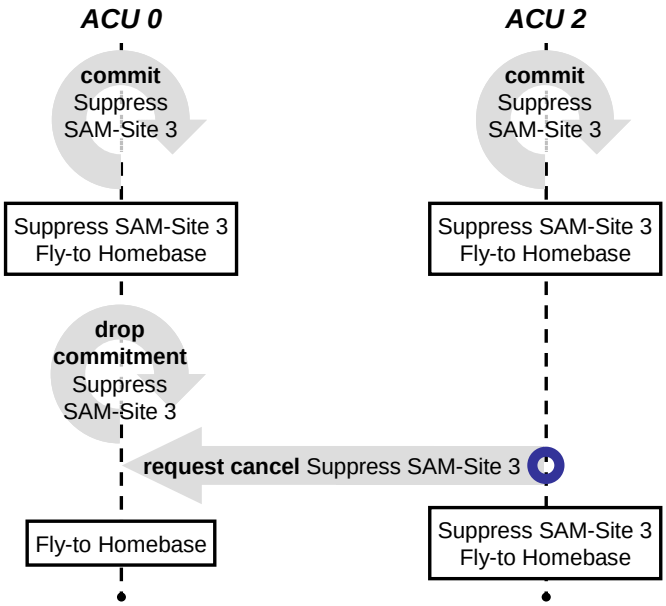


Abbildung 6-20: Zuweisung einer Aufgabe zur Sicherstellung der Aufgabenabdeckung im Team

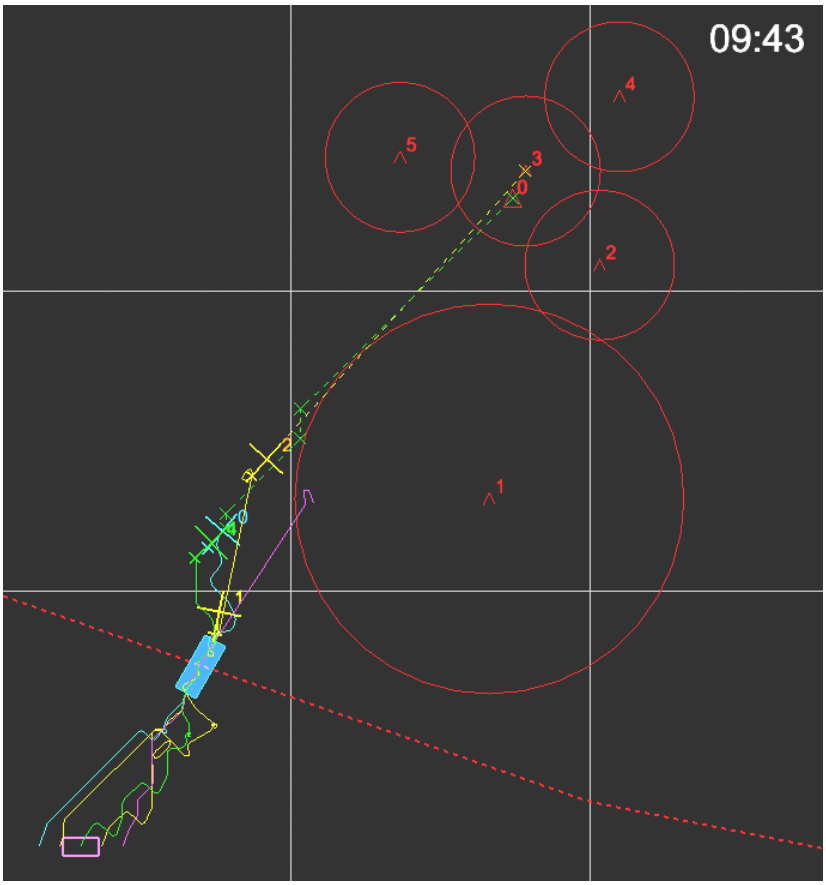


Abbildung 6-21: Lage nach Verlust von UCAV 3 und Bekämpfung von SAM-Stellung 0

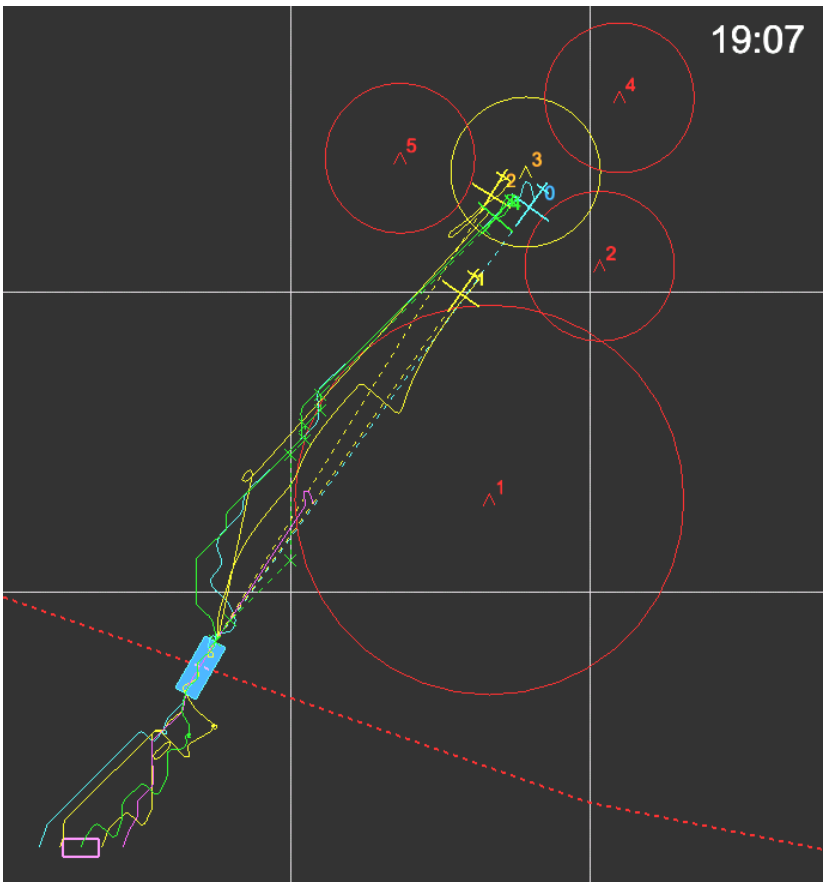


Abbildung 6-22: Beginn des Rückflugs nach erfolgreicher Bekämpfung des Angriffsziels

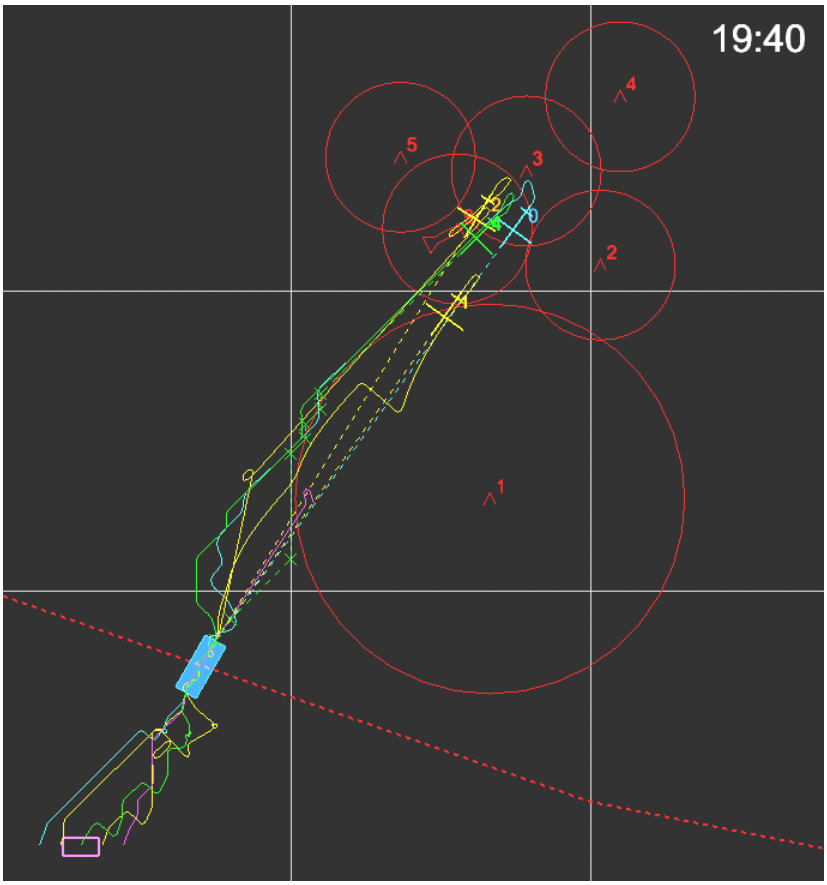


Abbildung 6-23:Aktivierung einer unbekannten SAM-Stellung

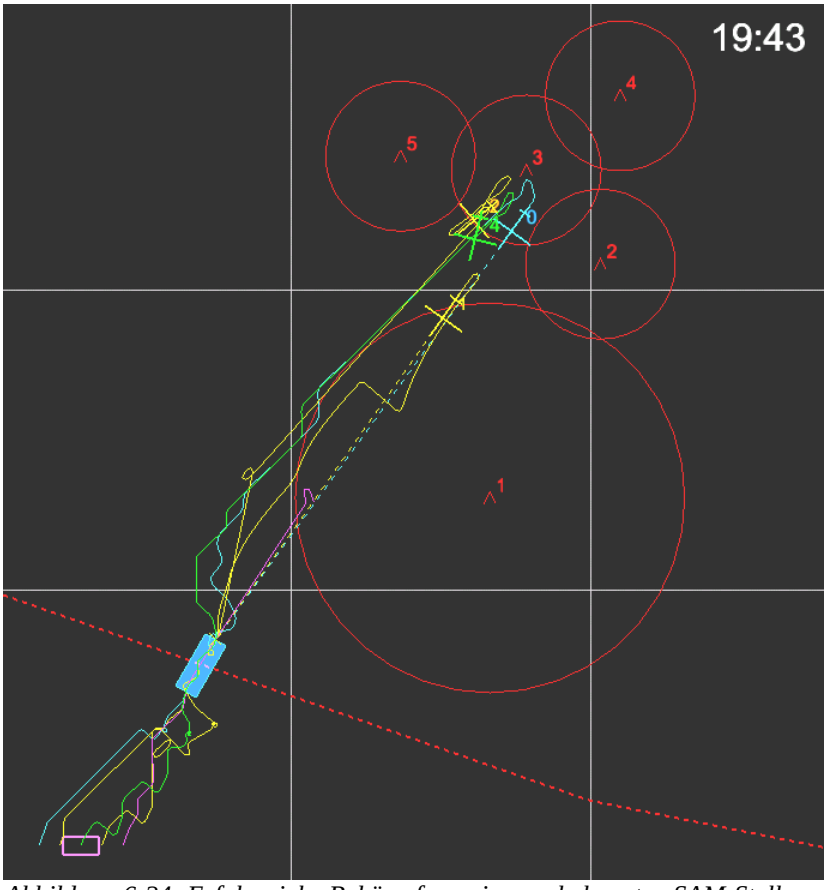


Abbildung 6-24: Erfolgreiche Bekämpfung einer unbekannten SAM-Stellung

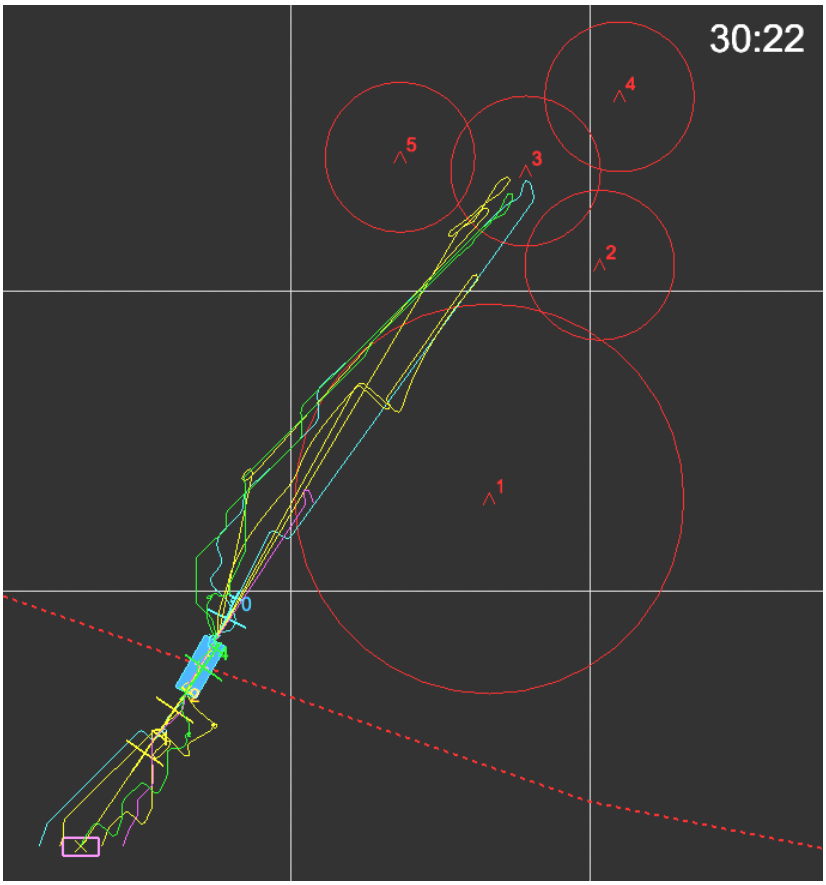


Abbildung 6-25: Nutzung des Korridors für den Rückflug zum Sützpunkt

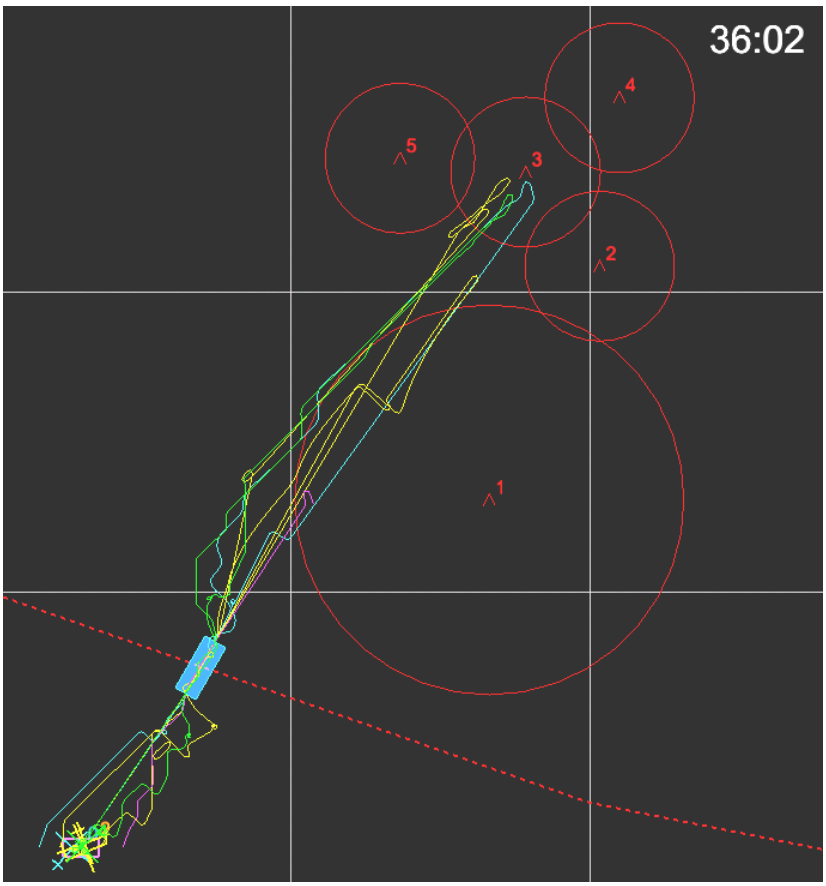


Abbildung 6-26: Rückkehr zum Stützpunkt

Zusammenfassung

Anhand dieses Beispiels für die Durchführung einer Mission im Rahmen eines umfassenden Szenarios kann also gezeigt werden, dass das implementierte Wissen auch in komplexen Szenarien auf funktionaler Ebene sinnvolles und erwünschtes Verhalten zeigt. Insbesondere kann das entwickelte Team aus *Supporting ACUs* im Gesamtkontext einer Mission sinnvolle Aufgaben an einzelne Teammitglieder zuweisen, wobei sowohl die Fähigkeiten der Teammitglieder als auch bereits eingegangene Verpflichtungen berücksichtigt werden und die Aufgabenverteilung an den tatsächlichen Verlauf der Mission und hierbei vor allem auch an unerwartete Ereignisse angepasst wird.

Wie an diesem und auch an ähnlichen Missionsszenarien festgestellt werden kann, ist das realisierte System also in der Lage, ohne zentrale Vorplanung oder situationsspezifisches Vorwissen erfolgreich zu agieren. Dies ist inbesondere auf die angelegte Fähigkeit zu wissensbasiertem Verhalten zurückzuführen. Dieser wesentliche Aspekt wird im Folgenden weiter diskutiert.

6.3 Wissensbasiertes Verhalten

Verhalten auf wissensbasierter Ebene im Sinne von Rasmussen zeichnet sich wie in Abschnitt 3.2.2.1 erläutert vor allem dadurch aus, dass es an explizit repräsentierten Zielen ausgerichtet ist. Im Gegensatz zur regelbasierten Verhaltensebene steht damit nicht im Vordergrund, wie in einer bestimmten Situation vorgegangen werden soll, sondern was erreicht werden soll. Weiterhin ermöglicht der Aufbau eines umfassenden Situationsverständnisses in Verbindung mit der Berücksichtigung von Zielen auf wissensbasierter Verhaltensebene statt einer starren Zuweisung von Situationen zu Aktionen eine größere Flexibilität im Hinblick auf die Situationskonfigurationen, in denen ein System sinnvoll agieren kann. Voraussetzung hierfür ist die Verfügbarkeit von in einer Situation relevantem Wissen immer dann, wenn es benötigt wird. Letzteres ist durch den im Rahmen der vorliegenden Arbeit gewählten Ansatz des Kognitiven Prozesses in Verbindung mit der Architektur COSA gegeben, da dieser Wissen von Verarbeitung trennt und zu jedem Zeitpunkt sämtliches zur Verfügung stehende Wissen auch einsetzen kann. Dies steht im diametralen Gegensatz zu konventionellen, prozedural programmierten Systemen, die sich in den entsprechenden Routinen befinden müssen, um dort hinterlegtes Wissen auch einsetzen zu können.

Wie in den vorhergehenden Kapiteln erläutert, basiert der im Rahmen dieser Arbeit entwickelte Funktionsprototyp auf dem Kognitiven Prozess als einem Paradigma, das vor allem für die Modellierung von Verhalten auf wissensbasierter Ebene ausgelegt ist (vgl. Abschnitt 3.2.3.2), und modelliert Verhalten auf dieser Ebene sowohl für die Zusammenarbeit zwischen mehreren ACUs als auch Aktivitäten einer einzelnen ACU. Die folgenden Abschnitte zeigen nun, dass dieses Verhalten Eigenschaften aufweist, wie sie typisch für die wissensbasierte Ebene sind, nämlich dass in einer Situation unterschiedliche Vorgehensweisen verfolgt werden können, die aber letztlich zum gleichen Ergebnis im Sinne der Ziele führen (Abschnitt 6.3.1) und dass Situationen gehandhabt werden können, die in dieser Form während des Entwicklungsprozesses nicht vorhergesagt und damit nicht berücksichtigt wurden (Abschnitt 6.3.2).

6.3.1 Zielorientierte Vorgehensplanung

Wie bereits dargestellt, besteht auf wissensbasierter Verhaltensebene keine direkte Verknüpfung zwischen spezifischen Situationen und dem jeweils anzuwendenden Vorgehen. Daher muss zunächst ermittelt werden, was erreicht werden soll, um anschließend

ein Vorgehen planen zu können, wie die definierte Zielsituation erreicht werden kann. Hierbei ist es zunächst nicht wesentlich, wie eine Lösung erzielt wird und ob die gewählte Vorgehensweise optimal hinsichtlich verschiedener Kriterien ist, sondern dass sich die Situation im Sinne der Ziele verändert.

Diese Forderung gilt sowohl im Rahmen des Erreichens individueller Ziele einer ACU als auch von Zielen, welche die Zusammenarbeit mehrerer ACUs im Team betreffen. Während jedoch eine einzelne ACU all die Ziele kennt und im Prinzip bei der Planung berücksichtigen kann, an deren Erreichen sie selbst interessiert ist, können mehrere ACUs in einem dezentral organisierten Team nicht den internen Zustand aller anderen ACUs kennen und in ihre Überlegungen mit einbeziehen. Dennoch ergibt sich durch die im Rahmen dieser Arbeit implementierten Ziele von Kooperation auch bei der Zusammenarbeit mehrerer ACUs Verhalten, das sich in vergleichbaren Situationen unterscheidet, aber letztlich zum gleichen Ergebnis im Sinne der Ziele führt.

Zur Verdeutlichung werden hier mehrere Läufe von Teilproblem 3 (vgl. Abschnitt 6.2.1.3, Abbildung 6-10) herangezogen, bei dem zwei SEAD-UCAVs (IDs 0 und 1) zwei SAM-Stellungen (IDs 0 und 1) bekämpfen sollen. Nach Eingang des Auftrags vom Operateur, alle SAM-Stellungen zu bekämpfen, müssen hier zunächst im Sinne der Ziele *Sei einer Aufgabe verpflichtet*, *Stelle Aufgabenabdeckung sicher*, *Vermeide redundante Aufgabenbearbeitung* und *Verteile Arbeitsbelastung* die beiden Aufgaben „Bekämpfe SAM-Stellung 0“ und „Bekämpfe SAM-Stellung 1“ jeweils einer ACU zugewiesen werden. Anschließend wird die Anzahl der verfügbaren HARMs von UCAV 1 im Sinne einer nicht vorhersehbaren Situationsänderung „künstlich“, d.h. durch externen Eingriff, auf Null gesetzt, so dass eine Umverteilung der Aufgaben stattfinden und UCAV 0 nun alle Aufgaben bearbeiten muss, wobei hier vor allem das Ziel *Stelle Aufgabenabdeckung sicher* von Bedeutung ist.

In Abbildung 6-27 bis Abbildung 6-31 sind verschiedene Vorgehensweisen der beteiligten ACUs dargestellt, um zu Missionsbeginn zu einer ausgewogenen Aufgabenverteilung im Team zu gelangen (für eine Erläuterung der Darstellungsweise siehe Seite 123). Die konkrete Zusammensetzung der dargestellten Abläufe aus einzelnen Aktionen wird dabei von diversen Faktoren beeinflusst.

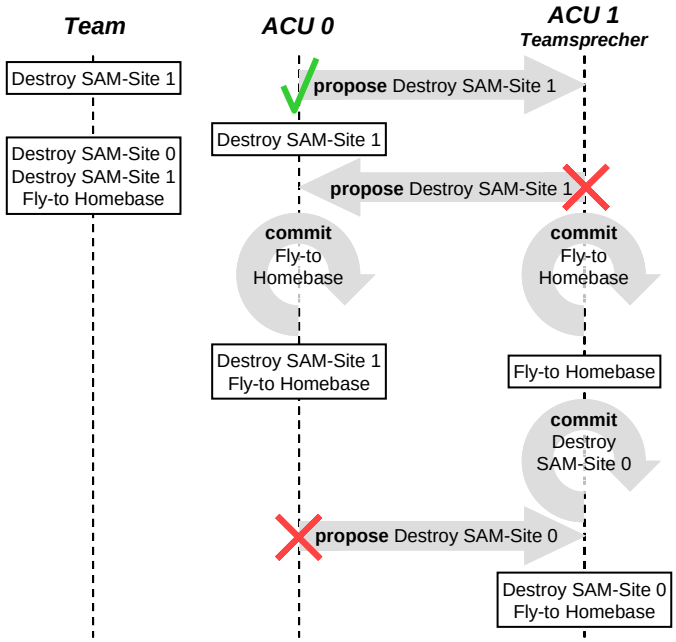


Abbildung 6-27: Initiale Zuweisung von Aufgaben – Vorgehensweise 1

Hierunter fallen zum Beispiel welche ACU die Rolle des Teamsprechers übernimmt, in welcher Reihenfolge die Verpflichtungen des Teams und einzelner Teammitglieder anderen bekanntgegeben werden, welche Handlungsalternativen ausgewählt werden, um die aktuell relevanten Zielsetzungen zu erreichen (vgl. auch Abbildung 4-4) und in welcher Reihenfolge diese ausgeführt werden. Ohne im Detail auf die einzelnen Abläufe einzugehen, soll hier lediglich darauf hingewiesen werden, dass sich in allen dargestellten Fällen über mehr oder weniger Zwischenschritte stets das gleiche Ergebnis im Sinne der Ziele *Stelle Aufgabenabdeckung sicher*, *Vermeide redundante Aufgabenzuweisung* und *Verteile Arbeitsbelastung* ergibt. Dieses zeichnet hier sich dadurch aus, dass letztlich jede ACU die Bekämpfung einer SAM-Stellung übernimmt. Wie die unterschiedliche Komplexität der einzelnen gefundenen Lösungswege in Abbildung 6-27 bis Abbildung 6-31 bereits erahnen lässt, ergeben sich unterschiedlich effiziente Prozedere, die gewünschte Zielsituation zu erreichen. Welche Lösung tatsächlich im Einzelfall gefunden wird, kann von mehr oder weniger zufälligen Faktoren abhängen. Hierbei ist kein prozedurales Wissen im Sinne eines optimalen Vorgehens hinterlegt. Eine Optimierung bzw. Steuerung des Vorgehens könnte durch Hinzufügen zusätzlichen Wissens erreicht werden, würde jedoch am Resultat, d.h. dem Erreichen der Zielkonfiguration, nichts ändern. Auf die hier dargestellte Weise wird somit gerade die Flexibilität des Ansatzes demonstriert, auf Basis von Zielen und Handlungsalternativen auch auf unterschiedlichen, nicht prozedural determinierten Wegen zu Lösungen zu kommen.

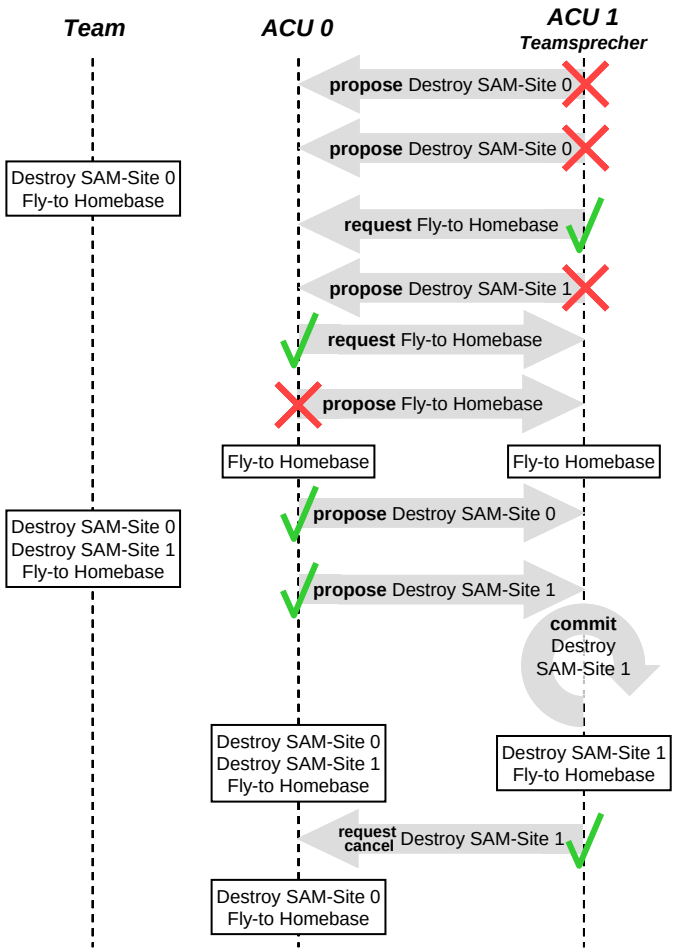


Abbildung 6-28: Initiale Zuweisung von Aufgaben – Vorgehensweise 2

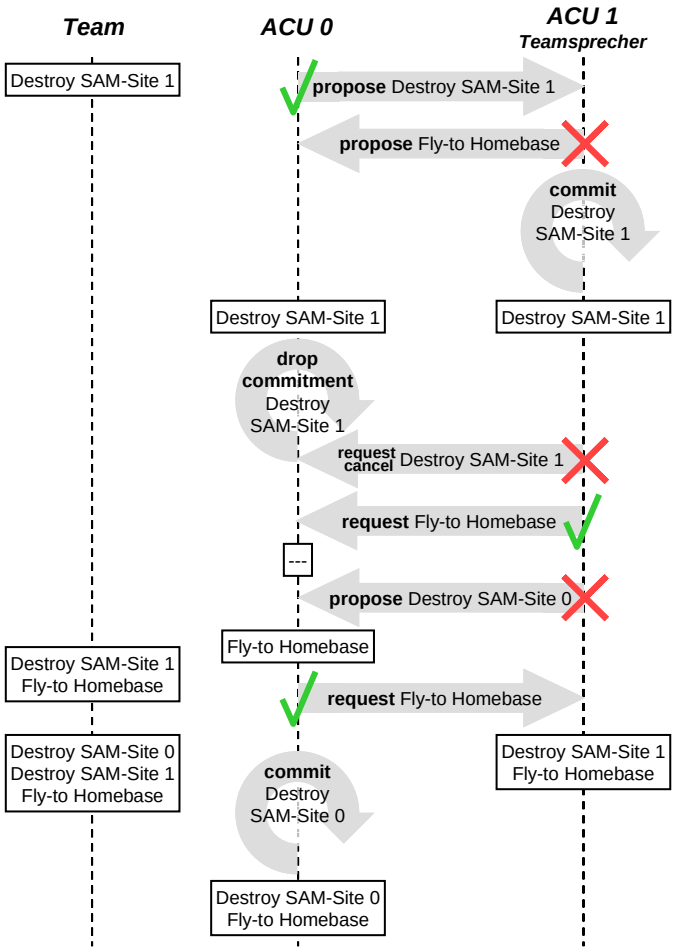


Abbildung 6-29: Initiale Zuweisung von Aufgaben – Vorgehensweise 3

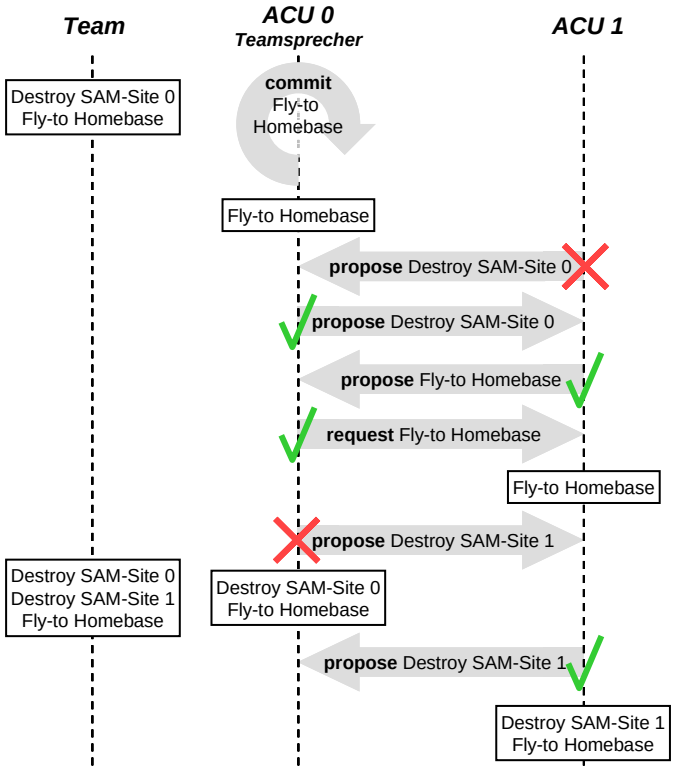


Abbildung 6-30: Initiale Zuweisung von Aufgaben – Vorgehenweise 4

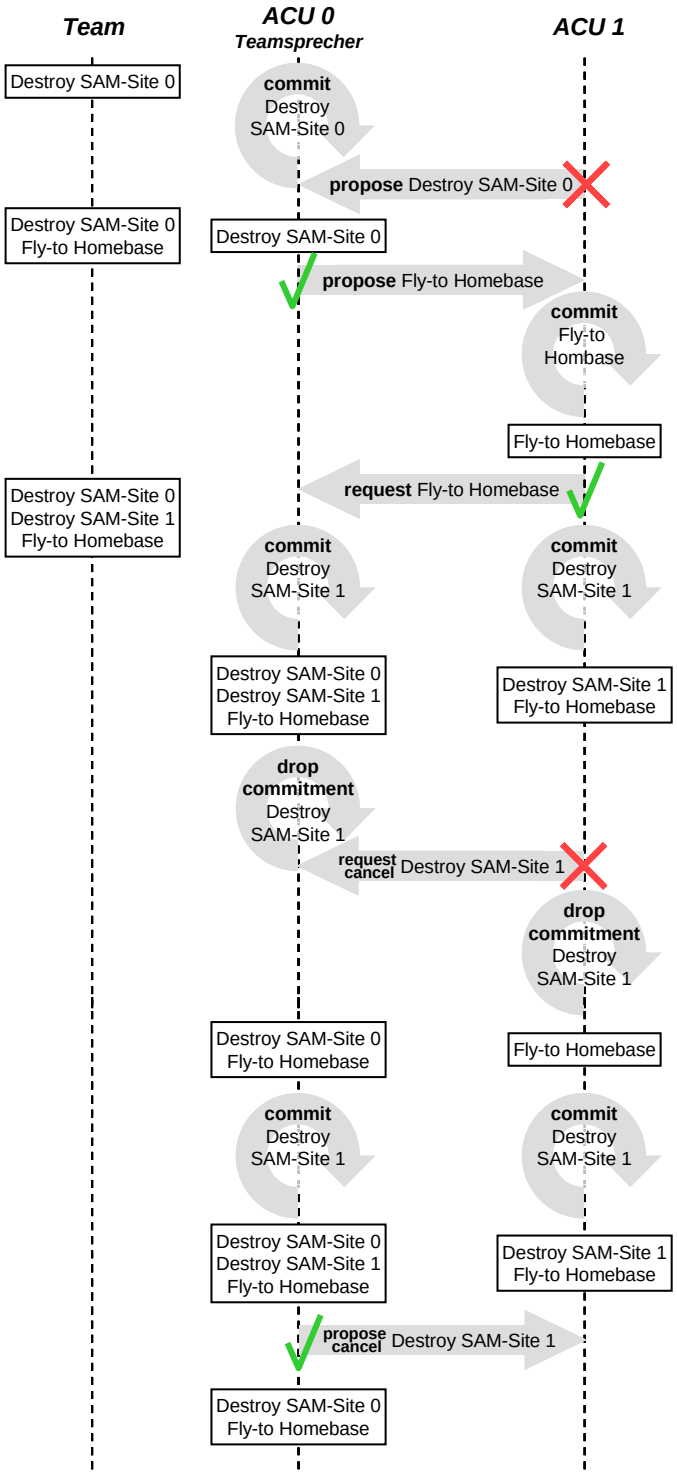


Abbildung 6-31: Initiale Zuweisung von Aufgaben – Vorgehensweise 5

Wie oben beschrieben wird im weiteren Verlauf jeder Mission eine Änderung der verfügbaren Ressourcen von ACU 1 insofern herbeigeführt als diese nun keine HARMs mehr zur Verfügung hat und somit ihre Verpflichtung zur Bekämpfung einer SAM-Stellung aufgeben muss. Vor dem Hintergrund der Verpflichtungen des Teams zur Bekämpfung beider SAM-Stellungen wird nun das Ziel *Stelle Aufgabenabdeckung sicher* aktiviert, das wiederum über verschiedene Vorgehensweisen erreicht werden kann. Im Einzelnen sind dies:

- ACU 1 teilt dem Team mit, dass sie nur noch zur Rückkehr zum Stützpunkt verpflichtet ist, woraufhin ACU 0 vorschlägt, die Aufgabe „Bekämpfe SAM-Stellung 1“ zu übernehmen und ACU 1 zeitgleich beim Team anfragt, ob diese Aufgabe übernommen werden kann. Da der Vorschlag von ACU 0 angenommen wird, ist sie schließlich zur vorgeschlagenen Aufgabe verpflichtet (siehe Abbildung 6-32).

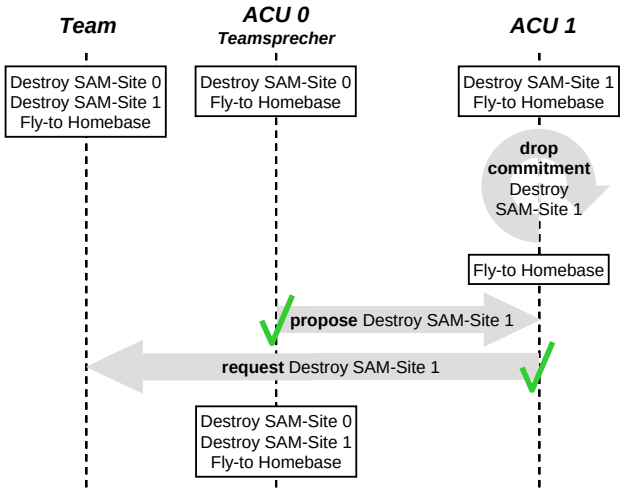


Abbildung 6-32: Aufgabenneuzuweisung im Team – Vorgehensweise 1

- Bevor ACU 1 dem Team mitteilt, dass sie nur noch zur Rückkehr zum Stützpunkt verpflichtet ist, fragt sie beim Team an, ob die Aufgabe „Bekämpfe SAM-Stellung 1“ übernommen werden kann. Daraufhin schlägt ACU 0 erfolgreich die Bearbeitung dieser Aufgabe vor und geht die entsprechende Verpflichtung ein (siehe Abbildung 6-33).

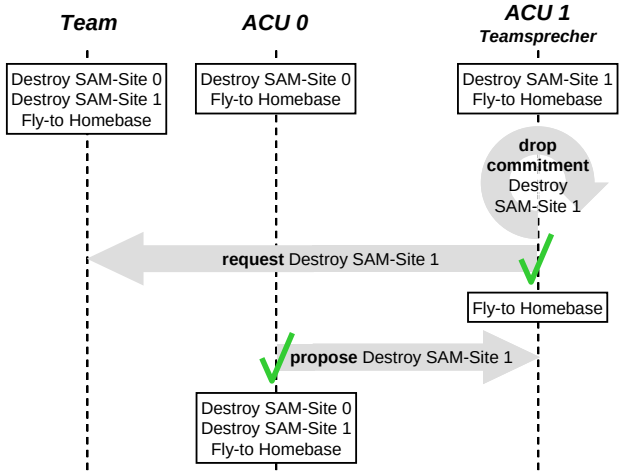


Abbildung 6-33: Aufgabenneuzuweisung im Team – Vorgehensweise 2

- Bevor ACU 1 dem Team mitteilt, dass sie nur noch zur Rückkehr zum Stützpunkt verpflichtet ist, fragt sie beim Team an, ob die Aufgabe „Bekämpfe SAM-Stellung 1“ übernommen werden kann. Daraufhin geht ACU 0 ohne vorherige Rückfrage beim Teammitglied die entsprechende Verpflichtung ein (siehe Abbildung 6-34).

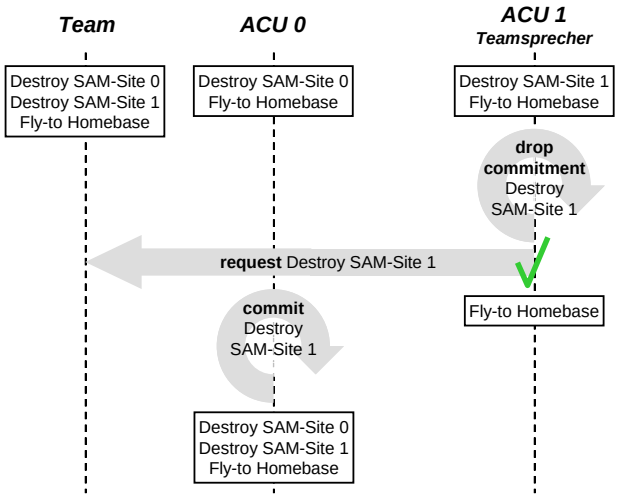


Abbildung 6-34: Aufgabenneuzuweisung im Team – Vorgehensweise 3

- ACU 1 teilt dem Team mit, dass sie nur noch zur Rückkehr zum Stützpunkt verpflichtet ist, woraufhin ACU 0 vorschlägt, die Aufgabe „Bekämpfe SAM-Stellung 0“ zu übernehmen. Da der Vorschlag von ACU 0 angenommen wird, ist diese schließlich zur vorgeschlagenen Aufgabe verpflichtet (siehe Abbildung 6-35).

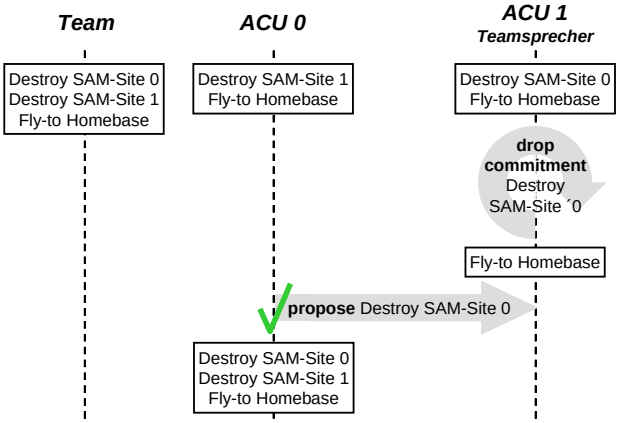


Abbildung 6-35: Aufgabenneuzuweisung im Team – Vorgehensweise 4

- ACU 1 teilt dem Team mit, dass sie nur noch zur Rückkehr zum Stützpunkt verpflichtet ist, woraufhin ACU 0 sich ohne vorherige Rückfrage verpflichtet, die Aufgabe „Bekämpfe SAM-Stellung 1“ zu übernehmen (siehe Abbildung 6-36).

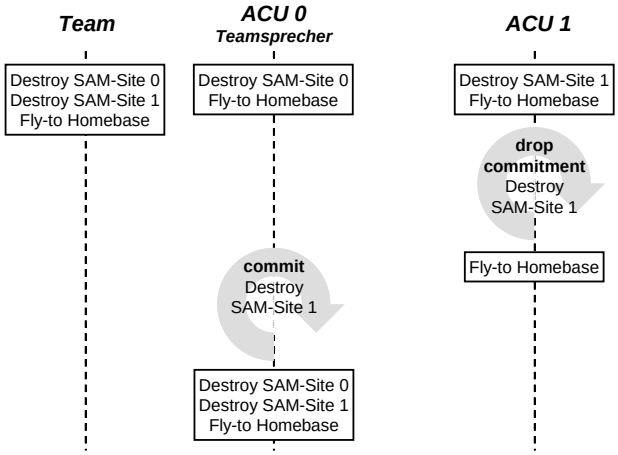


Abbildung 6-36: Aufgabenneuzuweisung im Team – Vorgehensweise 5

Wie bereits bei der initialen Aufgabenzuweisung resultieren auch hier sämtliche dargestellten Abläufe darin, dass die angestrebte Zielsituation erreicht wird und zwar hier indem ACU 0 die Verpflichtung von ACU 1 übernimmt.

Die hier dargestellten Vorgehensweisen zur Aufgabenzuweisung sowohl zu Beginn der Mission als auch während der Mission sind sicherlich keine effizienten Abläufe im Hinblick auf die Anzahl der Aktionen, die jeweils ausgeführt werden, bis die Aufgabenzuweisung letztlich erfolgt ist und könnten vermutlich insbesondere für die hier betrachtete Konfiguration „2 UCAVs und 2 SAM-Stellungen“ auf der regelbasierten Verhaltensebene deutlich effizienter gestaltet werden. Jedoch müsste hier wie bereits mehrmals diskutiert jede spezifische Situation explizit mit den auszuführenden Aktivitäten verknüpft werden, so zum Beispiel „Wenn ein UCAV zur Bekämpfung einer SAM-Stellung verpflichtet ist und ein anderes UCAV zur Bekämpfung einer anderen SAM-Stellung verpflichtet ist und das eine UCAV über keine HARMs mehr verfügt, während das andere UCAV noch genügend HARMs zur Bekämpfung beider SAM-Stellungen hat, dann soll letzteres UCAV die Verpflichtung des ersteren übernehmen.“ Mit diesem Ansatz ist es jedoch praktisch unmöglich, alle eventuell auftretenden Situationskonfigurationen abzudecken, vor allem wenn die Anzahl der beteiligten UCAVs und/oder SAM-Stellungen erhöht wird. Daher wird hier als vielversprechender Ansatz eine Kombination der regelbasierten Verhaltensebene für häufig auftretende Situationen mit der wissensbasierten Verhaltensebene angesehen, welche in der Lage ist, auch in nicht vorher spezifizierten Situationen zielgerichtete Vorgehensweisen und damit sinnvolles Verhalten hervorzubringen (siehe auch nächster Abschnitt).

6.3.2 Zielorientiertes Verhalten in unbekannten Situationen

Während im vorhergehenden Abschnitt exemplarisch gezeigt wurde, dass eine an Zielen ausgerichtete Vorgehensplanung zwar unterschiedliche Vorgehensweisen in ähnlichen Situationen vorschlagen kann, diese aber in der Regel zum gleichen Ergebnis im Sinne der Ziele führen, liegt der Fokus dieses Abschnittes auf zielorientiertem Verhalten in unbekannten Situationen. In diesem Zusammenhang soll gezeigt werden, dass die kontextunabhängige Repräsentation expliziter Handlungsziele in Verbindung mit einem adäquaten Verständnis der vorherrschenden Situation seitens der ACU auch in Situationen, die während des Entwicklungsprozesses so nicht berücksichtigt wurden, zu sinnvollem, d.h. an den Zielen ausgerichtetem Verhalten, führt. Dieses muss wie im vorherigen Abschnitt diskutiert nicht dem in dieser Situation als optimal erachteten Verhalten entsprechen, zeichnet sich jedoch dadurch aus, dass die jeweilige ACU unter bestmöglichem Einsatz ihres verfügbaren Wissens versucht, die vorherrschenden Ziele zu erreichen, statt undefiniertes und damit möglicherweise kontraproduktives Verhalten an den Tag zu legen.

Um zu zeigen, dass mit dem im Rahmen dieser Arbeit gewählten Ansatz solches zieleorientierte Verhalten in unbekannten Situationen möglich ist, wird im Folgenden das Verhalten des implementierten Funktionsprototyps in verschiedenen Situationen charakterisiert. Diese stellen wie die in Abschnitt 6.2.1 verwendeten Teilprobleme Ausschnitte des in Abschnitt 6.1.1 vorgestellten SEAD-/Attack-Szenarios dar. Im Einzelnen werden dabei die folgenden Situationsänderungen betrachtet: eine Änderung der taktischen Lage, eine Änderung der Teamstruktur durch Ausfall des Teamsprechers bzw. Hinzunahme eines zusätzlichen Teammitglieds und nicht mit den Zielen einer ACU konformes Verhalten einer konventionellen Automationsfunktion.

6.3.2.1 Änderung der taktischen Lage

Gegenstand dieses Abschnitts ist die Beschreibung des Verhaltens zweier ACUs, wenn sich im Rahmen des in Abschnitt 6.2.1.4 definierten Teilproblems die taktische Lage ändert. In allen betrachteten Fällen erfolgt zunächst die Aufgabenverteilung derart, dass das Attack-UCAV eine Verpflichtung zur Bekämpfung des Angriffsziels eingeht, während das SEAD-UCAV eine Verpflichtung zur Unterdrückung der (bekannten) SAM-Stellung eingeht, welche das Ziel schützt (vgl. z.B. Abbildung 6-37 links). Im Verlauf der Mission taucht nun eine weitere SAM-Stellung auf, so dass entsprechend der Situation Ziele aktiviert und Aktionen ausgeführt werden. Abbildung 6-37 stellt zunächst den Fall dar, dass sich eine zusätzliche SAM-Stellung aktiviert, welche das Ziel schützt.

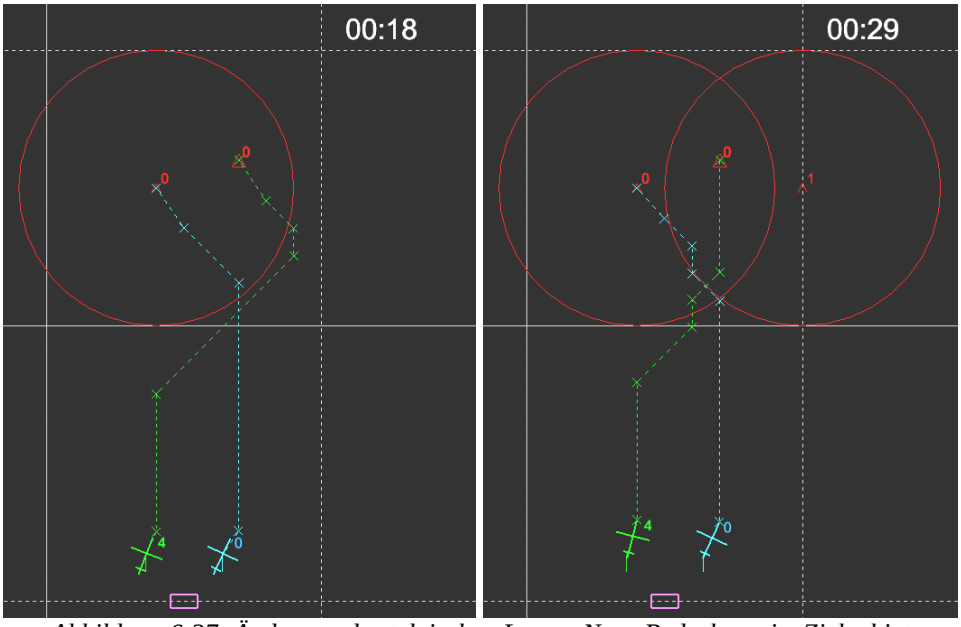


Abbildung 6-37: Änderung der taktischen Lage – Neue Bedrohung im Zielgebiet

Infolgedessen wird beim SEAD-UCAV das Ziel *Bearbeite unterstützende Aufgabe* und beim Attack-UCAV das Ziel *Vermeide Bedrohung* aktiviert, so dass das Attack-UCAV (ID 4) eine Anfrage an das SEAD-UCAV (ID 0) stellt, die neue SAM-Stellung 1 zu unterdrücken und das SEAD-UCAV in der Folge nicht nur zur Unterdrückung von SAM-Stellung 0, sondern auch von SAM-Stellung 1 verpflichtet ist (vgl. Tabelle 6-18).

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
4-18	*dialog-request*			
00:22	request	4	0	(task(name task-suppress)(object sam-site)(sam-site 1))
00:24	agree	0	4	
0-15	*dialog-inform*			
00:24	inform	0	4, 0	(information(actor 0)(type commitment)) (commitment(responsible(actor 0))(task(name task-suppress)(sam-site 0))) (commitment(responsible(actor 0))(task(name task-suppress)(sam-site 1))) (commitment(responsible(actor 0))(task(name task-protect)(ucav 4))) (commitment(responsible(actor 0))(task(name task-fly-to)(object homebase)))

Tabelle 6-18: Dialoge infolge einer neuen Bedrohung im Zielgebiet
(K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/4)

Taucht nun die neue Bedrohung nicht direkt im Zielgebiet, sondern auf dem Weg dorthin auf, so erfolgt zunächst eine Umplanung (vgl. Abbildung 6-38). Da die hieraus resultierenden Flugwege keine weitere Einschränkung im Hinblick auf die benötigte Flugzeit darstellen, werden hier keine weiteren Maßnahmen ergriffen.

Anders verhält es sich bei der in Abbildung 6-39 dargestellten Situation, bei der ebenfalls eine neue Bedrohung auf dem Weg ins Zielgebiet auftaucht, deren Umfliegen allerdings einen großen Umweg bedeuten würde. Auch hier werden die Ziele *Vermeide Bedrohung* und *Bearbeite unterstützende Aufgabe* aktiviert, woraufhin sich das SEAD-UCAV (ID 0) zur Unterdrückung der neu hinzugekommenen SAM-Stellung 1 verpflichtet (vgl. Tabelle 6-19).

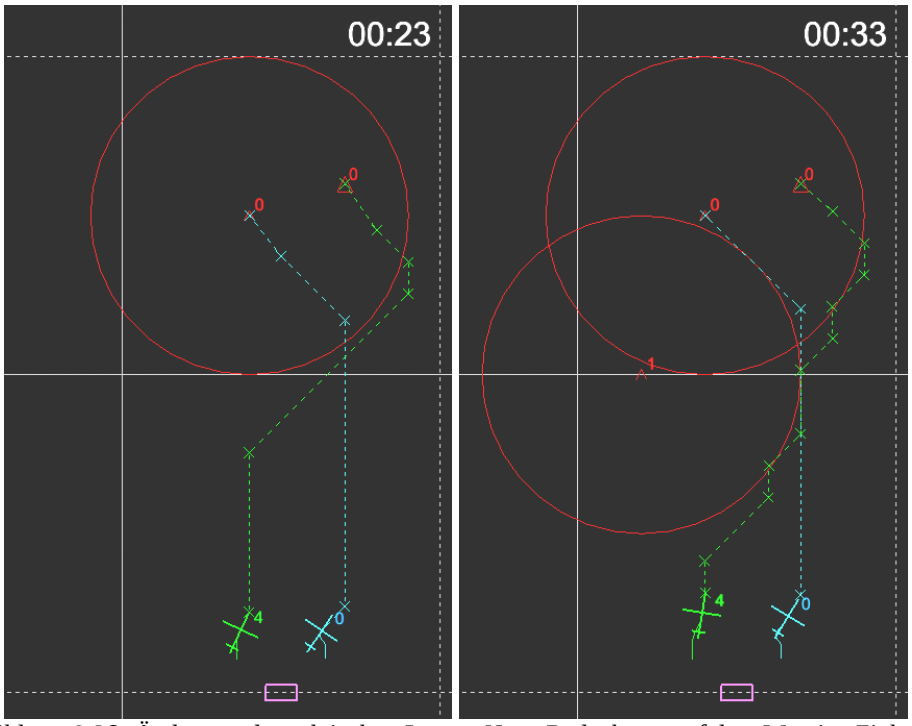


Abbildung 6-38: Änderung der taktischen Lage – Neue Bedrohung auf dem Weg ins Zielgebiet

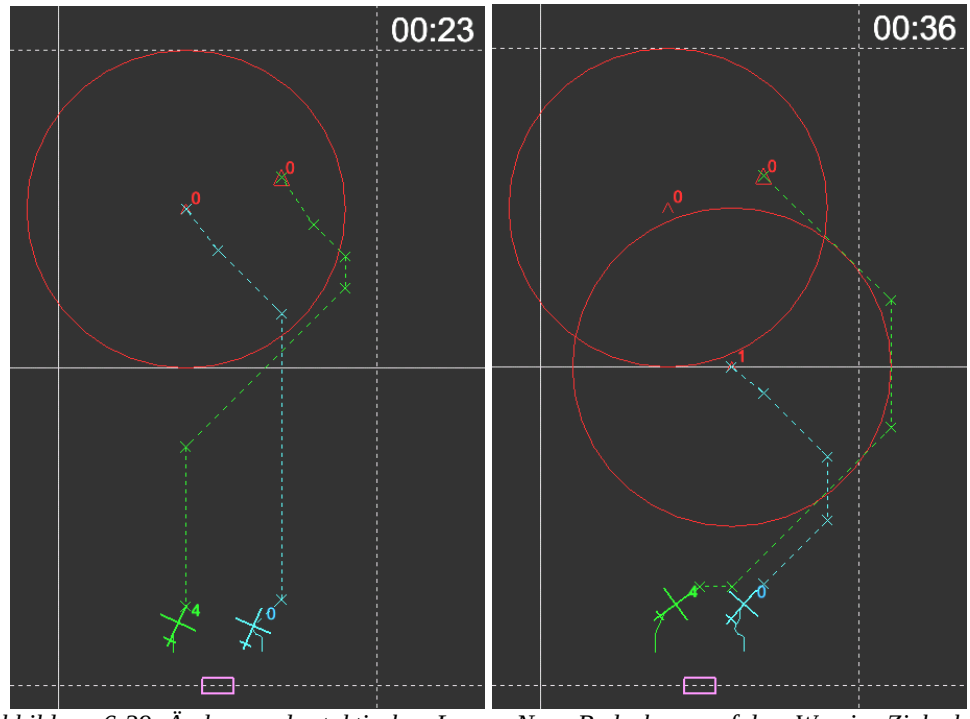


Abbildung 6-39: Änderung der taktischen Lage – Neue Bedrohung auf dem Weg ins Zielgebiet

K	*Protokoll*			
Zeit	*Perf.*	*S*	*E*	*Inhalt*
0-15	*dialog-inform*			
00:28	inform	0	4, 0	(information(actor 0)(type commitment)) (commitment(responsible(actor 0))(task(name task-suppress)(sam-site 1))) (commitment(responsible(actor 0))(task(name task-suppress)(sam-site 0))) (commitment(responsible(actor 0))(task(name task-protect)(ucav 4))) (commitment(responsible(actor 0))(task(name task-fly-to)(object homebase)))

Tabelle 6-19: Dialoge infolge einer neuen Bedrohung auf dem Weg ins Zielgebiet (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/4)

Ein vierter Fall ist in Abbildung 6-40 dargestellt. Hier befinden sich die UCAVs direkt im Gefährdungsbereich einer sich unerwartet aktivierenden SAM-Stellung, so dass das Ziel *Vermeide Bedrohung* bei beiden UCAVs aktiviert wird. Infolge einer Anfrage des Attack-UCAVs (vgl. Tabelle 6-20), welches zusätzlich aus dem Gefährdungsbereich fliegt, greift das SEAD-UCAV die bis dahin unbekannte SAM-Stellung 1 an. Nachdem sich diese temporär abgeschaltet hat, wird der Angriff auf das Ziel fortgesetzt.

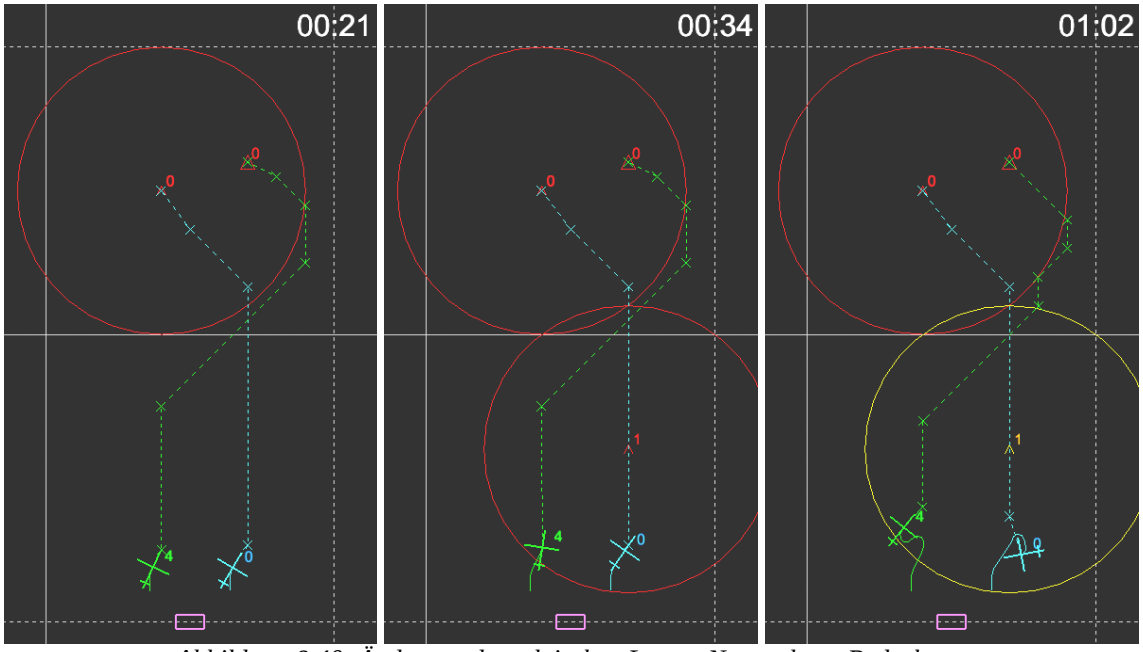


Abbildung 6-40: Änderung der taktischen Lage – Neue, aktute Bedrohung

K	*Protokoll*			
Zeit	*Perf.*	*S*	*E*	*Inhalt*
4-16	*dialog-request*			
00:38	request	4	0	(task(name task-suppress)(object sam-site)(sam-site 1))
00:40	agree	0	4	
0-14	*dialog-inform*			
00:43	inform	0	4, 0	(information(actor 0)(type commitment)) (commitment(responsible(actor 0))(task(name task-suppress)(sam-site 1))) (commitment(responsible(actor 0))(task(name task-suppress)(sam-site 0))) (commitment(responsible(actor 0))(task(name task-protect)(ucav 4))) (commitment(responsible(actor 0))(task(name task-fly-to)(object homebase)))

Tabelle 6-20: Dialoge infolge einer neuen, akuten Bedrohung (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0/4)

Anhand dieser unterschiedlichen Situationen, in denen eine neue SAM-Stellung zu einem nicht vorher bekannten Zeitpunkt an einem unbekannten Ort Teil des betrachteten Szenarios wird, kann somit exemplarisch gezeigt werden, dass auf Basis einer Inter-

pretation der geänderten Lage und der entsprechenden Ziele eine situationsangepasste Reaktion der ACUs erfolgt.

6.3.2.2 Änderung der Teamstruktur

Nachdem im vorhergehenden Abschnitt das Verhalten der UCAVs infolge einer Änderung der taktischen Lage charakterisiert wurde, wird nun eine unvorhergesehene Änderung der Teamstruktur betrachtet. Dazu wird ein Szenario betrachtet, bei dem ein Attack-UCAV (ID 4) und zwei SEAD-UCAVs (IDs 0 und 1) ein Angriffsziel bekämpfen sollen, das von zwei SAM-Stellungen geschützt wird (vgl. Abbildung 6-41). In diesem Zusammenhang werden der Ausfall des Teamsprechers und das Hinzukommen eines neuen Teammitgliedes betrachtet.

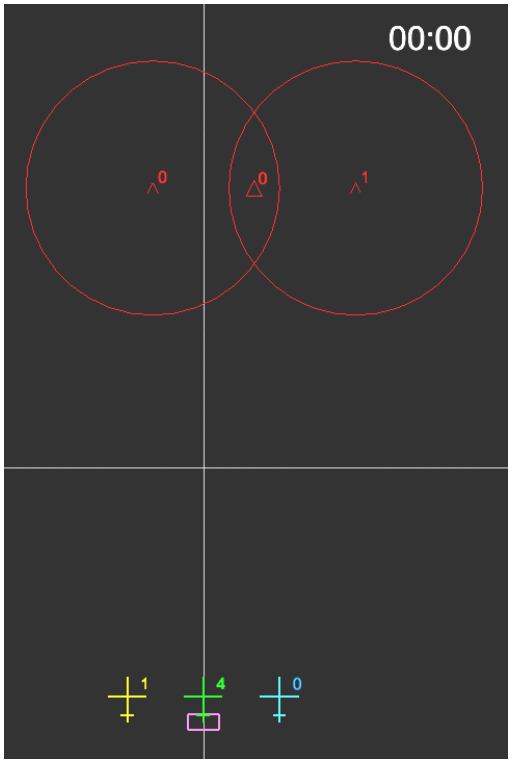


Abbildung 6-41: Ausgangssituation

Im Rahmen der ersten betrachteten Mission verpflichtet sich ein SEAD-UCAV, welches gleichzeitig Sprecher des gesamten Teams ist, zur Bekämpfung der beiden SAM-Stellungen, während das andere SEAD-UCAV das Attack-UCAV schützt. Nach einem Ausfall des Teamsprechers wird diese Rolle neu besetzt und das verbleibende SEAD-UCAV übernimmt die Bekämpfung der SAM-Stellungen.

Konkret wird zu Beginn der Mission UCAV 0 zum Sprecher des gesamten Teams bestehend aus den UCAVs mit den IDs 0, 1 und 4 und UCAV 1 zum Sprecher des SEAD-Sub-Teams bestehend aus den UCAVs mit den IDs 0 und 1 bestimmt. Ferner ist das Attack-UCAV (ID 4) zur Bekämpfung des Ziels, das SEAD-UCAV mit der ID 1 zur Eskortierung des Attack-UCAV und das SEAD-UCAV mit der ID 0 zur Unterdrückung der SAM-Stellungen 1 und 0 verpflichtet (vgl. Abbildung 6-42, 02:26).

Im Rahmen der Bekämpfung von SAM-Stellung 1 wird nun UCAV 0 getroffen (vgl. Abbildung 6-42, 05:39), so dass es weder seine weiteren Aufgaben im Missionskontext noch seine Rolle als Teamsprecher des gesamten Teams weiter ausüben kann. Daher werden die Ziele *Habe Teamsprecher* und *Stelle Aufgabenabdeckung sicher* aktiviert,

welche die Bestimmung von UCAV 1 als neuen Teamsprecher (vgl. Dialoge 4-26, 1-36, 1-38, 1-45 in Tabelle 6-22) und die Übernahme der Unterdrückung von SAM-Stellung 0 durch ACU 1 (vgl. Dialog 1-42 in Tabelle 6-22) zur Folge haben. Diese beiden neuen Aufgaben drücken sich für ACU 1 darin aus, dass sie nun zum einen Dialoge für das Team führen, d.h. hier zum Beispiel die Rückmeldung über die erfolgreiche Bekämpfung des Angriffsziels an den Operateur senden muss (vgl. Dialog 100-0 in Tabelle 6-21) und zum anderen ihr Vorgehen im Missionskontext ändern, d.h. konkret SAM-Stellung 0 angreifen muss (vgl. Abbildung 6-42, 06:06).

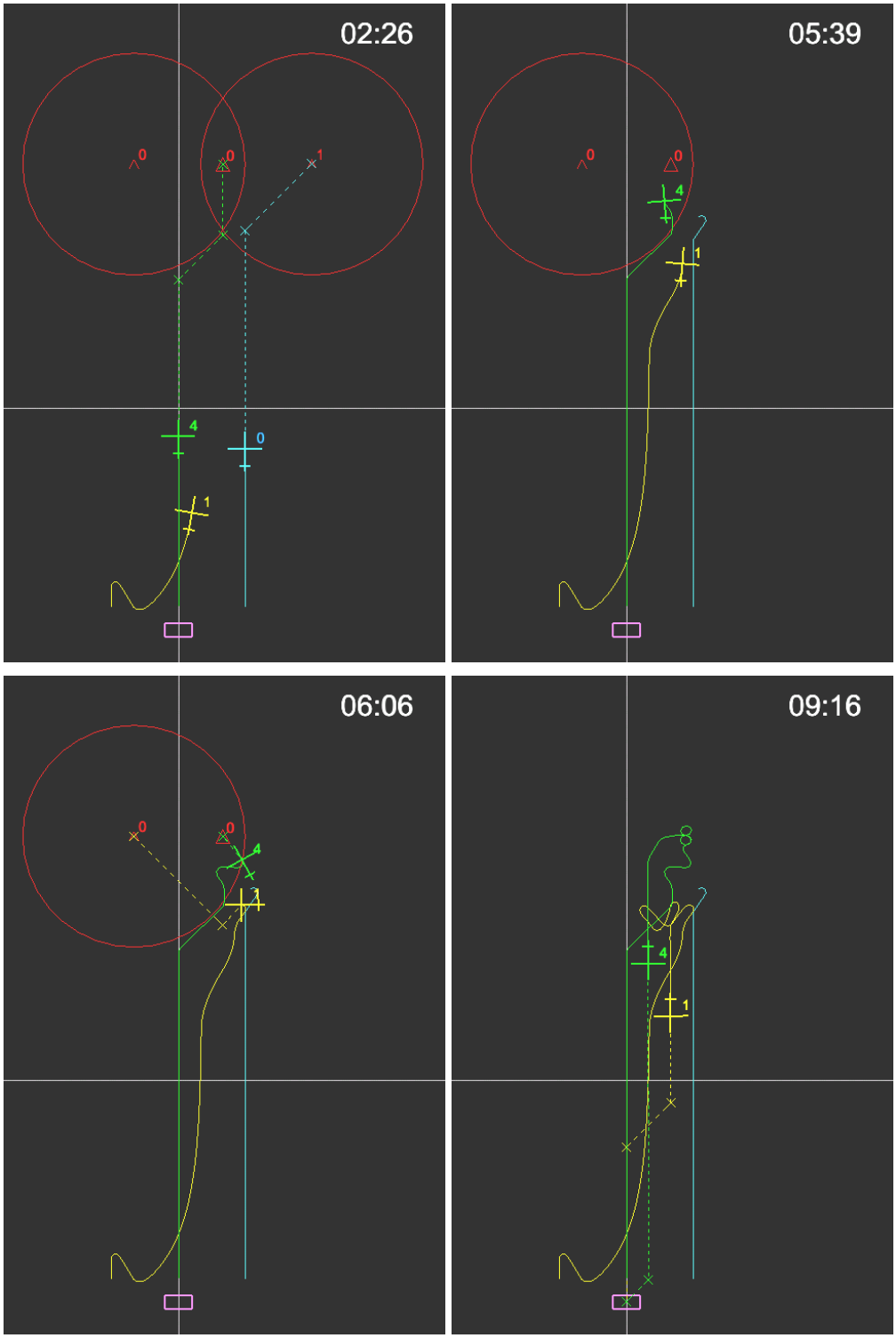


Abbildung 6-42: Ausfall Teamsprecher

K	*Protokoll*			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
100-0	*dialog-request*			
00:19	request	100	0, 1, 4	(mission-order(action destroy)(object target)(target 0))
00:20	agree	0, 1, 4	100	
07:35	inform	0, 1, 4	100	

Tabelle 6-21: Dialog zur Übermittlung des Auftrags und Rückmeldung über Erfolg der Durchführung (K=Kennung, Perf=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0,1,4, Operateur-ID: 100)

K	*Protokoll*			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
4-26	*dialog-inform*			
05:36	inform	4	4, 1, 0	(spokesman-election(team 4)(team 1)(team 0)) (spokesman-election-number(actor 4)(number 46))
1-36	*dialog-inform*			
05:39	inform	1	4, 1, 0	(spokesman-election(team 4)(team 1)(team 0)) (spokesman-election-number(actor 1)(number 55))
1-38	*dialog-propose*			
05:40	propose	1	4, 1, 0	(type spokesman)(actor 1) (spokesman-election(team 4)(team 1)(team 0)))
05:41	accept-proposal	4	1	
05:42	accept-proposal	1	4, 1, 0	
1-42	*dialog-inform*			
05:46	inform	1	4, 1, 0	(information(actor 1)(type commitment)) (commitment(responsible(actor 1))(task(name task-suppress)(object sam-site)(sam-site 0))) (commitment(responsible(actor 1))(task(name task-protect)(object ucav)(ucav 4))) (commitment(responsible(actor 1))(task(name task-fly-to)(object homebase)))
1-45	*dialog-inform*			
05:50	inform	1	4, 1, 0	(information(team 4)(team 1)(team 0)(type spokesman)) (spokesman(actor 1)(team 4)(team 1)(team 0))

Tabelle 6-22: Ausgewählte Dialoge nach Ausfall des Teamsprechers (K=Kennung, Perf=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0,1,4)

Die initiale Teamstruktur und Aufgabenverteilung in der zweiten betrachteten Mission, welcher das gleiche Szenario zugrunde liegt wie im ersten Lauf, entsprechen der oben beschriebenen: Es werden UCAV 4 zum Sprecher des gesamten Teams UCAV 1 zum Sprecher des SEAD-Sub-Teams bestimmt. Ferner ist das Attack-UCAV (ID 4) zur Bekämpfung des Ziels, das SEAD-UCAV mit der ID 1 zur Eskortierung des Attack-UCAV und das SEAD-UCAV mit der ID 0 zur Unterdrückung der SAM-Stellungen 1 und 0 verpflichtet (vgl. Abbildung 6-43, 01:35).

Im weiteren Verlauf der Mission wird nun ein weiteres Teammitglied (SEAD-UCAV mit ID 2) eingespielt, das insofern in die gemeinsame Durchführung der Mission eingebunden wird, als es die Bekämpfung einer der beiden SAM-Stellungen, zu denen ACU 0 verpflichtet ist, übernimmt.

Konkret findet nach dem Hinzukommen des neuen Teammitglieds zunächst diverse Kommunikation statt. So fragen die bereits vorhandenen Teammitglieder in Folge der Aktivierung des Ziels *Stelle Aufgabenabdeckung sicher* die Verpflichtung von ACU 2 zur Rückkehr zur Basis an (vgl. Dialog 4-35 in Tabelle 6-23), da diese Teamverpflichtung von allen Teammitgliedern bearbeitet werden muss. Ferner bittet ACU 2 auf Grund der Aktivierung des Ziels *Kenne Teammitglied* die anderen Teammitglieder, sie über

deren Fähigkeiten, Ressourcen und Verpflichtungen zu informieren (vgl. Dialoge 2-2, 2-4, 2-3, 2-8 in Tabelle 6-23). Da daneben auch das Ziel *Informiere Teammitglieder laufend* aktiv ist, stellt ACU 2 ihrerseits dem Team Informationen über die vorhandenen Ressourcen, Fähigkeiten und Verpflichtungen zur Verfügung (vgl. Dialoge 2-7, 2-11, 2-23, 2-29 in Tabelle 6-23).

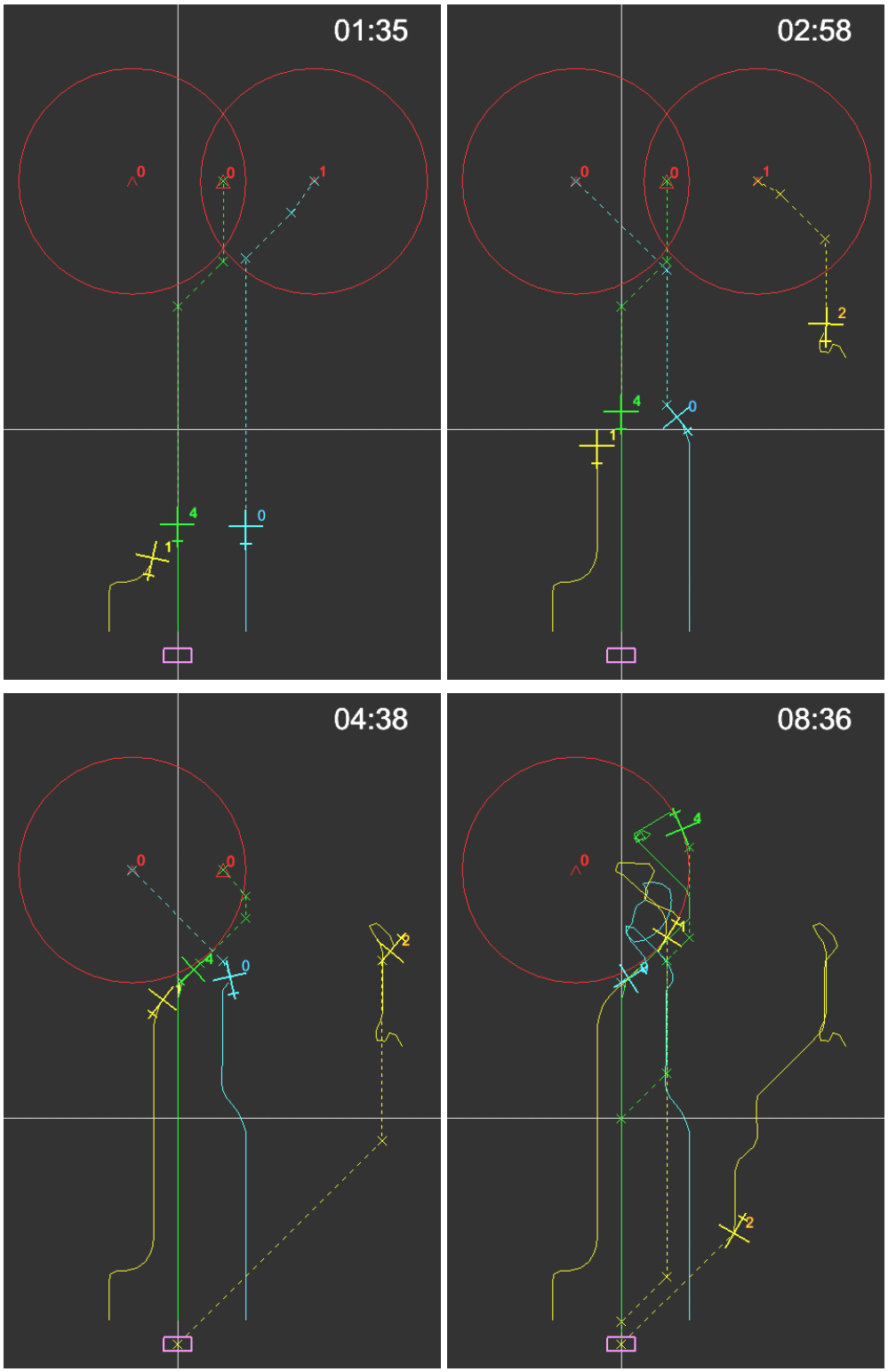


Abbildung 6-43: Hinzukommen eines neuen Teammitglieds

K	***Protokoll***			
Zeit	***Perf.***	***S***	***E***	***Inhalt***
4-35	*dialog-request*			
01:55	request	4	2	(task(name task-fly-to)(object homebase))
02:03	agree	2	4	
2-2	*dialog-query*			
02:01	query	2	0	(information(actor 0)(type capability))
02:02	agree	0	2	
02:11	inform	0	2	(capability(actor 0)(type sead))
2-4	*dialog-subscribe*			
02:01	subscribe	2	0	(information(actor 0)(type commitment))
02:05	agree	0	2	
02:05	inform	0	2	(commitment(responsible(actor 0))(task(name task-suppress)(sam-site 1))) (commitment(responsible(actor 0))(task(name task-suppress)(sam-site 0))) (commitment(responsible(actor 0))(task(name task-protect)(ucav 4))) (commitment(responsible(actor 0))(task(name task-fly-to)(object homebase)))
02:26	inform	0	2	(commitment(responsible(actor 0))(task(name task-suppress)(sam-site 0))) (commitment(responsible(actor 0))(task(name task-protect)(ucav 4))) (commitment(responsible(actor 0))(task(name task-fly-to)(object homebase)))
05:49	inform	0	2	(commitment(responsible(actor 0))(task(name task-protect)(object ucav))) (commitment(responsible(actor 0))(task(name task-fly-to)(object homebase)))
06:47	inform	0	2	(commitment(responsible(actor 0))(task(name task-fly-to)(object homebase)))
13:21	inform	0	2	
2-3	*dialog-query*			
02:02	query	2	4	(information(actor 4)(type capability))
02:03	agree	4	2	
02:07	inform	4	2	(capability(actor 4)(type attack))
2-7	*dialog-inform*			
02:02	inform	2	2, 0, 4, 1	(information(actor 2)(type resource)) (resource(actor 2)(type harm)(total-amount 2.0)(free 2.0))
2-11	*dialog-inform*			
02:06	inform	2	2, 0, 4, 1	(information(actor 2)(type capability)) (capability(actor 2)(type sead))
2-8	*dialog-subscribe*			
02:08	subscribe	2	1	(information(actor 1)(type resource))
02:11	agree	1	2	
02:12	inform	1	2	(resource(actor 1)(type harm)(total-amount 2.0)(free 2.0))
2-20	*dialog-propose*			
02:21	propose	2	0	(task(name task-suppress)(object sam-site)(sam-site 1))(type takeover)
02:22	accept-proposal	0	2	
02:23	inform	2	0	
2-23	*dialog-inform*			
02:28	inform	2	2, 0, 4, 1	(information(actor 2)(type commitment)) (commitment(responsible(actor 2))(task(name task-suppress)(sam-site 1))) (commitment(responsible(actor 2))(task(name task-fly-to)(object homebase)))
2-29	*dialog-inform*			
04:20	inform	2	2, 0, 4, 1	(content(information(actor 2)(type commitment))) (commitment (responsible(actor 2))(task(name task-fly-to)(object homebase)))

Tabelle 6-23: Ausgewählte Dialoge nach Hinzukommen eines neuen Teammitglieds (K=Kennung, Perf.=Performativ, S=Sender-ID, E=Empfänger-ID, UCAV-IDs: 0,1,2,4)

Sobald ACU 2 um die Verpflichtungen von ACU 0 weiß (vgl. Dialog 2-4) wird das Ziel *Verteile Arbeitsbelastung* relevant, so dass ACU 2 ACU 0 vorschlägt, die Unterdrückung von SAM-Stellung 1 zu übernehmen (Dialog 2-20 in Tabelle 6-23). Da dieser Vorschlag angenommen wird, ändern sich die Verpflichtungen von ACU 0 und 2 entsprechend (vgl. Dialoge 2-4 und 2-23 in Tabelle 6-23), so dass UCAV 0 nun SAM-Stellung 0 angreift, während UCAV 2 in Richtung SAM-Stellung 1 fliegt (vgl. Abbildung 6-43, 02:58).

Während SEAD-Teammitglieder ohne Verpflichtung zur Unterdrückung bzw. Bekämpfung einer SAM-Stellung üblicherweise das Attack-UCAV eskortieren, beginnt UCAV 2 nach erfolgreicher Bekämpfung von SAM-Stellung 1 unmittelbar den Rückflug zum Stützpunkt (vgl. Abbildung 6-43, 04:38), da ihm Wissen um die Verpflichtungen des gesamten Teams fehlt.

Auch wenn die Integration in das existierende Team damit nicht in allen möglichen Facetten durchgeführt wird, so wird doch das zur Verfügung stehende Wissen bestmöglich eingesetzt, um auf die unvorhergesehenen Änderungen der Teamstruktur im Sinne der implementierten Ziele von Teamarbeit zu reagieren, d.h. in beiden Fällen eine Anpassung der Aufgabenverteilung an die gegebene Situation vorzunehmen.

6.3.2.3 Nutzung einer konventionellen Automationsfunktion

Als abschließendes Beispiel dafür, dass auf wissensbasierter Verhaltensebene stets versucht wird, im gegebenen Situationskontext die aktuellen Handlungsziele zu erreichen, selbst wenn diese Situation so nicht vorgesehen ist, wird in diesem Abschnitt die Ansteuerung einer konventionellen Automationsfunktion – hier einer Routenplanungsfunktionalität – erläutert, deren Verhalten nicht konform mit den Zielen ist, welche die *Supporting ACU* gegenwärtig verfolgt. So will die ACU im konkreten Beispiel die potentielle Bedrohung durch eine temporär abgeschaltete SAM-Stellung vermeiden, während der Routenplaner diese nicht berücksichtigt.

Abbildung 6-44 stellt einen Ausschnitt aus dem in Abschnitt 6.2.1.1 erläuterten Teilproblem 1 (Lauf a) ab dem Zeitpunkt dar, zu dem das abgebildete UCAV den Rückflug zum Stützpunkt beginnen will. Zu diesem Zeitpunkt aktive Zielsetzungen sind nicht nur die Durchführung der Verpflichtung, zum Stützpunkt zurückzukehren sondern auch die Vermeidung der (potentiellen) Bedrohung durch die SAM-Stellung. Da erstgenanntes Ziel höher priorisiert ist, wird eine entsprechende Route durch die temporär abgeschaltete SAM-Stellung geplant und aktiviert (06:56). Diese SAM-Stellung wird bei der Routenplanung nicht berücksichtigt, so dass die geplante Route nahe an der SAM-Stellung vorbeiführt und das UCAV bei einer Reaktivierung der SAM-Stellung einer hohen Gefährdung ausgesetzt wäre. Da kein Wissen über eine solche Situation hinterlegt ist, wird zunächst nicht weiter reagiert. Während jedoch die Route abgeflogen wird, verringert sich der Abstand zur SAM-Stellung, so dass das Bedrohungspotential durch die SAM-Stellung zunimmt. Daher wird der Vermeidung der Bedrohung ein höheres Gewicht eingeräumt als dem Rückflug zum Stützpunkt und das UCAV dreht ab (07:12). Mit zunehmender Entfernung dreht sich diese Gewichtung wieder um und es wird abermals eine Route zum Stützpunkt geplant und aktiviert (07:23). Da sowohl dieser als auch der nächste Flugplan (07:47) wiederum zu nahe an der SAM-Stellung vorbeiführen, wiederholt sich die Umgewichtung der Ziele mit den entsprechend folgenden Handlungen, d.h. dem Ausweichmanöver (07:39, 08:14), bevor sich eine Route ergibt (08:22), auf welcher das UCAV mit der von ihm gewünschten Sicherheit den Bedrohungsbereich der SAM-Stellung verlassen kann (09:37).

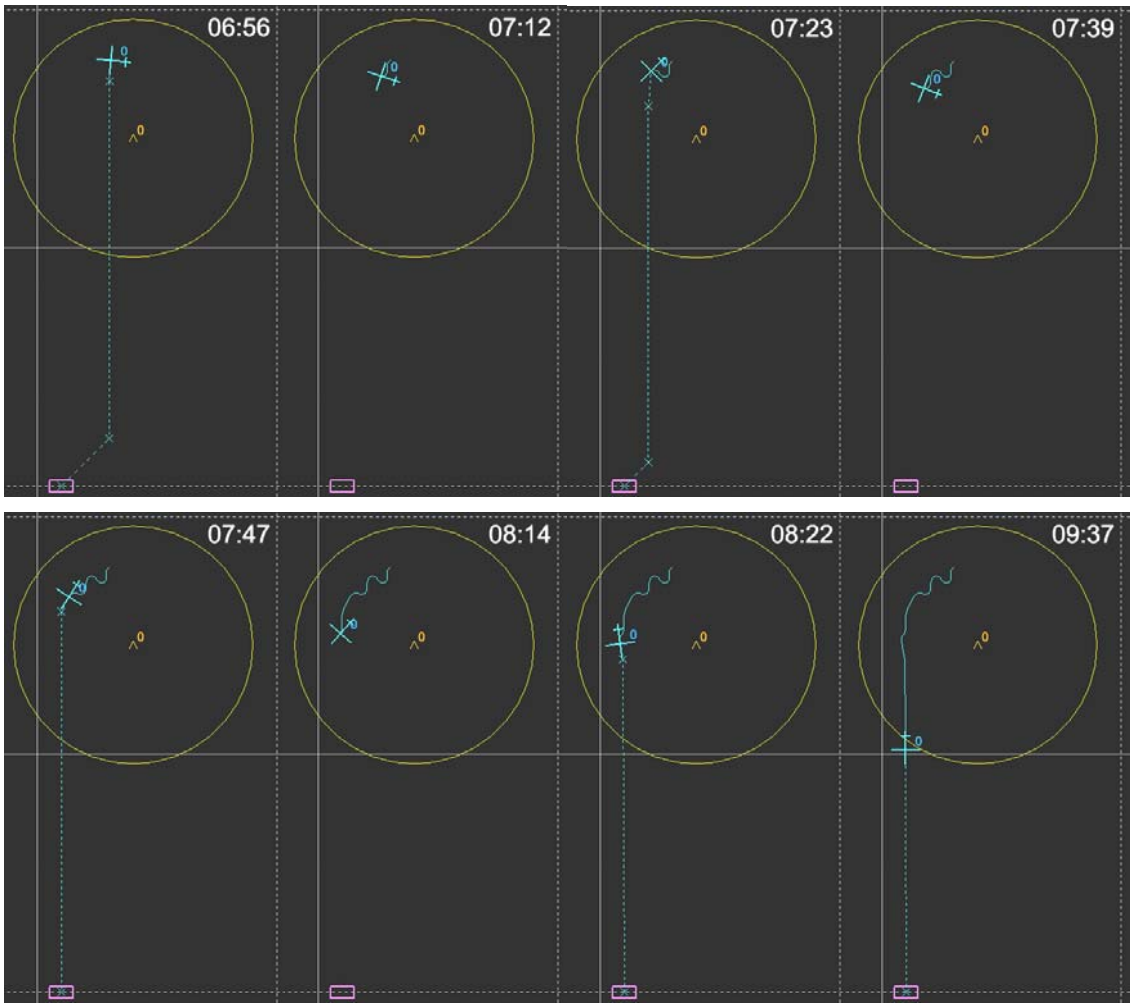


Abbildung 6-44: Einsatz verfügbarer Handlungsalternativen vor dem Hintergrund relevanter Ziele

Das hier erläuterte Verhalten des UCAVs entwickelt sich aus einer Ausgangssituation heraus, in der keine Handlungsalternative zur Verfügung steht, welche beide relevanten Zielsetzungen erreichen kann, d.h. zum Stützpunkt zu fliegen ohne die Bedrohung durch die SAM-Stellung zu groß werden zu lassen. Hinzu kommt eine Wissenslücke bezüglich der Beurteilung des Flugplans und möglicher Optionen, welches alternative Vorgehen gewählt werden könnte. Trotzdem ergibt sich hier durch das Zusammenwirken der explizit formulierten Ziele und den damit verbundenen bestmöglichen Einsatz der vorhandenen Handlungsoptionen ein Vorgehen, das im Wesentlichen den übergeordneten Anforderungen entspricht, d.h. ein Flug des UCAVs am Rande des Bedrohungsbereichs der SAM-Stellung hin zum Stützpunkt.

In dieser spezifischen Situation hätte eine ausgefeiltere Routenplanungsfunktionalität, welche sowohl das Bedrohungspotenzial der SAM-Stellungen als auch die Flugstrecke in die Planung mit einbezieht, vermutlich eine geeignetere Trajektorie erzeugt. Ein Beispiel für einen Ansatz, welcher auf einem numerischen Optimierungsverfahren beruht, das in einer Kostenfunktion derartige Zielsetzungen parametrisch berücksichtigt, findet sich in [Leuthäusser & Raupp, 1991]. Entsprechende Anwendungen sind in [Schulte et al., 1996] und [Schubert & Schulte, 2001] dargestellt. Hier werden jedoch relevante Ziele für die Routenplanung wie die genannte Bedrohungs- und Flugstreckenminimierung nur implizit hinterlegt, so dass keine Beurteilung dieser Kriterien vor dem Hintergrund der Gesamtsituation stattfinden kann und sich somit die bekannten Probleme von regelbasiertem Verhalten ergeben. Auch hier wird daher empfohlen, zwar solche Funk-

tionalitäten wann immer möglich im Hinblick auf Effizienz und Optimalität einzusetzen, aber zusätzlich im Hinblick auf Integrität und Flexibilität auf wissensbasierter Verhaltensebene die übergeordneten Zielsetzungen zu implementieren, um die Vorteile beider Herangehensweisen nutzen zu können. Im vorliegenden Beispiel würde dies zum einen die Möglichkeit eröffnen, entsprechende Parameter zum Beispiel einer Kostenfunktion an die aktuelle Situation anpassen zu können und zum anderen über eine Rückfallposition für Situationen zu verfügen, in denen keine optimierten, auf die jeweilige Situation zugeschnittenen Handlungsalternativen zur Verfügung stehen.

6.4 Mensch-ACU-Kooperation

Den Abschluss der Evaluierung des im Rahmen dieser Arbeit entwickelten Funktionsprototyps bildet eine Untersuchung dessen Eignung für Mensch-Maschine-Kooperation auf gleichberechtigtem, partnerschaftlichem Niveau. Derartige maschinelle Fähigkeiten werden zum Beispiel im Rahmen der Zusammenarbeit zwischen Operateur und *Operating ACUs* in der in Abbildung 2-10 dargestellten Arbeitssystemkonfiguration benötigt.

Da der entwickelte Funktionsprototyp *Supporting ACUs* abbildet, welche mit Teammitgliedern zusammenarbeiten können, die sich im gleichen Aufgabenkontext befinden und das gleiche Aufgabenspektrum abdecken, wird im Rahmen der hier beschriebenen Untersuchung die in Abbildung 6-45 dargestellte Arbeitssystemkonfiguration zu Grunde gelegt. Diese stellt einen Zwischenschritt auf dem Weg von der in Abbildung 2-9 dargestellten Konfiguration, auf welcher die Entwicklung der kooperativen *Supporting ACUs* basierte, hin zu der in Abbildung 2-10 abgebildeten dar. Die Betrachtung wird nämlich von einem rein maschinellen zu einem gemischt menschlich-maschinellen Team ausgeweitet, in dem aber alle Teammitglieder das gleiche Aufgabenspektrum abdecken. Konkret wird die in Abbildung 2-9 dargestellte Konfiguration um ein zweites Arbeitssystem erweitert, in dem ein menschlicher Pilot ein Flugzeug führt, wobei das Arbeitsziel dieses Arbeitssystems identisch mit dem Auftrag des Operateurs an das ACU-Team ist. Durch den Austausch entsprechender Nachrichten ergibt sich eine Zusammenarbeit zwischen den *Supporting ACUs* und dem Piloten im Hinblick auf das gemeinsame Ziel, d.h. die Erfüllung des Auftrags.

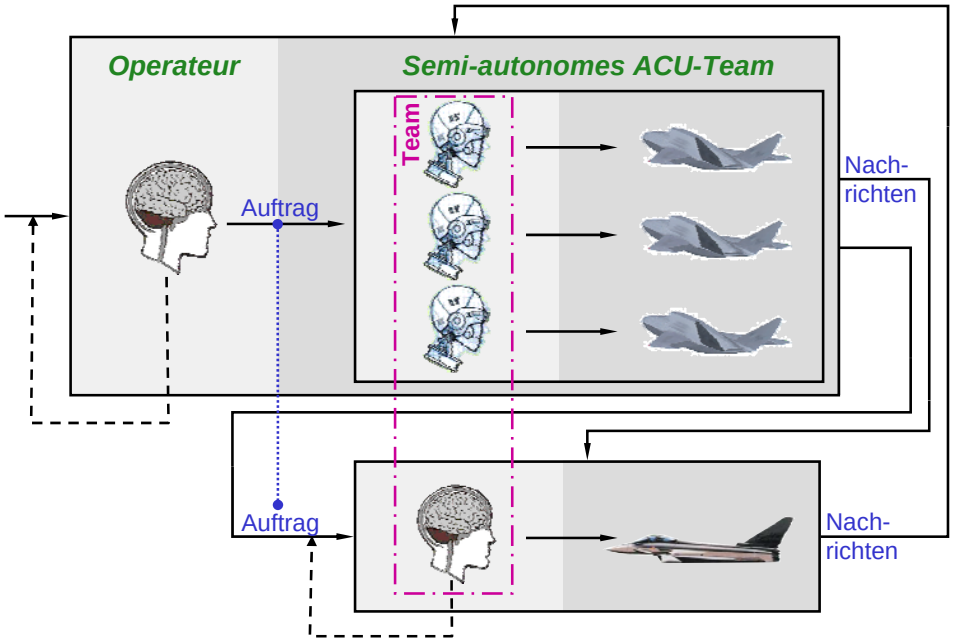


Abbildung 6-45: Arbeitssystemkonfiguration für die Untersuchung von Mensch-Maschine-Kooperation im gleichen Aufgabenkontext

Im Folgenden werden nun zunächst die Durchführung und Ergebnisse einer Experimentalkampagne beschrieben, der diese Arbeitssystemkonfiguration zugrunde liegt (Abschnitt 6.4.1 bzw. 6.4.2). Anschließend werden auf Basis beobachteter Probleme, die bei der Zusammenarbeit der im Rahmen dieser Arbeit entwickelten *Supporting ACUs* mit menschlichen Teammitgliedern auftraten, Anforderungen abgeleitet, die in zukünftige Entwicklungen kooperativer künstlicher kognitiver Einheiten mit einfließen können (Abschnitt 6.4.3).

6.4.1 Experimental Design

Die Untersuchung der Zusammenarbeit der *Supporting ACUs* mit menschlichen Teammitgliedern erfolgte anhand von vier Szenarien, welche sich aus den in Abschnitt 6.1.1 vorgestellten Szenarienelementen zusammensetzen. Hierbei wurde eines der UCAVs von einem menschlichen Piloten statt einer künstlichen kognitiven Einheit gesteuert. Alle Szenarien enthalten eine FLOT, einen Korridor, ein Angriffsziel und mehrere bekannte und unbekannte SAM-Stellungen sowie ein Attack- und ein bzw. drei SEAD-UCAVs (vgl. Abbildung 6-46).

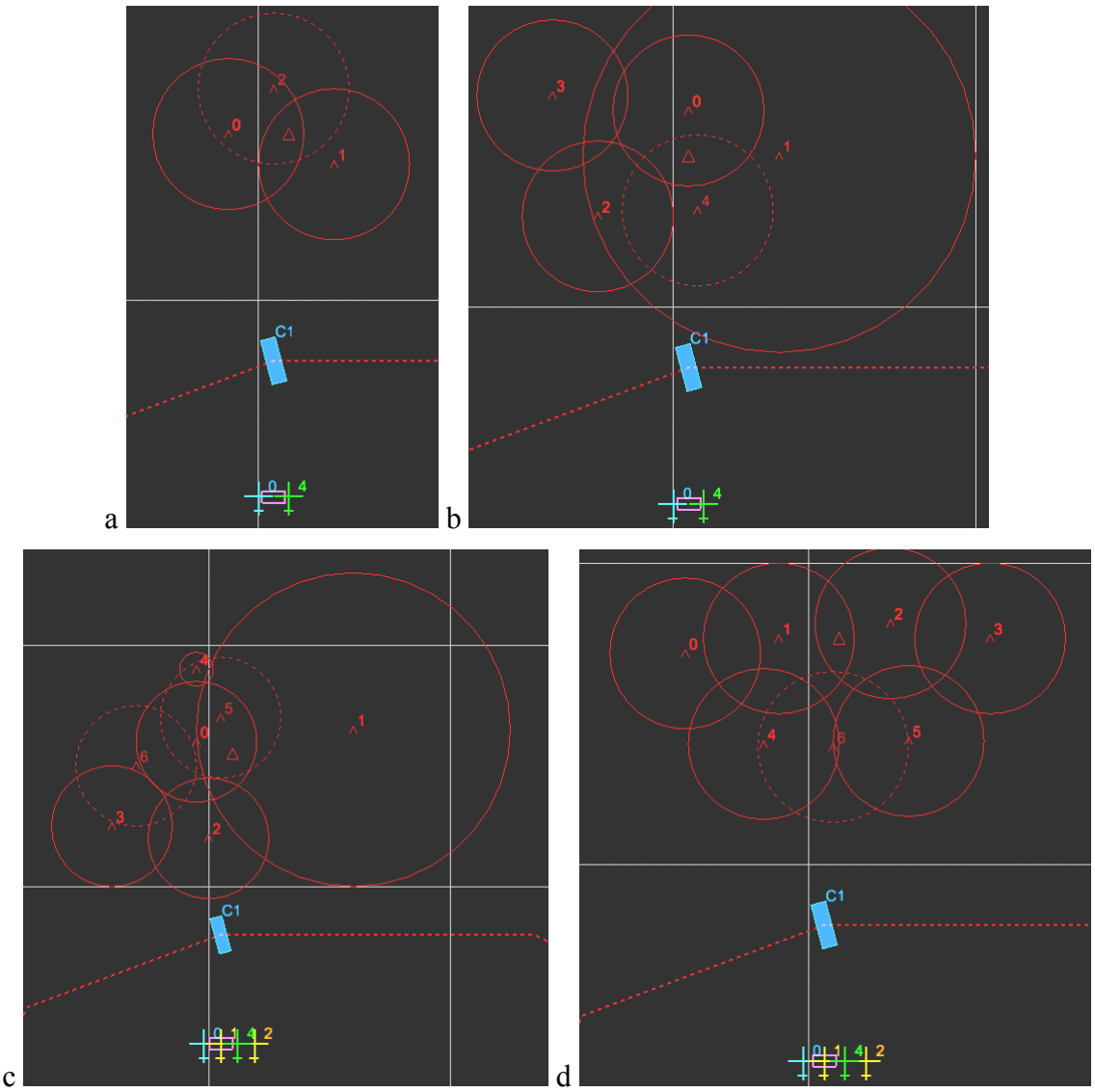


Abbildung 6-46: Szenarien zur Untersuchung von Mensch-Maschine Kooperation

Damit konnte sowohl eine rein heterogene Teamstruktur (1 Attack- und 1 SEAD-UCAV) als auch eine heterogene Teamstruktur mit einem homogenen SEAD-Subteam (1 Attack- und 3 SEAD-UCAVs) betrachtet werden. In allen Fällen bestand das Missionsziel darin, infolge eines Auftrags vom Operateur das Angriffsziel zu bekämpfen. Tabelle 6-24 gibt einen Überblick über die betrachteten Konfigurationen des Mensch-Maschine-Teams im Hinblick auf die Anzahl der Flugzeuge und die von Mensch bzw. ACU eingenommene funktionale Rolle im Team. Bei der Versuchsdurchführung wurden dabei die Konfigurationen 1 bis 4 in der dargestellten Reihenfolge durchlaufen, d.h. die Versuchspersonen führten zunächst das Attack-Flugzeug mit einer Unterstützung durch ein bzw. drei SEAD-UCAVs und wechselten anschließend in die SEAD-Rolle, wobei zunächst nur ein weiteres Attack-UCAV und schließlich zwei zusätzliche SEAD-UCAVs beteiligt waren. Dabei wurden in Konfiguration 1 und 3 jeweils die beiden in Abbildung 6-46 oben dargestellten Szenarien (a, b) durchlaufen, während in Konfiguration 2 und 4 jeweils die in Abbildung 6-46 unten dargestellten Szenarien (c, d) absolviert wurden.

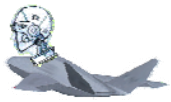
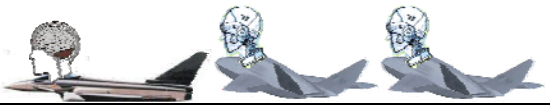
	Teamzusammensetzung	
Konfiguration	*Attack-AV*	*SEAD-AV(s)*
1		
2		
3		
4		

Tabelle 6-24: Teamzusammensetzungen (AV: Aerial Vehicle)

Hiermit wird es möglich, die Rolle des Menschen in verschiedenen Teamstrukturen bzw. an verschiedenen Positionen in einer Teamstruktur zu betrachten. So stellen Konfiguration 1 und 3 rein heterogene Teams dar, in denen der Mensch jeweils unterschiedliche funktionale Rollen (Attack bzw. SEAD) einnimmt. In Konfiguration 2 und 4 hingegen ist er Teil eines heterogenen Teams mit einem homogenen SEAD-Sub-Team, wobei er zunächst mit einem homogenen Sub-Team zusammenarbeiten muss (Konfiguration 2) bevor er Teil des homogenen Teams ist (Konfiguration 4).

In allen vier Konfigurationen übernimmt der Mensch die Rolle des Teamsprechers, d.h. die Kommunikation des gesamten Teams mit dem Operateur und in Konfiguration 4 zusätzlich die Kommunikation des SEAD-Teams mit dem Attack-UCAV.

Die Interaktion der Versuchspersonen mit dem Operateur, den maschinellen Teammitgliedern und seinem Flugzeug erfolgt mittels einer graphischen Benutzeroberfläche auf zwei Bildschirmen (siehe Abbildung 6-47 und Abbildung 6-48), die mittels der gleichen Schnittstellen wie die ACUs in die in Abschnitt 6.1.3 beschriebene Simulationsumgebung integriert sind. Abbildung 6-48 zeigt das *Navigation Display*, auf dem die bekannte taktische Lage sowie der Status des eigenen Flugzeugs dargestellt ist, dessen Kurs und Geschwindigkeit über Tastatureingaben gesteuert werden kann.

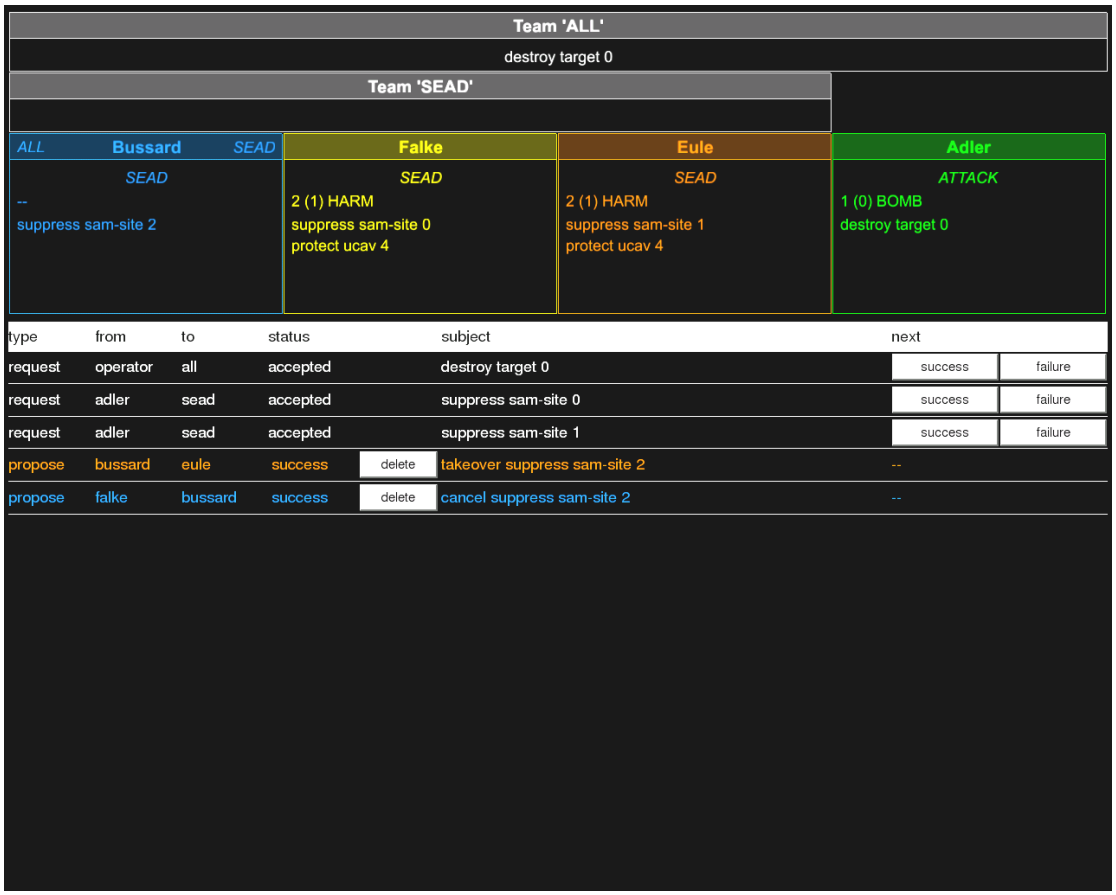


Abbildung 6-47: Benutzeroberfläche für Piloten – Team Display

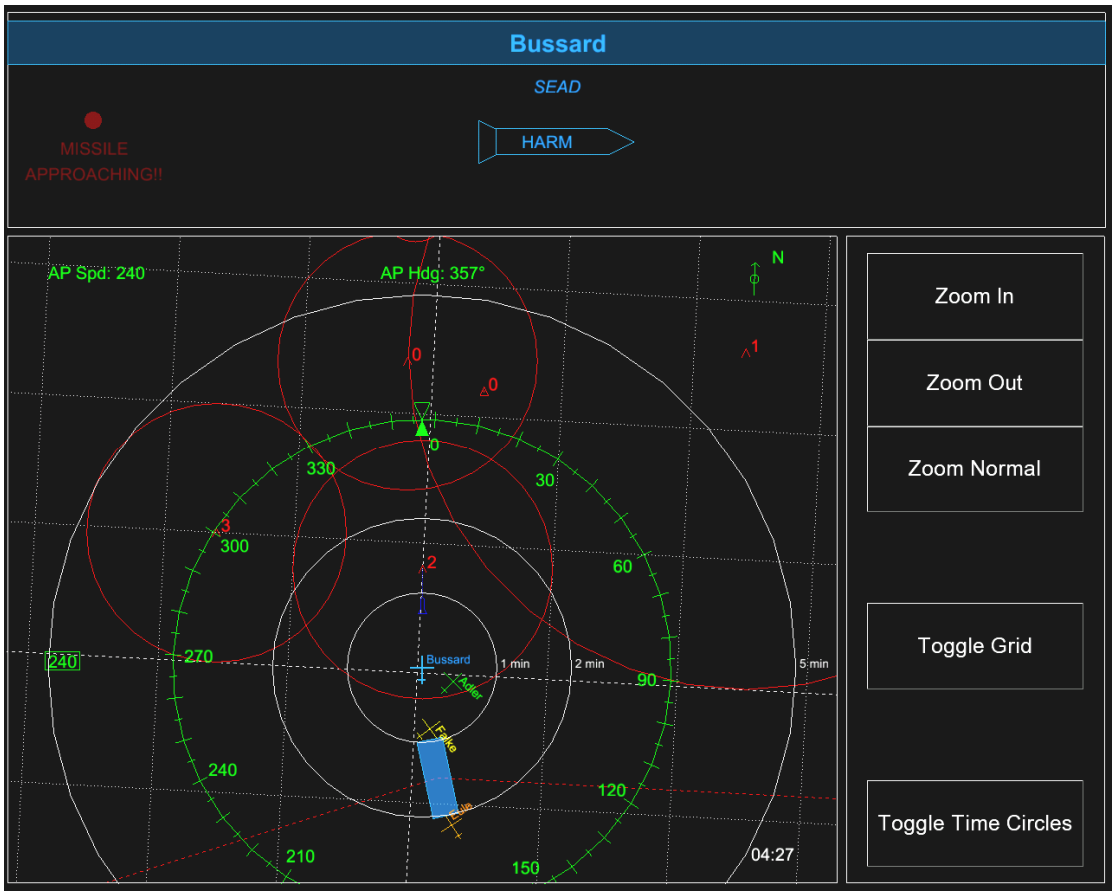


Abbildung 6-48: Benutzeroberfläche für Piloten – Navigation Display

Abbildung 6-47 zeigt das *Team Display* mit Hilfe dessen über Touchscreeneingaben mit dem Operateur und den Teammitgliedern interagiert werden kann. In der oberen Hälfte sind dabei die bekannten Verpflichtungen des gesamten Teams, des SEAD-Subteams sowie der einzelnen Teammitglieder ebenso wie deren Ressourcen dargestellt. In der unteren Hälfte findet sich eine Liste der geführten Dialoge mit Typ des Dialogs, Initiator, Teilnehmer, Zustand, Inhalt und möglichen Nachrichten, die als nächstes gesendet werden können. In diesem Display können zwei Arten von Eingaben gemacht werden: Zum einen können bereits begonnene Dialoge entsprechend der aktuellen Optionen fortgeführt werden. Zum anderen können Dialoge initiiert werden, d.h. proaktiv Informationen an Teammitglieder gesendet, Anfragen an einzelne Teammitglieder oder Teams gestellt oder Vorschläge unterbreitet werden.

Nach Abschluss einer jeden Mission wurde die Beanspruchung der Versuchsperson mit Hilfe des NASA-TLX Verfahrens (NASA Task Load Index, [Hart & Staveland, 1988]) bestimmt. Hierbei handelt es sich um ein subjektives, mehrdimensionales Bewertungsverfahren, mit dem die Beanspruchung eines Menschen bewertet werden kann. Resultat ist ein globales Beanspruchungsmaß, welches sich aus Einzelwertungen in sechs Unterskalen zusammensetzt, die mit unterschiedlichem Gewicht zur Gesamtbeanspruchung beitragen, nämlich

- Geistige Anforderungen,
- Körperliche Anforderungen,
- Zeitliche Anforderungen,
- Leistung,
- Anstrengung und
- Frustration.

Die im folgenden Abschnitt vorgestellten Ergebnisse basieren auf Versuchen mit fünf Versuchspersonen im Alter zwischen 31 und 50 Jahren, welche über fliegerische Ausbildung als Pilot bzw. Waffensystemoffizier verfügten und eine Flugerfahrung zwischen 550 und 3650 Stunden aufweisen konnten (siehe Tabelle 6-25). Je nach zeitlicher Verfügbarkeit führten diese Versuchspersonen dabei mindestens vier der acht beschriebenen Versuchsläufe durch.

		Versuchsperson				
		I	*II*	*III*	*IV*	*V*
Alter		31	35	50	46	42
Funktion		Luftfahrzeugführer	Luftfahrzeugführer	Waffensystemoffizier	Waffensystemoffizier	Luftfahrzeugführer
Zusatzqualifikation			Testflugberechtigung	Testflugberechtigung	Testflugberechtigung	Testflugberechtigung
Flugerfahrung in [h]	*T-37, T-38*	550	-	-	-	-
	Tornado		1500	2200	1400	-
	Phantom	-		700	500	-
	Mirage 2000	-		-	-	-
	F-18	-		-	-	-
	Breguet	-	-	-	-	2000
	Transall	-	-	-	-	1000
	sonstige	-	550	-	-	650

Tabelle 6-25: Überblick über Versuchspersonen

6.4.2 Ergebnisse

6.4.2.1 Leistung

Tabelle 6-26 gibt einen Überblick über die in den einzelnen Läufen erbrachte Leistung des Mensch-Maschine Teams in den betrachteten Konfigurationen und Szenarien (vgl. Tabelle 6-24 und Abbildung 6-46). Hierbei wird der Missionserfolg, d.h. ob das Angriffsziel erfolgreich bekämpft wurde oder nicht, die Anzahl der überlebenden UCAVs in Abhängigkeit von der Art des UCAVs und die Anzahl der dem Team zu Beginn und Ende der Mission zur Verfügung stehenden HARMs berücksichtigt. Sofern ein Durchlauf vorzeitig beendet wurde, sind die Werte in Klammern angegeben und geben den Zustand zu diesem Zeitpunkt wieder. Sie werden daher bei der Gesamtbetrachtung nicht berücksichtigt. Konkret wurde in 94 % der durchgeführten Läufe das Missionsziel erreicht, d.h. das Angriffsziel erfolgreich bekämpft und es überlebten im Mittel 90 % der UCAVs unabhängig davon, ob es sich um SEAD- oder Attack-UCAVs handelte. Weiterhin waren bei Missionsende im Durchschnitt noch 23 % der zu Beginn der Mission verfügbaren HARMs vorhanden.

Konfiguration	***Szenario***	***Versuchsperson***	***Missionserfolg***	***Überlebende UCAVs***			***HARMs***	
				SEAD	***Attack***	***gesamt***	***verfügbar***	***verbleibend***
1	a	I	ja	1	1	2	3	1
		II	ja	1	1	2		0
		III	ja	1	1	2		0
		V	ja	1	1	2		0
	b	I	ja	1	1	2	4	1
		II	ja	1	1	2		1
		III	ja	1	1	2		0
		IV	ja	1	1	2		1
		V	ja	1	1	2		0
2	c	I	ja	3	1	4	9	4
		II	ja	3	1	4		3
		III	ja	3	1	4		4
		IV	ja	3	1	4		3
		V	ja	3	1	4		5
	d	I	ja	3	1	4	9	2
		II	ja	3	1	4		3
		III	ja	2	1	3		2
		IV	ja	2	1	3		0
		V	ja	(2)	0	(2)		(2)
3	a	I	ja	1	1	2	3	0
		IV	ja	0	0	0		0
	b	I	ja	1	1	2	4	0
		III	ja	1	1	2		3
		IV	ja	1	1	2		0
		V	ja	1	1	2		2
4	c	I	ja	3	0	3	9	4
		III	ja	2	1	3		1
		IV	ja	(2)	(1)	(3)		(3)
		V	nein	(3)	(1)	(4)		(2)
	d	I	ja	2	1	3	9	0
		III	(nein)	(3)	(1)	(4)		(5)
		IV	ja	(2)	(1)	(3)		(1)
		V	nein	(3)	(1)	(4)		0
gesamt			*94 %*	*90 %*	*89 %*	*90 %*		*23 %*

Tabelle 6-26: Überblick über Leistung des Mensch-Maschine Teams (Werte in Klammern basieren auf Durchläufen, die vorzeitig beendet wurden, und werden bei der Gesamtbetrachtung nicht berücksichtigt)

Auf Grund dieser erzielten, sehr guten Leistung lässt sich pauschal die Aussage treffen, dass im Prinzip die Zusammenarbeit von Menschen mit den im Rahmen dieser Arbeit entwickelten *Supporting ACUs* möglich ist. In den nächsten Abschnitten wird diese Aussage auf Basis der bestimmten Beanspruchung sowie Anmerkungen der Versuchspersonen kritisch beleuchtet.

6.4.2.2 Beanspruchung

Die von den Versuchspersonen subjektiv empfundene Beanspruchung, welche mit Hilfe des NASA-TLX-Verfahrens bestimmt wurde, ist in Abbildung 6-49 in Abhängigkeit von der jeweiligen Konfiguration (vgl. Tabelle 6-24) dargestellt. Hierbei ist die Beanspruchung in den Konfigurationen mit vier UCAVs (2 bzw. 4) im Mittel leicht erhöht gegenüber den Konfigurationen mit zwei UCAVs (1 bzw. 3). Außerdem ist die Beanspruchung in der SEAD-Rolle (Konfiguration 3 und 4) jeweils größer als die in der Attack-Rolle (Konfigurationen 1 und 2).

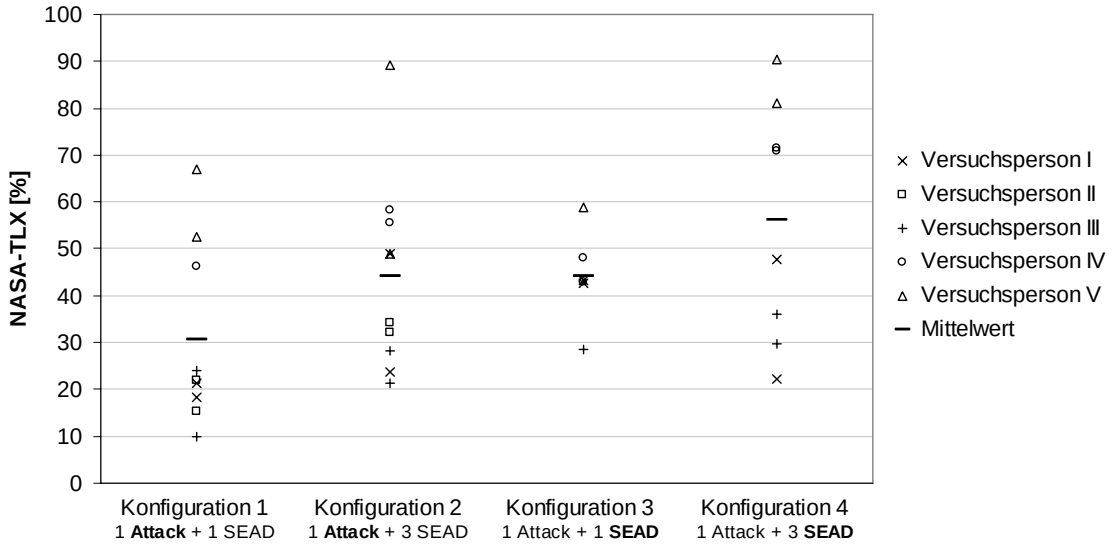


Abbildung 6-49: Beanspruchung der Versuchspersonen

Dass die empfundene Beanspruchung in Konfiguration 2 bzw. 4 gegenüber der in Konfiguration 1 bzw. 3 ansteigt entspricht insofern der Erwartung, als dort sowohl die Anzahl der Teammitglieder als auch die Komplexität der Szenarien größer ist. Dass der Anstieg der Beanspruchung nicht wie erwartet größer ausfällt, wird darauf zurückgeführt, dass die UAVs über kognitive und kooperative Fähigkeiten verfügen und daher gemeinsam mit dem menschlichen Teammitglied das gemeinsame Missionsziel verfolgen können anstatt auf detaillierte Kommandos vom Menschen angewiesen zu sein. Da dieser Effekt allerdings nicht bei allen Versuchspersonen bzw. in allen durchgeführten Missionen gleichermaßen ausgeprägt war, muss in weiteren Versuchsreihen geklärt werden, ob sich der beobachtete Trend verallgemeinern lässt.

Ähnliches gilt für die erhöhte Beanspruchung in der SEAD-Rolle gegenüber der Attack-Rolle in den Konfigurationen mit der gleichen Anzahl von UCAVs. Da die Änderung der mittleren Beanspruchung von Konfiguration 1 nach Konfiguration 3 in etwa der von Konfiguration 2 nach 4 sowie die Änderung von Konfiguration 3 nach 4 in etwa der von Konfiguration 1 nach 2 entspricht, wird hier die Änderung des primären Aufgabenspektrums und nicht die Einbindung in eine homogene statt heterogene Teamstruktur als Haupteinflussfaktor für die höhere Beanspruchung in der SEAD-Rolle angenommen. Diese umfasst nämlich statt der Bekämpfung eines dedizierten Targets die Bekämpfung

und Unterdrückung ausgewählter SAM-Stellungen unter Berücksichtigung der eigenen Ressourcensituation.

Aus welchen Beanspruchungsarten sich die von den Versuchspersonen empfundene Beanspruchung in den einzelnen Konfigurationen zusammensetzt ist in Abbildung 6-50 dargestellt. So betrug beispielsweise die Beanspruchung der Versuchspersonen in Konfiguration 1 im Mittel 31 %, wobei 36 % dieses Wertes durch geistige Anforderungen, 2 % durch körperliche Anforderungen, 24 % durch zeitliche Anforderungen, 15 % durch Leistung, 19 % durch Anstrengung und 4 % durch Frustration zustande kommen.

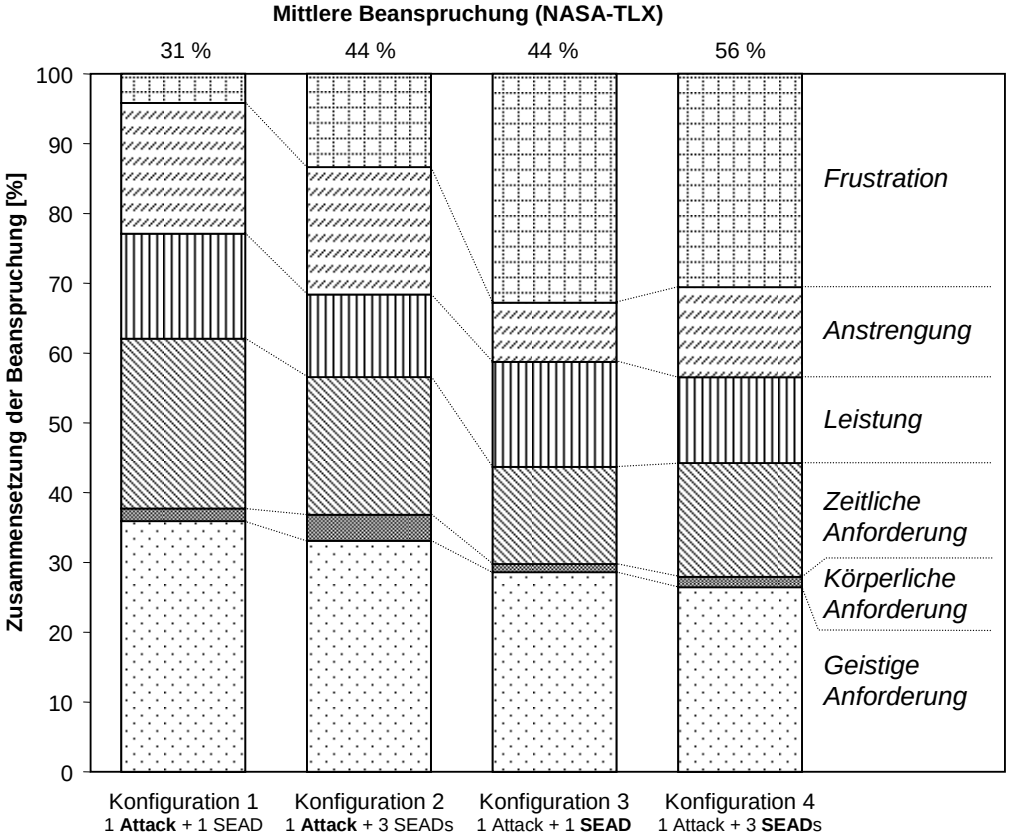


Abbildung 6-50: Zusammensetzung der Beanspruchung

Im Rahmen dieser Betrachtung bestätigt sich die Annahme, dass die Belastung der Versuchspersonen durch die Führung eines eigenen Flugzeugs wenn auch mit sehr eingeschränkten Interaktionsmöglichkeiten und insbesondere die Abstimmung mit anderen Teammitgliedern stets in einem hohen Anteil an geistiger Beanspruchung resultiert, während die körperliche Beanspruchung keine Rolle spielt. Daneben tragen zeitliche Anforderung, Leistung und Anstrengung in ähnlicher Höhe zur Gesamtbeanspruchung bei, wobei deren absolute Höhe und gegenseitige Gewichtung im Einzelfall von der Entwicklung der konkreten Mission abhängt (z.B. Anzahl und Entfernung neuer SAM-Stellungen, Erfolg bei der Bekämpfung von SAM-Stellungen (Unterdrückung bzw. Zerstörung), Bedrohung durch SAMs). Während sich dieses Bild soweit in allen Konfigurationen ähnelt und insbesondere auch mit der Erhöhung der Anzahl der Teammitglieder beim Übergang von Konfiguration 1 nach 2 bzw. von Konfiguration 3 nach 4 erhalten bleibt, ändert sich der Beitrag der Beanspruchungsart „Frustration“ in der SEAD-Rolle (Konfiguration 3 und 4) erheblich gegenüber der Attack-Rolle (Konfiguration 1 und 2). Dies wurde in den Interviews mit den Versuchspersonen darauf zurückgeführt, dass die ACUs in der Attack-Rolle deutlich stärker von dem erwarteten Verhalten abwichen als in der SEAD-Rolle und mehr taktisches Fehlverhalten zeigten. Da es zudem weder

einen gemeinsamen Missionsplan gab noch die Durchführung der Attack-Aufgabe beeinflusst werden konnte, fühlten sich die Versuchspersonen bei der Zusammenarbeit mit bzw. der Unterstützung des Attack-UCAVs irritiert, unsicher und entmutigt, was sich in der Beanspruchungsart „Frustration“ niederschlägt. Dieses Empfinden wurde in Konfiguration 4, in der die Versuchspersonen in das homogene SEAD-Team integriert waren, weiter verstärkt, da dort zusätzliche, schwer verständliche Dialogtypen gehandhabt werden mussten.

Insgesamt zeichnet sich für den Menschen im hier betrachteten Mensch-Maschine Team ein mittleres Beanspruchungsniveau mit einem Schwerpunkt auf geistigen Anforderungen ab, von dem erwartet wird, dass es sich in realen Arbeitsumgebungen wie zum Beispiel einem Cockpit oder einer Bodenkontrollstation weiter steigert. Dort liegt nämlich eine höhere Belastung durch Faktoren wie beispielweise die Führung eines realen Flugzeugs, eine größere Komplexität der Umgebung mit Gelände, Hindernissen, Lufträumen etc. oder notwendige Funkkommunikation vor. Daher werden im nächsten Abschnitt auf Basis von Interviews mit Versuchspersonen Anforderungen an die Fähigkeiten der hier betrachteten ACUs formuliert, die darauf abzielen, die Zusammenarbeit im Mensch-Maschine Team zu verbessern. Damit soll ermöglicht werden, dass auch in realen Umgebungen eine zufriedenstellende Leistung bei der Durchführung individueller Aufgaben und damit letztlich auch eines Auftrags an ein Team erreicht werden kann.

6.4.3 Anforderungen an ACUs im Mensch-Maschine Team

Ausgangspunkt für die sich an die Versuchsläufe anschließenden Interviews mit den Versuchspersonen war die Identifikation konkreter Probleme, die im Rahmen der Versuchsdurchführung im Kontext der Zusammenarbeit von Mensch und ACU aufgetreten waren und zwar mit dem Ziel, Anforderungen an ACUs abzuleiten, die diesen Problemen entgegenwirken. Somit sind die im Folgenden dargestellten Anforderungen in erster Linie für das Anwendungsfeld militärischer Air-to-Ground-Attack-Missionen zu verstehen, wobei angenommen wird, dass sie sich im Einzelfall auf ähnlich strukturierte Missionen bzw. Applikationen übertragen lassen. Generell wurde von den Versuchspersonen oft die Zusammenarbeit in menschlichen Teams, die heute derartige Missionen durchführen, als erstrebenswertes Vorbild angeführt, da sich dort etablierte Strukturen bewährt haben und die Menschen an diese Abläufe gewohnt sind. Durch eine Berücksichtigung dieses Sachverhalts kann zumindest im ersten Schritt eine größere Effizienz erwartet werden, wenn heutige Crews mit UAVs statt mit anderen bemannten Flugzeug-Crews zusammenarbeiten sollen.

Vor diesem Hintergrund werden in den nächsten Abschnitten nötige Fähigkeiten von ACUs in den Bereichen Teamstruktur, Abstraktionsgrad der Interaktion, Automationsgrad, Zusammenarbeit im Team, Aufgabendurchführung und Kommunikation im Team charakterisiert, wobei abschließend mögliche Einsatzfelder für ein Assistenzsystem in der Ausprägung *supplementation* (vgl. Abschnitt 2.1.2.2) aufgezeigt werden.

6.4.3.1 Teamstruktur

Im Gegensatz zu der im Rahmen dieser Arbeit etablierten Teamstruktur, bei der alle Teammitglieder gleichberechtigt auftreten, wird im Hinblick auf die Zusammenarbeit bemannter und unbemannter Kräfte im militärischen Umfeld die Einführung einer hierarchischen Teamstruktur angestrebt. Wesentlicher Unterschied hierbei ist die Rolle des Teamsprechers bzw. -leiters, welcher im gleichberechtigten Team aus *Supporting ACUs* lediglich eine leicht herausgehobene Stellung einnimmt, indem er zum Beispiel für das Team mit dem Operateur kommuniziert und Teamverpflichtungen eingeht, aufgibt und

dem Team mitteilt. Der Teamleiter eines hierarchischen Teams hingegen kommuniziert nicht nur mit externen Stellen, sondern trifft Entscheidungen, die das Team betreffen, und kann den Teammitgliedern Vorgaben machen, die diese höher priorisieren müssen als eigene Überlegungen. Während gefordert wird, dass Mensch-Maschine Teams hierarchisch organisiert sind und ein Mensch die Rolle des Teamleiters einnimmt, besteht im Hinblick auf rein maschinelle Sub-Teams wie es beispielsweise das SEAD-Team in Konfiguration 2 darstellt, keine Präferenz hinsichtlich der Teamstruktur.

Im Einzelnen soll der Teamleiter den Teammitgliedern bzw. Sub-Teams Vorgaben machen können, welche Aufgaben zu bearbeiten sind, wer welche Aufgabe bearbeiten soll und wie einzelne Aufgaben bearbeitet werden sollen. Derartige Vorgaben dürfen nur abgelehnt werden, wenn sie nicht eingehalten werden können (z.B. fehlende Bewaffnung, Treibstoffmangel). Im Rahmen der Vorgaben sollen die Teammitglieder oder auch Sub-Teams aber in der Lage sein, selbständig und sinnvoll zu agieren. Weiterhin darf unter gewissen Umständen von Vorgaben abgewichen werden, wenn zum Beispiel das Überleben des eigenen Flugzeugs gefährdet ist.

Allerdings sollen die Teammitglieder nicht von Vorgaben des Teamleiters abhängig sein, diese im Missionskontext bewerten und abhängig von der konkreten Situation den Teamleiter auch auf entscheidungsrelevante Information hinweisen, die dieser unter Umständen nicht in seine Entscheidung miteinbezogen hat. Insbesondere soll auch von Teammitgliedern die Initiative ergriffen werden, indem diese zum Beispiel zu Missionsbeginn Vorschläge hinsichtlich der der zu bearbeitenden Aufgaben und Aufgabenverteilung machen, welche jedoch von Vorgaben des Teamleiters übersteuert werden können.

6.4.3.2 Abstraktionsgrad der Interaktion

Die Interaktion zwischen Mensch und ACUs soll auf verschiedenen Abstraktionsniveaus möglich sein. Hierbei sind inbesondere die folgenden Möglichkeiten vorzusehen:

- *Vorgabe von Aufgaben an ein Team* aus ACUs (z.B. „Schaffen eines bedrohungsfreien Korridors bis zum Angriffsziel“, „Unterdrückung von SAM-Stellung 1 in 5 min und anschließend SAM-Stellung 2 in 10 min“)
- *Vorgabe von Aufgaben an eine einzelne ACU*, auch um die Aufgabenverteilung in einem Team zu beeinflussen (z.B. „Unterdrückung von SAM-Stellung 1 in 5 min“)
- *Vorgabe an eine einzelne ACU, wie eine Aufgabe durchgeführt werden soll* (z.B. „Flug über Wegpunkt X2“, „Fliege 2 min voraus“, „Greife jetzt SAM-Stellung an“, „Ignoriere Bedrohung durch SAM-Stellung 3“)

Welche konkreten Aufgaben in welcher Detaillierungstiefe und mit welchen Randbedingungen hierbei erforderlich sind, hängt von der spezifischen Mission ab und muss im Einzelfall definiert werden. Seitens der ACUs müssen diese dann im Missionskontext verstanden und entsprechend umgesetzt werden können.

Generell wird im Hinblick auf die Führung mehrerer UAVs von einer Bodenkontrollstation oder sogar einem Cockpit aus eine Interaktion auf einem möglichst hohen Abstraktionsniveau angestrebt, um die Belastung für den Menschen auf einem akzeptablen Niveau zu halten. Voraussetzung hierfür ist allerdings eine zuverlässige Bearbeitung der Aufgaben im Missionskontext durch die ACUs, was nur durch ein umfassendes Verständnis der Aufgabenstruktur, Abhängigkeiten zwischen einzelnen Aufgaben und der taktischen Lage sowie der Zusammenarbeit und Zusammenhänge im Team möglich ist. Insbesondere im Hinblick auf mögliche Vorgaben in einem hierarchisch organisierten Team ist trotzdem die Möglichkeit zur Interaktion auf allen oben beschriebenen

Abstraktionsgraden vorzusehen. Hierbei wird allerdings davon ausgegangen, dass das Abstraktionsniveau stets über dem einer Führung auf Wegpunktebene liegt, da erwartet wird, dass dies in den angestrebten Konfigurationen nicht handhabbar und nicht nötig ist.

In diesem Zusammenhang ist das Problem zu adressieren, wie mit Vorgaben auf unterschiedlichen Abstraktionsniveaus umgegangen werden soll. Während zunächst detailliertere Vorgaben grundsätzlich einzuhalten sind, sind Situationen vorstellbar, in denen eine Emanzipation im Sinne übergeordeter Zielsetzungen erwünscht ist (z.B. Vorgabe nicht mehr relevant; Änderung der taktischen Lage, die der Mensch nicht erkennt bzw. er keine freien Ressourcen mehr für adäquate Änderungen detaillierter Vorgaben an die ACUs hat; Angriff durch eine SAM-Stellung, auf den wegen einer Vorgabe nicht reagiert werden dürfte). Mögliche Ansätze in diesem Kontext sind, dass eine Vorgabe nicht mehr relevant ist, sofern sie erfüllt ist oder eine neue gegeben wird, oder aber in den entsprechenden Situationen ein adäquater Vorschlag an den Teamleiter oder sogar ein selbständiges Hinwegsetzen über Vorgaben erfolgt, sofern z.B. Sicherheitsziele gefährdet sind.

6.4.3.3 Automationsgrad

Grundsätzlich sollen ACUs in der Lage sein, im Rahmen von Missionsrandbedingungen, die im Einzelfall zum Beispiel durch ein Missionsbriefing oder auch durch detaillierte Vorgaben eines menschlichen Teamleiters definiert sein können, weitgehend eigenständig sinnvoll und zielgerichtet zu agieren. Dennoch soll es möglich sein, einen solchen hohen Automationsgrad einzuschränken, so dass insbesondere zwei Automationsgrade relevant sind, wenn die Aspekte „Auswahl zu bearbeitender Aufgaben“, „Verteilung von Aufgaben innerhalb eines Teams“ und „Durchführung von Aufgaben“ betrachtet werden. Diese zeichnen sich dadurch aus, dass Rückfrage beim Menschen gehalten wird, bevor eine Aktion eingeleitet wird, oder Aktivitäten selbständig ausgewählt und durchgeführt werden.

Welcher Ansatz dabei sinnvoll ist, ist im Einzelfall von der konkreten Situation abhängig, wobei zum Beispiel die Missionsphase in Zusammenhang mit der dort herrschenden Beanspruchung des Menschen (z.B. Transitflug im Gegensatz zu Endanflug auf Ziel) und die Zeitkritikalität einer Aktion (z.B. Selbstverteidigung) eine wichtige Rolle spielen. In diesem Zusammenhang könnte der Automationsgrad in einem ersten Schritt für diverse Aktionen abhängig von verschiedenen Kriterien festgelegt bzw. durch den Menschen adaptierbar gestaltet werden. Langfristig hingegen ist eine selbständige Adaption der ACUs auf Basis eines umfassenden Verständnisses des Missionskontextes und insbesondere der menschlichen Beanspruchung denkbar. Im Umfeld dieser Problematik kann dabei auf umfassende Forschungsarbeiten auf dem Gebiet der adaptierbaren und adaptiven Automation (vgl. [Scerbo, 2005]) zurückgegriffen werden.

6.4.3.4 Zusammenarbeit im Team

Im Hinblick auf die angestrebte Zusammenarbeit im Mensch-Maschine Team, welche durch die Interaktion auf hohem Abstraktionsniveau und eine hohe Eigenständigkeit der ACUs gekennzeichnet ist, ist ein gemeinsamer Vorgehensplan von Mensch und ACUs hinsichtlich der Durchführung der Mission notwendig. Dieser kann von einem Teammitglied unter Umständen in Absprache mit anderen Teammitgliedern gemeinsam erarbeitet werden. Von dem gemeinsamen Verständnis eines derartigen Missionsplans wird insofern eine Verbesserung der Zusammenarbeit erwartet, als basierend hierauf eine Antizipation des Verhaltens anderer Teammitglieder und damit insbesondere in der

SEAD-Rolle eine darauf angepasste Unterstützung des Attack-Flugzeugs vorgenommen werden kann (z.B. Auswahl zu unterdrückender SAM-Stellungen in Abhängigkeit vom geplanten Flugweg).

Die ACUs müssen somit fähig sein, einen derartigen Missionsplan in die Planung ihres individuellen Vorgehens mit einzubeziehen und gegebenenfalls auf Basis eines Verständnisses der taktischen Lage und Missionsziele einen solchen Missionsplan zu erarbeiten bzw. dessen Ausarbeitung durch gezielte Bereitstellung von Information oder das Einbringen konkreter Vorschläge zu unterstützen. Ähnliche Fähigkeiten sind auch bei der Verteilung von Aufgaben im homogenen Mensch-Maschine Team, wie es in Konfiguration 4 betrachtet wurde, nötig.

Weiterhin ist im Hinblick auf eine Antizipation des Verhaltens von Teammitgliedern und ein darauf abgestimmtes eigenes Vorgehen Information über andere Teammitglieder über deren Ressourcen, Fähigkeiten und Verpflichtungen hinaus nötig. Diese muss zum einen von ACUs proaktiv oder auf Anfrage bereitgestellt werden und zum anderen in das Verständnis der Situation miteinbezogen werden. Beispiele hierfür sind die aktuell verfolgte Absicht wie zum Beispiel die Durchführung eines Selbstverteidigungsmanövers im Gegensatz zur längerfristigen Verpflichtung zur Durchführung missionsrelevanter Aufgaben, der Status einer Aufgabe wie zum Beispiel der Zeitpunkt zu dem die Unterdrückung einer SAM-Stellung erwartet wird oder das geplante Vorgehen bei der Durchführung einer Aufgabe wie zum Beispiel die Wahl des Flugwegs am Rand des Bedrohungsbereichs einer SAM-Stellung entlang.

6.4.3.5 Aufgabendurchführung

Die Auswahl der zu bearbeitenden Aufgaben und deren Verteilung auf Teammitglieder im Missionsverlauf durch die im Rahmen dieser Arbeit entwickelten ACUs wurde von den Versuchspersonen im Allgemeinen als sinnvoll empfunden. Die Fähigkeiten der ACUs bei der konkreten Bearbeitung individueller Aufgaben, welche nicht den Schwerpunkt der vorliegenden Arbeit bildeten, bedürfen jedoch einer Weiterentwicklung und zwar vor allem hinsichtlich der folgenden Aspekte:

- Vermeidung taktischen Fehlverhaltens, z.B. hinsichtlich des Aufenthalts im Bedrohungsbereich von SAM-Stellungen
- Berücksichtigung zeitlicher Abhängigkeiten zwischen und Wichtig-/Dringlichkeit von Aufgaben bei deren Zuweisung und Durchführung, z.B. bei der Koordination der Nutzung des Korridors als gemeinsam genutzter Ressource oder der Reihenfolge der Durchführung von Aufgaben
- Antizipation des Verhaltens anderer Teammitglieder im Hinblick auf das weitere eigene Vorgehen, z.B. bei der Wahl des Zeitpunkts, zu dem ein Einflug in den Bedrohungsbereich einer SAM-Stellung stattfindet

6.4.3.6 Kommunikation im Team

Während die Vorgabe formaler Abläufe von den Versuchspersonen als akzeptabel und nicht störend empfunden wurde, ist eine Anpassung des Vokabulars an den im Rahmen militärischer Missionen gebräuchlichen Wortschatz nötig. Von diesem wird erwartet, dass er ein intuitiveres Verständnis der Bedeutung ausgetauschter Nachrichten ermöglicht als die von der FIPA vorgegebenen Performative. Bei der Definition dieses Wortschatzes müssen weiterhin Anforderungen durch die gewünschten Interaktionsniveaus (vgl. Abschnitt 6.4.3.2) und die im Rahmen der Zusammenarbeit im Team auszutauschende Information (vgl. 6.4.3.4) berücksichtigt werden. Seitens der ACUs ist hier

sowohl die Fähigkeit notwendig, diesen Wortschatz zu verstehen als auch ihn semantisch und syntaktisch korrekt bei der Generierung von Nachrichten anzuwenden.

Im Hinblick auf einen möglichst effizienten Ablauf der Kommunikation im Mensch-Maschine Team ist insbesondere eine Reduktion der Häufigkeit ausgetauschter Nachrichten bzw. initiierter Dialoge anzustreben, um nicht unnötig Aufmerksamkeits- und Verarbeitungsressourcen des Menschen zu binden. In diesem Zusammenhang werden zwei Fähigkeiten von ACUs als relevant angesehen, nämlich die Nutzung und das Verständnis von Abkürzungen und das implizite Ableiten von Informationen aus der Situation bzw. anderen Dialogen. Abkürzungen beziehen sich dabei sowohl auf eine Straffung des Ablaufs von Dialogen als auch die Nutzung von Codewörtern für häufig kommunizierte Sachverhalte (vgl. NATO Brevity Code). Die Fähigkeit implizit Informationen abzuleiten ermöglicht es, sinnvoll im Missionskontext agieren zu können ohne auf explizite Mitteilungen anderer Teammitglieder angewiesen zu sein. Beispiele hierfür sind das Ableiten von Verpflichtungen eines Teammitglieds, wenn dieses eine Anfrage angenommen hat oder es keine andere Möglichkeit gibt, Aufgaben im Team zu verteilen, weil beispielsweise nur das Attack-UCAV das Angriffsziel bekämpfen kann.

6.4.3.7 Assistenzsystem

Wie in Kapitel 2 erläutert ist die Einführung eines Assistenzsystems als eine *Operating ACU*, die den Menschen bei seinen Aufgaben unterstützt, stets sinnvoll. Im Rahmen der Experimentalkampagne wurden hierbei einige konkrete Unterstützungsmöglichkeiten identifiziert, die im Folgenden kurz charakterisiert werden. Grundsätzlich finden sich diese sowohl im Bereich der Primäraufgabe, d.h. bei der Führung des eigenen Flugzeugs im Missionskontext, als auch der Sekundäraufgabe, d.h. der Zusammenarbeit mit ACUs in einem Team. Durch dieses breite Aufgabenspektrum wird vor allem die Aufmerksamkeitslenkung auf die dringlichste Aufgabe im Sinne der ersten Onkenschen Grundforderung an Assistenzsysteme (vgl. Abschnitt 2.1.2.2) als unabdingbar angesehen, da zum Beispiel beobachtet wurde, dass Versuchspersonen Dialoge bearbeiteten, die als nicht zeitkritisch einzustufen waren, während gleichzeitig eine Änderung der taktischen Lage oder ein Angriff von einer SAM-Stellung zu lange unbemerkt blieb.

Im Zusammenhang mit Kooperation im Mensch-Maschine Team zeichnet sich ein Unterstützungsbedarf in folgenden Bereichen ab:

- Aufrechterhalten von Situationsbewusstsein hinsichtlich der Aufgaben und des Zustands anderer Teammitglieder
- Aufgabenverteilung und Informationsfluss im Team und
- Dialogführung

Der erste Aspekt zielt hierbei darauf ab, dass der Mensch zu jeder Zeit Bescheid weiß, welches Teammitglied welche Aufgabe bearbeitet und in welchem Zustand es sich befindet (z.B. Verfügbarkeit von Waffen im Hinblick auf eine mögliche Zuweisung neuer Aufgaben). Der zweite Aspekt soll Probleme vermeiden, die auftreten, wenn beispielsweise eine Aufgabe einem ungeeigneten Teammitglied zugewiesen wird, Nachrichten falsch verstanden oder an einen falschen Adressaten geschickt werden oder für andere Teammitglieder notwendige Information diesen nicht rechtzeitig mitgeteilt wird. Der dritte Aspekt schließlich soll vor allem das Führen von Dialogen erleichtern, die sich über einen langen Zeitraum erstrecken (z.B. Rückmeldung über den Missionserfolg), indem zum Beispiel ein Hinweis gegeben wird, wenn der Mensch eine Nachricht senden muss oder dies sogar vom Assistenzsystem übernommen wird.

6.4.4 Fazit

Durch die in diesem Abschnitt vorgestellte Experimentalkampagne konnte gezeigt werden, dass die im Rahmen dieser Arbeit entwickelten *Supporting ACUs* im Prinzip auch mit menschlichen Teammitgliedern kooperieren können und im Sinne des Missionserfolgs eine gute Leistung erbringen.

Dennoch ergeben sich eine Reihe von Anforderungen, die bei der Hinzunahme eines menschlichen Teammitglieds vor allem im Hinblick auf die Durchführung militärischer Missionen berücksichtigt werden müssen. Hierunter fallen eine Anpassung der Teamstruktur mit einer herausgehobenen Stellung des Menschen und die Möglichkeit, auf verschiedenen Abstraktionsebenen Anfragen zu stellen und Informationen auszutauschen. Weiterhin wurden eine situationsangepasste Adaption des Automationsgrades und eine Verbesserung der taktischen Fähigkeiten der ACUs bei der Durchführung konkreter Aufgaben diskutiert. Von besonderer Wichtigkeit ist schließlich die Anpassung der Kommunikation im Team an menschliche Eigenheiten, d.h. insbesondere beschränkte Verarbeitungs- und Aufmerksamkeitsressourcen, da Kommunikation die Basis für Kooperation im Mensch-Maschine Team darstellt.

Diese Anforderungen können jedoch weitgehend in das Konzept integriert werden, das den *Supporting ACUs* zugrunde liegt. Die wissensbasierte, symbolistische Implementierung von zielebasiertem Verhalten auf wissensbasierter Verhaltensebene ermöglicht es nämlich zum einen, Verhalten zu erklären und damit dem Menschen zusätzlich notwendige Information bereitzustellen. Zum anderen kann das dem Verhalten zugrunde liegende Wissen lokal, d.h. im Rahmen einzelner Wissensmodelle, erweitert bzw. geändert werden, um zum Beispiel zu einer besseren Einschätzung der taktischen Lage zu gelangen oder Vorgaben eines menschlichen Teammitglieds bei der Auswahl von Handlungsoptionen zu berücksichtigen.

7 Zusammenfassung und Ausblick

Sowohl Menschen als auch Maschinen müssen sich in immer komplexer werdenden Arbeitsumgebungen zurechtfinden und dabei insbesondere auch mit anderen Menschen und/oder Maschinen kooperieren, um die zu bearbeitenden Aufgabenstellungen erfolgreich meistern zu können. Dabei treten immer öfter Situationen auf, die während des Entwicklungsprozesses in der spezifischen Konstellation nicht berücksichtigt wurden. Menschen sind in der Lage, auch in solchen Situationen, in denen keine direkte Verknüpfung mit einer aktuell zu bearbeitenden Aufgabe besteht, sinnvoll, d.h. zielgerichtet, zu agieren. Sie verfügen nämlich auf der so genannten wissensbasierten Verhaltensebene über die Fähigkeit, eine Situation umfassend zu verstehen, Ziele auszuwählen, die erreicht werden sollen, und ein Vorgehen zu planen, von dem erwartet wird, dass es sie der gewünschten Situationskonfiguration näher bringt. Die Fähigkeiten von Maschinen hingegen beschränken sich meist auf die Planung und vor allem Ausführung von Aktivitäten in bekannten Situationen.

Ziel der vorliegenden Arbeit war es, im Hinblick auf eine immer wichtiger werdende Zusammenarbeit von Mensch und Maschine auch in unbekannten Situationskonfigurationen zunächst maschinelle Fähigkeiten zur Kooperation auf wissensbasierter Verhaltensebene umzusetzen, um damit die Grundlage für Mensch-Maschine Kooperation auf Basis eines umfassenden Verständnisses der Gesamtsituation zu schaffen.

Um das Zusammenwirken von Mensch und Maschine im Anwendungsfeld der UAV-Flugführung strukturiert betrachten zu können, wurde zunächst eine Analyse des Arbeitssystems „Führung mehrerer UAVs durch einen menschlichen Operateur“ durchgeführt. Hierbei wurden insbesondere verschiedene Möglichkeiten diskutiert, neben dem Menschen so genannte künstliche kognitive Einheiten als weitere intelligente Systemkomponenten an Bord der UAVs in das Arbeitssystem zu integrieren. Ausgangspunkt für die weiteren Betrachtungen war eine Konfiguration, in der ein menschlicher Operateur einem Team aus künstlichen kognitiven Einheiten Teilaufträge zuweisen kann, die diese eigenständig in Kooperation bearbeiten. Langfristig wird jedoch angestrebt, dass die künstlichen kognitiven Einheiten nicht nur untereinander sondern auch mit Menschen auf Basis eines partnerschaftlichen Verhältnisses kooperieren.

Um eine solche Integration menschlicher Teammitglieder in das maschinelle Team zu erleichtern, wurde für die Umsetzung rationalen, zielgerichteten Verhaltens im Rechner der Theorieansatz des Kognitiven Prozesses zugrunde gelegt. Dieser unterstützt insbesondere die Nachbildung der wissensbasierten Verhaltensebene des Menschen auf Basis einer expliziten Repräsentation von Handlungszielen. Den zweiten Teil des Konzepts bilden Methoden zur Kooperation, wobei vor allem die Teilaspekte der Koordination und Kommunikation im Team von Interesse sind.

Die beiden Konzeptanteile maschineller Zusammenarbeit auf wissensbasierter Verhaltensebene, nämlich künstliche Kognition und Kooperation, werden zusammengeführt, indem Kooperation entsprechend den Merkmalen analysiert wird, die gemäß dem Kognitiven Prozess notwendig sind, um wissensbasiertes Verhalten darzustellen. Hierbei nehmen Ziele von Kooperation wie zum Beispiel das Herbeiführen einer ausgewogenen Aufgabenverteilung im Team, das Vermeiden redundanter Aufgabenbearbeitung durch mehrere Teammitglieder oder das Bereitstellen relevanter Information für andere Team-

mitglieder eine zentrale Rolle ein. Daneben werden Handlungsalternativen identifiziert, welche eingesetzt werden können, um derartige Ziele zu erreichen. Außerdem wird beschrieben, wie solche Handlungsalternativen ausgeführt werden können und welches Wissen notwendig ist, um zu einem Verständnis der Situation zu gelangen, auf dessen Basis die Relevanz der Ziele bestimmt sowie das weitere Vorgehen geplant werden kann.

Die Implementierung eines Funktionsprototyps, d.h. konkret einer künstlichen kognitiven Einheit mit kooperativen Fähigkeiten, erfolgte auf Grundlage der kognitiven Systemarchitektur COSA und der Programmiersprache CPL. Erstere stellt applikationsunabhängige Anteile des Kognitiven Prozesses in einem Framework zur Verfügung und erlaubt damit eine Fokussierung auf die Umsetzung der gewünschten Funktionalität. Die Programmiersprache CPL unterstützt die Modellierung von Wissen auf Basis mentaler Begriffe wie zum Beispiel Zielen. Damit wird das Programmiermodell an das Denkmodell des Menschen angenähert, so dass die für Kooperation auf wissensbasierter Verhaltensebene identifizierten Konzepte ohne weitere Transformationen direkt in ein Rechnermodell umgesetzt werden können.

Abschließend wurde der implementierte Funktionsprototyp anhand eines Anwendungsbeispiels evaluiert. Hierbei handelt es sich um eine kooperative SEAD-/Attack-Mission, bei der fünf UAVs gemeinsam ein Ziel bekämpfen sollen, wobei dieses Missionsziel wegen unterschiedlicher Fähigkeiten und Ressourcen der UAVs nur in Zusammenarbeit erreicht werden kann. Jedes dieser UAVs wurde dabei von einer Instanz des Funktionsprototyps, d.h. einer künstlichen kognitiven Einheit mit kooperativen Fähigkeiten, geführt. In einem ersten Schritt wurde nachgewiesen, dass die geforderte Funktionalität mit dem gewählten Ansatz abgebildet werden kann, nämlich dass die künstlichen kognitiven Einheiten in der Lage sind, im Hinblick auf das Erreichen des Missionsziels zusammenzuarbeiten. In einem zweiten Schritt wurden die Errungenschaften im Hinblick auf die Realisierung von Kooperation auf wissensbasierter Verhaltensebene dargestellt. Hierbei ermöglicht die explizite Repräsentation von Zielen in Verbindung mit dem Aufbau eines umfassenden Situationsverständnisses und dem Wissen um mögliche Handlungsoptionen, dass nicht für jede spezifische Situation definiert werden muss, welche Aktionen dort auszuführen sind. Vielmehr wird durch die Ziele beschrieben, was erreicht werden soll und ein Vorgehen bestimmt, wie diese Zielsituation erreicht werden kann. Dieser Ansatz stellt insbesondere in Situationen, für die eigentlich bekannt ist, was zu tun ist, nicht die effizienteste Vorgehensweise dar und ergibt nicht immer optimales Verhalten. Dadurch dass das vorhandene Wissen aber auch in unbekannten Situationen stets bestmöglich im Sinne des Erreichens der Ziele eingesetzt wird, kann letztlich eine größere Flexibilität im Hinblick auf die Situationskonfigurationen erreicht werden, in denen ein System sinnvoll agieren kann.

Schließlich erfolgte eine Bewertung der für maschinelle Kooperation ausgelegten künstlichen kognitiven Einheiten im Hinblick auf deren Eignung für Mensch-Maschine Kooperation. Hierzu wurde eine Experimentalkampagne durchgeführt, bei der im oben skizzierten Anwendungsbeispiel ein UAV durch ein bemanntes Flugzeug ersetzt wurde. Hierbei konnte gezeigt werden, dass eine Zusammenarbeit im Prinzip möglich ist und gute Ergebnisse im Sinne des Missionserfolgs erzielt werden. Um jedoch die Zusammenarbeit mit maschinellen Teammitgliedern für den Menschen angenehmer zu gestalten und die gemeinsam erzielbare Leistung weiter zu steigern, wurde auf Basis der Experimente eine Reihe von Anforderungen identifiziert, denen zukünftige künstliche kognitive Einheiten im Mensch-Maschine Team genügen sollen. Diese Anforderungen können weitgehend in das im Rahmen dieser Arbeit zugrunde gelegte Konzept integriert

werden, so dass hiermit eine tragfähige Basis für weitere Arbeiten geschaffen werden konnte.

Eine Weiterentwicklung der künstlichen kognitiven Einheiten im Hinblick auf Mensch-Maschine-Kooperation im Arbeitssystem kann dabei unter verschiedenen Gesichtspunkten erfolgen (vgl. auch Abschnitt 2.1.2.2). Von besonderem Interesse ist in diesem Zusammenhang zum einen die Betrachtung der Zusammenarbeit in einem Mensch-Maschine-Team, in dem alle Teammitglieder ähnliche Aufgabenstellungen bearbeiten. Zum anderen kann eine Ausweitung der Fähigkeiten im Hinblick auf die Unterstützung eines menschlichen Teammitglieds durch ein so genanntes Assistenzsystem erfolgen. In diesem Kontext können und sollen insbesondere Modelle des Menschen in die Verhaltensentscheidung der künstlichen kognitiven Einheiten mit einbezogen werden (z.B. Beanspruchung des Menschen im Aufgabenkontext, vgl. [Donath & Schulte, 2006]), so dass eine situationsangepasste Adaption des Automationsgrades und eine angemessene Unterstützung des Menschen möglich wird.

Eine weitere Ausweitung der Fähigkeiten des hier vorgestellten Funktionsprototyps kann erfolgen, indem nicht nur Menschen in die Betrachtung mit einbezogen werden, sondern auch Einsatzszenarien verfeinert und andere Anwendungsgebiete untersucht werden. Hierbei ist zunächst eine Anpassung und Ausweitung des domänenspezifischen Wissens der künstlichen kognitiven Einheiten im Hinblick auf andere luftfahrttechnische Applikationen naheliegend, wie zum Beispiel die Zusammenarbeit mehrerer bemannter und unbemannter Hubschrauber im Rahmen von militärischen und zivilen Missionen. Daneben ist jedoch auch eine Anwendung in anderen Bereichen wie zum Beispiel der Fahrzeugtechnik oder Produktionstechnik vorstellbar. In diesem Zusammenhang kann eine Weiterentwicklung auch dahingehend erfolgen, dass eine Erprobung nicht nur in simulierten sondern auch realen Umgebungen stattfindet, indem zum Beispiel Flugversuche mit UAV-Demonstratoren (siehe z.B. [Meitinger et al., 2005][Höse et al., 2007][Kriegel et al., 2007]) durchgeführt werden.

Im Hinblick auf einen Einsatz in solchen realen Umgebungen, der unter anderem mit beschränkten Rechnerressourcen an Bord der Fluggeräte und erhöhten Anforderungen an die Geschwindigkeit der Verhaltensentscheidung einhergeht, ist es von besonderem Interesse, eine Performanzsteigerung des Systems zu untersuchen. Diese kann zum einen durch eine Optimierung und Weiterentwicklung des verwendeten Frameworks erzielt werden (vgl. [Matzner et al., 2008]). Zum anderen kann bei der Modellierung des Verhaltens nicht nur die wissensbasierte Verhaltensebene, sondern zusätzlich auch die regel- und ggf. sogar fertigkeitsbasierte Verhaltensebene berücksichtigt werden. Damit könnten die Vorteile der einzelnen Ebenen unter Vermeidung der Nachteile kombiniert werden. So könnten beispielsweise vorgefertigte Vorgehensweisen in bekannten Situationen schnelles und effizientes Verhalten erzeugen, während dieses Verhalten parallel auf wissensbasierter Verhaltensebene vor dem Hintergrund der zu erreichenden Ziele auf Basis eines umfassenden Situationsverständnisses evaluiert und ggf. steuernd eingegriffen werden könnte. Zusätzlich ergäbe sich durch die Berücksichtigung wissensbasierten Verhaltens die bereits erwähnte Flexibilität im Hinblick auf das Auftreten unbekannter Situationskonfigurationen. In diesem Zusammenhang ist auch die Anwendung maschineller Lernverfahren denkbar, um zum Beispiel auf wissensbasierter Ebene erarbeitete Problemlösungen künftig regelbasiert anwenden zu können.

Darüber hinaus ist auch eine vertiefte Betrachtung von Methoden im Umfeld der Wissensakquisition und Wissensmodellierung anzustreben, um die Entwicklung von Systemen, welche wie der hier entwickelte Funktionsprototyp auf umfangreichen Wis-

sensbasen beruhen, zu vereinfachen. In diesem Kontext relevante Ansätze sind zum Beispiel Methoden zum automatischen Wissenserwerb, bei denen Rechner in die Lage versetzt werden, auf Basis umfangreichen Datenmaterials und gegebenenfalls domänenspezifischen Hintergrundwissens für eine Applikation relevantes Wissen zu extrahieren. Auch Methoden, die eine weitere Annäherung von Expertenwissen und Wissensrepräsentation im umgesetzen System ermöglichen, sind in diesem Zusammenhang von großem Interesse. Hierbei ist es zum Beispiel denkbar, bei der Erhebung von Expertenwissen bereits die angestrebte Struktur der Wissensrepräsentation in der sich anschließenden Implementierung zu berücksichtigen.

Dieser Überblick über einige, zukünftig mögliche Anwendungsbereiche und Forschungsthemen im Umfeld kognitiver Kooperation auf wissensbasierter Verhaltensebene vermittelt einen Eindruck davon, für welche Themengebiete die im Rahmen dieser Arbeit erzielten Ergebnisse relevant sind. Die hier umgesetzte Fähigkeit zu maschineller Kooperation und die Realisierung wissensbasierten Verhaltens im Rechner stellen eine gute Ausgangsbasis für weitere Arbeiten insbesondere im Bereich der Mensch-Maschine-Kooperation dar.

Literaturverzeichnis

[ACT-R Homepage]
ACT-R Homepage. Online-Ressource: http://act-r.psy.cmu.edu/. Letzter Zugriff: 11.09.2007.

[Adam et al., 2003]
Volkmar Adam, Peter Hecker, Bernd Korn & Peter Stütz (2003). Procedures and Technologies for Remotely Controlled and Autonomous UAV Guidance within Civil Airspace. In: ONERA (Hrsg.) *Dynamic Flow Control, Unmanned Aerial Vehicles, Payload, Control and Mission Management, Space R&T and Missions for Earth Observation*, ODAS 2003, Toulouse, 4.-6. Juni 2003.

[Adolf, 2007]
Florian M. Adolf (2007). A Sequence and Supervisory Control System for Onboard Mission Management of an Unmanned Helicopter. In: *Proceedings of the 1st CEAS European Air and Space Conference*. Berlin, 10.-13. September 2007.

[Anderson, 2001]
John R. Anderson (2001). *Kognitive Psychologie*. Heidelberg, Berlin: Spektrum Akademischer Verlag.

[Anderson et al., 2004]
John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere & Yulin Qin (2004). An Integrated Theory of Mind. *Psychological Review*, *111(4)*, S. 1036-1060.

[Ashby, 1956]
William R. Ashby (1956). *An Introduction to Cybernetics*. London: Chapman & Hall. Erhältlich online (1999): http://pcp.vub.ac.be/books/IntroCyb.pdf

[Augier & Feigenbaum, 2003]
Mie Augier & Edward Feigenbaum (2003). Herbert A. Simon. In: *Proceedings of the American Philosophical Society*, *147(2)*, Juni 2003, S. 193-198.

[Austin, 1962]
John L. Austin (1962). *How to Do Things With Words*. Oxford: Oxford University Press.

[Balzert, 2000]
Helmut Balzert (2000). *Lehrbuch der Software-Technik*. Heidelberg: Spektrum Akademischer Verlag.

[Baxter & Horn, 2003]
Jeremy W. Baxter & Graham S. Horn (2003). A Multi-agent System for Executing Group Tasks. In: V. Palade, R. J. Howlett & L. C. Jain (Hrsg.), *KES 2003* (S. 697-703). Berlin, Heidelberg: Springer.

[Baxter & Horn, 2005]
Jeremy W. Baxter & Graham S. Horn (2005). Controlling Teams of Uninhabited Air Vehicles. In: *Proceedings of 4th International Joint*

Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '05). Utrecht, NL, 25.-29. Juli 2005.

[Beard et al., 2002]
Randal W. Beard, Timothy W. McLain, Michael A. Goodrich & Erik P. Anderson (2002). Coordinated Target Assignment and Intercept for Unmanned Air Vehicles. *IEEE Transactions on Robotics and Automation*, *18(6)*, S. 911-922.

[Biggers & Ioerger, 2001]
Keith E. Biggers & Thomas R. Ioerger (2001). Automatic Generation of Communication and Teamwork within Multi-Agent Systems. *Applied Artificial Intelligence*, *15(10)*, S. 875-916.

[Billings, 1997]
Charles E. Billings (1997). *Aviation Automation – the Search for a Human-Centered Approach*. Mahwah, NJ: Erlbaum.

[Bond & Gasser, 1988]
Alan H. Bond & Les Gasser (1988). *Readings in Distributed Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann Publishers.

[Borghoff & Schlichter, 2000]
Uwe M. Borghoff & Johann H. Schlichter (2000). *Computer Supported Cooperative Work: Introduction to Distributed Applications*. Berlin, Heidelberg: Springer.

[Bratman, 1992]
Michael E. Bratman (1992). Shared Cooperative Activity. In: *The Philosophical Review*, *101(2)*. S. 327-341.

[van Breda & van Erp, 2006]
Leo van Breda & Jan B. F. van Erp (2006). Supervising UMVs: Improving Operator Performance through Anticipatory Interface Concepts. In: *NATO RTO-Meeting Proceedings HFM-135: Human Factors of Uninhabited Military Vehicles as Force Multipliers*. Biarritz, Frankreich, 9.-11. Oktober 2006.

[Buchanan et al., 1983]
Bruce G. Buchanan, David Barstow, Robert Bechtal, James Bennett, William Clancey, Casimir Kulikowski, Tom Mitchell & Donald A. Waterman (1983). Constructing an Expert System. In: F. Hayes-Roth, D. A. Waterman & D. B. Lenat (Hrsg). *Building Expert Systems*. Reading, MA: Addison-Wesley.

[Buss et al., 2007]
Martin Buss, Michael Beetz & Dirk Wollherr (2007). CoTeSys – Cognition for Technical Systems. In: *Proceedings of the 4th COE Workshop on Human Adaptive Mechatronics (HAM)*. Tokyo, Japan, 2.-3. März 2007.

[Byrne, 2003]
Michael D. Byrne (2003). Cognitive Architecture. In: J. Jacko & A. Sears (Hrsg.) *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. Mahwah, NJ: Lawrence Erlbaum. S. 97-117.

[Byrne & Kirlik, 2002]
Michael D. Byrne & Alex Kirlik (2002). *Integrated Modeling of Cognition and*

the Information Environment: Closed-Loop, ACT-R Modeling of Aviation Taxi Errors and Performance. Technical Report AHFD-02-19/NASA-02-10, Aviation Human Factors Division, Institute of Aviation, University of Illinois, Illinois, December 2002. Erhältlich online: http://www.humanfactors.uiuc.edu/Reports&PapersPDFs/TechReport/02-19.pdf (Letzter Zugriff: 10.5.2007)

[Byrne & Kirlik, 2003]
Michael D. Byrne & Alex Kirlik (2003). *Using Computational Cognitive Modeling to Diagnose Possible Sources of Aviation Error*. Technical Report AHFD-03-14/NASA-03-4, Aviation Human Factors Division, Institute of Aviation, University of Illinois, Illinois, June 2003. Erhältlich online: http://www.humanfactors.uiuc.edu/Reports&PapersPDFs/TechReport/03-14.pdf (Letzter Zugriff: 10.5.2007)

[Chandler et al., 2002]
Phillip R. Chandler, Meir Pachter, Kendall E. Nygard & Dharba Swaroop (2002). Cooperative Control for Target Classification. In R. Murphey & P. M. Pardalos (Hrsg.), *Cooperative Control and Optimization*. Kluwer Academic Publishers.

[Cohen & Levesque, 1990]
Philip R. Cohen & Hector J. Levesque (1990). Intention is choice with commitment. *Artificial Intelligence*, *42*, S. 213-261.

[Cohen & Levesque, 1991]
Philip R. Cohen & Hector J. Levesque (1991). Teamwork. *Noûs*, *25(4)*, S. 487-512.

[CoTeSys Homepage]
CoTeSys – Cognition for Technical Systems Homepage. Internetressource: http://www.cotesys.de. Letzter Zugriff: 05.12.2007.

[Cummings & Mitchell, 2005]
Mary L. Cummings & Paul J. Mitchell (2005). Managing Multiple UAVs through a Timeline Display. In: *Proceedings of the InfoTech@Airspace*. AIAA-2005-7060. Arlington, VA, 26.-29. September 2005.

[Dargar et al., 2002]
Anup Dargar, Ahmed Kamel, Gordon Christensen & Kendall E. Nygard (2002). *An Agent Based Framework for UAV Collaboration*. In: *Proceedings of the ISCA 11th International Conference on Intelligent Systems*.

[Dittrich et al., 2003]
Jörg Dittrich, Andreas Bernatz & Frank Thielecke (2003). Intelligent Systems Research Using a Small Autonomous Rotorcraft Testbed. In: *2nd AIAA "Unmanned Unlimited" Systems, Technologies, and Operations*. AIAA 2003-6561. San Diego, CA, 15.-18. September 2003.

[Donath & Schulte, 2006]
Diana Donath & Axel Schulte (2006). Weiterentwicklungsmöglichkeiten kognitiver und kooperativer Operateurassistenz: Ein Konzeptansatz für modellbasierte adaptive Automation. *48. Fachausschusssitzung Anthropotechnik T5.4 Cognitive Systems Engineering in der Fahrzeug- und Prozessführung*. Fraunhofer-IITB, Karlsruhe, 24.-25. Oktober 2006.

[Doran et al., 1997]
Jim E. Doran, Stan Franklin, Nicholas R. Jennings & Tim Norman (1997). On Cooperation in Multi-Agent Systems. *The Knowledge Engineering Review*, *12(3)*, S. 309-314.

[Draper et al., 2006]
Mark H. Draper, Gloria Calhoun, Jeremy Nelson, Austen Lefebvre, Heath Ruff (2006). Synthetic Vision Overlay Concepts for Uninhabited Aerial Vehicle Operations: Evaluation of Update Rate on Four Operator Tasks. In: *NATO RTO-Meeting Proceedings HFM-135: Human Factors of Uninhabited Military Vehicles as Force Multipliers*. Biarritz, Frankreich, 9.-11. Oktober 2006.

[Dogan, 2003]
Atilla Dogan (2003). Probabilistic Approach in Path Planning for UAVs. In: *Proceedings of IEEE International Symposium on Intelligent Control*. Houston, Oktober 2003.

[Durfee et al, 1992]
Edmund H. Durfee, Victor R. Lesser & Daniel D. Corkill (1992). Distributed Problem Solving. In: Stuart C. Shapiro (Hrsg.) *Encyclopedia of Artificial Intelligence*. John Wiley. S. 379-388.

[Eisenführ & Weber, 2003]
Franz Eisenführ & Martin Weber (2003). *Rationales Entscheiden*. Berlin: Springer. 4. Auflage.

[Endsley, 1995]
Mica R. Endsley (1995). Toward a Theory of Situation Awareness in Dynamic Systems. In: *Human Factors*, *37(1)*, S. 32-64.

[Fan & Yen, 2004]
Xiaocong Fan & John Yen (2004). Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents. *Journal of Physics of Life Reviews*, *1(3)*, S. 173-201.

[Ferber, 1999]
Jacques Ferber (1999). *Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence*. Harlow: Addison-Wesley.

[Ferber, 2001]
Jacques Ferber (2001). *Multiagentensysteme – Eine Einführung in die Verteilte Künstliche Intelligenz*. München: Addison-Wesley.

[Finin et al., 1994]
Tim Finin, Richard Fritzson, Don McKay & Robin McEntire (1994). KQML as an Agent Communication Language. In: *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM '94)*, ACM Press, November 1994.

[FIPA Homepage]
Homepage der Foundation for Intelligent Physical Agents. Online-Ressource: http://www.fipa.org (Letzter Zugriff: 11.09.2007).

[FIPA Nachrichtenformat, 2002]
FIPA ACL Message Structure Specification (2002). Dokumentennummer

SC00061G. Erhältlich online: http://www.fipa.org/specs/fipa00061/ SC00061G.pdf (Letzter Zugriff: 11.09.2007).

[FIPA Performative, 2002]
FIPA Communicative Act Library Specification (2002). Dokumentennummer SC00037J. Erhältlich online: http://www.fipa.org/specs/fipa00037/ SC00037J.pdf (Letzter Zugriff: 11.09.2007).

[FIPA Propose Interaktion, 2002]
FIPA Propose Interaction Protocol Specification (2002). Dokumentennummer SC00036H. Erhältlich online: http://www.fipa.org/specs/fipa00036/ SC00036H.pdf (Letzter Zugriff: 11.09.2007).

[FIPA Query Interaktion, 2002]
FIPA Query Interaction Protocol Specification (2002). Dokumentennummer SC00027H. Erhältlich online: http://www.fipa.org/specs/fipa00027/ SC00027H.pdf (Letzter Zugriff: 11.09.2007).

[FIPA Request Interaktion, 2002]
FIPA Request Interaction Protocol Specification (2002). Dokumentennummer SC00026H. Erhältlich online: http://www.fipa.org/specs/fipa00026/ SC00026H.pdf (Letzter Zugriff: 11.09.2007).

[FIPA Subscribe Interaktion, 2002]
FIPA Subscribe Interaction Protocol Specification (2002). Dokumentennummer SC00035H. Erhältlich online: http://www.fipa.org/ specs/fipa00035/SC00035H.pdf (Letzter Zugriff: 11.09.2007).

[Fitts, 1951]
Paul M. Fitts (1951). *Human Engineering for an Effective Air Navigation and Traffic Control System*. Washington, D.C.: National Academy of Sciences.

[Fleishman & Zaccaro, 1992]
Edwin A. Fleishman & Stephen J. Zaccaro (1992). Toward a Taxonomy of Team Performance Functions. In R. W. Swezey & E. Salas (Hrsg.), *Teams: Their Training and Performance* (S. 31-56). Norwood, New Jersey: Ablex Publishing Corporation.

[Frampton & Keirl, 2006]
Robert A. Frampton & John M. Keirl (2006). Autonomy and its Application to Unmanned Systems. In: *Proceedings of the 1st Moving Autonomy Forward Conference*. Grantham, UK, 21.-22. Juni 2006.

[Frey, 2005]
Andreas Frey (2005). *Überwachung und Kontrolle in einem künstlichen kognitiven System zur autonomen Fahrzeugführung*. Dissertation, Universität der Bundeswehr München. Berlin: Köster.

[Frey et al., 2001]
Andreas Frey, Andreas Lenz, Henrik Putzer, Anton Walsdorf & Reiner Onken (2001). In-Flight Evaluation of CAMA – The Crew Assistant Military Aircraft. In: *Deutscher Luft- und Raumfahrtkongress*, Hamburg, 17.-20. September 2001.

[Gellert & Nowak, 2002]
Manfred Gellert & Claus Nowak (2002). *Teamarbeit, Teamentwicklung,*

Teamberatung. Ein Praxisbuch für die Arbeit in und mit Teams. Meezen: Christa Limmer.

[Gerlach, 1996]
Marc Gerlach (1996). *Schnittstellengestaltung für ein Cockpitassistenzsystem unter besonderer Berücksichtigung von Spracheingabe*. Dissertation, Universität der Bundeswehr München, Fortschritt-Bericht VDI, Reihe 12 Nr. 273, Düsseldorf: VDI-Verlag.

[Gluck et al., 2005]
Kevin A. Gluck, Jerry Ball, Glenn F. Gunzelmann, Michael A. Krusmark, Don R. Lyon & Nancy J. Cooke (2005). A Prospective Look at a Synthetic Teammate for UAV Applications. In: *Proceedings of the American Institute of Aeronautics and Astronautics Infotech@Aerospace Conference*. Arlington, VA, 26.-29. September 2005. AIAA 2005-6970.

[Görz, 1993]
Günther Görz (Hrsg.) (1993). *Einführung in die künstliche Intelligenz*. Bonn: Addison-Wesley.

[Grosz & Kraus, 1996]
Barbara J. Grosz & Sarit Kraus (1996). Collaborative plans for complex group action. *Artificial Intelligence*, *86(2)*, S. 269-357.

[Groth et al., 2006]
Conrad Groth, Claudia Meitinger, Diana Donath & Axel Schulte (2006). Missionsauftragsanalyse in COSA als Funktionsmodul eines Pilotenassistenzsystems. In: *Deutscher Luft- und Raumfahrtkongress 2006*. Braunschweig, 6.-9. November 2006.

[Gu et al., 2004]
Dawei Gu, Waseem A. Kamal & Ian Postlethwaite (2004). A UAV Waypoint Generator. In: *Proceedings of AIAA 1st Intelligent Systems Technical Conference*. Chicago, September 2004.

[Hart & Staveland, 1988]
Sandra G. Hard & Lowell E. Staveland (1988). Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In: P. A. Hancock & N. Meshkati (Hrsg.) *Human Mental Workload*. Amsterdam: North Holland Press, S. 139-184.

[Hegazy et al., 2005]
Tamir Hegazy, Ben Ludington & George Vachtsevanos (2005). Reconnaissance and Surveillance in Urban Terrain with Unmanned Aerial Vehicles. In: *Proceedings of the 16th IFAC World Congress*. Prag, CZ.

[Hollnagel & Woods, 2005]
Erik Hollnagel & David D. Woods (2005). *Joint Cognitive Systems. Foundations of Cognitive Systems Engineering*. Boca Raton, FL: Taylor & Francis.

[Höse et al., 2007]
Dennis Höse, Michael Kriegel & Axel Schulte (2007). Development of a Microcontroller Based Sensor Acquisition System for Uninhabited Aerial Vehicles. In: *Proceedings of the 1st CEAS European Air and Space Conference*. Berlin, 10.-13. September 2007.

[Howitt & Richards, 2006]
Sara Howitt & Dale Richards (2006). Human Interaction with Teams of Autonomous UAVs. In: *Proceedings of the 1st Moving Autonomy Forward Conference 2006*. Lincoln, UK, 21.-22. Juni 2006.

[IEEE, 2007]
IEEE Transactions on Control Systems Technology. Special Issue on Multivehicle Systems Cooperative Control with Application. July 2007. Volume 15, Number 4. ISSN 1063-6536.

[d'Inverno et al., 1998]
Mark d'Inverno, David Kinny, Michael Luck & Michael Wooldridge (1998). A Formal Specification of dMARS. In: Singh, Rao & Wooldridge (Hrsg.). *Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages*. Lecture Notes in AI, 1365. S. 155-176. Berlin: Springer.

[Jarasch et al., 2006]
Gregor Jarasch, Matthias Friedrich, Kai Harth, Martin Momberg, Simon Schärer, Achim Schönhoff & Dana Wetteborn (2006). System Design des Barracuda Flight Control Systems. In: *Deutscher Luft- und Raumfahrtkongress 2006*. Braunschweig, 6.-9. November 2006.

[Jarasch & Schulte, 2008]
Gregor Jarach & Axel Schulte (2008). Satisfying Integrity Requirements for Highly Automated UAV Systems by a Systems Engineering Approach to Cognitive Automation. In: *Proceedings of the 27th Digital Avionics Systems Conference*. St. Paul, MN, 26.-30. Oktober 2008.

[Jennings, 1994]
Nicholas R. Jennings (1994). *Cooperation in Industrial Multi-Agent Systems*. Singapore: World Scientific.

[Jennings, 1995]
Nicholas R. Jennings (1995). Controlling Cooperative Problem Solving in Industrial Multi-agent Systems Using Joint Intentions. *Artificial Intelligence, 75*, S. 195-240.

[Jennings, 1996]
Nicholas R. Jennings (1996). Coordination Techniques for Distributed Artificial Intelligence. In: G. M. P. O'Hare & N. R. Jennings (Hrsg.). *Foundations of Distributed Artificial Intelligence*. Chichester: Wiley. S. 187-210.

[Johannsen, 1993]
Gunnar Johannsen (1993). *Mensch-Maschine-Systeme*. Berlin, Heidelberg: Springer.

[Johnson, 1997]
Todd R. Johnson (1997). Control in ACT-R and Soar. In: M. Shafto & P. Langley (Hrsg.), *Proceedings of the 19th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum Associates. S. 343-348.

[Jones et al., 1993]
Randolph Jones, Milind Tambe, John E. Laird & Paul S. Rosenbloom (1993). Intelligent automated agents for flight training simulators. In: *Proceedings of*

the 3rd Conference on Computer Generated Forces and Behavioral Representation.

[Jones, 1996]
Gary Jones (1996). Trip Report (Not Quite a Conference Report) on the Architectures of Soar and ACT-R and How They Model Human Behaviour. In: *Artificial Intelligence and Simulation of Behaviour Quarterly 96*. Winter 1996, S. 41-44.

[Jones et al., 1999]
Randolph M. Jones, John E. Laird, Paul E. Nielsen, Karen J. Coulter, Patrick Kenny & Frank V. Koss (1999). Automated Intelligent Pilots for Combat Flight Simulation. *AI Magazine*, *20(1)*, S. 27-41.

[Karbach & Linster, 1990]
Werner Karbach & Marc Linster (1990). *Wissensakquisition für Expertensysteme*. München: Hanser.

[Kriegel et al., 2007]
Michael Kriegel, Claudia Meitinger & Axel Schulte (2007). Operator Assistance and Semi-Autonomous Functions as Key Elements of Future Systems for Multiple UAV Guidance. In: *Proceedings of the 7th International Conference on Engineering Psychology and Cognitive Ergonomics* (in Verbindung mit HCI International 2007). Peking, China, 22.-27. Juli 2007.

[Kriegel & Schulte, 2006]
Michael Kriegel & Axel Schulte (2006). Work System Analysis of the Integration of Autonomous Functions and Intelligent Operator Assistance in UAV Guidance. In: *NATO RTO-Meeting Proceedings HFM-135: Human Factors of Uninhabited Military Vehicles as Force Multipliers*. Biarritz, Frankreich, 9.-11. Oktober 2006.

[Kuwata et al., 2007]
Yoshiaki Kuwata, Arthur Richards, Tom Schouwenaars & Jonathan P. How (2007). Distributed Robust Receding Horizon Control for Multivehicle Guidance. In: *IEEE Transactions on Control Systems Technology*, *15(4)*, S. 627-641.

[Laird et al., 1987]
John E. Laird, Allen Newell & Paul S. Rosenbloom (1987). Soar: An Architecture for General Intelligence. *Artificial Intelligence*, *33*, S. 1-64.

[Laird et al., 1994]
John E. Laird, Randolph M. Jones & Paul E. Nielsen (1994). Coordinated Behavior of Computer Generated Forces in TacAir-Soar. In: *Proceedings of the 4th Conference on Computer Generated Forces and Behavioral Representation*.

[Laird et al., 1998]
John E. Laird, Randolph M. Jones & Paul E. Nielsen (1998). Knowledge-based Multiagent Coordination. In: *Presence*, *7(6)*, Dezember 1998, S. 547-563.

[Laird, 2006]
John E. Laird (2006). *The Soar 8 Tutorial*. Erhältlich online: http://prdownloads.sourceforge.net/soar/Soar-Suite-8.6.3.exe?download (Letzter Zugriff: 18.03.2008). University of Michigan.

[Langley & Rogers, 2005]
Pat Langley & Seth Rogers (2005). An Extended Theory of Human Problem Solving. In: *Proceedings of the 27th Annual Meeting of the Cognitive Science Society*. Stresa, Italien.

[Leuthäusser & Raupp, 1991]
Ulrich Leuthäusser & Friedhelm Raupp (1991). An efficient method for three-dimensional route planning with different strategies and constraints. In: *Air Vehicle Mission Control and Management*. AGARD CP 504, Amsterdam, 22.-25. Oktober 1991.

[Lehman et al., 1998]
Jill F. Lehman, John E. Laird & Paul S. Rosenbloom (1998). A Gentle Introduction to Soar, an Architecture for Human Cognition. In D. Scarborough & S. Sternberg (Hrsg.), *Invitation to Cognitive Science* (Vol. 4). Cambridge, MA: MIT Press.

[Lehman et al., 2006]
Jill F. Lehman, John E. Laird & Paul S. Rosenbloom (2006). A Gentle Introduction to Soar, an Architecture for Human Cognition: 2006 Update. Erhältlich online: http://ai.eecs.umich.edu/soar/sitemaker/docs/ misc/GentleIntroduction-2006.pdf (Letzter Zugriff: 12.5.2007)

[Levesque et al., 1990]
Hector J. Levesque, Philip R. Cohen & José H.T. Nunes (1990). On Acting Together. In: *Proceedings of the National Conference on Artificial Intelligence (AAAI-1990)*.

[Maes, 1990]
Pattie Maes (1990). Situated Agents Can Have Goals. *Robotics and Autonomous Systems*, *6*, S. 49-70.

[Malone & Crowston, 1994]
Thomas W. Malone & Kevin Crowston (1994). The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, *26(1)*, S. 87-119.

[von Martial, 1992]
Frank von Martial (1992). *Coordinating Plans of Autonomous Agents.* LNAI 610. Berlin: Springer.

[Matzner et al., 2008]
Alexander Matzner, Mark Minas & Axel Schulte (2008). Efficient Graph Matching with Application to Cognitive Automation. In: A. Schürr, M. Nagl, A. Zündorf (Hrsg.): *Proceedings of the 3rd International Workshop on Applications of Graph Transformation with Industrial Relevance (AGTIVE '07)*. Berlin: Springer.

[Meitinger & Schulte, 2004]
Claudia Ertl & Axel Schulte (2004). System Design Concept for Co-operative and Autonomous Mission Accomplishment of UAVs. (Kognitiver Systemansatz für die Entwicklung eines Flugführungssystems für autonom und kooperativ agierende UCAVs). In: *Deutscher Luft- und Raumfahrtkongress 2004*. Dresden, 20.-23. September 2004.

[Meitinger & Schulte, 2005]
Claudia Ertl & Axel Schulte (2005). Enabling Autonomous UAV Co-operation

by Onboard Artificial Cognition. In: *Proceedings of the 1st International Conference on Augmented Cognition* (in Verbindung mit HCI International 2005). Las Vegas, USA, 22.-27. Juli 2005.

[Meitinger et al., 2005]
Claudia Ertl, Michael Kriegel & Axel Schulte (2005). Experimental Set-Up for the Development of Autonomous Capabilities and Operator Assistance in UAV Guidance. In: *Proceedings of the 6th Conference on Engineering Psychology & Cognitive Ergonomics* (in Verbindung mit HCI International 2005). Las Vegas, USA, 22.-27. Juli 2005.

[Meitinger & Schulte, 2006a]
Claudia Meitinger & Axel Schulte (2006). Human-Centred Automation for UAV-Guidance: Oxymoron or Tautology? – The Potential of Cognitive and Co-operative Systems – In: *Proceedings of the 1st Moving Autonomy Forward Conference 2006*. Lincoln, UK, 21.-22. Juni 2006.

[Meitinger & Schulte, 2006b]
Claudia Meitinger & Axel Schulte (2006). Cognitive Machine Co-operation as Basis for Guidance of Multiple UAVs. In: *NATO RTO-Meeting Proceedings HFM-135: Human Factors of Uninhabited Military Vehicles as Force Multipliers*. Biarritz, Frankreich, 9.-11. Oktober 2006.

[Meitinger & Schulte, 2007]
Claudia Meitinger & Axel Schulte (2007). *Algorithmenentwicklung und Simulation zur Schwarmführung von UCAVs*. Institutsbericht Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Institut für Systemdynamik und Flugmechanik UniBwM / LRT13 / IB / 2007-1.

[Mitchell, 1997]
Tom M. Mitchell (1997). *Machine Learning*. New York: McGraw-Hill.

[Moitra et al., 2003]
Abha Moitra, Robert M. Mattheyses, Virginia A. DiDomizio, Louis J. Hoebel, Robert J. Szczerba & Boris Yamrom (2003). Multivehicle Reconnaissance Route and Sensor Planning. In: *IEEE Transactions on Aerospace and Electronic Systems*, *39(4)*, S. 799-812.

[Moormann et al., 2007]
Dieter Moormann, Udo Korte, Balazs Fischer & Markus Kurze (2007). The Barracuda AutoFlight Module for Autonomous Flight. In: *Proceedings of the 17th IFAC Symposium on Automatic Control in Aerospace*. Toulouse, FR, 25.-29. Juni 2007.

[Newell, 1990]
Allen Newell (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

[Newell & Simon, 1972]
Allen Newell & Herbert A. Simon (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

[Niermeyer, 2001]
Rainer Niermeyer (2001). *Teamarbeit. Führen und Erfolge sichern*. Freiburg: Haufe.

[Noy & McGuinness, 2001]
Natalya F. Noy & Deborah L. McGuinness (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 & Stanford Medical Informatics Technical Report SMI-2001-0880. Erhältlich online: http://protege.stanford.edu/publi cations/ontology_development/ontology101.pdf (Letzter Zugriff: 15.6.2007)

[Onken, 1994]
Reiner Onken (1994). Basic Requirements Concerning Man-Machine Interactions in Combat Aircraft. In: *Proceedings of the Workshop on Human Factors / Future Combat Aircraft*. 1994.

[Onken, 2002]
Reiner Onken (2002). Cognitive Co-operation for the Sake of the Human-Machine Team Effectiveness. In: *NATO RTO-Meeting Procedures MP-088, HFM-084: The Role of Humans in Intelligent and Automated Systems*, Warschau, Polen, 2002.

[Onken & Schulte, in Vorb.]
Reiner Onken & Axel Schulte (in Vorb.) *System-ergonomic Design of Cognitive Automation in Work Systems – Dual-Mode Cognitive Design of Vehicle Guidance and Control Work Systems*. Berlin: Springer.

[Parunak et al., 2003]
Van Dyke Parunak, Sven A. Brueckner & James J. Odell (2003). Swarming Coordination of Multiple UAVs for Collaborative Sensing. In: *Proceedings of the 2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations (Aerospace, Land, and Sea) Conference and Workshop*. 8.-15. September 2003.

[Pearson & Laird, 2003]
Douglas J. Pearson & John E. Laird (2003). Example-driven Diagrammatic Tools For Rapid Knowledge Acquisition. In: *Proceedings of the 2nd International Conference on Knowledge Capture* (K-CAP 2003).

[Pettersson & Doherty, 2004]
Per O. Pettersson & Patrick D. Doherty (2004). Probabilistic Roadmap Based Path Planning for an Autonomous Unmanned Aerial Vehicle. In: *Workshop on Connecting Planning and Theory with Practice*. Whistler, Juni 2004.

[Piepenburg, 1991]
Ulrich Piepenburg (1991). Ein Konzept von Kooperation und die technische Unterstützung kooperativer Prozesse. In: Horst Oberquelle (Hrsg.) *Kooperative Arbeit und Computerunterstützung – Stand und Perspektiven*. Stuttgart: Verlag für Angewandte Psychologie. S. 79-98.

[Platts et al., 2004]
Jon Platts, Erik Berglund, Martin Hagström, Petter Ögren, Isabella Panella, Simon Howell & Andy McCallum (2004). *GARTEUR Flight Mechanics Autonomy in UAVs Design Challenge*. GARTEUR FM AG14.

[Platts et al., 2007]
Jon Platts, Petter Ögren, Patrick Fabiani, Vittorio di Vito & Rüdiger Schmidt (2007). *Final Report of GARTEUR FM AG14*. GARTEUR/TP-157.

[Prévôt et al., 1995]
Thomas Prévôt, Marc Gerlach, Wilhelm Ruckdeschel, Thomas Wittig & Reiner Onken (1995). Evaluation of Intelligent On-board Pilot Assistance in In-flight Field Trials. In: *6th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems*. Massachusetts Institute of Technology. Cambridge, MA. June 1995.

[Pschyrembel, 2002]
Willibald Pschyrembel & Robert Braun (Hrsg.) (2002). *Pschyrembel Klinisches Wörterbuch*. 259. Auflage 2002. Berlin: de Gruyter.

[Puppe, 1991]
Frank Puppe (1991). *Einführung in Expertensysteme*. Berlin: Springer.

[Putzer, 2004]
Henrik Putzer (2004). *Ein uniformer Architekturansatz für kognitive Systeme und seine Umsetzung in ein operatives Framework*. Dissertation, Universität der Bundeswehr München. Berlin: Köster.

[Putzer & Onken, 2003]
Henrik Putzer & Reiner Onken (2003). COSA – a generic cognitive system architecture based on a cognitive model of human behaviour. *Cognition Technology and Work*, *5*, S. 140-151.

[QinetiQ, 2006]
Pressemitteilung QinetiQ (2006). QinetiQ completes world's first in-flight demo of multiple unmanned aircraft autonomy system. 28.11.2006. http://www.qinetiq.com/home/newsroom/news_releases_homepage/2006/4th_quarter/QinetiQ_world_first_inflight_demo_of_multiple_UAV_system.html (Letzter Zugriff: 27.02.2007).

[QinetiQ, 2007]
Pressemitteilung QinetiQ (2007). World first as fast jet pilot directs multiple unmanned aircraft. 3.4.2007. http://qinetiq.co.uk/home/newsroom/news_releases_homepage/2007/2nd_quarter/World_first_as_fast_jet_pilot_directs_multiple_unmanned_aircraft.html (Letzter Zugriff: 25.5.2007).

[Rao & Georgeff, 1995]
Anand Rao & Michael P. Georgeff (1995). BDI Agents: From Theory to Practice. In: *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*. San Francisco, CA, 12.-14. Juni 1995.

[Rasmussen, 1983]
Jens Rasmussen (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.

[Rathinam et al., 2004]
Sivakumar Rathinam, Marco Zennaro, Tony Mak und Raja Sengupta (2004). An Architecture for UAV Team Control. In: *Proceedings of the IFAC Conference on Intelligent Autonomous Vehicles*. Portugal, Juli 2004.

[Reason, 1991]
James Reason (1991). *Human Error*. Cambridge: Cambridge University Press.

[REFA, 1984]
REFA. Verband für Arbeitsstudien und Betriebsorganisation (1984). *Methodenlehre des Arbeitsstudiums*. Band 1-3. 7. Auflage. München: Carl Hanser.

[Reimer, 1991]
Ulrich Reimer (1991). *Einführung in die Wissensrepräsentation*. Teubner: Stuttgart.

[Rich & Knight, 1991]
Elaine Rich & Kevin Knight (1991). *Artificial Intelligence*. New York: McGraw-Hill.

[Ritter et al., 2002]
Frank E. Ritter, Nigel R. Shadbolt, David Elliman, Richard M. Young, Fernand Gobet & Gordon D. Baxter (2002). *Techniques for Modeling Human Performance in Synthetic Environments: A Supplementary Review*. HSIAC State of the Art Report 02-02. Erhältlich online: http://acs.ist.psu.edu/papers/SOAR-Jun03.pdf (Letzter Zugriff: 10.05.2007)

[Ritter & Kim, 2006]
Frank E. Ritter & Jong W. Kim (2006). Soar: Frequently Asked Questions List. (Version 23.6.2006). Onlineressource: http://acs.ist.psu.edu/soar-faq/soar-faq.html (Letzter Zugriff: 12.5.2007)

[Ritter & Kim, 2007]
Frank E. Ritter & Jong W. Kim (2007). ACT-R: Frequently Asked Questions List. Onlineressource: http://acs.ist.psu.edu/projects/act-r-faq/act-r-faq.html (Letzter Zugriff: 10.5.2007, Version: 12.4.2007)

[Rosenbloom et al., 1994]
Paul S. Rosenbloom, W. Lewis Johnson, Randolph M. Jones, Frank Koss, John E. Laird, Jill. F. Lehman, Robert Rubinoff, Karl B. Schwamb & Milind Tambe (1994). Intelligent Automated Agents for Tactical Air Simulation: A Progress Report. In: *Proceedings of the 4th Conference on Computer Generated Forces and Behavioral Representation*.

[von Rosenstiel, 2003]
Lutz von Rosenstiel (2003). *Grundlagen der Organisationspsychologie*. 5. Auflage. Stuttgart: Schäffer-Poeschel.

[von Rosenstiel et al., 2005]
Lutz von Rosenstiel, Walter Molt & Bruno Rüttinger (2005). *Organisationspsychologie*. 9. Auflage. Stuttgart: Kohlhammer.

[Russell & Norvig, 2003]
Stuart Russel & Peter Norvig (2003). *Artificial Intelligence – A Modern Approach*. Upper Saddle River, NJ: Prentice Hall.

[Ryan et al., 2004]
Allison Ryan, Marco Zennaro, Adam Howell, Raja Sengupta und J. Karl Hedrick (2004). An Overview of Emerging Results in Cooperative UAV Control. In: *Proceedings of the 43rd IEEE Conference on Decision and Control*. Atlantis, 14.-17. Dezember 2004.

[SAE, 1996]
SAE (1996). *Certification Considerations for Highly-Integrated Or Complex Aircraft Systems*. Dokumentennummer ARP 4754, November 1996.

[Salas et al., 1992]
Eduardo Salas, T. L. Dickinson, Sharolyn A. Converse & Scott I. Tannenbaum (1992). Toward an Understanding of Team Performance and Training. In R. W. Swezey & E. Salas (Hrsg.), *Teams: Their Training and Performance* (S. 3-29). Norwood, New Jersey: Ablex Publishing Corporation.

[Scerbo, 2005]
Mark W. Scerbo (2005). Adaptive Automation. Online-Ressource: http://www.cs.colorado.edu/~mozer/courses/6622/papers/aachpt05-12-15.htm (Letzter Zugriff: 8.3.2008).

[Schmidt, 2007]
Sebastian Schmidt (2007). *Analyse einer Hubschrauber-Schwarmoperation mittels der KP-Methode.* Diplomarbeit Universität der Bundeswehr München, Fakultät für Luft- und Raumffahrttechnik. LRT13/D/07-8.

[Schneider, 1996]
Helmut Schneider (1996). *Lexikon zu Team und Teamarbeit*. Köln: Wirtschaftsverlag Bachem.

[Schreiber et al., 2000]
Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, Nigel Shadbolt, Walter Van de Velde & Bob Wielinga (2000). *Knowledge Engineering and Management – The CommonKADS Methodology*. Cambridge, MA: MIT Press.

[Schubert & Schulte, 2001]
Andreas Schubert & Axel Schulte (2001). Automatische Generierung eines pilotengerechten Tiefflugprofils mit empirischer Modellparameteradaption für die Flugführung. In: *Deutscher Luft- und Raumfahrtkongress 2001*. Hamburg, 17.-20. September 2001.

[Schulte et al., 1996]
Axel Schulte, Jürgen Rogozik & Wolfgang Klöckner (1996). Visualizing of Planning and Decision-aiding Functions for Crew Assistance in Military Aircraft Operations. In: Jacques G. Verly (Hrsg.). *Enhanced and Synthetic Vision 1996*. SPIE Proceedings Volume 2736. S. 164-175. Orlando, FL.

[Schulte, 2006]
Axel Schulte (2006). Manned-Unmanned Missions: Chance or Challenge? In: *The Journal of the JAPCC (Joint Air Power Competence Centre)*. Kalkar. Ausgabe 3/2006.

[Schulte et al., 2008]
Axel Schulte, Claudia Meitinger & Reiner Onken (2008). Human Factors in the Guidance of Autonomous Vehicles: Oxymoron or Tautology? The Potential of Cognitive and Co-operative Operator Assistant Systems. In: *Cognition, Technology and Work.* ISSN 1435-5566.

[Schurr et al., 2005]
Nathan Schurr, Steven Okamoto, Rajiv T. Maheswaran, Paul Scerri, & Milind Tambe (2005). Evolution of a Teamwork Model. In R. Sun (Hrsg.), *Cognition*

and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation. Cambridge: Cambridge University Press.

[Scott et al., 2006]
Stacey D. Scott, Stéphane Mercier, Mary L. Cummings & Enlie Wang (2006). Assisting Interruption Recovery in Supervisory Control of Multiple UAVs. In: *Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting*. San Francisco, 16.-20. Oktober 2006.

[Searle, 1969]
John R. Searle (1969). *Speech Acts: An Essay in the Philosophy of Language.* Cambridge: Cambridge University Press.

[Shannon & Weaver, 1948]
Claude Shannon & Warren Weaver (1948). *The Mathematical Theory of Communication*. Urbana, University of Illinois Pres.

[Sheridan, 1992]
Thomas B. Sheridan (1992). *Telerobotics, Automation, and Human Supervisory Control*. Cambridge, MA: MIT Press.

[Shima et al., 2007]
Tal Shima, Steve Rasmussen & Dave Gross (2007). Assigning Micro UAVs to Task Tours in an Urban Terrain. In: *IEEE Transactions on Control Systems Technology*, *15(4)*, S. 601-612.

[Simon, 1969]
Herbert A. Simon (1969). *The Sciences of the Artificial*. Cambridge, MA: MIT Press.

[SoarTech, 2002]
SoarTechnology (2002). Soar: A Comparison with Rule-based Systems. Onlineressource: http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/SoarRBSComparison.pdf (Letzter Zugriff: 23.05.2007)

[SoarTech, 2005]
SoarTechnology (2005). TacAir-Soar. Onlineressource: http://www.soartech.com/projects/TacAir-Soar.pdf (Letzter Zugriff: 23.05.2007)

[Solso, 2005]
Robert L. Solso (2005). *Kognitive Psychologie*. Heidelberg: Springer.

[Stelling, 1999]
Dirk Stelling (1999). *Teamarbeit in Mensch-Maschine Systemen*. Göttingen: Hogrefe.

[Stone & Veloso, 1999]
Peter Stone & Manuela Veloso (1999). Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. *Artificial Intelligence*, *110(2)*, S. 241-273.

[Strohal & Onken, 1998]
Michael Strohal & Reiner Onken (1998). Intent and Error Recognition as part of a knowledge-based cockpit assistant. In: *Proceedings of SPIE – The International Society for Optical Engineering*. Orlando, Florida. 13.-16. April 1998.

[Stütz & Schulte, 2000]
Peter Stütz & Axel Schulte (2000). Evaluation of the Cockpit Assistant

Military Aircraft CAMA in flight trials. In: *3rd International Conference on Engineering Psychology and Cognitive Ergonomics*. Edinburgh, UK, 25.-27. Oktober 2000.

[Stütz et al., 2001]
Peter Stütz, Volkmar Adam & Achim Schönhoff (2001). Missionsmanagement für ein hochfliegendes, unbemanntes Luftfahrzeug im kontrollierten Luftraum: Betriebsverfahren und Avionikkonzept. In: *Deutscher Luft- und Raumfahrtkongress 2001*. Hamburg, 17.-20. September 2001.

[Stütz, 2004]
Peter Stütz (2004). Ein autonomes Missionsmanagementsystem für unbemannte Luftfahrzeuge: Konzept und Anwendung. In: *Deutscher Luft- und Raumfahrtkongress 2004*. Dresden, 20.-23. September 2004.

[Swezey & Salas, 1992]
Robert W. Swezey & Eduardo Salas (1992). Guidelines for Use in Team-Training Development. In: R.W. Swezey & E. Salas (Hrsg.) *Teams: Their Training and Performance*. Norwood, NJ: Ablex Publishing Corporation.

[Taatgen et al., 2006]
Niels Taatgen, Christian Lebiere & John Anderson (2006). Modeling Paradigms in ACT-R. In: Ron Sun (Hrsg.), *Cognition and Multi-Agent Interaction*. Cambridge: Cambridge University Press. S. 29-52.

[Tambe, 1997]
Milind Tambe (1997). Towards flexible Teamwork. *Journal of Artificial Intelligence Research*, *7*, S. 83-124.

[Taylor, 2001]
Robert M. Taylor (2001). Cognitive Cockpit Systems Engineering: Pilot Authorisation and Control of Tasks. In: *Proceedings of the 8th European Conference on Cognitive Science Approaches to Process Control (CSAPC '01)*. München, DE, 24.-26. September 2001.

[Thielecke et al., 2004]
Frank Thielecke, Jörg Dittrich & Andreas Bernatz (2004): ARTIS - Ein VTOL UAV Demonstrator. In: *Deutscher Luft- und Raumfahrtkongress 2004*, Dresden, 20.-23. September 2004.

[Tidhar et al., 1998]
Gil Tidhar, Clinton Heinze & Mario Selvestrel (1998). Flying Together: Modelling Air Mission Teams. *Applied Intelligence*, *8*. S. 195-218.

[Tidhar & Sonenberg, 2000]
Gil Tidhar & Liz Sonenberg (2000). Organized Distributed Systems (Extended Abstract). In O. Etzion & P. Scheuermann (Hrsg.), *Cooperative Information Systems – Proceedings of 7th International Conference CoopIS 2000* (S. 126-131). Berlin, Heidelberg: Springer.

[Tvaryanas et al., 2005]
Anthony P. Tvaryanas, William T. Thompson & S. H. Constable (2005). *US Military Unmanned Aerial Vehicle Mishaps: Assessment of the Role of Human Factors Using HFACS*.

[Tvaryanas & Thompson, 2006]
Anthony P. Tvaryanas & William T. Thompson (2006). Unmanned Aircraft System (UAS) Operator Error Mishaps: An Evidence-based Prioritization of Human Factors Issues. In: *NATO RTO-Meeting Proceedings HFM-135: Human Factors of Uninhabited Military Vehicles as Force Multipliers*. Biarritz, Frankreich, 9.-11. Oktober 2006.

[V-Modell XT, 2008]
V-Modell XT, Version 1.2.1.1. Erhältlich online: http://ftp.tu-clausthal.de/ftp/institute/informatik/v-modell-xt/Releases/1.2.1.1/Dokumentation/V-Modell-XT-Gesamt.pdf (Letzter Zugriff: 02.09.2008)

[Walker, 2006]
David Walker (2006). Autonomy in Action – The UAV Surrogate. In: *Proceedings of the 1st Moving Autonomy Forward Conference 2006*. Lincoln, UK, 21.-22. Juni 2006.

[Walsdorf et al., 1997]
Anton Walsdorf, Reiner Onken, Herbert Eibl, Hartmut Helmke, Reiner Suikat & Axel Schulte (1997). The Crew Assistant Military Aircraft (CAMA). In: *The Human-Electronic Crew: The Right Stuff? 4th Joint GAF/RAF/USAF Workshop on Human-Computer Teamwork*. Kreuth, September 1997.

[Walsdorf, 2002]
Anton Walsdorf (2002). *Zentrale, objektorientierte Situationsrepräsentation angewandt auf die Handlungsziele eines Cockpitassistenzsystems*. Dissertation Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik.

[Weiß & Jakob, 2005]
Gerhard Weiß & Ralf Jakob (2005). *Agentenorientierte Softwareentwicklung*. Berlin, Heidelberg: Springer.

[Wickens, 1984]
Christopher D. Wickens (1984). Engineering Psychology and Human Performance. Columbus, OH: Merrill.

[Williams, 2004]
Kevin W. Williams (2004). *A Summary of Unmanned Aircraft Accident/Incident Data: Human Factors Implications*. FAA Civil Aerospace Medical Institute, Oklahoma City, OK, USA.

[Williams, 2006]
Kevin W. Williams (2006). *Human Factors Implications of Unmanned Aircraft Accidents: Flight Control Problems*. FAA Civil Aerospace Medical Institute, Oklahoma City, OK, USA.

[Winograd & Flores, 1986]
Terry Winograd & Fernando Flores (1986). *Understanding Computers and Cognition*. Norwood, New Jersey: Ablex Publishing Corporation.

[Wooldridge, 2002]
Michael Wooldridge (2002). *An Introduction to MultiAgent Systems*. Chichester: Wiley.

[Wooldridge & Jennings, 1998]
Michael Wooldridge & Nicholas R. Jennings (1998). Pitfalls of Agent-Oriented Development. In: In K. P. Sycara & M. Wooldridge (Hrsg.). *Agents '98: Proceedings of the Second International Conference on Autonomous Agents*. ACM Press, Mai 1998.

[Wray & Jones, 2006]
Robert E. Wray & Randolph M. Jones (2006). Considering Soar as an Agent Architecture. In: Ron Sun (Hrsg.) *Cognition and Multi-Agent Interaction*. Cambridge, NY: Cambridge University Press.

[Yakir & Kaminka, 2007]
Ari Yakir & Gal Kaminka (2007). An Integrated Development Environment and Architecture for Soar-Based Agents. In: *Innovative Applications of Artificial Intelligence (IAAI-07).*

[Yen et al., 2001]
John Yen, Jianwen Yin, Thomas R. Ioerger, Michael S. Miller, Dianxiang Xu & Richard A. Volz (2001). CAST: Collaborative Agents for Simulating Teamwork. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01).*

A Modelle des a-priori Wissens

Während in Kapitel 5 das Zusammenspiel zwischen den wesentlichen Modellen der Pakete „Kommunikation“, „Kooperation“ und „Problemlösung“ erläutert wurde, sind hier tabellarisch sämtliche Klassen des implementierten Funktionsprototyps aufgelistet, um einen Eindruck von der Art und Struktur des a-priori Wissens zu vermitteln.

Modellart	***Klasse***
Wunsch	end
Handlungsalternative	means

Tabelle A-1: Modelle des Paketes "Problemlösung"

Modellart	***Klassen***
Umweltmodell	conversation-id
	dialog
	dialog*mapping
	dialog-state
	dialog-transition
	dialog-cancel : dialog
	dialog-inform : dialog
	dialog-instruct : dialog
	dialog-propose : dialog
	dialog-query : dialog
	dialog-request : dialog
	dialog-subscribe : dialog
Wunsch	continue-dialog
	abort-unnecessary-dialog
Handlungsalternative	cancel
	inform
	propose
	query
	request
	subscribe
	send-message
Anweisungsmodell	exec*send-message

Tabelle A-2: Modelle des Paketes "Kommunikation"

Modellart	***Klasse***
Umweltmodell	actor
	actor*mapping
	agenda-item
	required-time
	commitment
	commitment*mapping
	evaluation
	known-information
	mission-order
	mission-order*mapping
	resource
	resource*mapping
	required-resource
	shared-resource
	shared-resource*mapping
	assignment
	capability
	capability*mapping
	required-capability
	task
	task-fly-to : task
	task-destroy : task
	task-suppress : task
	task-protect : task
	task*mapping
	spokesman
	spokesman*mapping
	spokesman-election-number
	spokesman-election-number*mapping
	spokesman-election
	spokesman-election*mapping
	team
	team*mapping
	task-coverage
	workload
	information-type
	information-type*mapping

Tabelle A-3: Umweltmodelle des Pakets „Kooperation

Modellart	***Klasse***
Wunsch	assign-shared-resource
	avoid-redundant-task-completion
	balance-workload
	do-not-have-commitment
	ensure-task-coverage
	have-spokesman
	have-greatest-number
	know-number
	keep-agenda-achievable
	keep-others-informed
	know-team-member
	have-commitment
	sort-commitment-into-agenda
	support-team-member
	wait-for-shared-resource
Handlungsalternative	commit
	drop-commitment
	insert-commitment

Tabelle A-4: Modelle des Paketes "Kooperation"

Modellart	***Klasse***
Soar-Regeln	preserve
Umweltmodell	ucav
	ucav*mapping
	harm
	harm*mapping
	homebase
	homebase*mapping
	sam
	sam-site
	sam-site*mapping
	target
	target*mapping
	time
	corridor
	corridor*mapping
	position
	position*mapping
	distance
	flightplan
	attitude
	attack-attitude : attitude
	leave-attitude : attitude
	anti-collision-attitude : attitude
	attack-location
	threat
	covering
	on-way

Tabelle A-5: Modelle des Paketes "Umgebung"

Modellart	***Klasse***
Umweltmodell	mission-status
	escort-direction
Wunsch	be-at
	comply-with-commitment
	consider-threat-status
	have-active-flightplan
	have-attitude
	have-flightplan
	know-coordinates
	no-airborne-harm
	sam-site-active
Handlungsalternative	activate-flightplan
	determine-coordinates
	escort
	fire-weapon
	fly-to
	loiter
	plan-flightplan
	turn
Anweisungsmodell	exec-activate-flightplan
	exec-escort
	exec-fire-weapon
	exec-fly-to
	exec-loiter
	fp-comment
	exec-plan-flightplan
	exec-turn

Tabelle A-6: Modelle des Paketes "Mission"

Modellart	***Klasse***
Wunsch	priority

Tabelle A-7: Modelle des Paketes "Priorisierung"

Modellart	***Klasse***
Wunsch	avoid-collision
	avoid-flight-into-terrain
	avoid-missile
	avoid-threat
	minimise-threat
Handlungsalternative	climb
Anweisungsmodell	exec-climb

Tabelle A-8: Modelle des Paketes "Sicherheit"

Modellart	***Klasse***
Umweltmodell	instruction
	instruction*mapping
Wunsch	comply-with-instruction
Handlungsalternative	delete-instruction

Tabelle A-9: Modelle des Paketes "Überwachung"

Abkürzungsverzeichnis

ACT-R	Adaptive Control of Thought – Rational
ACU	Artificial Cognitive Unit
AV	Aerial Vehicle
BDI	Belief Desire Intention
CommonKADS	Common Knowledge Acquisition and Document Structuring
COSA	Cognitive System Architecture
CoTeSys	Cognition for Technical Systems
CPL	Cognitive Programming Language
DARPA	Defense Advanced Research Projects Agency
DFG	Deutsche Forschungsgemeinschaft
dMARS	distributed Multi-Agent Reasoning System
FIPA ACL	Foundation for Intelligent Physical Agents Agent Communication Language
FLOT	Forward Line of Own Troops
FMS	Flight Management System
GARTEUR	Group for Aeronautical Research and Technology Europe
HARM	High-Speed Anti-Radiation Missile
KP	Kognitiver Prozess
KP-Methode	Kognitive Prozess-Methode
KQML	Knowledge Query and Manipulation Language
LTM	Long Term Memory
MIDS	Multi-Functional Information Distribution System
NATO	North Atlantic Treaty Organization
NFS	Network File System
OCU	Operating ACU
PRS	Procedural Reasoning System
R/X	Receive
RPV	Remotely Piloted Vehicle
SAM	Surface to Air Missile
SCU	Supporting ACU
SEAD	Suppression of Enemy Air Defense
STEAM	Shell for Teamwork
T/X	Transmit
UAV	Uninhabited Aerial Vehicle
UCAV	Uninhabited Combat Aerial Vehicle
WASLA-HALE	Weitreichende Abbildende Signalerfassende Luftgestützte Aufklärung – High Altitude Long Endurance
WM	Working Memory