Prävention und Beseitigung von Fehlerursachen im Kontext von unbemannten Fahrzeugen

Aron Schnakenbeck*, Christoph Sieber, Luis Miguel Vieira da Silva, Felix Gehlhoff
Institut für Automatisierungstechnik
Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg
Hamburg, Deutschland
aron.schnakenbeck@hsu-hh.de

Alexander Fay
Lehrstuhl für Automatisierungstechnik
Ruhr-Universität Bochum
Bochum, Deutschland

***Zusammenfassung*—Unbemannte Fahrzeuge sind durch zunehmende Autonomie in der Lage in unterschiedlichen unbekannten Umgebungen zu operieren. Diese Flexibilität ermöglicht es ihnen Ziele eigenständig zu erfüllen und ihre Handlungen dynamisch anzupassen ohne starr vorgegebenen Steuerungscode. Allerdings erschwert ihr autonomes Verhalten die Gewährleistung von Sicherheit und Zuverlässigkeit bzw. der Verlässlichkeit, da der Einfluss eines menschlichen Bedieners zur genauen Überwachung und Verifizierung der Aktionen jedes Roboters begrenzt ist. Daher werden Methoden sowohl in der Planung als auch in der Ausführung von Missionen für unbemannte Fahrzeuge benötigt, um die Sicherheit und Zuverlässigkeit dieser Fahrzeuge zu gewährleisten. In diesem Artikel wird ein zweistufiger Ansatz vorgestellt, der eine Fehlerbeseitigung während der Missionsplanung und eine Fehlerprävention während der Missionsausführung für unbemannte Fahrzeuge sicherstellt. Die Fehlerbeseitigung basiert auf formaler Verifikation, die während der Planungsphase der Missionen angewendet wird. Die Fehlerprävention basiert auf einem regelbasierten Konzept, das während der Missionsausführung angewendet wird. Der Ansatz wird an einem Beispiel angewendet und es wird diskutiert, wie die beiden Konzepte sich ergänzen und welchen Beitrag sie zu verschiedenen Aspekten der Verlässlichkeit leisten.**

***Index Terms*—Unbemannte Fahrzeuge, GRAFCET, Model Checking, Laufzeitverifikation**

I. Einleitung

Unbemannte Fahrzeuge (UxV) werden in den unterschiedlichsten Umgebungen eingesetzt, etwa im Wasser, an Land, in der Luft oder auch im Weltraum. Ihre Vielseitigkeit in Bezug auf Fähigkeiten und Erscheinungsformen eröffnet immer weitere Anwendungsszenarien [1]. Die kontinuierliche technologische Entwicklung fördert den Einsatz von autonomen UxV, die ein zunehmendes Maß an autonomem Verhalten aufweisen. Autonome UxVs zeichnen sich dadurch aus, dass sie in teilweise oder vollständig unbekannten Umgebungen operieren können. Dies ermöglicht den Einsatz von unbemannten Bodenfahrzeugen (UGV) oder unbemannten Luftfahrzeugen (UAV). Diese führen keinen streng vordefinierten Steuerungscode aus, sondern haben die Fähigkeit selbstbestimmt zu erkennen, zu handeln und zu reagieren, um vorgegebene Ziele zu erreichen. Besonders vielversprechend ist die Kombination mehrerer heterogener autonomer UxVs zu einem Verbund. Einzelne Fahrzeuge können sich gegenseitig ergänzen und Schwächen kompensieren, sodass komplexere Missionen ausgeführt werden können [2]. Selbst für größere Verbünde ermöglicht diese Autonomie die Unabhängigkeit von einem menschlichen Bediener und reduziert den Bedarf an Kontrolle und Überwachung auf ein Minimum.

Ein wichtiger Aspekt zur Sicherstellung der Autonomie eines Systems (hier autonome UxVs) ist dessen Verlässlichkeit (engl. *dependability*), sodass Autonomie nur mit verlässlichen Robotern erreicht werden kann. Die Verlässlichkeit eines Systems wird von Avizienis et al. [3] als Fähigkeit definiert, einen Serviceausfall zu vermeiden, der häufiger und schwerwiegender auftritt, als hinzunehmen ist. Da die Autonomie autonomer UxVs möglichst wenig menschliche Eingriffe und Überwachung vorsieht, können nicht alle auftretenden Fehler durch menschliches Eingreifen verhindert werden, und es kann nicht auf alle auftretenden Fehler durch einen menschlichen Bediener reagiert werden. Daher muss die akzeptable Fehlerhäufigkeit für autonome UxVs als extrem niedrig eingestuft werden, was die Bedeutung der Verlässlichkeit erhöht.

Avizienis et al. [3] definieren die Attribute Verfügbarkeit, Zuverlässigkeit (im Sinne von engl. *reliability*), Sicherheit (im Sinne von engl. *safety*), Wartbarkeit und Integrität zur Erreichung der Verlässlichkeit. Die Attribute *Verfügbarkeit* und *Zuverlässigkeit* sind notwendig, damit ein UxV wirklich autonom sein kann. Wenn das UxV nicht betriebsbereit oder in der Lage ist, einen korrekten Dienst fortzusetzen, kann keine Autonomie erreicht werden. *Sicherheit* muss von autonomen UxV selbst gewährleistet werden. Das Fehlen von menschlichen Eingriffen, die im Zweifel schwerwiegende Fehler verhindern, muss von den autonomen UxV selbst kompensiert werden. Eine weitere Anforderung an autonome UxVs ist deren *Wartbarkeit*. Reagiert ein autonomes UxV auf Umweltveränderungen, bspw. mit Systemanpassungen, so muss sichergestellt werden, dass das UxV Wartungen auf autonome Weise durchführen kann. Allerdings muss auch sichergestellt werden, dass die Änderungen durch solche Wartungen nicht zu neuen Fehlerursachen führen und die *Integrität* des Systems verletzen.

Zur Sicherstellung der Vielzahl dieser Attribute, und damit der Sicherstellung der Verlässlichkeit, bedarf es mehrerer, teils sehr verschiedener Methoden. Fokus dieser Arbeit sind die Aspekte der Zuverlässigkeit und Sicherheit, für deren Sicher-

stellung zwei unabhängige Konzepte vorgeschlagen werden: eine Methode zur Fehlerbeseitigung während der Planungsphase und eine Methode zur Fehlerprävention während der Ausführungsphase einer Mission.

Nach der Beschreibung von Grundlagen zur Missionsplanung für autonome UxVs in Abschnitt II wird in Abschnitt III-A ein formaler Verifikationsansatz vorgestellt. Dieser ermöglicht eine Verifikation von Missionsplänen während der Planungsphase und gegebenenfalls die Beseitigung von gefundenen Fehlerursachen aus den Plänen. Die Missionspläne werden in die Spezifikationssprache GRAFCET transformiert, um sie mittels statischer Analysen und Model Checking untersuchen zu können. Abschnitt III-B präsentiert ein Konzept, um sicherzustellen, dass autonome UxVs während der anschließenden Missionsausführung trotz minimaler menschlicher Überwachung gegebene Sicherheitsvorgaben einhalten, um so die Fehlerprävention sicherzustellen. Dieser zweistufige Ansatz ist notwendig, da autonome UxVs in einer dynamischen Umgebung operieren und nicht alle möglichen Ereignisse in der Planungsphase abgedeckt werden können. Der Ansatz wird in Abschnitt IV auf eine exemplarische Mission angewendet, die von zwei autonomen UxVs ausgeführt wird, bevor in Abschnitt V diskutiert wird, wie die beiden vorgestellten Verifikationskonzepte zusammenhängen und welchen Beitrag die Ansätze zur Erreichung der Verlässlichkeit leisten.

II. Grundlagen

Dieser Beitrag betrachtet autonome UxV, die sich selbstständig bewegen und in einer unbekannten und unkontrollierbaren Umgebung geeignete Aktionen wählen müssen [4]. Der gemeinsame Einsatz von autonomen UxV in einem Verbund ermöglicht die Durchführung einer Vielzahl komplexer *Szenarien*. Im Kontext dieses Artikels beschreibt ein Szenario eine Konstellation von Bedingungen und Umständen, wie die aktuell verfügbaren UxV, sowie ein übergeordnetes Ziel, das erreicht werden soll [5]. Um das übergeordnete Ziel eines Szenarios zu erreichen, werden *Missionen* verwendet. Eine Mission besteht aus einer Abfolge von einem oder mehreren *Missionskommandos* und den erforderlichen Parametern eines Missionskommandos, die einem einzelnen UxV zugewiesen werden. Zum Beispiel kann eine Mission eines autonomen UxV nur aus einem Missionskommando `Bewegung` mit den Parametern `pos_x` und `pos_y` der Zielposition in x- und y-Koordinaten sowie einem Parameter `vel` der Geschwindigkeit, mit der das UxV das Ziel ansteuern soll, bestehen. Diese Missionen werden aus einem Szenario abgeleitet [5].

Das Ableiten oder Planen von Missionen für einen Verbund autonomer UxV für ein bestimmtes Szenario kann sehr komplex werden und zu einer Vielzahl möglicher Lösungen führen, sodass eine automatisierte Planung erstrebenswert ist. Automatisierte Planung wurde im Bereich der KI-Planung viele Jahre lang untersucht, mit dem Ziel, eine Abfolge von Aktionen (im Kontext dieser Arbeit Missionskommandos) zu finden, die von einem Anfangszustand zu einem gewünschten Zielzustand führen [6]. Die am weitesten verbreitete Sprache im Bereich der KI-Planung ist die *Planning Domain Definition Language* (PDDL) [7]. PDDL ist eine domänenunabhängige Sprache, die zur Beschreibung von Planungsproblemen verwendet wird, indem sowohl die Domäne als auch das Problem separat beschrieben werden. Die Domäne beschreibt hauptsächlich die von jeder Ressource bereitgestellten Aktionen mit ihren Vorbedingungen und Wirkungen. Das Problem definiert den Anfangszustand sowie den Zielzustand [7]. Zur Lösung eines solchen Planungsproblems werden Planer eingesetzt, die unter anderem Satisfiability Modulo Theories (SMT) verwenden, indem sie die Planungsprobleme als Erfüllbarkeitsprobleme in SMT formulieren und dann Solver verwenden, um sie zu lösen [8]. Wenn alle Gleichungen durch Zuweisung von Werten zu den Variablen erfüllt werden können, existiert ein Plan, der vom Startzustand zum Endzustand führt [8]. Eine Herausforderung bei der Umsetzung solcher Ansätze mit PDDL in der KI-Planung ist der Aufwand, der für die Erstellung eines solchen Planungsproblems erforderlich ist. Ein weiteres Problem ist, dass der Einsatz von KI-Planung in realen Anwendungen selten ist, da die Ausdruckskraft von PDDL nicht ausreicht [9].

Darüber hinaus gibt es Ansätze, die eine automatisierte Planung auf Basis von Informationsmodellen anstreben. Ansätze wie in [10] konzentrieren sich darauf, Funktionen von autonomen UxV formal zu beschreiben, um die folgenden zwei Aspekte anzugehen: Einerseits kann die Heterogenität verschiedener UxV in einem Verbund überwunden werden und einzelne UxV in einem Verbund können aufgrund des Informationsmodells bei Bedarf ausgetauscht werden. Andererseits kann die automatisierte Planung erleichtert werden. Solche Modelle sind komplex und deren Erstellung zeitaufwendig, sodass Ansätze zur automatischen Erstellung eines solchen Modells wie in [11] vorteilhaft sind. Es mangelt jedoch noch an Ansätzen, die eine automatisierte Planung basierend auf einem Informationsmodell durchführen. Bisher lag der Fokus auf dem Informationsmodell und Methoden zur Generierung des Modells, wobei formales Schließen oder KI-Planung für die automatisierte Planung vorgeschlagen wurden. Erste Ansätze in diese Richtung werden beispielsweise in [12] vorgestellt, bei denen ein Planungsproblem als Erfüllbarkeitsproblem in SMT aus einem Fähigkeitsmodell automatisch generiert und anschließend gelöst wird.

Aufgrund der Komplexität der Erstellung von Planungsproblemen einerseits und andererseits des Mangels an geeigneten Mitteln für die automatisierte Planung basierend auf formalen Modellen werden Missionen für autonome UxV oft noch manuell geplant. Sowohl manuelle als auch automatisierte Planung führt immer zu einem *Plan*, wie in Listing 1 gezeigt. Grundsätzlich umfasst ein Plan eine Abfolge von Aktionen mit den entsprechenden Parametern und stellt eine mögliche Lösung für ein definiertes Problem dar, um das zuvor festgelegte Ziel zu erreichen. Ein Plan besteht aus verschiedenen Zeitpunkten. Zu jedem Zeitpunkt können eine oder mehrere Aktionen aufgelistet werden, die zu diesem Zeitpunkt ausgeführt werden sollen. Wenn mehrere Aktionen einem Zeitpunkt zugeordnet sind, wie in Listing 1 für den Zeitpunkt `1` gezeigt, werden diese Aktionen parallel ausgeführt. Für jede

Aktion werden die erforderlichen Parameter angegeben.

Listing 1: Allgemeine Struktur eines Plans bestehend aus verschiedenen Zeitpunkten und entsprechenden Aktionen mit den erforderlichen Parametern.

```
0: Aktion_x Par_a
1: Aktion_b Par_x Par_c Par_f
   Aktion_c Par_d
2: Aktion_a Par_b
```

Missionen werden verwendet, um die Informationen aus einem Plan auf die einzelnen autonomen UxV zu übertragen. Missionskommandos entsprechen den Aktionen und sind eindeutig autonomen UxV zugeordnet, sodass eine Mission eines einzelnen autonomen UxV aus seinen Missionskommandos in der Reihenfolge besteht, in der sie im Plan vorkommen. *Bedingungen* sind erforderlich, wenn ein autonomes UxV ein Missionskommando erst nach Ausführung eines bestimmten Missionskommandos durch ein anderes UxV ausführen soll. Um fehlerhafte Pläne und damit Fehler in der Missionsausführung zu vermeiden, müssen solche Pläne in der Planungsphase verifiziert werden, was insbesondere für manuell erstellte Pläne gilt. Jedoch müssen auch automatisch erstellte Pläne je nach gewähltem Ansatz verifiziert werden, da beispielsweise komplexe Einschränkungen nicht immer in PDDL ausgedrückt werden können. Darüber hinaus müssen selbst korrekte Pläne für autonome UxV zur Laufzeit weiter verifiziert werden. Pläne beschreiben nur einen groben Ablauf, da autonome UxV in einer unbekannten Umgebung arbeiten. Das bedeutet, dass ihr tatsächliches Verhalten leicht vom Plan abweichen oder ihn erweitern kann, ohne den Zweck des Plans zu gefährden. Autonome UxV müssen auf ihre Umgebung reagieren und daher beispielsweise ihren Weg zu einem Wegpunkt selbstständig bestimmen, indem sie Hindernisse durch Anpassung der Koordinaten umgehen. Dementsprechend müssen zwei Dinge sichergestellt werden:

1) Pläne müssen formal verifiziert werden, bevor sie an die UxV übergeben werden, um Fehler während der Ausführung im Voraus zu verhindern.
2) In der Ausführungsphase sind die UxV vielen Dynamiken ausgesetzt, daher muss ihr tatsächliches Verhalten auch während der Ausführung verifiziert werden.

III. Verifizierungsmethoden

Dieser zweistufige Ansatz wird im Folgenden vorgestellt.

A. Beseitigung von Fehlerursachen vor Missionsausführung

Um sicherzustellen, dass die Pläne fehlerfrei sind, wird formale Verifikation als Methode zur Beseitigung von Fehlerursachen eingesetzt. Die Beseitigung von Fehlerursachen wird von Avizienis et al. [3] als Mittel zur Reduzierung der Anzahl und Schwere von Fehlerursachen beschreiben. Luckcuck et al. [13] präsentieren eine Übersicht, wie formale Methoden im Kontext von autonomen UxV eingesetzt werden. Unter anderem werden Ansätze untersucht, die formale Methoden während des Entwurfs von Missionsplänen nutzen, welche später von Verbünden ausgeführt werden, wie z.B. in [14]–[16]. Diese Ansätze konzentrieren sich jedoch mehr auf die Erstellung der Spezifikation, anstatt formale Verifikation zur Sicherstellung von Anforderungen einzusetzen.

Um eine formale Verifikation zu ermöglichen, müssen die Pläne in einer formalen Sprache modelliert werden. In dieser Arbeit wird GRAFCET [17] als graphische Modellierungssprache genutzt. GRAFCET wurde ursprünglich für die Modellierung von Steuerungsverhalten im Bereich der industriellen Fertigung entwickelt. GRAFCET ist für die Modellierung von Plänen geeignet, da gleichzeitig ablaufende Missionen über nebenläufige Sequenzen und Abhängigkeiten zwischen den Missionen über interne Variablen modelliert werden können. Zudem ist GRAFCET durch die graphische Repräsentation leicht verständlich und in der Automatisierung weit verbreitet. In der Vergangenheit haben einige Autoren dieses Beitrages bereits Methoden zur Verifikation von GRAFCET untersucht [18], [19], die im Folgenden auf Missionsplänen angewendet werden. Ein erster Schritt ist die Transformation der Missionspläne in GRAFCET. Anschließend werden im GRAFCET-Modell gewünschte Verhaltenseigenschaften verifiziert. Die zu verifizierenden Eigenschaften müssten dazu formalisiert werden, bei der Anwendung von Model Checking beispielsweise in einer temporalen Logik wie Computation Tree Logic (CTL) [20]. Nach Verifikation der Missionspläne können diese von den UxVs ausgeführt werden.

1) Transformation von Missionsplänen in GRAFCET: Die Regeln zur Transformation eines Plans in ein entsprechendes GRAFCET-Modell sind in Abbildung 1 dargestellt. Gestrichelte Linien stellen Platzhalterelemente dar, die mittels einer anderen Transformationsregel aus Abbildung 1 erzeugt werden. Für jede Mission im Plan wird eine neue Sequenz von Schritten (grafisch dargestellt durch ein Quadrat) in dem GRAFCET-Modell generiert, beginnend mit einem Anfangsschritt (doppelt umrandetes Quadrat) und endend mit einer Schlusstransition (grafisch dargestellt durch eine horizontale Linie). Für jeden Missionsbefehl wird ein Schritt in die Sequenz eingefügt, geordnet nach dem Zeitpunkt, an dem er im Plan auftritt. Während ein Schritt des GRAFCET-Modells aktiv ist, simuliert eine sogenannte Aktion die Ausführung eines Missionsbefehls `<command>`. Wenn das UxV den Befehl beendet hat, setzt es das Signal `<commandFinished>` auf *true*, der Schritt wird deaktiviert, und der nächste Schritt wird aktiviert, in dem der nächste Missionsbefehl ausgeführt wird.

Für jeden Missionsbefehl, der eine Bedingung (`<command_m>` in Abb. 1) enthält, wird eine interne boolesche Variable (`<condVar>`) eingeführt, die anzeigt, ob die Bedingung erfüllt ist. Die vorgelagerte Transitionsbedingung des Schritts, der dem jeweiligen Missionsbefehl entspricht, wird mithilfe eines $\wedge$-Operators um diese Variable erweitert. Die entsprechende Variable wird auf *true* gesetzt, nachdem der Befehl `<command_n>` abgeschlossen wurde, der die Bedingung erfüllt. Dies geschieht mit Hilfe einer sogenannten speichernd wirkenden Aktion, die `<conVar>` auf *true* setzt, wenn der zugehörige Schritt deaktiviert wird (dargestellt durch den Pfeil). Die Anwendung der Transformationsregeln führt zu einem

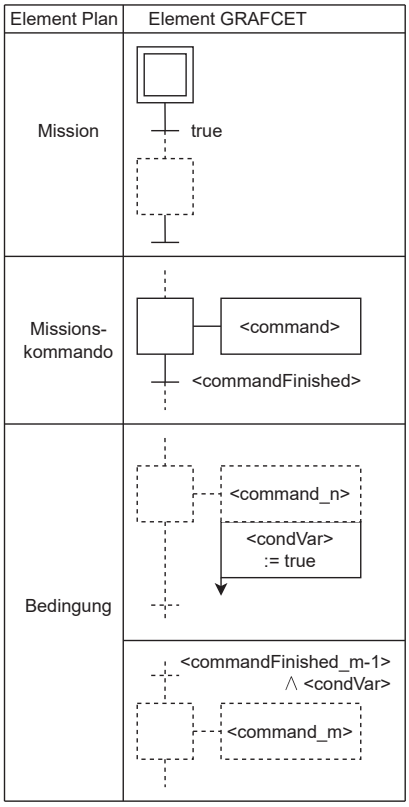

Abbildung 1: Regeln zur Transformation eines Plans (linke Seite) in ein GRAFCET-Modell (rechten Seite).

GRAFCET-Modell, das eine formale Darstellung des Plans ist und verifiziert werden kann.

2) Eigenschaften von Missionsplänen: Um die GRAFCET-Modelle zu verifizieren, müssen Eigenschaften definiert und vom Anwender verifiziert werden:

- Strukturelle Anforderungen: Es könnte verlangt werden, dass Missionen verschiedene Arten von Befehlen enthalten, wie das Greifen eines Objekts, das Fahren zu einem Ort oder das Ablegen eines Objekts. Ebenso könnte gefordert werden, dass Missionen mit einem Startbefehl beginnen und mit einem Stoppbefehl enden. Diese Eigenschaften werden mit Hilfe einer graphbasierten Analyse überprüft, die eine Tiefensuche nach Schritten durchführt, die solchen Missionsbefehlen entsprechen.
- Chronologische Reihenfolge der Befehle: Einige Arten von Befehlen hängen logisch von anderen ab, z. B. muss auf ein Greifen das Ablegen eines Objektes folgen. Diese Eigenschaften können mit Hilfe von temporalen Logiken formalisiert und mittels Model Checking [18] verifiziert werden. Die Formalisierung solcher Eigenschaften in temporalen Logiken erfordert ein höheres Maß an Fachwissen. Da jedoch jede Mission aus modularen und wiederverwendbaren Missionsbefehlen besteht, ist es möglich, eine bibliotheksartige Liste von Eigenschaften zu definieren, die wiederverwendet werden können. Eine solche Befehlsfolge kann in CTL als $AG(\psi_1 \rightarrow AF\psi_2)$ formalisiert werden, was bedeutet, dass ψ_2 zu einem Zeitpunkt nach ψ_1 eintreten muss.
- Freiheit von Deadlocks: Wenn zwei UxVs gegenseitig auf die Beendigung einer bestimmten Aufgabe warten, um fortzufahren, könnte dies zu einem Deadlock führen. Mögliche Deadlocks können ebenfalls durch Model Checking erkannt werden.
- Abwesenheit von sicherheitskritischen Situationen: Es muss sichergestellt werden, dass bestimmte Befehle nicht gleichzeitig ausgeführt werden können, z. B. dass UxVs nicht zur gleichen Zeit den gleichen Gegenstand greifen oder dass sie nicht gleichzeitig zum selben Ort fahren. Da die UxVs die Befehle selbständig ausführen, hängt ihr genaues Verhalten von den Implementierungsdetails ab. Solche Situationen können zu einem Deadlock führen, z. B. wenn zwei UxVs versuchen, einen Zielort zu erreichen, aber gleichzeitig eine Kollision vermeiden wollen. Um diese Art von Situationen auf Missionsplanebene zu erkennen, können die entsprechenden Eigenschaften entweder mit Hilfe von Model Checking analysiert werden, oder es kann eine statische Analyse verwendet werden, wie in [19] vorgeschlagen.
  Für das Model Checking können diese Situationen mithilfe von Invarianten nachgewiesen werden. Für einen Beispielplan, der zwei Missionsbefehle enthält, die ein Greifen desselben Objekts vorsehen, muss sichergestellt werden, dass die korrespondierenden Schritte im GRAFCET-Modell nicht gleichzeitig aktiv sein können: $AG\neg(step\_1 \wedge step\_2)$.

B. Fehlerprävention während der Missionsausführung

Nach Avizienis et al. [3] umfasst Fehlerprävention zum einen die Vermeidung der Entstehung von Fehlern und zum anderen die Vermeidung der weiteren Auswirkungen bestehender Fehler. Der im vorangegangenen Kapitel erstellte und verifizierte Missionsplan räumt den einzelnen autonomen UxVs bewusst maximale Handlungsfreiheit ein. Dies hat zur Folge, dass die Missionsausführung nicht vollständig vorhersehbar und daher nur bedingt kontrollierbar ist. Erschwert wird dies zusätzlich durch unbekannte und sich ändernde Umgebungen, z. B. ist die Lage sicherheitsrelevanter Gebiete im Voraus nicht bekannt und kann sich zudem verändern. Sind solche Gebiete außerdem nicht direkt relevant für das Missionsziel, ist es möglich, dass ein autonomes UxV, trotz korrekter Wahrnehmung des Gebietes, zugehörige Informationen nicht verarbeitet. Wird z. B. ein Sperrgebiet lediglich wahrgenommen, ohne daraufhin den eigenen Pfad entsprechend anzupassen, kann dies zu unsicherem Verhalten führen. Daher ist es ratsam, den UxV parallel zum Missionsplan auch Missionsauflagen zu übermitteln. Diese Auflagen legen verbotene, hier sicherheitskritische, Verhaltensweisen fest. Eine missionsunspezifische Formulierung erleichtert die Wiederverwendung von Auflagen [21]. In [22] stellen die Autoren einen regelbasierten Ansatz zur Laufzeitverifikation vor, mit dem sie unsicheres Verhalten von unbemannten Luftfahrzeugen (UAV) erkennen konnten. Es wurden jedoch nur Sicherheitsverstöße erkannt und gemeldet,

die bereits aufgetreten waren. Eine Meldung veranlasste dann einen menschlichen Nutzer Gegenmaßnahmen einzuleiten.

Das hier vorgestellte Konzept zur missionsbegleitenden Fehlerprävention greift den Ansatz von [22] auf und erweitert ihn um eine einerseits präventiv erkennende und andererseits selbstständig reagierende Komponente. Um die Entstehung eines Fehlers mit Hilfe eines regelbasierten Systems wirksam zu verhindern, sind vier Schritte notwendig: (I) Formulierung des Fehlers, (II) Formulierung des drohenden Fehlers, (III) Fähigkeit, den Fehler aus der aktuellen Situation vorherzusagen, (IV) Gegenreaktion auf den drohenden Fehler. Diese vier Schritte werden im Folgenden anhand eines einfachen Beispiels erläutert. Zu diesem Zweck werden ein unbemanntes Bodenfahrzeug (UGV) und ein Sperrgebiet betrachtet. Das UGV darf das Sperrgebiet nicht befahren. Der zugehörige Fehler wird wie folgt formuliert: WENN die Position des UGV im Sperrgebiet liegt, DANN liegt ein Fehler vor (I). Die aktuelle Position des UGV ist somit die Fehlerursache. Ein drohender Fehler kann nun durch die Betrachtung der bevorstehenden Position formuliert werden. WENN die bevorstehende Position des UGV im Sperrgebiet liegt, DANN liegt ein drohender Fehler vor (II). Durch die bewusste Verwendung der Variable *bevorstehende Position* innerhalb der Regel kann der zugehörige Wert auf mehrere Arten ermittelt werden. Im besten Fall sind dem UGV bereits Informationen über zukünftige Ziele und Wegpunkte bekannt. Im ungünstigsten Fall muss die bevorstehende Position anhand der aktuellen Position, des Kurses und der Geschwindigkeit für einen zu

Abbildung 2: Situation des betrachteten Anwendungsfalls mit a) UGV, b) UAV, c) Paket, d) Fabrikhalle und e) Zielposition.

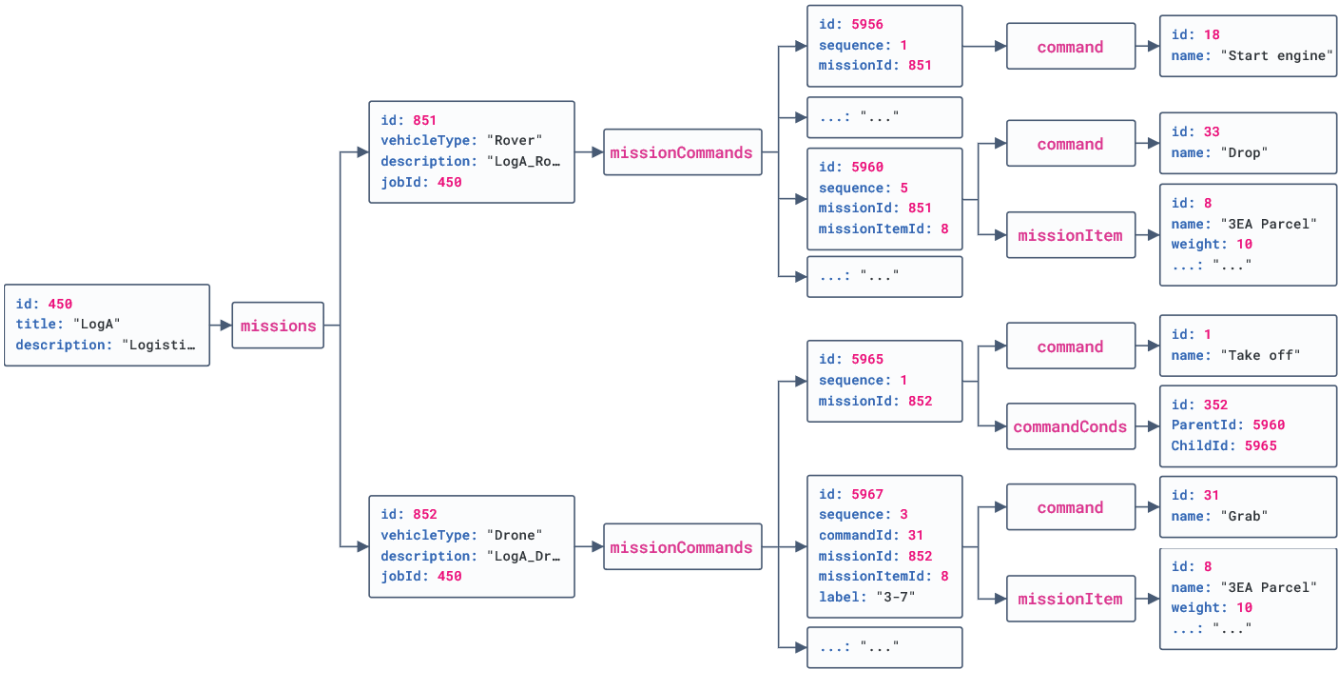

Abbildung 3: Auszug des Missionsplans im JSON-Format für das Szenario in Abbildung. 2.

wählenden Zeithorizont berechnet werden, z. B. mit einer separaten Regel (III). Der Zeithorizont sollte so festgelegt werden, dass eine angemessene Reaktionszeit sowie Zeit für eine Gegenreaktion (z. B. Abbremsen) berücksichtigt wird. Allerdings kann aufgrund der relativ langen Reaktionszeit des Menschen trotz rechtzeitiger Benachrichtigung ein Fehler auftreten, bevor eine Gegenmaßnahme eingeleitet wurde. Daher kann es sinnvoll sein, am Ende der in (II) formulierten Regel bereits eine Gegenreaktion zu definieren: WENN die bevorstehende Position des UGV im Sperrgebiet liegt, DANN liegt ein drohender Fehler vor UND das UGV reduziert seine Geschwindigkeit um 50 Prozent (IV). Die hier exemplarisch vorgeschlagene Geschwindigkeitsreduzierung ermöglicht es dem UGV, einen alternativen Pfad zu bestimmen. Wenn auch dieser Pfad das Sperrgebiet kreuzt, wird die Geschwindigkeit so lange weiter reduziert, bis das UGV beim Einfahren in das Gebiet zum Stillstand kommt. Es ist zweckmäßig, dass in [22] vorgeschlagene Konzept der Benachrichtigung des menschlichen Nutzers auch bei automatischen Gegenreaktionen beizubehalten, um auch ein menschliches Eingreifen grundsätzlich zu ermöglichen.

IV. Umsetzung und Implementierung

In diesem Abschnitt wird das Potenzial des Ansatzes und das Zusammenspiel der beiden vorgestellten Konzepte anhand eines einfachen Anwendungsfalles demonstriert. Die entsprechende Implementierung wurde mit der in [5] vorgestellten, ROS2-basierten Systemarchitektur durchgeführt. Diese ermöglicht die Trennung von Missionsplanung und -ausführung, ohne sich im Detail mit dem Aufbau, der Sensorik oder Aktorik der einzelnen UxVs auseinandersetzen zu müssen.

Im gewählten Anwendungsfall wird ein UxV-Verbund, bestehend aus einem UGV und einem UAV, verwendet, um den Transport eines Pakets durchzuführen. Dieses Paket befindet sich zunächst an einer Startposition in einer Fabrikhalle. Sein Zielort ist von einem Zaun umgeben. Da weder das UGV noch das UAV allein in der Lage sind, das Paket zu transportieren, muss das Paket außerhalb des eingezäunten Bereichs übergeben werden. Abbildung 2 zeigt die Ausgangssituation des Anwendungsfalls. Das UGV befindet sich neben dem Paket. Außerhalb befindet sich das UAV neben der Zielposition.

Das erwartete Verhalten des Verbunds besteht darin, den Transport des Pakets selbstständig durchzuführen. Der Missionsplan, der dieses Verhalten repräsentiert, wurde manuell über eine Weboberfläche geplant, die den Plan in einem JSON-Format speichert. Die resultierende JSON-Datei für das Beispielszenario ist auszugsweise in Abb. 3 in einer Graphdarstellung gezeigt. Der Plan umfasst zwei Missionen: Eine für das UGV (oben) und eine für das UAV (unten). Jede Mission besteht aus einer ähnlichen Abfolge von mehreren Missionsbefehlen: *start/takeOff* $\rightarrow$ *drive/flyTo* $\rightarrow$ *grab* $\rightarrow$ *drive/flyTo* $\rightarrow$ *drop* $\rightarrow$ *drive/flyTo* $\rightarrow$ *stop/land*. Da das Paket jedoch zuerst vom UGV transportiert werden muss, wartet das UAV, bis das UGV das Paket an einer Übergabeposition abgesetzt hat. Daher wird dem *takeOff*-Befehl der UAV eine

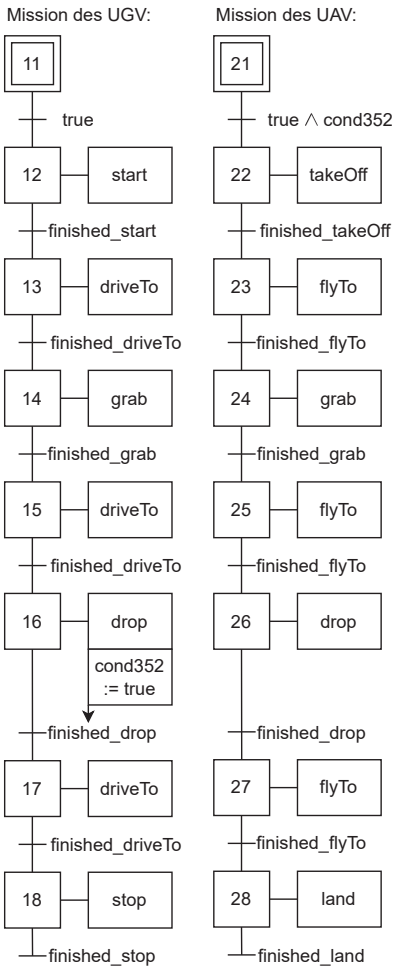

Abbildung 4: GRAFCET-Modell des Plans für den Anwendungsfall.

Befehlsbedingung hinzugefügt: Das UAV kann erst abheben, wenn das UGV das Paket abgelegt hat.

A. Verifikation der Missionspläne

Für die Verifikation des Plans während der Entwurfsphase wird der in Abschnitt III-A vorgestellte Ansatz auf den erstellten Plan angewendet, der in Abbildung 3 dargestellt ist. Die Anwendung der GRAFCET-Transformationsregeln resultiert in einem GRAFCET-Modell, das in Abbildung 4 dargestellt ist.

Exemplarisch werden zwei Eigenschaften verifiziert: Erstens muss sichergestellt werden, dass der Plan nicht erfordert, dass sich die UxVs näher als in einem bestimmten Sicherheitsabstand zueinander bewegen. Die Formalisierung dieser Eigenschaften kann durch Informationen aus der JSON-Datei aus Abbildung 3 unterstützt werden, in der die Zielkoordinaten z. B. für die *driveTo* und *flyTo* Befehle gespeichert sind. Wenn die Zielkoordinaten für Befehle verschiedener UxVs übereinstimmen, dürfen die entsprechenden Schritte im GRAFCET-Modell nicht gleichzeitig aktiv sein. Für das GRAFCET-Modell in Abbildung 4 bedeutet dies, dass die

Schritte 13, 14 nicht gleichzeitig mit den Schritten 23, 24 aktiv sein dürfen. In CTL kann das mit den Invarianten $AG\neg(step\_13 \wedge step\_23)$, $AG\neg(step\_14 \wedge step\_23)$, usw. formuliert werden. Zweitens muss sichergestellt werden, dass der Plan durch die Modellierung von Bedingungen keine Verklemmung hervorruft. Für jede induzierte Bedingungsvariable (in diesem Fall nur *cond352*) kann mit Hilfe der CTL-Formel $EF(cond352 == true)$ überprüft werden, ob sie schließlich erfüllt ist. Wie in [18] vorgestellt, wurde das GRAFCET-Modell in ein Transitionssystem überführt und mit dem Model Checker ITS-Tools[1] verifiziert, wodurch die formalisierten Eigenschaften nachgewiesen werden konnten.

B. Laufzeitverifikation der Missionsausführung

Die mit GRAFCET verifizierten und fehlerfreien Missionspläne können nun in ihrer ursprünglichen Form (vgl. Abb. 3) an UGV und UAV übermittelt werden. Sie gewähren den UxVs maximale Handlungsfreiheit bei der Ausführung ihrer Missionen. Wie in der Einleitung erwähnt, muss aufgrund der Autonomie der UxVs sichergestellt werden, dass während der Missionsausführung keine Fehler auftreten. Exemplarisch wird hier die Gefahr durch das Eindringen in Sperrgebiete als Fehlerursache betrachtet. Insbesondere im Zusammenhang mit UAVs werden Sperrgebiete (ugs. Geofence) zur Einhaltung von Abständen, z. B. zu Objekten und Einrichtungen, verwendet [23]. Im Kontext autonomer UxV ist es nicht zweckmäßig, feste Pfade festzulegen, um Sperrgebiete zu vermeiden. Stattdessen vermeiden autonome UxV diese Sperrgebiete effektiv, indem sie ihre Pfade selbstständig bestimmen. Abbildung 2 zeigt ein Sperrgebiet, hervorgehoben durch Pylonen. Es werden zwei einfache regelbasierte Auflagen erstellt, um sicherzustellen, dass weder das UGV noch das UAV in dieses Gebiet eindringen. Analog zu dem Ansatz in [22] werden diese Regeln innerhalb einer Ontologie mit der SWRL-Regelsprache [24] modelliert und zur Laufzeit ausgewertet. Tabelle I und Tabelle II zeigen zwei SWRL-Regeln. Regel 1 in Tabelle I erkennt vorausschauend, ob das UGV in das Sperrgebiet einfährt, und reduziert die Geschwindigkeit, um gegebenenfalls eine Neuplanung des Pfades zu ermöglichen. Wenn das UGV dennoch in das Sperrgebiet einfährt, weil es seine Geschwindigkeit nicht rechtzeitig weit genug reduziert hat oder keine Gegenreaktion erfolgte, wird es durch Regel 2 in Tabelle II gestoppt, indem seine Geschwindigkeit auf null gesetzt wird.

Für Regel 1 verlangt SWRL, dass die aktuelle Geschwindigkeit *?velocity* bereits in Zeile 5 in der Prämisse der Regel abgefragt wird, damit die neue Geschwindigkeit *?newvelocity* in der Konklusion berechnet werden kann. Da Regel 2 die Geschwindigkeit nicht berechnet, sondern fest zuweist, ist hier eine vorherige Abfrage nicht notwendig. Das UAV erhält ähnliche Regeln, die die Geschwindigkeit reduzieren oder einen Schwebeflug (engl. Loiter) bewirken. Während der Ausführung der Missionen aktualisieren beide UxVs laufend ihre jeweilige Wissensbasis und analysieren die Einhaltung ihrer jeweiligen Regeln.

[1] https://lip6.github.io/ITSTools-web/

Tabelle I: Die Regel R1 verlangsamt das UGV, wenn es in ein Sperrgebiet einfahren wird.

Line No.	SWRL-Atom
1	UGV(?myUGV)
2	^ hasImpendingPosition(?myUGV, ?position)
3	^ RestrictedArea(?RA)
4	^ iswithin(?position, ?RA)
5	^ Velocity(?myUGV, ?velocity)
6	→ ImpendingFault(?myUGV, “The UGV may enter a restricted area”)
7	^ Velocity(?myUGV, ?newvelocity) ^ swrlb:multiply(0.5 , ?velocity, newvelocity)

Tabelle II: Die Regel R2 bewirkt, dass das UGV anhält, sobald es sich im Sperrgebiet befindet.

Line No.	SWRL-Atom
1	UGV(?myUGV)
2	^ hasPosition(?myUGV, ?position)
3	^ RestrictedArea(?RA)
4	^ iswithin(?position, ?RA)
5	→ Fault(?myUGV, “The UGV is within a restricted area”)
6	^ Velocity(?myUGV, 0.0)

Durch die Kombination der beiden Konzepte wird sichergestellt, dass der Packstücktransport kohärent strukturiert ist und keine sicherheitsrelevanten Störungen bei der Durchführung des Einsatzes auftreten. Das UGV holt das Paket zunächst aus der Werkshalle und transportiert es zur Übergabeposition. Die Sicherheitsauflagen verhindern einen geradlinigen Transport durch das Sperrgebiet. Sobald das Paket an der Übergabeposition abgeladen ist, startet das UAV und nähert sich, um das Paket zu übernehmen, während das UGV zurückkehrt. Abbildung 5 zeigt diese Szene im Rahmen der Mission. Nachdem das UAV das Paket am Zielort abgeladen hat, ist die Mission erfolgreich beendet.

V. Fazit

In diesem Artikel wurden Aspekte der Verlässlichkeit für autonome UxV untersucht. Aus der angestrebten Freiheit in der Missionsausführung und der Vielfalt von UxV ergeben sich besondere Anforderungen an den Umgang mit Fehlern. Diese Dualität von Selbstständigkeit einerseits und der Notwendigkeit der wirksamen Kontrolle andererseits erschwert die Verifikation sowohl im Rahmen der Missionsplanung als auch der Missionsausführung. Die Teilaspekte der Verlässlichkeit, insbesondere Zuverlässigkeit und Sicherheit, wurden mit Konzepten zur Fehlerbeseitigung und Fehlerprävention behandelt. Dennoch sind weder die Fehlerbeseitigung vor der Mission noch die Fehlerprävention während der Mission allein in der Lage, fehlerfreie Missionen zu gewährleisten. Die Verifikation der Missionspläne basiert nur auf unvollständigen und statischen Annahmen über die Umgebungsbedingungen, sodass trotz der funktional logischen Abfolge der Missionen deren Erfolg nicht garantiert ist. Die Verifikation der Missionsausführung berücksichtigt dagegen stärker die Umwelt und die aktuellen Bedingungen. Die damit verbundenen Regeln

Abbildung 5: Übergabe des c) Pakets. a) UGV kehrt in die Fabrikhalle zurück und b) UAV nähert sich der Übergabeposition.

sind jedoch nicht darauf ausgelegt, den Erfolg der Mission zu gewährleisten.

Doch selbst die Kombination der beiden Konzepte erfüllt die Aspekte der Verlässlichkeit nur bedingt. Diese erfordert weitere Überlegungen zu den anderen, nicht angesprochenen Aspekten der Verlässlichkeit, wie z. B. der Wartbarkeit, wie in [3] definiert. Die Autoren bewerten eine einzelne, allumfassende Methode zur Gewährleistung der Verlässlichkeit für unzweckmäßig und empfehlen einen modularen, kombinierbaren Ansatz zur Verifizierung verschiedener Aspekte.

In dem gezeigten Anwendungsfall ergänzen sich beide vorgestellte Konzepte, sodass die Mission erfolgreich durchgeführt werden kann. Die Autoren befürworten die Entwicklung weiterer Konzepte, insbesondere für Fehlertoleranz und Fehlervorhersage, wie sie in [3] definiert sind. Wenn solche Konzepte auf modularer Basis entwickelt werden, können sie leicht zu größeren Konstrukten kombiniert werden, was einen ganzheitlichen Ansatz für die Verlässlichkeit von autonomen UxVs erleichtert.

Danksagung

Diese Forschungsarbeit aus dem Projekt RIVA wird durch dtec.bw – Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr gefördert. dtec.bw wird von der Europäischen Union – NextGenerationEU finanziert.

Literatur

[1] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, "Artificial Intelligence for Long-Term Robot Autonomy: A Survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4023–4030, 2018.

[2] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–31, 2019.

[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[4] ISO 8373:2021(E), "Robotics – Vocabulary," 2021.

[5] C. Sieber, L. M. Vieira da Silva, A. Fay, T. Brogt, G. Strobel, S. Berkowitz, and L. Zembrot, "A Universal Approach to Command and Control Heterogeneous Autonomous Robots," *dtec.bw-Beiträge der Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg: Forschungsaktivitäten im Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr dtec.bw*, p. 276–280, 2022.

[6] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*, 4th ed., ser. Pearson Series in Artificial Intelligence. Hoboken: Pearson, 2021.

[7] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson *et al.*, "PDDL— The Planning Domain Definition Language," Technical report, Tech. Rep., 1998.

[8] M. Cashmore, D. Magazzeni, and P. Zehtabi, "Planning for Hybrid Systems via Satisfiability Modulo Theories," *Journal of Artificial Intelligence Research*, vol. 67, pp. 235–283, 2020.

[9] A. Rogalla, A. Fay, and O. Niggemann, "Improved Domain Modeling for Realistic Automated Planning and Scheduling in Discrete Manufacturing," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2018, pp. 464–471.

[10] L. M. Vieira da Silva, A. Köcher, and A. Fay, "A capability and skill model for heterogeneous autonomous robots," *at - Automatisierungstechnik*, vol. 71, no. 2, pp. 140–150, 2023.

[11] L. M. Vieira da Silva, A. Köcher, P. Topalis, and A. Fay, "A Python Framework for Robot Skill Development and Automated Generation of Semantic Descriptions," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2023.

[12] A. Köcher, L. M. Vieira da Silva, and A. Fay, "Automated Process Planning Based on a Semantic Capability Model and SMT." [Online]. Available: http://arxiv.org/pdf/2312.08801

[13] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal Specification and Verification of Autonomous Robotic Systems: A Survey," *ACM Comput. Surv.*, vol. 52, no. 5, 2019.

[14] M. Kloetzer, X. C. Ding, and C. Belta, "Multi-robot deployment from LTL specifications with reduced communication," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 4867–4872.

[15] V. Hilaire, P. Gruer, A. Koukam, and O. Simonin, "Formal specification approach of role dynamics in agent organisations: Application to the satisfaction-altruism model," *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, no. 05, pp. 615–641, 2007.

[16] K. Talamadupula, G. Briggs, T. Chakraborti, M. Scheutz, and S. Kambhampati, "Coordination in human-robot teams using mental modeling and plan recognition," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2957–2962.

[17] IEC International Electrotechnical Commission, "Grafcet specification language for sequential function charts," IEC 60848, 2013-02.

[18] R. Mroß, A. Schnakenbeck, M. Völker, A. Fay, and S. Kowalewski, "Transformation of GRAFCET Into GAL for Verification Purposes Based on a Detailed Meta-Model," *IEEE Access*, vol. 10, pp. 125 652–125 665, 2022.

[19] A. Schnakenbeck, R. Mroß, M. Völker, S. Kowalewski, and A. Fay, "A Control Flow based Static Analysis of GRAFCET using Abstract Interpretation," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–7.

[20] C. Baier and J.-P. Katoen, *Principles of Model Checking*, ser. The MIT Press. London, England: MIT Press, Apr. 2008.

[21] C. Sieber, L. M. Vieira da Silva, and A. Fay, *Agilität durch Auflagen – Unterstützung der Missionsplanung für autonome Roboter*. VDI Verlag, 01 2023, pp. 691–704.

[22] C. Sieber, L. M. Vieira da Silva, K. Grünhagen, and A. Fay, "Rule-Based Verification of Autonomous Unmanned Aerial Vehicles," *Drones*, vol. 8, no. 1, 2024.

[23] M. N. Stevens and E. M. Atkins, "Geofencing in immediate reaches airspace for unmanned aircraft system traffic management," in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 2140.

[24] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean *et al.*, "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Member submission*, vol. 21, no. 79, pp. 1–31, 2004.