http://dx.doi.org/10.25673/116041

SysML-Profil zur automatisierten Generierung von PDDL-Beschreibungen aus Systemmodellen

Hamied Nabizada [1], Lasse Beers [1], Maximilian Weigand [1], Felix Gehlhoff [1] und Alexander Fay [2]

Abstract: In diesem Beitrag wird die Entwicklung eines dedizierten SysML-Profils vorgestellt, das die Einbettung von Planning Domain Definition Language (PDDL) Konstrukten direkt in SysML-Modelle ermöglicht. Dieser Ansatz unterstützt die automatisierte Prozessablaufplanung, indem es eine Schnittstelle zwischen Systemmodellierung und Planungsalgorithmen schafft. Durch die Anwendung des Profils in einem Anwendungsfall aus dem Flugzeugbau wird gezeigt, wie Planungsbeschreibungen effizient erstellt und für Optimierungsprozesse in PDDL-Solvern genutzt werden können.

Keywords: AI Planning, Modellbasierte Systementwicklung, PDDL, SysML

1 Einleitung

Die modellbasierte Systementwicklung (MBSE) hat sich als Alternative zur traditionellen, dokumentenbasierten Systementwicklung etabliert, indem sie den Einsatz von Modellen zur detaillierten und durchgängigen Beschreibung von Produktionssystemen fördert [HS21]. Durch diese einheitliche Datenbasis wird eine effiziente Zusammenarbeit der beteiligten Gewerke im Entwicklungsprozess geschaffen [SS20].

Methoden der MBSE, wie z.B. die Systems Modeling Language (SysML)-gestützten Modellierungs-Vorgehensweisen von [We16] oder [Ei15], bieten strukturierte Ansätze zur Erstellung von Systemmodellen. Diese strukturierten Vorgehensweisen reduzieren die Heterogenität der Inhalte und der Darstellung einzelner Teilmodelle des Systems und fördern somit die Wiederverwendbarkeit und Vergleichbarkeit der erstellten Modelle [Es07]. Allerdings erfordern diese Methoden oft viele manuelle Schritte, wie z.B. die Zuordnung von vielen einzelnen Prozessschritten zu möglichen technischen Ressourcen, die diese Prozesse umsetzen können, welches ein hohes Maß an Expertenwissen benötigt. Dies trifft insbesondere zu, wenn verschiedene Systemvarianten verglichen werden sollen, um die ideale Konfiguration für spezifische Optimierungsziele zu bestimmen. Bei der Analyse verschiedener Systemvarianten wird sowohl die Gesamtstruktur als auch das Verhalten

1 Helmut-Schmidt-Universität / Universität der Bundeswehr, Institut für Automatisierungstechnik, Holstenhofweg 85, 22043 Hamburg, Deutschland, hamied.nabizada@hsu-hh.de, https://orcid.org/0000-0001-8251-837X; lasse.beers@hsu-hh.de, https://orcid.org/0000-0003-0946-7742; maximilian.weigand@hsu-hh.de, https://orcid.org/0009-0000-1602-4873; felix.gehlhoff@hsu-hh.de, https://orcid.org/0000-0002-8383-5323

2 Ruhr-Universität Bochum, Lehrstuhl für Automatisierungstechnik, Universitätsstrasse 150, 44801 Bochum, Deutschland, alexander.fay@rub.de, https://orcid.org/0000-0002-1922-654X

der Subsysteme betrachtet. Die beiden Aspekte, die Gesamtstruktur des Systems und das Verhalten der einzelnen Subsysteme, müssen in Beziehung zueinander gesetzt werden, um untersuchen zu können, wie Änderungen in einem Teil des Systems das Verhalten des Gesamtsystems beeinflussen. Zum Beispiel kann eine Variante eines Robotik-Systems, die über mehrere parallel arbeitende Endeffektoren verfügt, einen Produktionsprozess schneller durchführen als eine Variante mit nur einem Endeffektor. Allerdings sind bei der erstgenannten Variante Leerlaufzeiten der zusätzlichen Endeffektoren zu erwarten. Sollte in einem Optimierungsszenario die Maximierung der Ressourcenauslastung angestrebt werden, so würde der zusätzliche Endeffektor diesem Ziel widersprechen.

Im Rahmen der Systementwicklung muss ermittelt werden, welche Systemvariante am besten geeignet ist, um eine festgelegte Menge von Prozessen unter Berücksichtigung von Prozessbedingungen effizient durchzuführen [Tö17]. Dafür kann beispielsweise die Durchlaufzeit herangezogen werden. Um diese berechnen zu können, wird ein Prozessablaufplan benötigt, der die genaue Abfolge der auszuführenden Aktionen definiert und diese den entsprechenden Subsystemen zuordnet. Eine Automatisierung dieser Prozessablaufplanung ist erstrebenswert, um den zeitlichen Aufwand für die Bewertung verschiedener Systemvarianten zu reduzieren.

Um diese Automatisierung zu ermöglichen, ist es notwendig spezifische Planungsaspekte in die Systemmodellierung zu integrieren. Planungsaspekte beziehen sich auf die Definition von Aktionen, die zur Erreichung spezifischer Ziele innerhalb des modellierten Systems notwendig sind. Diese Aspekte sind zentral im Forschungsbereich des *AI Plannings*, der sich mit der Entwicklung von Algorithmen und Sprachen zur Lösung von Planungsproblemen beschäftigt [GNT16]. Aus diesem Forschungsbereich stammt die *Planning Domain Definition Language (PDDL)*, die sich als de-facto Standard für die Beschreibung von Planugsproblemen etabliert hat [Ma22b]. Daher wird in diesem Beitrag ein speziell entwickeltes *SysML*-Profil für PDDL vorgestellt. Dieses Profil ermöglicht es, PDDL-spezifischen Konstrukte direkt in SysML-Modelle zu integrieren und somit die Automatisierung der Prozessplanung in der Systementwicklung unterstützt.

In Abschnitt 2 werden zunächst die technischen Grundlagen dargelegt und eine Übersicht über verwandte Arbeiten gegeben. Das entwickelte SysML-Profil für PDDL wird in Abschnitt 3 vorgestellt. Anschließend erfolgt in Abschnitt 4 die Anwendung dieses Profils auf die Strukturmontage im Flugzeugbau. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick in Abschnitt 5 ab.

2 Grundlagen und verwandte Arbeiten

2.1 SysML und Profilmechanismus

Die Unified Modeling Language (UML) ist eine von der Object Management Group (OMG) standardisierte grafische Modellierungssprache, die ursprünglich für die Softwareentwick-

lung entwickelt wurde. Sie bietet eine breite Vielfalt an Elementen und Diagrammen, um verschiedene Aspekte eines Systems zu modellieren [OM19]. Diese Elemente und Diagramme ermöglichen die grafische Darstellung der Struktur, des Verhaltens und der Interaktion eines Softwaresystems.

Der UML-Profilmechanismus, ein zentraler Bestandteil des UML-Metamodells, ermöglicht die Erweiterung der UML mit spracheigenen Mitteln. Auf diese Weise kann die Sprache durch die Integration domänenspezifischer Konstrukte und Notationen an die spezifischen Anforderungen bestimmter Systeme angepasst werden [We08]. Das bekannteste Beispiel für ein UML-Profil ist die Sprache SysML. Sie wurde speziell für die Modellierung komplexer Systeme entwickelt und hat sich mittlerweile als führende Modellierungssprache im Bereich der MBSE etabliert [Ma22a]. Der Profilmechanismus bietet jedoch auch die Möglichkeit zur individuellen Erweiterung der Sprache. Ein Profil definiert dabei Stereotypen, Tagged Values, Derived Properties, Constraints und weitere Konstrukte, die zur Erfüllung der spezifischen Anforderungen notwendig sind [Se15].

2.2 Planning Domain Definition Language (PDDL)

Die Planning Domain Definition Language (PDDL) [Ha19] ist eine standardisierte Sprache, die speziell für die Beschreibung von Planungsproblemen im Bereich der symbolischen KI entwickelt wurde. Sie strukturiert die Problemstellung, indem sie zwischen *Domänen* (**Domain**) und *Problemen* (**Problem**) unterscheidet: Die Domäne stellt das strukturelle Gerüst für Plaungsaufgaben bereit, indem sie allgemeine Definitionen enthält, die über einzelne Planungsprobleme hinweg gültig sind. Ein *Problem* hingegen konkretisiert die Planungsaufgabe innerhalb einer bestimmten Domäne. Es legt konkrete Objekte fest, beschreibt den Anfangszustand und definiert den gewünschten Zielzustand. Dadurch wird die Domäne auf eine spezifische Situation angewendet, die gelöst werden soll.

Die Kernelemente einer Domänenbeschreibung umfassen u.a. Objekttypen, Prädikate und Aktionen. *Typen* (**Type**) dienen zur strukturierten Organisation von Objekten in der Planungsdomäne und helfen bei der klaren Zuordnung von Eigenschaften und Aktionen zu spezifischen Objektklassen. *Aktionen* (**Action**) sind zentrale Bestandteile in PDDL und beschreiben mögliche Handlungen innerhalb der Planungsdomäne. Jede Aktion definiert spezifische *Parameter*, die variabel sind und die beteiligten Objekte spezifizieren, sowie *Vorbedingungen*, die erfüllt sein müssen, bevor die Aktion ausgeführt werden kann. Die *Effekte* beschreiben die Veränderungen im Zustand der Planungsdomäne nach der Ausführung der Aktion. Dies ermöglicht es, aus einer Reihe von Aktionen Pläne zu erstellen, die auf das Erreichen eines gewünschten Zielzustands ausgerichtet sind. *Prädikate* (**Predicate**) in PDDL dienen dazu, spezifische Bedingungen oder Zustände in der Planungsdomäne zu beschreiben. Sie sind wesentliche Elemente zur Formulierung von Vorbedingungen und Effekten von Aktionen. Ein Prädikat kann als eine Art Aussage verstanden werden, die entweder wahr oder falsch ist und sich auf die Objekte oder Zustände bezieht, die durch die Aktionen beeinflusst werden. Durch die Verwendung von Prädikaten können

Planungssysteme logische Zusammenhänge innerhalb der Domäne überprüfen und entscheiden, ob bestimmte Aktionen unter den gegebenen Umständen zulässig und wirksam sind. *Funktionen* (**Function**) speichern numerische Informationen, die auch quantitative Zustände wie Kosten oder Distanzen darstellen können. Diese werden genutzt, um quantitative Vorbedingungen für Aktionen zu definieren und deren Effekte quantitativ zu bewerten, was die Grundlage für die Optimierung von Plänen nach spezifischen Kriterien schafft.[3]

2.3 Verwandte Arbeiten

In [HVC13] wird eine SysML-Taxonomie für Montageaufgaben vorgestellt, die zur Beschreibung von Systemfähigkeiten herangezogen wird. Diese Systemfähigkeiten bilden die Basis für die Ableitung von PDDL-Aktionen. Der Beitrag stellt zwar eine PDDL-Beschreibung zur Steuerung eines Robotiksystems in Montageumgebungen vor, jedoch erfolgt die Erstellung der PDDL-Beschreibung manuell und ist auf die Taxonomie beschränkt.

Viera et al. präsentieren in [Vi23] einen fähigkeitsbasierten Ansatz zur Generierung von PDDL-Beschreibungen. Auf Basis einer Ontologie werden benötigte Fähigkeiten zur Herstellung von Produkten und angebotene Fähigkeiten von Produktionsressourcen zugeordnet, woraus dann PDDL-Beschreibungen automatisch generiert und verarbeitet werden. Dieser Ansatz erfordert jedoch ein spezifisches Fähigkeitsmodell und beruht nicht auf einem Systemmodell aus der MBSE.

In [Ri21] wird ein Ansatz vorgestellt, um Planungsprobleme in Hierarchical Domain Definition Language (HDDL), einem Dialekt von PDDL, zu vereinfachen. Dafür vergleichen die Autoren zunächst HDDL und SysML Elemente und leiten anschließend einen Workflow für die Modellierung von HDDL Modellen ab. Die Beschreibung erfolgt jedoch konzeptionell und eine Transformation der Elemente muss manuell durchgeführt werden.

Der Ansatz von Batarseh und McGinnis [BM12] nutzt ein SysML-Profil zur automatisierten Generierung von Simulationsmodellen für Arena, basierend auf SysML-Systemmodellen. Dieser Ansatz zielt darauf ab, die Modellierungskomplexität zu reduzieren und die Effizienz zu steigern. Planungsprobleme werden jedoch nicht addressiert.

In [Wa19] wird ein Ansatz beschrieben, der das Modell eines Fertigungssystems automatisiert in einen Produktionsplan umwandelt. Dieser Ansatz nutzt dafür die ISA-95 und generiert daraus automatisiert PDDL-Beschreibungen. Daher ist dieser Ansatz speziell auf Fertigunsgssysteme ausgelegt, die nach der ISA-95 modelliert sind. Daher ist sowohl eine Anwendbarkeit auf Systeme, die nicht nach dieser Norm modelliert sind, sowie eine Integration in SysML-basierte Systemmodelle nicht möglich.

Die Modellierung von PDDL-Domänenbeschreibungen wird häufig als besonders herausfordernd, zeitaufwändig und fehleranfällig betrachtet [Li23]. Dennoch bietet keiner der

3 Eine vollständige Erläuterung der Planungssprache kann [Ha19] entnommen werden.

bestehenden Ansätze eine direkte Integration von Planungslogiken in SysML-basierte Systemmodelle, die in der Regel detaillierte Informationen über das System bereitstellen und somit eine wertvolle Basis für PDDL-Domänenbeschreibungen liefern können. Daher zeigt dieser Beitrag, wie entsprechende Elemente aus dem Systemmodell mittels eines SysML-Profils für PDDL mit planungsrelevanten Informationen annotiert werden können, um eine automatisierte Generierung von PDDL-Domänen zu ermöglichen.

3 SysML-Profil für PDDL

Die Integration von PDDL in das SysML-Umfeld erfordert die Entwicklung eines dedizierten SysML-Profils, das die spezifischen Konstrukte und Zusammenhänge von PDDL abbildet. Als Grundlage für dieses Profil dient die Backus-Naur-Form (BNF)-Definition von PDDL 3.1 [Ko11], durch die die Kompatibilität mit dem PDDL-Standard sichergestellt werden kann. Wie in [BW13] dargestellt wird, ist die automatisierte Überführung der BNF-Definition in ein Metamodell, wie beispielsweise das Profil, jedoch nicht ohne Weiteres möglich, da die BNF-Definition nicht ausreichend ist, um die syntaktischen und semantischen Unterschiede zwischen den beiden Modellierungssprachen zu überbrücken. Daher erfolgte die Entwicklung des Profils manuell, stets unter Einhaltung der BNF-Definition von PDDL 3.1, um die Kernkonstrukte von PDDL nach SysML zu übertragen.

Besonders für die Beschreibung der PDDL-Domäne liefert ein Systemmodell die erforderlichen Informationen. Die Daten für die Problembeschreibung stammen üblicherweise aus einem Produktmodell, das nicht direkt Teil des Systemmodells ist. Das Profil kann sowohl die Domänen- als auch die Problembeschreibung darstellen, vorausgesetzt, das Produktmodell wird in die Systemmodellierungsumgebung integriert. Dieser Beitrag fokussiert auf die Darstellung der Domänenbeschreibung.

Die zentralen Konstrukte der Domänenbeschreibung umfassen Aktionen, Prädikate, Funktionen und Typen, die als Grundlage für die Planungsaufgaben in PDDL dienen. Jedes dieser Elemente wurde im Rahmen des SysML-Profils als Stereotyp modelliert. Stereotypen erlauben die Erweiterung bestehender UML-/SysML-Metamodellelemente bzw. UML-/SysML-Metaklassen um zusätzliche Eigenschaften, die für PDDL-spezifische Zusammenhänge notwendig sind. Ein Auszug dieser Stereotypen für die Beschreibung von PDDL-Domänen wird in Tabelle 1 aufgelistet und den entsprechenden UML-/SysML-Metaklassen zugeordnet.[4]

In PDDL definiert die **Domain** die Umgebung und die Regeln, innerhalb derer die Planung stattfindet. Dies umfasst spezifische Aktionen, Typen und Bedingungen, die in Planungsprozessen benötigt werden. Für die Modellierung in SysML wird das entsprechende Element durch den Stereotyp `<<PDDL_Domain>>` repräsentiert, welches die UML-Metaklassen `Package` oder `Model` erweitert. Indem der übergeordnete Ordner des Systemmodells als PDDL-Domain

4 Das vollständige Profil ist zur weiteren Einsicht und Nutzung verfügbar unter `https://github.com/hsu-aut/SysML-Profile-PDDL`.

Tab. 1: Zuordnung von PDDL-Elementen zu UML-/SysML-Metaklassen

PDDL-Element	PDDL-Stereotyp	UML-/SysML-Metaklassen
Domain	<<PDDL_Domain>>	Model, Package
Type	<<PDDL_Type>>	Class
Predicate	<<PDDL_Predicate>>	ObjectFlow, ControlFlow
Function	<<PDDL_Function>>	ObjectFlow, ControlFlow
Action	<<PDDL_Action>>	CallBehaviorAction

stereotypisiert wird, ist es möglich die PDDL-Domain aus dem Systemmodell abzuleiten, da die planungsspezifische Struktur und deren Logik sowie benötigte Attribute Teil des Systemsmodells sind. Dafür werden sogenannte derivedProperties verwendet, welche auf die weiteren PDDL-Domain Elemente wie **Type** oder **Action** verweisen.

Ein **Type** in PDDL definiert eine Klasse von Objekten, ähnlich wie Datentypen in Programmiersprachen verwendet werden. Mit dem entsprechenden Stereotyp <<PDDL_Type>> wird die UML-Metaklasse Class erweitert, da diese genutzt wird, um strukturelle Aspekte des Systemmodells abzubilden, beispielsweise die Komponenten des Systems. Durch die Erweiterung der UML-Metaklasse Class können Objekteigenschaften modelliert sowie Typenhierachien definiert werden. Solche Hierachien erlauben es, Objekte in Über- und Untertypen zu strukturieren, wodurch Subtyp-Strukturen innerhalb des Systems abgebildet und verwaltet werden können.

Eine **Action** beschreibt in PDDL eine spezifische Aktion, die ausgeführt werden kann, um den Zustand des Planungssystems zu verändern. In Systemmodellen wird in der Regel ein Aktivitätsdiagramm verwendet, um das Verhalten des Systems zu beschreiben [Be24]. Daher erweitert der Stereotyp <<PDDL_Action>>, welcher in Abbildung 1 dargestellt ist, die UML-Metaklasse CallBehaviorAction, die in bestehenden Systemmodellen zur Abbildung von Prozessen bzw. Systemfunktionen genutzt wird. Jede Aktion in PDDL definiert außerdem die notwendigen Vorbedingungen, die erfüllt sein müssen, bevor die Aktion ausgeführt werden kann, den Effekt, welcher aus der Aktion resultiert, sowie Parameter in Form von typisierten Variablen, welche in den Vorbedingungen und Effekten verwendet werden.

Im Profil werden diese Zusammenhänge durch die Erweiterung des Stereotyps mit Hilfe eines Customization Elements umgesetzt. Innerhalb des Customization Elements ist es möglich derivedProperty hinzuzufügen. Ein derivedProperty ist eine Eigenschaft, dessen Wert durch seine Beziehung zu anderen Elementen definiert ist. Beispielsweise verweist das derivedProperty Precondition auf alle Object- und ControlFlows mit den Stereotypen <<PDDL_Predicate>> oder <<PDDL_Function>>, welche in ein Element mit dem Stereotypen <<PDDL_Action>> eingehen, und das derivedProperty <<PDDL_Effect>> auf alle Object- und Controlflows, die aus dem <<PDDL_Action>> Element herausgehen. Auf diese Weise können die einzelnen Elemente des PDDL-Profils miteinander verknüpft und in Beziehung gesetzt werden.

Es ist ebenso möglich, die Nutzung von Stereotypen einzuschränken. So wird bspw. die `possibleOwner` Eigenschaft des `Customization` Elements verwendet, um die Nutzung des <<`PDDL_Action`>> Stereotyps einzuschränken. In diesem Fall darf der Stereotyp nur innerhalb der <<`PDDL_Domain`>>, `Packages` sowie `Actvities` genutzt werden. Dies dient als Richtlinie für den Anwender und verhindert die Nutzung des Elements an einer falschen Stelle.

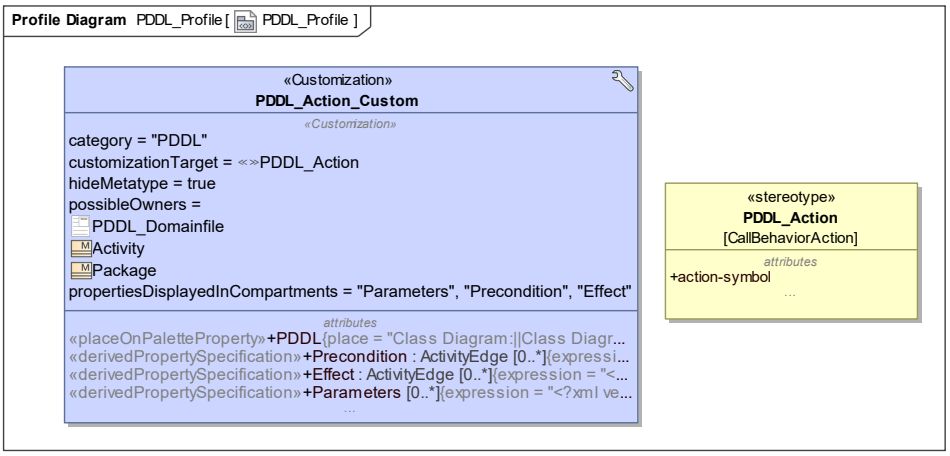

Abb. 1: <<`PDDL_Action`>> inklusive dem `Customization`-Element

In PDDL wird ein **Predicate** verwendet, um Bedingungen oder Zustände innerhalb einer Planungsdomäne zu beschreiben, die entweder zutreffen oder nicht zutreffen. Diese Prädikate werden für die Definition der Vorbedingungen und Effekte von Aktionen genutzt. Der entsprechende Stereotyp <<`PDDL_Predicate`>> erweitert die UML-Metaklassen `ObjectFlow` und `ControlFlow`. Sie werden genutzt, um Prädikate als Vorbedingungen oder Effekte für PDDL-Aktionen zu modellieren.

Im Gegensatz dazu repräsentiert eine **Function** in PDDL einen numerischen Wert. Dieses Element wird verwendet, um eine Vorbedingung oder einen Effekt einer Aktion abbilden zu können, entsprechend erweitert der <<`PDDL_Function`>> dieselben Metaklassen wie der Stereotyp <<`PDDL_Predicate`>>.

Das vorgestellte Profil ermöglicht eine PDDL-spezifische Erweiterung bestehender SysML-Sytemmodelle. Durch die Verwendung von `derivedProperties`, um die Zusammenhänge der einzelnen PDDL Elemente dynamisch zu erfassen, führen Änderungen an das Systemmodell, sofern sie mit dem PDDL-Profil annotiert sind, automatisch auch zu Änderungen der entsprechenden PDDL-Elemente. Dies sichert die Konsistenz zwischen der Systementwicklung und der Planungslogik.

Das so angereicherte Systemmodell bildet die Grundlage für die automatisierte Generierung von PDDL-Beschreibungen. Mit diesem angereicherten Systemmodell können die relevan-

ten Planungsinformationen extrahiert und einem Algorithmus zur Generierung übergeben werden. Die Beschreibung der Methode zur systematischen Anreicherung von Systemmodellen sowie ein Algorithmus zur Generierung von PDDL-Beschreibungen basierend auf den angereicherten Systemmodellen werden in separaten Veröffentlichungen beschrieben.

4 Anwendungsfall im Flugzeugbau

Zur Validierung des entwickelten Profils wurde ein Anwendungsfall aus der Strukturmontage im Flugzeugbau herangezogen, bei dem ein verfahrbarer UR10-Roboterarm innerhalb des Flugzeugrumpfs positioniert ist. Dieses Robotik-System wird eingesetzt, um verschiedene Kollare auf Niete zu drehen, wobei jeder Kollartyp einen spezifischen Endeffektor benötigt.[5]

Im Rahmen des Anwendungsfalles sollte ein Prozessablaufplan zur Optimierung der Durchlaufzeit identifiziert werden. Dabei war zu berücksichtigen, dass eine Umkonfiguration der Hardware erforderlich ist, sobald ein anderer Endeffektor für das Aufdrehen der Kollare benötigt wird.

In Abbildung 2 wird ein Ausschnitt des Aktivitätsdiagramms aus dem angereicherten Systemmodell gezeigt, der das Zusammenspiel zwischen dem modellierten Systemverhalten und dem PDDL-Profil veranschaulicht.

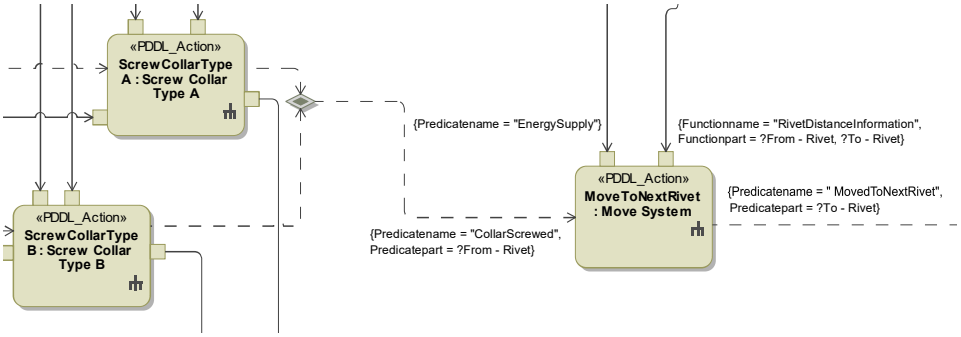

Abb. 2: Ausschnitt aus dem angereicherten Systemmodell

Der Ausschnitt zeigt drei <<PDDL_Action>>-Elemente sowie die dazugehörigen Vorbedingungen und Effekte. Die <<PDDL_Action>>-Elemente „ScrewCollarType A“ bzw. „ScrewCollarType B“ beschreiben den Prozessschritt für das Aufdrehen der Kollare. Sobald dieser Prozessschritt abgeschlossen ist, bewegt sich das System in der <<PDDL_Action>> „MoveToNextRivet“ zum nächsten Niet. Diese Schritte werden zyklisch vielfach durchlaufen.

Um die Aktion „MoveToNextRivet “ auszuführen, die den Endeffektor von einem Niet zum nächsten führt, fließen folgende Vorbedingungen in die Aktion ein:

5 Eine Vorgehensweise zur Erstellung des zugehörigen Systemmodells ist in [Be23] dargelegt.

1. Das System muss mit Energie versorgt sein. Dafür fließt das Prädikat „EnergySupply“ in die Aktion hinein.
2. Der Kollar, an dem sich der Endeffektor aktuell befindet, muss bereits verschraubt sein. Dafür sind die Aktionen „ScrewCollarTypeA“ bzw. „ScrewCollarTypeB“ zuständig, die das Prädikat „CollarScrewed“ auf „True“ setzen, sobald die Aktion abgeschlossen ist.
3. Die Koordinaten der Niete und die damit verbundene Distanz zwischen den Nieten ist in der `Function` „RivetDistanceInformation“ definiert und wird von dort aus an die MoveToNextRivet `<<PDDL_Action>>` übergeben. Ein PDDL-Solver kann auf der Basis der Distanzen und Niettypen den Niet identifizieren, welcher als nächstes bearbeitet werden sollte.

Sobald die Aktion durchgeführt wurde, hat sie als Effekt, dass der Endeffektor sich auf einer neuen Arbeitsposition befindet. Das entsprechende Prädikat „MovedToNextRivet“ wird mit der neuen Nietposition auf „True“ gesetzt.

Als Modellierungssoftware wurde im Anwendungsfall Magic Systems of Systems Architect (MSoSA) für die Erstellung des Systemmodells sowie für die Erstellung des Profils verwendet. Das Systemmodell wurde anschließend innerhalb dieser Software mit dem Profil angereichert. Durch die Anreicherung des Systemmodells kann es systematisch nach Planungsinformationen abgefragt werden. Zur automatisierten Generierung der Planungsbeschreibungen kann beispielsweise der *Report Wizard* in MSoSA verwendet werden, der auf die Velocity Engine aufbaut. Dafür werden Templates in der Velocity Template Language (VTL) benötigt. Im Rahmen des Anwendungsfalls wurden entsprechende VTL-Templates für die PDDL-Domain und PDDL-Problem geschrieben. Diese Templates beinhalten die syntaktische Struktur von PDDL mit entsprechenden Platzhaltern, die durch die Engine gefüllt werden. In List. 1 wird ein Ausschnitt der damit erzeugten Domänenbeschreibung gezeigt, der die zuvor beschriebene Aktion aus Abbildung 2 in PDDL-Syntax zeigt.

```
(:action MoveToNextRivet
    :parameters (?From - Rivet ?To - Rivet)
    :precondition (and
                    (CollarScrewed ?From)
                    (EnergySupply)
                  )
    :effect (and
                (MovedToNextRivet ?To )
                (not (CollarScrewed ?From ))
                (increase (total-cost) (RivetDistanceInformation ?From ?To))
            )
)
```

List. 1: PDDL-Syntax für die Aktion "MoveToNextRivet"

Anschließend kann diese standardisierte Beschreibung an einen Solver zur Lösung des Planungsproblems übergeben werden. Für den Anwendungsfall kam beispielsweise der Solver *Delfi* [Ka18] zum Einsatz, der von [Mu22] bereitgestellt wird. Mit dem resultierenden Plan des Solvers kann das System abschließend hinsichtlich Optimierungszielen wie der Minimierung der Durchlaufzeit oder dem erwarteten Energieverbrauch untersucht werden. Die daraus gewonnenen Daten dienen dem Systementwickler als Basis für Entscheidungen in der Systemauslegung. Sollte beispielsweise statt eines UR10 ein UR16 in Betracht gezogen werden, lässt sich ein direkter Vergleich dieser beiden Varianten durchführen. Dafür muss das Systemmodell mit den Informationen für den UR16 erweitert werden, jedoch kann die gleiche Planungslogik ohne Mehraufwand wiederverwendet werden.

5 Zusammenfassung und Ausblick

Im Rahmen dieses Beitrags wurde ein SysML-Profil vorgestellt, das speziell darauf ausgerichtet ist, die Modellierung und Planung komplexer Produktionssysteme durch Integration von Planungslogiken mittels PDDL zu verbessern. Das entwickelte Profil ermöglicht eine PDDL-spezifische Erweiterung bestehender SysML-basierter Systemmodelle, was zu einer konsistenten und synchronisierten Aktualisierung von Planungs- und Systeminformationen führt. Das heißt, dass Änderungen im Systemmodell automatisch zu Änderungen in den PDDL-Komponenten führen. Die Anwendbarkeit des Profils wurde in einem realen Anwendungsfall im Flugzeugbau erfolgreich validiert.

Die Nutzung dieses Profils setzt jedoch eine Vorgehensmethode voraus, die beschreibt, welche Schritte notwendig sind um Systemmodelle systematisch mit diesem Profil zu erweitern. Zudem ist ein Algorithmus erforderlich, der die angereicherten Systemmodelldaten liest und in PDDL-Syntax umwandelt. Diese automatisierte Generierung von PDDL-Beschreibungen wurde bereits experimentell mit VTL-Templates implementiert. Sie beschränkt sich jedoch auf die Verwendung in der Modellierungssoftware MSoSA. Da Informationen für die PDDL-Problembeschreibung in der Regel nicht direkt im Systemmodell enthalten sind, sondern sich oft in einem Produktmodell befinden, empfiehlt sich ein Ansatz, der diese Informationen aus verschiedenen Engineering-Artefakten extrahieren kann. Insbesondere weil die Informationen aus der Domänen- und Problembeschreibung in Beziehung gesetzt werden müssen, werden zukünftige Arbeiten darauf abzielen, diese algorithmische Generierung zu standardisieren und flexible Transformationsprogramme zu entwickeln, die beispielsweise auf den Ansatz von [WF22] basieren, um weitere Datenquellen berücksichtigen zu können. Sowohl die Vorgehensmethode als auch der Algorithmus sind Gegenstand zukünftiger Veröffentlichungen.

Förderhinweis

Die Autoren [Projekt iMOD] bedanken sich für die Förderung bei dtec.bw – Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr. dtec.bw wird von der Europäischen Union – NextGenerationEU finanziert.

Literatur

[Be23] Beers, L.; Weigand, M.; Nabizada, H.; Fay, A.: MBSE Modeling Workflow for the Development of Automated Aircraft Production Systems. In: 28th International Conference on Emerging Technologies and Factory Automation (ETFA). S. 1–8, 2023.

[Be24] Beers, L.; Nabizada, H.; Weigand, M.; Gehlhoff, F.; Fay, A.: A SysML Profile for the Standardized Description of Processes during System Development. In: International Systems Conference (SysCon). Im Erscheinen, 2024.

[BM12] Batarseh, O.; McGinnis, L. F.: System modeling in SysML and system analysis in Arena. In: Proceedings of the Winter Simulation Conference (WSC). S. 1–12, 2012.

[BW13] Bergmayr, A.; Wimmer, M.: Generating Metamodels from Grammars by Chaining Translational and By-Example Techniques. In: MDEBE@ MoDELS. S. 22–31, 2013.

[Ei15] Eigner, M.; Dickopf, T.; Schulte, T.; Schneider, M.: mecPro2-Entwurf einer Beschreibungssystematik zur Entwicklung cybertronischer Systeme mit SysML. Tag des Systems Engineering (Hanser)/, S. 163–172, 2015.

[Es07] Estefan, J. A.: Survey of Model-based Systems Engineering (MBSE) Methodologies. Incose MBSE Focus Group 25/8, S. 1–12, 2007.

[GNT16] Ghallab, M.; Nau, D.; Traverso, P.: Automated Planning and Acting. Cambridge University Press, 2016.

[Ha19] Haslum, P.; Lipovetzky, N.; Magazzeni, D.; Muise, C.: An Introduction to the Planning Domain Definition Language. Springer International Publishing, Cham, 2019.

[HS21] Henderson, K.; Salado, A.: Value and benefits of model–based systems engineering (MBSE): Evidence from the literature. Systems Engineering 24/1, S. 51–66, 2021.

[HVC13] Huckaby, J.; Vassos, S.; Christensen, H. I.: Planning with a task modeling framework in manufacturing robotics. In: International Conference on Intelligent Robots and Systems. S. 5787–5794, 2013.

[Ka18] Katz, M.; Sohrabi, S.; Samulowitz, H.; Sievers, S.: Delfi: Online Planner Selection for Cost-Optimal Planning. In: Ninth International Planning Competition. S. 57–64, 2018.

[Ko11] Kovacs, D. L.: Complete BNF description of PDDL 3.1. Language Specification, Department of Measurement and Information Systems, Budapest University of Technology and Economics/, 2011.

[Li23] Lindsay, A.: On Using Action Inheritance and Modularity in PDDL Domain Modelling. Proceedings of the International Conference on Automated Planning and Scheduling 33/1, S. 259–267, 2023.

[Ma22a] Ma, J.; Wang, G.; Lu, J.; Vangheluwe, H.; Kiritsis, D.; Yan, Y.: Systematic Literature Review of MBSE Tool-Chains. Applied Sciences 12/7, S. 3431, 2022.

[Ma22b] Mayr-Dorn, C.; Egyed, A.; Winterer, M.; Salomon, C.; Fürschuß, H.: Evaluating PDDL for programming production cells: a case study. In: 4th International Workshop on Robotics Software Engineering (RoSE). S. 17–24, 2022.

[Mu22] Muise, C.; Pommerening, F.; Seipp, J.; Katz, M.: PLANUTILS: Bringing Planning to the Masses. In: 32nd International Conference on Automated Planning and Scheduling (ICAPS 2022). System Demonstrations. 2022.

[OM19] OMG: Systems Modeling Language (SysML™ 1.6), 2019.

[Ri21] Rimani, J.; Lesire, C.; Lizy-Destrez, S.; Viola, N.: Application of MBSE to model Hierarchical AI Planning problems in HDDL. In: International Conference on Automated Planning and Scheduling (ICAPS). 2021.

[Se15] Seidl, M.; Scholz, M.; Huemer, C.; Kappel, G.: UML @ Classroom: An Introduction to Object-Oriented Modeling. Springer, 2015.

[SS20] Schmidt, M. M.; Stark, R.: Model-Based Systems Engineering (MBSE) as computer-supported approach for cooperative systems development. In: Proceedings of 18th European Conference on Computer-Supported Cooperative Work. 2020.

[Tö17] Törmänen, M.; Hägglund, A.; Rocha, T.; Drenth, E.: Integrating Multi-Disciplinary Optimization into the Product Development Process using Model-Based Systems Engineering (MBSE). In: NAFEMS World Congress. 2017.

[Vi23] Vieira da Silva, L. M.; Heesch, R.; Köcher, A.; Fay, A.: Transformation eines Fähigkeitsmodells in einen PDDL-Planungsansatz. at-Automatisierungstechnik 71/2, S. 105–115, 2023.

[Wa19] Wally, B.; Vyskočil, J.; Novák, P.; Huemer, C.; Šindelár, R.; Kadera, P.; Mazak, A.; Wimmer, M.: Flexible production systems: Automated generation of operations plans based on ISA-95 and PDDL. IEEE Robotics and Automation Letters 4/4, S. 4062–4069, 2019.

[We08] Weilkiens, T.: Systems engineering with SysML/UML: Modeling, analysis, design. Morgan Kaufmann/Elsevier, Amsterdam und Boston, 2008.

[We16] Weilkiens, T.: SYSMOD - The Systems Modeling Toolbox: Pragmatic MBSE with SysML. Lulu.com, 2016.

[WF22] Weigand, M.; Fay, A.: Creating Virtual Knowledge Graphs from Software-Internal Data. In: 48th Annual Conference of the IEEE Industrial Electronics Society. S. 1–6, 2022.