Wissensbasierte Entwurfsbewertung der Produktgestalt mittels Multi-Agentensystemen

Von der Fakultät für Maschinenbau
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur
Dr.-Ing.
genehmigte

Dissertation

von
M.Eng. Stefan Plappert

2023

1. Referent: Prof. Dr.-Ing. Roland Lachmayer
2. Referent: Prof. Dr.-Ing. Thomas Vietor
Tag der Promotion: 09. Juni 2023

In Erinnerung an Cornelia Plappert.

Vorwort

Die vorliegende Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Produktentwicklung und Gerätebau (IPeG) der Gottfried Wilhelm Leibniz Universität Hannover (LUH) entstanden.

Einen besonderen Dank möchte ich Prof. Dr.-Ing. Roland Lachmayer für die wissenschaftliche Betreuung, den vertrauensvollen Umgang und die konstruktiven Anregungen aussprechen. Weiterhin möchte ich mich ganz herzlich bei Prof. Dr.-Ing. Thomas Vietor für die Übernahme des Koreferats und Prof. Dr.-Ing. Bernd-Arno Behrens für die Übernahme des Prüfungsvorsitzes danken.

Ferner möchte ich mich bei meinen Kolleginnen und Kollegen am IPeG für die professionelle Zusammenarbeit und die gemeinsam verbrachte Zeit bedanken. Über die Arbeitszeit hinaus sind tiefgreifende Freundschaften, insbesondere mit Lukas Hoppe und Behrend Bode entstanden. Für die vielen fachlichen, sowie sehr anregenden Gespräche und die Unterstützung beim Schreiben dieser Arbeit gilt mein besonderer Dank Dr.-Ing. Paul Christoph Gembarski. Des Weiteren möchte ich dem Team der Forschungsgruppe „Knowledge-based Systems Engineering“ mit Lukas Hoppe, Patrik Müller, Kevin Herrmann, Carl Steinnagel und Johannes Budau für den stets guten Zusammenhalt danken.

Den Studierenden, die ich während der Arbeit betreuen durfte und deren Ergebnisse teils in die Arbeit eingeflossen sind, gilt ebenfalls mein Dank. Namentlich hervorzuheben ist Christian Becker und dessen Arbeiten zur Implementierung des Multi-Agentensystems.

Ganz herzlich möchte ich mich auch bei meinen Eltern für die bedingungslose Unterstützung in allen Lebensphasen bedanken. Ich danke ebenfalls meiner Freundin Katharina Norén für die uneingeschränkte Unterstützung.

Stefan Plappert Hannover, im Juni 2023

Kurzfassung

Beim Gestalten legen die Konstruierenden eine Vielzahl der Produkteigenschaften fest, welche Einfluss auf den gesamten Produktlebenszyklus haben. Daher verlangt ein moderner Produktentwicklungsprozess, dass ein Produkt nicht nur aufgrund seiner Funktionserfüllung optimiert wird, sondern auch vor dem Hintergrund des Design for X (DfX), welches die Berücksichtigung von Anforderungen aus dem gesamten Produktlebenszyklus beinhaltet. Die zunehmende Komplexität in der Produktentwicklung und der fehlende Wissensaustausch, beispielsweise zwischen der Entwicklung und Fertigung, führen zu unnötigen Iterationsschleifen und hohen Kosten.

Im Rahmen dieser Arbeit wird eine wissensbasierte Entwurfsbewertung der Produktgestalt mittels Multi-Agentensystemen vorgestellt, um den Konstruierenden ein digitales Assistenzsystem zur Überprüfung von CAD-Modellen hinsichtlich der DfX-Anforderungen zur Verfügung zu stellen. Hierzu werden, zum einen für die Repräsentation des domänenspezifischen Wissens wissensbasierte Entwicklungs- und Konstruktionssysteme untersucht und zum anderen für die Durchführung von dezentralen Schlussfolgerungsmechanismen und Verhandlungen das Multi-Agentensystem vorgestellt. Das methodische Vorgehen *MaSE4D* zur Entwicklung von Multi-Agentensystemen ermöglicht, ausgehend vom initialen Systemkontext, eine strukturierte Zusammensetzung der dezentral verteilten Wissensquellen, um eine ganzheitliche Bewertung in der Domäne der Produktgestaltung vorzunehmen. Durch einen generalisierten Aufbau einer Software-Architektur ist es zudem möglich ein 3D-Modell in einer CAD-Entwicklungsumgebung, mittels graphenbasierter Featureerkennung, zu analysieren und automatisiert anzupassen. Des Weiteren werden Templates für die Programmierung von Agenten zur Verfügung gestellt, welche die dezentrale Wissensverwaltung, die Kommunikation unter den Agenten sowie deren Verwaltung im Kontext der Produktgestaltung unterstützen.

Die Anwendbarkeit und Validierung des methodischen Vorgehens und der Entwicklungsumgebung für das Multi-Agentensystem erfolgt durch die Bewertung von zehn unterschiedlichen Frästeilentwürfen.

Schlagwörter: Multi-Agentensysteme, Entwurfsbewertung, Design for X, Wissensbasierte Systeme, Assistenzsysteme für die Produktentwicklung.

Abstract

Title: Knowledge-based evaluation of embodiment design using multi-agent systems

During the design process, the designers determine numerous product properties that have an influence on the entire product life cycle. Therefore, a modern product development process requires that a product is not only optimized based on its functional fulfillment but also regarding Design for X, which includes the consideration of requirements from the entire product life cycle. The increasing complexity in product development and the lack of knowledge exchange between development and manufacturing lead to iteration loops and higher costs.

In the context of this work, a knowledge-based evaluation of embodiment design using multi-agent systems is presented to provide designers with a digital assistance system for the verification of CAD models regarding Design for X. For this purpose, knowledge-based engineering systems are investigated for the representation of domain-specific knowledge on the one hand. On the other hand, multi-agent systems are introduced for the execution of decentralized reasoning mechanisms and negotiations. The methodological approach *MaSE4D* for the development of multi-agent systems allows, starting from the initial system context, a structured composition of the decentralized distributed knowledge sources to perform a holistic evaluation in the domain of product design. A generalized software architecture allows analyzing and automatically adapting a 3D model in a CAD development environment using graph-based feature recognition. Furthermore, templates for programming agents are provided, which support decentralized knowledge management, communication among agents, as well as their management in the context of product design.

The applicability and validation of the multi-agent system's methodological approach and development environment is performed by evaluating ten different milling designs.

Keywords: multi-agent systems, design review, Design for X, knowledge-based systems, assistance systems for product development.

Inhaltsverzeichnis

Abkürzungen und Formelzeichen

Allgemeine Abkürzungen

CAD	*Computer-aided Design*
CAE	*Computer-aided Engineering*
CAx	*Computer-aided Everything*
DAI	*Distributed Artificial Intelligence*
DfX	*Design for X*
DSR	*Design Science Research*
DSRC	*Design Science Research Cycles*
IT	Informationstechnik
KBE	*Knowledge-based Engineering*
KBES	*Knowledge-based Engineering Systems*
KBS	*Knowledge-based Systems*
KI	Künstliche Intelligenz
MAS	Multi-Agentensysteme

Abkürzungen für die Programmierung

AAG	*Attributed Adjacency Graph*
API	*Application Programming Interface*
B-Rep	*Boundary Representation*
CSG	*Constructive Solid Geometry*
CSP	*Constraint-Satisfaction Problem*
GUI	*Graphical User Interface*
IGES	*Initial Graphics Exchange Specification*
ITS	Intelligentes Tutorensystem
MBD	*Model-based Definition*
OOP	Objektorientierte Programmierung

STEP	*Standard for the Exchange of Product Model Data*
UML	*Unified Modeling Language*

Abkürzungen zu Multi-Agentensystemen

ACL	*Agent Communication Language*
AIL	*Action-Item-List*
AMS	*Agent Management System*
AO	agentenorientiert
AP	*Agent Platform*
ASL	*AgentSpeak Language*
BDI	*Belief-Desire-Intention*
brf	*Belief Revision Function*
DF	*Directory Facilitator*
FIPA	*Foundation for Intelligent Physical Agents*
Gaia	*Generic Architecture for Information Availability*
JADE	*Java Agent Development Environment*
JATLite	*Java Agent Template Lite*
JID	*Jabber Identifier*
KADS	*Knowledge Acquisition Documentation and Structuring*
KQML	*Knowledge Query and Manipulation Language*
MaSE	*Multiagent Systems Engineering*
MaSE4D	*Multiagent Systems Engineering for Engineering Design*
MOKA	*Methodology and Tools Oriented to Knowledge-based Engineering Applications*
OO	Objektorientiert
ProKon	*Proactive Support of Design Engineering Process with Agent-based Systems*
ROADMAP	*Role Oriented Analysis and Design for Multi-Agent Programming*
SPADE	*Smart Python Agent Development Environment*
WB	wissensbasiert
XMPP	*Extensible Messaging and Presence Protocol*

1 Einleitung

Die Produktentwicklung ist seit dem Beginn der 90er-Jahre einem Veränderungsprozess unterlegen [VDI2218], der sich u. a. durch den steigenden Kostendruck, die Erhöhung der Qualitäts- und Umweltanforderungen, die Verkürzung von Entwicklungszeiten sowie werkstofftechnische Fortschritte und technologische Innovationen in Fertigung und Montage auszeichnet. Diese mitunter sehr unterschiedliche Produktionsbedingungen müssen bei der fertigungsgerechten Auslegung von Produkten beachtet werden. Gesteigert wird dieser Umstand durch stark zunehmende Globalisierungstendenzen im Fertigungsbereich. Hinzu kommt die geografisch verteilte Entwicklung mit heterogenen Systemen, die einen ganz erheblichen Einfluss auf den Gestaltungsprozess haben und somit zu einem Vorgehen führen, bei dem Konstruierende, weit mehr als früher, über die Fertigung und Montage hinaus, den gesamten Produktlebenszyklus in ihre Arbeit einbeziehen müssen [Andr07, Matt19]. Zudem hängt die Produktgestaltung stark von der Erfahrung und den Fähigkeiten der Konstruierenden ab, sprich von der Suche nach geeigneten Lösungskonzepten, der Erstellung eines ersten Ausführungsentwurfs und der Detaillierung von Teilen und Baugruppen, unter Berücksichtigung bestehender Beschränkungen, z. B. aus der Fertigung oder Logistik [Bave16, Ullm09, Vajn09].

Laut Franke [Fran76] kann das Gestalten von Produkten als Optimieren in einem vieldimensionalen Gleichungssystem aufgefasst werden. In diesem Gleichungssystem stellen die Gestalt- und Werkstoffeigenschaften die direkt festgelegten unabhängigen Variablen dar und die zu erreichenden Anforderungen die abhängigen Variablen, welche um die zu erfüllenden Bedingungen ergänzt werden. Abbildung 1.1 zeigt beispielhaft die direkt und indirekt festgelegten Produkteigenschaften. Die Konstruierenden beeinflussen, neben den direkt festzulegenden Gestalt- und Werkstoffeigenschaften, auch indirekt die Eigenschaften des Produkts. Daher verlangt ein moderner Produktentwicklungsprozess u. a., dass ein Produkt nicht nur aufgrund seiner Funktionserfüllung, sondern auch für alle der eigentlichen Produktentwicklung nachfolgenden Phasen des Produktlebenszyklus zu optimieren ist. Dieses Vorgehen wird *Design for X* (DfX)[1]

[1] Design for X stellt eine Reihe von Gestaltungsrichtlinien für die Produktentwicklung, zur Verfolgung verschiedener Ziele, z. B. Kosten-, Gewichts- oder Fertigungsziele [VDI2221-1], dar.

genannt, da das Produkt so bspw. auch fertigungs-, montage-, wartungs-, transport- und recyclinggerecht ausgelegt werden muss [EIGN14, UGUR17]. Somit müssen die Konstruierenden über immer mehr Wissen bei der Produktgestaltung verfügen, um den steigenden Anforderungen aus den der Produktentwicklung nachfolgenden Prozessen gerecht zu werden. Da dieses Wissen meist dezentral in den Unternehmen verteilt ist, ergeben sich einige Herausforderungen während der Gestaltung, auf welche im folgenden Kapitel weiter eingegangen wird.

Abbildung 1.1: Direkt und indirekt festgelegte Produkteigenschaften beim Gestalten und ihr Bezug zu Anforderungen und Bedingungen [VDI2223]

1.1 Herausforderungen während der Gestaltung

Während der Gestaltung wird eine Lösung, von der Aufgabenstellung bis zur konkreten, vollständig determinierten Beschreibung des technischen Gebildes, schrittweise konkretisiert [KLOS90]. Um effektiv und effizient zu gestalten, müssen die Konstruierenden den Gestaltungsprozess daher im Detail so steuern, dass er in der zur Verfügung stehenden Zeit zu einem guten Ergebnis kommt [VDI2223]. Dieses Vorgehen erweist sich in der Praxis häufig als schwierig, da folgende Herausforderungen bei den Gestaltungstätigkeiten gehandhabt werden müssen:

Fehlendes fertigungsspezifisches Wissen während der Gestaltung: Zur Erstellung eines durchdachten und realistischen Entwurfs ist für jeden Fertigungsprozess oft ein detailliertes Expertenwissen notwendig [ULLM01]. Es fehlen allerdings häufig Kenntnisse über fertigungsspezifisches Wissen, wie z. B. den Bauraum in der Maschine, zur Vermeidung von Kollisionen oder die Verfügbarkeit der Werkzeuge, zur Herstellung des entworfenen Bauteils. Zudem

vergeben die Konstruierenden die Toleranzen für Bauteile, wissen aber nicht immer, welche Auswirkungen die Toleranzvorgaben auf die Fertigungsprozesse und -schritte haben und somit den Fertigungsaufwand, insbesondere die Kosten, nicht bewerten können [FENG05, DOEL20].

Fehlende Kommunikation zwischen Fertigung und Entwicklung: Grabowski und Geiger [GRAB97] haben durch mehrere Befragungen von Konstruierenden gezeigt, dass die Kommunikation zwischen der Entwicklung und der Fertigung eine große Herausforderung in der Zusammenarbeit darstellt. Ein Mangel an Kommunikation führt zu unnötigen Iterationen, welche die Entwicklungszeit und damit die Kosten des Gestaltungsprozesses erhöhen [BEND21]. Die meisten Konstruierenden beklagen, dass sie zu viel Zeit damit verbringen, die benötigten Informationen von der Fertigung zu beschaffen, sei es informell durch Besprechungen, Telefongespräche sowie Lektüre oder formell durch organisatorische, computerbasierte Informationen [JIAN10]. Da die Konstruierenden diese Aufgaben häufig als zermürbend bezeichnen [MÜL12], wird eine detaillierte Klärung der Herstellbarkeit nur unzureichend durchgeführt. Zudem geht die Entwicklungsabteilung oft davon aus, dass die Fertigungsabteilung Vorschläge für Konzepte, z. B. zur Reduzierung der Teilezahl oder der Prozessschritte, formuliert. Die Fertigungsabteilung stellt hingegen das eigentliche Konzept häufig gar nicht infrage, sondern optimiert nur das bestehende Konzept [DOEL20].

Steigende Komplexität in der Produktentwicklung: Aufgrund kürzerer Entwicklungszyklen, zunehmender Produktkomplexität und verteilter Entwicklung steigt die Zahl der Aufgaben und das zu ihrer Erfüllung erforderliche Wissen, mit welchem sich Ingenieurinnen und Ingenieure befassen müssen [HUTH20, JIA04]. Dies zeigt sich in der Geometrie der Produkte, den Anforderungen des Kunden und auch in den Systemen, die benötigt werden [GEMB19]. Somit erhöht die Individualisierung den Aufwand für die Entwurfsbewertung, da im Hinblick auf die Losgröße 1 jeder Entwurf individuell überprüft werden muss. Weiterhin steigt durch den Wunsch nach Individualisierung sowohl der Detaillierungsgrad, als auch das Verlangen nach einer Automatisierung des Gestaltungsprozesses [DUEH20].

Kurze Markteinführungszeiten und unsichere Entscheidungen: Unter der großen Anzahl von Entscheidungen, die während der Projektlaufzeit getroffen werden, gehören die Entscheidungen während der Produktentwicklung zu den wichtigsten, da sie einen großen Einfluss auf die Qualität der Produkteigenschaften haben [DOST16]. Insbesondere durch Faktoren wie den zunehmenden Wettbewerb und die kürzere Markteinführungszeit müssen nun diverse Fachexpertinnen und -experten parallel an der Produktentwicklung arbeiten und sich hierfür vermehrt abstimmen [JIA04]. Aufgrund des Zeitdrucks werden in der Praxis häufig kurzfristige und unsichere Entscheidungen getroffen, was zu erheblichen negativen Konsequenzen wie

Terminverzögerungen, Einschränkungen der Funktionalität, Kostenüberschreitungen und Produktqualitätsmängeln führen kann [LEND09].

Mangelnde Dokumentation von Entscheidungen: Neben der Dokumentation der eigentlichen Produktgestalt ist die Beschreibung der Vorgehensweise bei der Modellierung und insbesondere eine ausführliche Beantwortung der Frage: „Warum etwas so und nicht anders gemacht wurde?“ anzustreben [MATT19]. An dieser Stelle sind die Konstruierenden, Berechnungsingenieurinnen und -ingenieure oder Fertigungsplanenden aktuell noch selbst gefordert, ihre Überlegungen, die entwickelten Alternativen und die Gründe für die letztlich getroffene Entscheidung zu dokumentieren. Allerdings werden diese Entscheidungen nur unzulänglich dokumentiert, da es u. a. noch wenig Unterstützung von Softwareprogrammen gibt.

Die Mehrbelastung der Konstruierenden durch die beschriebenen Herausforderungen führt wiederum zu Fehlern, welche im schlimmsten Fall erst beim Kunden aufgedeckt werden. Um diese Fehler frühzeitig zu erkennen und Folgefehler zu vermeiden, sollten Kontrollschleifen zur Überprüfung möglichst kurz sein [BEND21]. Um die Konstruierenden hierbei zu unterstützen und sicherzustellen, dass alle Anforderungen der nachgelagerten Prozesse berücksichtigt wurden, werden Entwurfs- bzw. Entwicklungsbewertungen durchgeführt.

1.2 Digitale Entwurfsbewertung der Produktgestalt

Die Durchführung einer *Entwurfs- bzw. Entwicklungsbewertung* (engl. design review)[2] in einer kollaborativen Umgebung ist eine Technik, die dazu dient, die Gestaltung gleichzeitig aus verschiedenen Perspektiven zu überprüfen und so die Lücke zwischen der Entwicklungsabteilung und anderen am Produktentwicklungsprozess beteiligten Stakeholdern zu schließen [MEDA06]. Entwicklungsbewertungen sind einer der Kontrollmechanismen der Produktentwicklung, die dazu dienen, bestehende Lösungen zu validieren, die Qualität der durchgeführten Arbeiten zu überprüfen, anstehende Aktivitäten zu steuern und neue Lösungen vorzuschlagen [HORV20, HUET07, WANG17]. Der gesamte Produktlebenszyklus muss hier vorweggenommen und der Gestaltentwurf hinsichtlich der Erfüllung der Anforderungen und des Einhaltens von Bedingungen überprüft werden [ADWE20]. Analyse- und Bewertungsschritte sollten vorzugsweise in einem interdisziplinären Team durchgeführt werden, um die unterschiedlichen Erfahrungen der Teammitglieder für eine umfassende Beurteilung zu nutzen [VDI2223]. Dieser Schritt ist besonders für die Fertigung wichtig und beinhaltet u. a. die Kontrolle der Einhaltung

[2] Die Entwicklungsbewertung beschreibt den Prozess einer geplanten, dokumentierten unabhängigen Bewertung einer existierenden Konstruktion oder eines vorgeschlagenen Konstruktionsentwurfs [DIN61160].

von Normen, die fertigungsgerechte Bemaßung, die erforderlichen Fertigungsangaben und die Beachtung der Beschaffungsgesichtspunkte [EIGN14]. Generell versprechen die Entwurfsbewertungen eine hohe Fehlervermeidung und ermöglichen, durch die Verhandlung verschiedener unternehmerischer Bereiche, das Finden von Kompromissen, z. B. zwischen Funktionalität und Herstellbarkeit des Produktes. Die Durchführung von Entwurfsbewertungen für einfache Einzelteile ist durch den hohen Ressourcen- und Personaleinsatz nicht wirtschaftlich. Um dennoch auch bei diesen Bauteilen die Vorteile der Entwurfsbewertung nutzen zu können, wird im Folgenden der Prozess zur Durchführung von Entwurfsbewertungen in einer digitalen und interaktiven Entwicklungsumgebung untersucht.

Die jüngsten Fortschritte im Bereich der Künstlichen Intelligenz (KI) fördern die Entwicklung von autonomen Entscheidungsprozessen in Systemen, die in der realen Welt operieren [ALZE20]. Hierbei muss jedoch immer der Kontext, in welchem das Produkt entworfen wird, berücksichtigt werden. Die optimale Lösung für das gleiche Produkt kann sich z. B. in unterschiedlichen betrieblichen Situationen unterscheiden, sodass das Entscheidungsunterstützungssystem in der Lage sein muss, diesen Kontext bei der Entwurfsbewertung einzubeziehen. Bei solch einer vielseitigen Unterstützung ist es erforderlich, dass sich die Schnittstellen zu den Anwendenden nicht mehr überwiegend statisch, sondern immer intuitiver verhalten und sich dabei permanent den individuellen Gegebenheiten anpassen können [VAJN09]. Für eine rechnerunterstützte Bewertungsumgebung spricht zudem, dass die Mitglieder des Projektteams bei den regulären formellen Sitzungen häufig vor dem Bewertungsteam über ihre Arbeit berichten müssen, was einer typischen Situation der „Rechtfertigung“ entspricht [ECKE05]. Dieser Aspekt wird behoben, da bei einer rechnerunterstützten Entwurfsbewertung, die Konstruierenden selbstständig ihren Entwurf überprüfen lassen können. Des Weiteren ist einer der Schlüsselfaktoren der Erwerb von Fachwissen und das Sammeln von Erfahrungen von Konstruierenden mit langjährigen Kompetenzen innerhalb des Systems, sodass dieses Wissen auch steigender beruflicher Fluktuation dem Unternehmen erhalten bleibt [QUIN15].

Um die Entwicklungsbewertung in eine digitale Entwurfsbewertung zu überführen, sollte die Eingabe in den Prozess, also der Unterlagensatz bzw. das CAD-Modell und das Ergebnis des Prozesses, die Dokumentation der Bewertung sowie die Ableitung von Empfehlungen und Maßnahmen ebenfalls digital möglich sein. Des Weiteren müssen einige organisatorische Anpassungen vorgenommen werden. So entfällt die Planung der Sitzung, da die virtuelle Entwurfsbewertung unabhängig von Orts- und Zeitrestriktionen durchgeführt werden kann und so die Flexibilität erhöht wird. Zudem wird die Entwicklungsleiterin bzw. der Entwicklungsleiter aus der bestehenden Entwicklungsbewertung durch die Anwendenden bzw. Konstruierenden ersetzt, welche selbstständig die virtuelle Entwurfsbewertung durchführen und infolgedessen

für die Änderungen an der Produktgestalt verantwortlich sind. Um die virtuelle Entwurfsbewertung als Rahmenwerk für Konstruierende, für die Bewertung der Produktgestalt, zur Verfügung stellen zu können, müssen folgende Herausforderungen adressiert werden:

- Das Entwicklungsteam muss in Form von virtuellen Akteurinnen und Akteuren abgebildet werden, inklusive des spezifischen Domänenwissens.
- Die Bewertung der Produktgestalt muss aufgrund unterschiedlicher Perspektiven erfolgen.
- Das System muss automatisiert Empfehlungen für eine Optimierung der Produktgestalt bereitstellen und die Durchführung von Maßnahmen ermöglichen.

Infolge der gewonnenen Erkenntnisse benötigen Konstruierende geeignete Unterstützungsmechanismen und Werkzeuge, um Informationen und Wissen aus allen Bereichen des Unternehmens zu synthetisieren und zu verwalten [Chir06]. Mit anderen Worten erfordert die Unterstützung der Entscheidungsfindung die Verwaltung von Daten, Modellen und Wissen sowie die damit verbundene Beurteilung, auf der die Entscheidungen basieren [Ullm01].

1.3 Aufgabenstellung und Aufbau der Arbeit

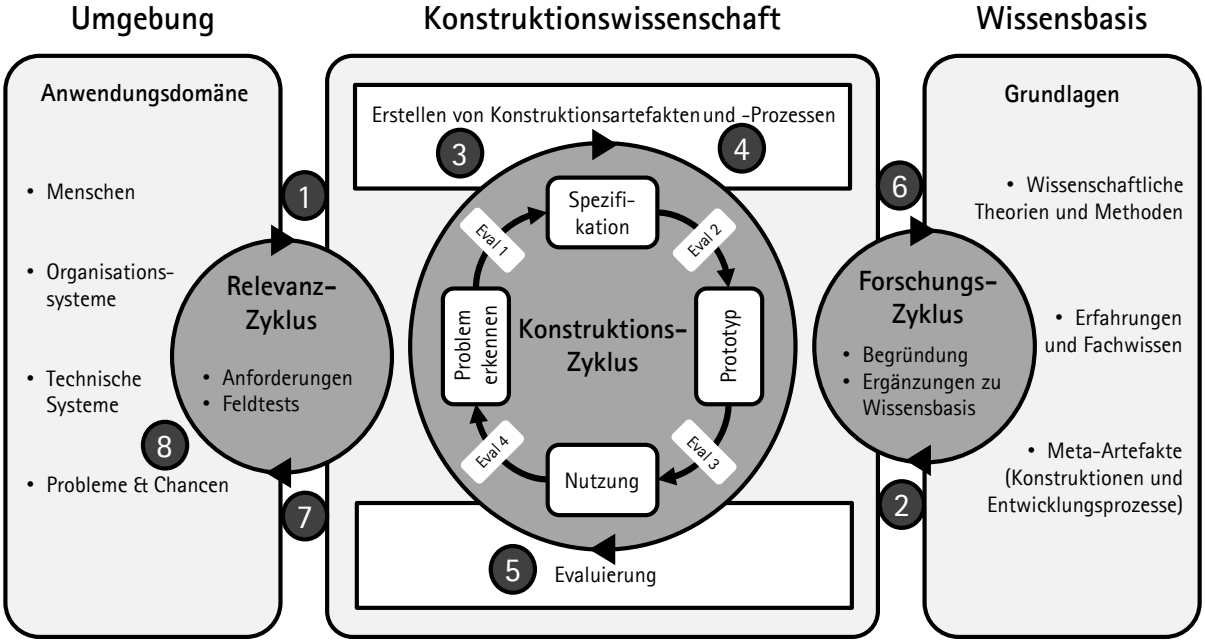
Aufgrund der Herausforderungen im modernen Entwicklungsprozess und dem daraus resultierenden Bedarf von Unterstützungssystemen in der Entwurfsbewertung, insbesondere mit Unterstützung durch intelligente Softwareprogramme, erscheint eine Implementierung eines Assistenzsystems für Konstruierende in einer CAD-Entwicklungsumgebung nützlich [Plap22d]. Hierfür bietet die agentenbasierte Technologie ein natürliches Mittel zur Modellierung eines Problems auf der Grundlage des Paradigmas der verteilten KI, da reale Entitäten und ihre Interaktionen direkt durch autonome Problemlösungsagenten dargestellt werden können, welche zusätzlich die aktuelle Wissenssuche zu einem aktiven persönlichen Assistenten machen [Jenn95, Jian10]. Der Bedarf an vielfältigen, hochentwickelten und sich schnell ändernden Fähigkeiten und Kenntnissen macht das Multi-Agentensystem (MAS) besonders geeignet für Gestaltentwürfe, welche sehr wissensintensiv sind [Land97]. Im Rahmen dieser Arbeit wird der Einsatz von MAS als Entwurfsbewertungssystem, im Hinblick auf die fertigungsgerechte Gestaltung, betrachtet. Hierfür werden die folgenden wissenschaftlichen Fragestellungen untersucht:

- In welcher Weise lässt sich ein methodisches Vorgehen für die Entwicklung von Multi-Agentensystemen für die Produktgestaltung ableiten und wie sieht hierfür ein generalisierter Aufbau einer Software-Architektur aus?
- Wie kann dezentrales Domänenwissen erfasst, formalisiert und kommuniziert werden, um Entwurfsbewertungen der Produktgestalt zu operationalisieren und zu digitalisieren?
- Wie lässt sich ein entsprechendes Assistenzsystem für Konstruierende ausgestalten, um ein 3D-Modell in einer CAD-Entwicklungsumgebung zu analysieren und automatisiert anzupassen?

Die Aufgabenstellung umfasst, neben dem Teilbereich der Gestaltmodellierung (Anwendungsdomäne), die Analyse, Verarbeitung und Speicherung von kontextspezifischem Wissen (Wissensbasis) sowie die automatisierte Entwurfsbewertung mittels Multi-Agentensystemen (Konstruktionswissenschaft[3]). Dies spiegelt auch die Wahl des Forschungsansatzes aus der Domäne der Informationstechnik (IT) wider, welcher die *Design Science Research Cycles* (DSRC) von Hevner und Chatterjee [HEVN04] mit der *Design Science Research* (DSR) von Sonnenberg und vom Brocke [SONN12] kombiniert. Durch die Verwendung der Forschungszyklen aus der DSRC werden die Domänen miteinander verknüpft. Der Relevanzzyklus initiiert die Konstruktionswissenschaft mit dem Anwendungskontext, der nicht nur die Anforderungen an die Forschung als Input liefert, sondern auch Akzeptanzkriterien für die endgültige Bewertung der Forschungsergebnisse definiert. Der Forschungszyklus übergibt die Erfahrungen und das Fachwissen, welche den Stand der Technik im Anwendungsbereich der Forschung definieren, sowie die neu entwickelten Artefakte und Prozesse. Der interne Konstruktionszyklus ist das Herzstück eines jeden konstruktionswissenschaftlichen Projekts. Die Methode der DSR beschreibt einen zyklischen Prozess mit vier Aktivitäten *Problemdefinition*, *Spezifikation*, *Prototyp* und *Nutzung*. Des Weiteren kann dieser Prozess in beide Richtungen durchlaufen werden. Zwischen den Aktivitäten gibt es Evaluationsaktivitäten, welche das zuletzt entwickelte Artefakt überprüfen und somit den Nutzen nachweisen. Die inhaltliche und strukturelle Implementierung des kombinierten Ansatzes im Rahmen dieser Arbeit ist in Abbildung 1.2 dargestellt.

In Kapitel zwei wird der Stand der Technik und Forschung für wissensbasierte Informations- und Entwicklungssysteme im Kontext der KI erarbeitet. Hierzu wird die Formalisierung und Verwaltung von Gestaltungswissen sowie die CAD-Integration inklusive automatisierter Featureerkennung aufgezeigt. Als Ansatz zur dezentralen Schlussfolgerung werden MAS eingeführt und hierfür agentenorientierte Methoden, Architekturen für die Interaktion zwischen den Agen-

[3]Die Konstruktionswissenschaft unterstützt ein pragmatisches Forschungsparadigma aus der Informationstechnik, das die Schaffung innovativer IT-Artefakte zur Lösung von Problemen der realen Welt fordert [HEVN04].



Forschungsansatz (Kapitel)

1) **Forschungsfragen / Produktanforderungen** (1+3)
2) **Absicherung durch Wissen, Methoden und Erfahrungen** (2)
3) **Definition Konstruktionsartefakt /-repräsentation** (3+4)
4) **Entwicklungsprozess für Konstruktionsartefakt** (3+5)
5) **Evaluierung / Konstruktionsoptimierung** (4+5)
6) **Neues Wissen** (Literatur, Meta-Artefakte, Methoden) (4+5)
7) **Anwendungstests / Messgrößen / Implementierung** (6)
8) **Beweis der Forschungsfragen** (7)

Abbildung 1.2: Forschungsansatz dieser Arbeit in Anlehnung an [Hevn04, Sonn12] und Gegenüberstellung der Kapitel

ten und Formen der Kommunikation und Kollaboration vorgestellt. Zudem werden Werkzeuge bzw. Software-Rahmenwerke zur Programmierung von MAS miteinander verglichen. Anschließend werden verwandte Arbeiten vorgestellt, welche MAS in der Produktentwicklung einsetzen.

Im dritten Kapitel werden Herausforderungen für die Entscheidungsunterstützung bei der Entwurfsbewertung der Produktgestalt abgeleitet, welche anschließend durch eine Spezifikation des MAS konkretisiert werden. Zur Unterstützung bei der Entwicklung von MAS werden bestehende Methoden analysiert und Forschungslücken für die Anwendung in der Domäne der Produktgestaltung hergeleitet. Darauf basierend wird für die Problemanalyse ein qualitativer Vergleich der bestehenden Ansätze für MAS in der Produktentwicklung durchgeführt.

In Kapitel vier wird der Entwicklungsprozess für ein MAS zur Entwurfsbewertung beschrieben. Hierzu wird der initiale Systemkontext, durch die Definition von Anwendungsfällen des späteren Systems und der Verfeinerung durch Rollen analysiert, sodass diese in einer nachfolgenden Entwurfsphase detailliert abgebildet und durch die Verwendung von Agenten in das anzuwendende Gesamtsystem implementiert werden können.

Kapitel fünf definiert das Konstruktionsartefakt des Forschungsansatzes, in dem eine Software-Architektur und Templates für die Programmierung von Agenten zur Verfügung gestellt werden. Durch die Definition eines Entwurfsbewertungsagenten kann die dezentrale Wissensverwaltung, die Kommunikation unter den Agenten sowie deren Verwaltung im Kontext der Produktgestaltung unterstützt werden. Zudem kann durch den Einsatz von Perzeptions- und Handlungsfähigkeiten eine Interaktion mit dem CAD-Modell durchgeführt werden.

Die Anwendbarkeit des methodischen Vorgehens zur Entwicklung eines MAS in der Produktgestaltung und der Templates zur Entwurfsbewertung bilden den Schwerpunkt von Kapitel sechs. Dort wird das Vorgehen zur Bewertung der Herstellbarkeit einer Fräskonstruktion anhand eines Beispiels detailliert beschrieben und mittels Benchmarks von zehn Fräsentwürfen validiert.

Die Arbeit schließt mit einer Zusammenfassung, einer kritischen Würdigung und einem Ausblick.

2 Stand der Technik und Forschung

2.1 Wissensbasierte Systeme und Künstliche Intelligenz

Wissensbasierte Systeme sind aus dem modernen Leben nicht mehr wegzudenken. Auch wenn es Anwendenden häufig nicht bewusst ist, interagieren oder profitieren sie fast täglich von Software, welche basierend auf integriertem Wissen bei der Entscheidungsfindung hilft, Prozesse steuert und kontrolliert, eine Fehleranalyse durchführt oder andere Tätigkeiten übernimmt. Wissensbasierte Systeme (engl. knowledge-based systems (KBS)) stellen spezifisches Domänenwissen dar, indem sie die Aktionen einer Expertin oder eines Experten beim Durchlaufen eines Lösungsprozesses simulieren und komplizierte Probleme mithilfe von KI lösen [Milt08].

2.1.1 Wissensbasierte Informationssysteme

Die Aufgabe eines KBS ist die Simulation der intelligenten Verarbeitung von Informationen und dem darauf basierenden Handeln. Hierfür ist es stets notwendig, Wissen zu erfassen, darzustellen und zu verwerten. Die Systeme unterscheiden sich dabei in der Form der Wissensrepräsentation und den angewendeten Inferenzmechanismen sowie in der Art der zu lösenden Problemstellungen. Die Struktur eines KBS ist laut Hopgood [Hopg21] in drei Hauptkomponenten unterteilt: die Wissensbasis, die Inferenzmaschine und eine Schnittstelle zur Umgebung (Abbildung 2.1). Die Wissensbasis dient als Datenbank für das System und enthält alle erforderlichen Regeln und Informationen. Zur Verknüpfung der Wissensbasis mit dem vorliegenden Problem wird eine Inferenzmaschine eingesetzt, die gezielt auf das Expertenwissen zugreift und so den Lösungsprozess steuert und überwacht. Für die Interaktion mit dem KBS wird eine Schnittstelle zur Umgebung verwendet, die es erlaubt, die Wissensbasis zu erweitern und die Problemdefinition zu realisieren. Als zusätzliche Komponenten für ein KBS können ein Wissensakquisitionsmodul und ein Erklärungsmodul eingesetzt werden. Das Erklärungsmodul erklärt die Entscheidungen des KBS, sodass die Anwendenden sie überprüfen können

[HOPP20b, PLAP20b]. Das Wissensakquisitionsmodul wird genutzt, um Wissen durch die Interaktion mit dem Anwendenden oder der Expertin bzw. dem Experten zu erfassen und strukturiert in der Wissensbasis abzuspeichern.

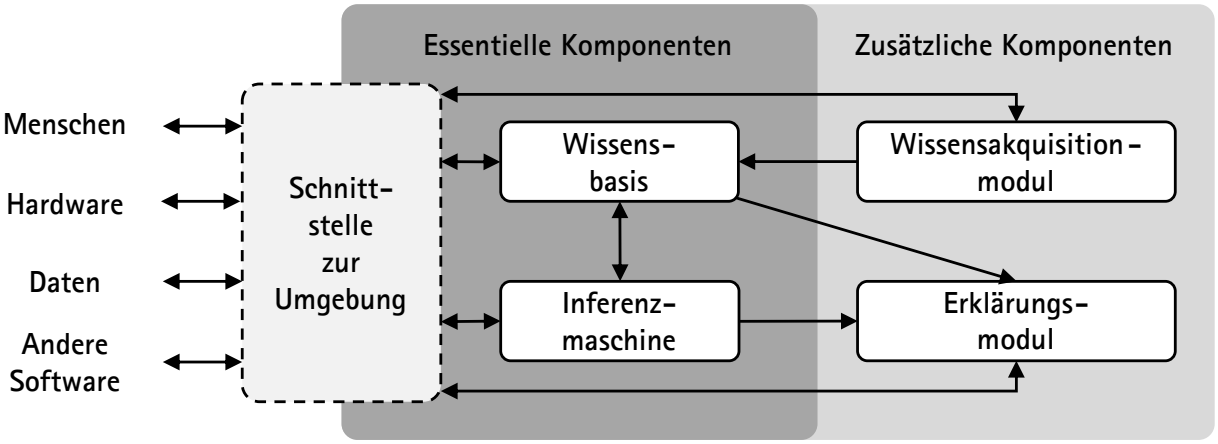


Abbildung 2.1: Komponenten eines wissensbasiertes System (in Anlehnung an [HOPG21])

Es gibt zwei mögliche Ansätze für die Darstellung von Wissen innerhalb eines wissensbasierten Systems. Zum einen die direkte und explizite Modellierung von Wissen in Form von Regeln, Constraints und Fallbasen und zum anderen die indirekte und implizite Modellierung der Randbedingungen und Restriktionen, sodass ein Optimierungsalgorithmus zur Erkundung des Lösungsraums eingesetzt werden kann [GEMB21, PLAP22c]. Im Allgemeinen beschreibt der Lösungsraum hierbei die Menge aller möglichen Lösungen für eine gegebene Menge an Anforderungen bzw. Problemstellungen [HUBK84]. Dem Lösungsraum steht laut Ponn und Lindemann [PONN11] der Anforderungsraum gegenüber. Dieser bildet die Gesamtheit der gestellten Anforderungen ab und wird während des Entwicklungsprozesses fortlaufend aktualisiert. Lösungsraum und Anforderungsraum werden also während des Entwicklungsprozesses regelmäßig abgeglichen und erneuert. Für die explizite Form der Wissensrepräsentation in wissensbasierten Systemen können grundsätzlich drei Techniken für die Schlussfolgerung verwendet werden: regelbasiertes, modellbasiertes und fallbasiertes Schließen [SABI98].

Regelbasiertes Schließen

Wird ein Regelwerk explizit als Grundlage für einen Inferenzmechanismus verwendet, wobei die Wissensdarstellung auf Wenn-Dann-Sonst-Anweisungen basiert, wird von regelbasiertem Schließen gesprochen [SABI98]. Der durch die Regeln abgebildete Entscheidungsbaum wird dabei entweder durch Vorwärts- oder Rückwärtsverkettung durchlaufen. Während bei der Vorwärtsverkettung der Entscheidungsbaum, beginnend von der Wurzel, durchlaufen und anhand von Randbedingungen, durch Abarbeiten von Regeln, auf eine Lösung geschlossen wird, wird bei der Rückwärtsverkettung der Entscheidungsraum genutzt, um bekannte Lösungen

des Baumes hinsichtlich ihrer Eignung bezüglich der spezifischen Fragestellung zu überprüfen [Lupa09, Saut17]. Bei jedem Hinzufügen von neuen Regeln muss die Konsistenz der Regelbasis überprüft werden, sodass bei steigender Anzahl der Regeln die Anpassung des Regelwerks immer komplizierter wird [Gemb19].

Modellbasiertes Schließen

Beim modellbasierten Schließen wird der mögliche Lösungsraum durch ein einschränkungsbasiertes physikalisches und/oder logisches Modell oder durch die Darstellung von Zuweisung und Ressourcenverbrauch beschrieben [Rocc12, Sabi98]. Allgemein lässt sich das modellbasierte Schließen anhand des Modellcharakters unterscheiden in logik-, constraint- und ressourcenbasierte Ansätze [Saut17].

Bei den constraint-basierten Ansätzen werden mittels Constraints, welche als eine Beziehung zwischen Domänen zu verstehen sind, denen Kombinationen von Werten zugewiesen werden, Beziehungsnetzwerke aufgebaut, welche das physikalische oder logische Modell repräsentieren [Stum98]. Um mit diesen Beziehungsnetzwerken zu schlussfolgern, werden sogenannte Constraint-Satisfaction Probleme (CSP) gelöst. In einem CSP können die großen Lösungsräume mithilfe der Constraint-Programmierung eingeschränkt und überschaubar gemacht werden, was eine deklarative Beschreibung und effektive Lösung kombinatorischer Optimierungsprobleme in Bereichen wie der Planung ermöglicht [Bart05, Bart10]. Um ein CSP zu lösen, durchsuchen die meisten Algorithmen systematisch den Raum aller möglichen Zuordnungen von Werten zu Variablen [Petr12].

Das ressourcenbasierte Schließen stellt einen Spezialfall vom constraint-basierten Schließen dar. Hierbei wird ein Gleichgewichtszustand, aus den nachgefragten Ressourcen der Komponenten bzw. der Umwelt zu den Ressourcen, welche die Komponenten und die Umwelt maximal liefern können, angestrebt [Gemb19]. Der Begriff der Ressource umfasst alle Arten umfassender (kumulativer) physischer, technischer und kommerzieller Entitäten, sowohl real als auch virtuell, die von einer Komponente eines Systems bereitgestellt und von einer anderen Komponente verbraucht oder vorübergehend genutzt werden können [Hein96].

Fallbasiertes Schließen

Beim fallbasierten Schließen (engl. case-based reasoning (CBR)) wird das Wissen nicht explizit modelliert, sondern in Form von Beispielen bereits bewährter Lösungen gespeichert. Hierbei wird ein neuer Fall bzw. ein neues Problem mit einer Fall- oder Wissensbasis verglichen und auf seine Eignung zur Lösung des neuen Falls hin bewertet [Aamo94]. Nach erfolgreicher

Anwendung des neuen Falls wird dieser wieder zur Fallbasis hinzugefügt, sodass die Fallbasis mit zunehmender Anwendung größer wird. Ein einfaches fallbasiertes System kann eine Reihe von Fällen zusammenstellen, die am besten passen oder einzelne bestehende Fälle abrufen. Anspruchsvollere Systeme können bestehende Fälle mischen oder modifizieren, um sie an neue Situationen anzupassen. Das grundlegende Prinzip des CBR basiert auf der Annahme, dass ähnliche Probleme auf ähnliche Weise gelöst werden [Beie19]. Ausgehend von diesen Definitionen besteht der klassische CBR-Zyklus aus den folgenden Phasen: Abrufen, Wiederverwenden, Überarbeiten und Behalten [Berg03].

2.1.2 Wissensbasierte Entwicklungs- und Konstruktionssysteme

Wissensbasiertes Konstruieren (engl. knowledge-based engineering (KBE)) bezeichnet die Technologie zur Erfassung und systematischen Wiedergabe von produkt- und verfahrenstechnischem Wissen [Vajn09]. Mit dem Einsatz von KBE wird angestrebt, Zeit und Kosten der Produktentwicklung, durch die Automatisierung von sich wiederholenden und nicht kreativen Gestaltaufgaben sowie durch die Unterstützung der multidisziplinären Optimierung in allen Phasen des Gestaltprozesses zu reduzieren [Rocc12]. Nach Milton [Milt08] ist ein wissensbasiertes Konstruktionssystem (KBES) eine spezielle Art von KBS, das um Fähigkeiten der computergestützten Analyse (CAE) und der rechnerunterstützte Konstruktion (engl. computer-aided design (CAD)) erweitert wird. Chapman und Pinfold [Chap01] präzisieren dieses und sehen in den KBE-Werkzeugen einen evolutionären Schritt in der rechnerunterstützten Entwicklung, der durch die Aggregation von Methoden der objektorientierten Programmierung (OOP), der KI sowie des CAD definiert wird. KBES eignen sich besonders für die Abbildung von Konstruktionslösungsräumen, wenn eine Kombination aus Konfiguration, Domänenwissen und Geometrie zur Problemlösung bzw. zur Erfüllung der Aufgabenstellung vorliegt [Plap20a, Vajn09]. Die Gesamtheit dieses für eine KBE-Anwendung relevanten Wissens wird allgemein auch als Wissensbasis bezeichnet und durch Wissensträger wie die CAD-Geometrie, in Form von Features, Parametern, Verknüpfungen oder Templates, Stücklisten und Teileverwaltung, spezialisierte Wissensbasen in Form von Regel- und Constraint-Sammlungen und Softwareprogrammen, durch Auslegungsmakros und Berechnungsalgorithmen repräsentiert [VDI5610-2]. Diese Eigenschaften ermöglichen es auch komplexe Aufgabenstellungen systematisch abzubilden. Ferner können technische Entscheidungen identifiziert und Vorgehensweisen für den Modellentwurf verdeutlicht werden [Verh12]. Allerdings betrachten die meisten Forschungsarbeiten die Modellierung konfigurierbarer Produkte als ein Anordnungsproblem einer vordefinierten Menge von Komponenten (Lösungsräume) zu einer gültigen Produktstruktur

[Ostr12]. Diese großen Lösungsräume werfen das Problem auf, dass bei einer Änderung oder Erweiterung des Lösungsraums eine Konsistenzprüfung durchgeführt werden muss. Diese Konsistenzprüfung kann bei der Prüfung des gesamten Lösungsraums sehr zeitaufwendig sein, insbesondere wenn der Einfluss einer Änderung nicht auf einen bestimmten Bereich begrenzt werden kann [Plap22b].

Eine geeignete Methode für die Entwicklung von KBES stellt MOKA (Methodology and tools Oriented to Knowledge-Based Engineering Applications) dar, welche den Pfad von der Wissensidentifizierung bis zur Implementierung in ein CAx-System unterstützt. Die MOKA-Methodik unterscheidet zwischen dem formalen und dem informalen Wissensmodell sowie zwischen den Phasen der *Wissenserfassung*, der *Wissensformalisierung* und der *Wissensverpackung* [Skar07]. Die grundlegenden Arbeitsschritte in den jeweiligen Phasen sind in der Abbildung 2.2 dargestellt. In der Wissenserfassungsphase wird das informale Wissensmodell erstellt. Nach der Identifikation von relevantem Wissen wird dies zunächst überprüft und gerechtfertigt. Anschließend wird das Wissen in einer für Menschen gut verständlichen Sprache gesammelt [Brim00]. Auf Basis der Ergebnisse der ersten Phase findet eine Formalisierung des Wissens statt. Ziel ist es, das Wissen in einer maschinenlesbaren Form für die spätere Implementierung vorzubereiten [Skar07]. Das formale Modell wird in einer grafischen und objektorientierten Form festgehalten. Dies dient der Abstraktion des in der dritten Phase zu erstellenden Quellcodes, welcher die praktische Realisierung des wissensbasierten Systems darstellt [Brim00]. Die bei der Formalisierung des Wissens entstehenden Ergebnisse werden im formalen Wissensmodell zusammengefasst, welches der Implementierung in eine computergestützte Umgebung, in der Phase der Wissensverpackung, entspricht.

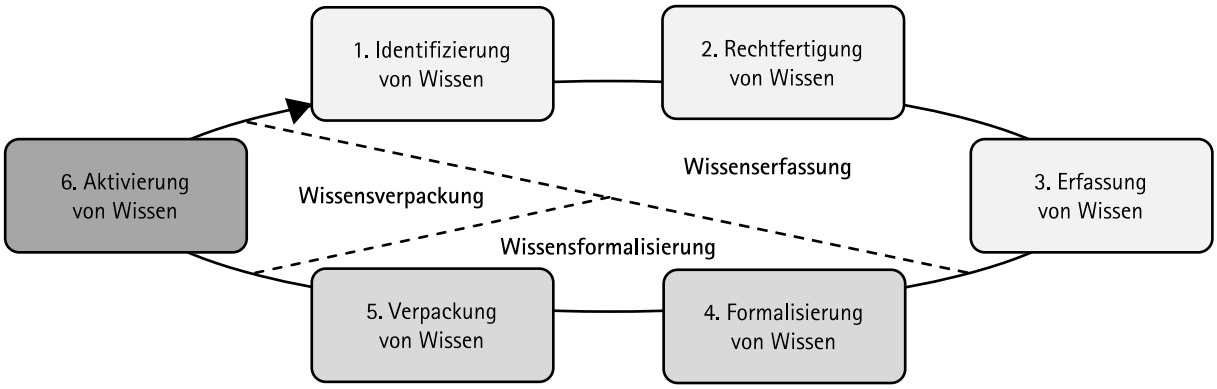


Abbildung 2.2: Der Lebenszyklus der KBE-Entwicklung nach MOKA (in Anlehnung an [Stok01])

Eine weitere Möglichkeit Wissen im Kontext der Produktentwicklung zu dokumentieren, sind Konstruktionsregeln (auch Gestaltungsrichtlinien genannt), welche Handlungsempfehlungen für die Konstruierenden darstellen. Sie haben sich in einer Vielzahl von Gestaltaufgaben

technischer Produkte bewährt und sind somit für einen größeren Benutzerkreis von Interesse [VDI2223]. Zudem geben sie den Konstruierenden konkrete Hinweise auf potenzielle Schwachstellen in Entwürfen und zeigen auf, wie diese durch geeignete Verbesserungen beseitigt werden können, sodass sie als Checkliste, zum Überprüfen von Entwürfen vor der Freigabe zum Ausarbeiten, dienen [VDI2223]. Ingenieurorientierte Constraints ermöglichen die Abbildung von Formeln, mit denen sich Gestaltungswissen, z. B. für die Auslegung von Maschinenelementen, hinterlegen lässt [VDI5610-2]. Hinsichtlich der informationstechnischen Unterstützung von Konstruktionstätigkeiten, unterscheidet Gembarski zwischen Fachwissen und Kontrollwissen [GEMB19].

KBES können anhand ihrer Verfahrensweisen in synthetische und analytische Systeme unterteilt werden. Synthetische Systeme bauen ausgehend von dem im System integrierten Wissen das CAD-Modell eines Produkts auf. Analytische Systeme untersuchen vorhandene CAD-Modelle eines Bauteils oder einer Baugruppe, beruhend auf dem integrierten Wissen bezüglich spezifischer Merkmale, Probleme oder Anwendungsfälle. Eine Kombination dieser beiden Verfahren ist möglich und kann beispielsweise zu einer Anpassung von CAD-Daten auf der Grundlage von Analyseergebnissen führen. Synthetische KBES eignen sich vorwiegend für die Anpassungs- und Variantenkonstruktion, welche besonders in Kleinserienproduktionen häufig auftritt. Diese Vorgehensweise wird insbesondere für Produktkonfiguratoren, welche eine regelbasierte Konfiguration individualisierter Produkte durchführen und Konstruktionsautomatisierungssysteme (engl. design automation), bei welchen einfache Konstruktionsarbeiten z. B. durch Makroprogrammierung automatisiert werden, eingesetzt [GEMB19, LACH17]. Analytische KBES zeigen ein noch größeres Anwendungsgebiet auf. Durch die Vielfalt der Fachgebiete des Ingenieurswesens und der daraus resultierenden Breite an themenspezifischem Wissen, erscheinen die Anwendungsfälle von untersuchend operierenden KBES kaum begrenzt. Dabei operieren analytische KBES häufig in drei Funktionsbereichen: Dem Einlesen von fallspezifischen Daten, dem darauf basierende Auswerten durch die Problemlösungskomponente im Zusammenspiel mit der Wissensbasis des Systems sowie dem Funktionsbereich der Erklärung oder Repräsentation der Ergebnisse [HOPP20a]. Die hierfür notwendigen Bestandteile eines KBES müssen nicht linear durchlaufen werden. Wie eine Schnittstelle zwischen KBES und CAD-Systemen zur Analyse und Synthese von 3D-Modellen ausgestaltet werden kann, wird im Folgenden beschrieben.

2.1.3 CAD-Integration und Featureerkennung

Die Kernfunktion eines CAD-Werkzeugs besteht in der Abbildung geometrischer Merkmale und in der Abbildung von Abhängigkeiten eines Produktes durch ein Rechnermodell, wobei die Modelle einheitlich, übersichtlich, nachvollziehbar und leicht veränderbar aufgebaut werden sollen [VDI2209]. Diese Werkzeuge enthalten Elemente für die Geometriedarstellung, z. B. Drahtgitter-, Flächen- und Volumenmodelle und Abbildungen der Produktstruktur, durch eine Objektstruktur, die aus Teiledateien und Baugruppendateien besteht. Während Teiledateien dazu dienen, Geometrieelemente und Abhängigkeiten eines Einzelteils zu definieren und zu organisieren, erfolgt die übergeordnete Positionierung und Verknüpfung der Einzelteile in den Baugruppendateien [Saut17]. Mit der Schnittstelle für die Applikationsprogrammierung (engl. application programming interface (API)) ist es möglich, nahezu alle Modellinformationen aus einem CAD-Programm auszulesen, zu nutzen oder zu verändern [Plap20b, Reic20]. Somit kann die Arbeit mit dem CAD-System erleichtert und automatisiert werden [VDI2249].

Featurebasierte CAD-Modelle werden heute vorwiegend in der Konstruktion, in der Arbeitsplanung und im Qualitätsmanagement eingesetzt, jedoch eignen sie sich aufgrund der Integration von Gestalt und phasenspezifischen Informationen ebenfalls zur Einbeziehung von Informationen, die sich auf spätere Phasen des Produktlebenszyklus wie Fertigung, Nutzung, Wartung, Demontage und Recycling beziehen [VDI2218]. Ganz allgemein beschreiben Features eine Aggregation von Geometrieelementen und/oder Semantik eines Produkts und stellen damit eine eigene Klasse von Bausteinen, als Informations- und Integrationsobjekte, im rechnerunterstützten Produktentwicklungsprozess dar. Die erste Kategorie von Features, welche außerhalb der reinen Geometriemodellierung in der Praxis eingesetzt wurde, sind Fertigungsfeatures für spanende Fertigungsverfahren. Die Fertigungsfeatures werden, hinsichtlich Gestalt und Topologie, in erster Linie durch die betreffenden Bearbeitungsverfahren bestimmt und repräsentieren, neben der fertigungsrelevanten Geometrie, sämtliche fertigungsspezifischen Toleranzen, Oberflächenbeschreibungen sowie ergänzende Angaben [Zhan01]. Die Auswahl der Features, die in einer Aufspannung gefertigt werden, hängt von der geforderten Genauigkeit, der Zugangsrichtung der Features, der Anzahl und der Richtungen der Maschinenachsen ab [VDI2218]. Innerhalb der featurebasierten Arbeitsplanerstellung werden Fertigungsfeatures, wie z. B. Bohrungen, Absätze oder Nuten verwendet (Abbildung 2.3). Neben der Beschreibung des Werkstücks durch Fertigungsfeatures, müssen die Bearbeitungsmöglichkeiten, die im Unternehmen zur Verfügung stehen und das Planungswissen, rechnerintern abgebildet sein.

Ausgehend von bestimmten Merkmalen des Entwurfs, wird entweder auf diese zugeordneten Eigenschaften geschlossen, in der Sprache der Feature-Technologie *Featureerkennung* (engl.

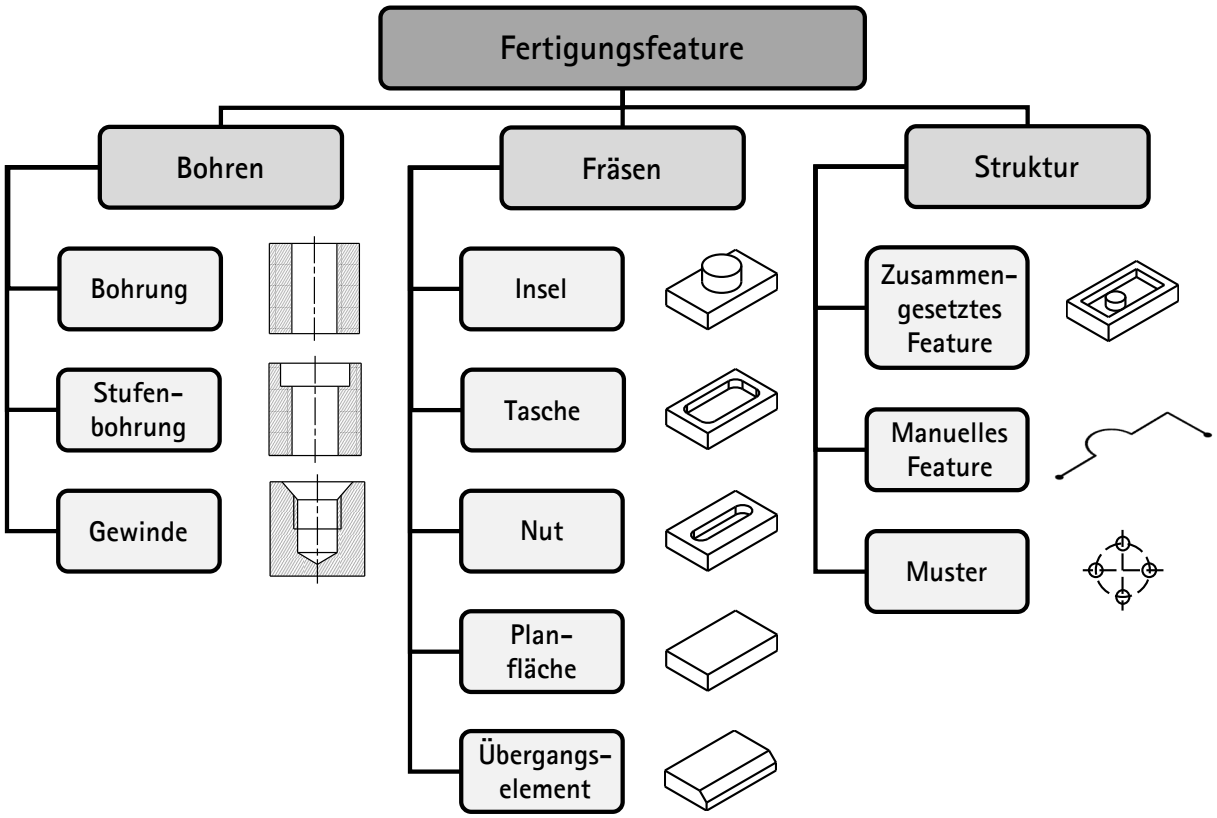


Abbildung 2.3: Fertigungsfeature für die Bohr- und Fräsbearbeitung (in Anlehnung an [VDI2218])

feature recognition) (Analyse) oder es werden bestimmte vorgegebene Eigenschaften zugeordneter Lösungsmerkmale, vom Programm automatisch, als *featurebasierte Modellierung* (engl. feature-based modeling) festgelegt (Synthese) [Webe00]. Die Methoden der automatischen Featureerkennung basieren auf der Idee, aus den einzelnen Elementen des CAD-Geometriemodells mittels Softwareunterstützung signifikante Identifikationsmerkmale eines Features zu erkennen [VDI2218]. Die Featureerkennung betrachtet das Modell aus verschiedenen Blickwinkeln und muss die Informationen entsprechend lesen und übernehmen [Foug18], um aus einem Festkörpermodell eine Menge von Fertigungsmerkmalen zu interpretieren. Hierbei lässt sich das Vorgehen bei der Featureerkennung in drei Aspekte unterteilen [Wu96]: (1) die Featuredefinition, (2) die Featurerepräsentation und (3) der Featureerkennungsmechanismus. Das Modul der Featuredefinition dient dazu, die zu erwartenden Geometrien in mehrere Klassen, bzw. Features zu zerlegen, welche zudem Informationen, z. B. über die Herstellung des Features, enthalten. Die Featurerepräsentation wandelt die einfachen geometrischen Informationen im CAD-Eingabemodell in eine höhere Form, mit geometrischen und topologischen Attributen, um, um die Featuresuche und -zuordnung zu erleichtern. Bei den Mechanismen zur Featureerkennung werden die Features zunächst anhand bestimmter Trennungsregeln für die weitere Analyse extrahiert. In der Literatur zu diesem Thema wurden zahlreiche Techniken,

wie regelbasierte, graphenbasierte, volumenzerlegende und auf künstlichen neuronalen Netzen basierende Systeme, entwickelt [SHI20]. Unter all diesen Techniken gilt der graphenbasierte Ansatz als die erfolgreichste Methode, die erstmals von Joshi und Chang [JOSH88], als Konzept eines attributierten Adjazenzgraphen (AAG), beschrieben wurde.

Häufig wird Featureerkennungssoftware für neutrale Datenformate verwendet, welche eine vereinfachte und begrenzte Menge an Informationen, im Vergleich zu den Daten des jeweiligen nativen Datenformats, enthalten. Sie können allerdings von anderen CAD-Systemen ausgelesen und genutzt werden [BEND21]. Dabei ist der Umfang der möglichen Manipulationsschritte jedoch stark begrenzt, da das Modell als eine Sammlung von Flächen, Kanten und Punkten importiert wird, die nicht weiter an Modellierungswerkzeuge gebunden sind. Diese Darstellung der Modelle wird auch als *Boundary Representation Model*, kurz B-Rep-Modell, oder als Totkörper bezeichnet. In der Abbildung 2.4 ist beispielhaft ein Probemodell und dessen zugehörige B-Rep-Struktur zu sehen. Dabei sind die verschiedenen Flächen über ihre Zugehörigkeit zu bestimmten Kanten, Punkten oder Oberflächen, welche jeweils durch einen eindeutigen Schlüssel (TransientKey) bestimmt sind, miteinander verbunden. Weiterhin werden Informationen über die jeweilige Geometrie der Entitäten abgespeichert. Beispielsweise können Flächen als planar, zylindrisch oder torusförmig erkannt werden. Die Lage der Flächen im Koordinatensystem wird in der Form von Koordinatendaten zugehöriger Punkte definiert. Mit der Vektorsumme aller Normalenvektoren wird das Volumen in Abhängigkeit der umschließenden Flächen geprüft. Diese Summe ist nur dann Null, wenn die Oberfläche vollständig geschlossen ist [VDI2209]. Die Austauschformate STEP (Standard for Exchange for Product Data) und IGES (Initial Graphics Exchange Specification) nutzen B-Rep-Modelle zur Darstellung der Oberflächen. In den meisten heutigen 3D-CAD-Anwendungen werden hybride Modelle verwendet, welche eine Mischung aus konstruktiver Festkörpergeometrie (engl. constructive solid geometry (CSG)) und B-Rep-Modellen darstellen [EIGN14]. CSG-Modelle entstehen durch Boole'sche (mengen-theoretische) Verknüpfungen (Vereinigung, Subtraktion/Differenz, Durchschnitt/Verschneidung) von Grundvolumina (Primitives), welche im sogenannten CSG-Baum gespeichert werden [VDI2209].

In verschiedenen graphenbasierten Darstellungsschemata sind Features in der Regel vordefinierte Schablonen und werden durch die Informationen zur erforderlichen Anzahl von Flächen für die Zusammensetzung eines Zielfeatures sowie die topologischen und geometrischen Beziehungen, zwischen den zusammengesetzten Flächen, eingeschränkt. In graphischen Darstellungsschemata, wie dem AAG, stellen Knoten und Kanten normalerweise die Flächen und Kanten aus der B-Rep-Struktur dar, die mit einigen Attributen, wie Konvexität und Konkavität der Kanten, Art der Fläche, Rechtwinkligkeit, Parallelität oder Tangentialität von Kanten und

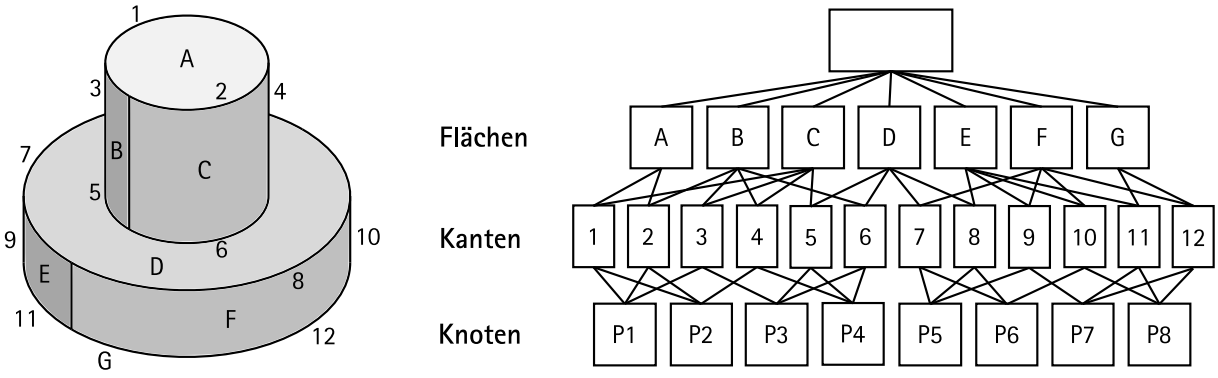


Abbildung 2.4: Probemodell (links) und dazugehörige B-Rep-Struktur (in Anlehnung an [EL-M03])

Flächen, verbunden sind [SHI20]. Ein AAG kann als Graph G=(N,A,T) definiert werden, wobei N die Menge der Knoten, A die Menge der Kanten und T die Menge der Attribute zu den Kanten in A ist [ZHU15]. Hierbei stellt ein Knoten eine bestimmte Fläche eines Modells dar und eine Kante die Verbindung zwischen zwei Flächen, einschließlich Konkavität oder Konvexität. Anschließend werden die Features als Teilgraphen aus dem vollständigen Graphen extrahiert. In Abbildung 2.5 ist ein AAG für ein Fräsbauteil abgebildet, bei welchem nicht nur die Konnektivität der einzelnen Flächen in einem Modell dargestellt ist, sondern auch die Charakteristik der Verbindung.

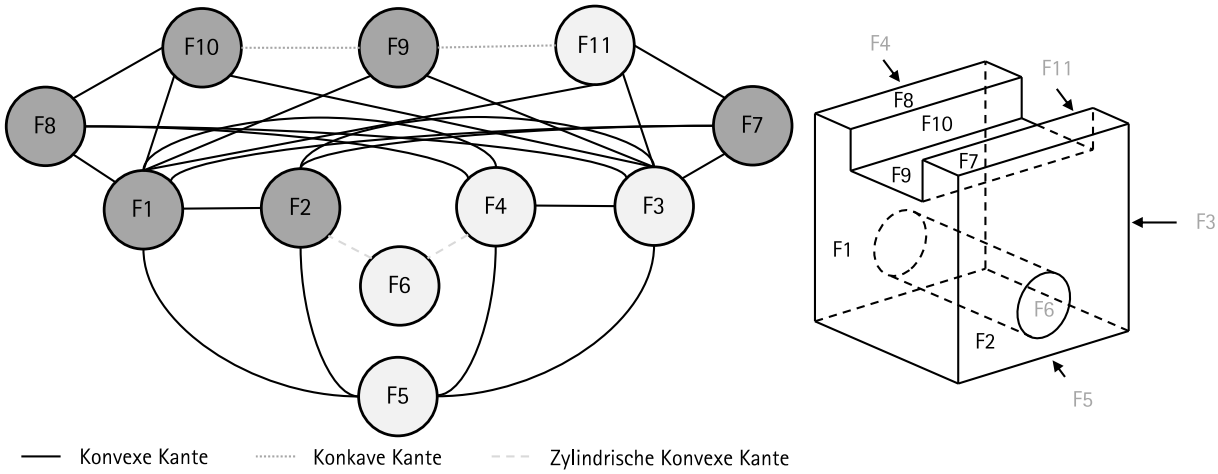


Abbildung 2.5: Attributierter Adjazenzgraph für ein Fräsbauteil

Im Unterschied zum featurebasierten CAD, können bei der featurebasierten Modellierung Methoden zur Formvervollständigung genutzt werden, um halbfertige Bauteile automatisch zu vervollständigen [KRAH21]. Es ist allerdings zu bedenken, dass es sich nur um einen eingeschränkten Synthesebegriff handelt, denn es können mit diesem Konzept lediglich bekannte Zusammenhänge zwischen vordefinierten Eigenschaften und vordefinierten Features reproduziert und geschickt ausgenutzt werden. „Reine Synthese“, d. h. die Erzeugung vorher überhaupt

nicht bekannter Features, womöglich noch aufgrund neuer, bei der Erstellung des Programms nicht vorausgedachter Eigenschaften, ist nicht möglich [Lach13, Webe00].

2.2 Agenten und Multi-Agentensysteme

Die in Kapitel 2.1.2 beschriebenen KBES werden in der Regel als zentrale Systeme aufgebaut. Zwar ermöglichen die meisten Konfiguratoren das Einlesen von Informationen aus unterschiedlichen Quellen, aber die Inferenz bzw. Schlussfolgerung findet zentral statt. Dies führt zu aufwendigen Konsistenzprüfungen, sobald Veränderungen an dem System vorgenommen werden. Um diesem Umstand entgegenzuwirken, wurden IT-Lösungen entwickelt, welche auf dem Paradigma der verteilten künstlichen Intelligenz (engl. distributed artificial intelligence (DAI)) und Softwareagenten basieren [Dost16]. Beispielsweise haben MAS von Wissenschaftlerinnen und Wissenschaftlern verschiedener Disziplinen große Aufmerksamkeit erhalten, da diese ein Mittel zur Lösung komplexer Probleme darstellen, indem sie in kleinere Aufgaben unterteilt werden. Die einzelnen Aufgaben werden autonomen Einheiten, den sogenannten Agenten, zugewiesen [Dorr18]. Die Haupteigenschaften von Agenten, wie Autonomie, Verteilung, Anpassungsfähigkeit, Flexibilität, Kooperation und Kommunikation [Wool09], ermöglichen einerseits die Fähigkeit, Heterogenität und Verteilung von Agenten, Wissen und Ereignissen, im Entwurfsraum effizient zu handhaben und andererseits die Erleichterung des Austauschs von Informationen und die gemeinsame Nutzung von Ressourcen durch Kommunikation und Kooperation zwischen Agenten [Chou15]. Somit stellen MAS eines der vielversprechendsten technologischen Paradigmen für die Entwicklung offener, verteilter, kooperativer und intelligenter Softwaresysteme dar [Hao06].

In der Literatur gibt es keine allgemeingültige Definition für einen Agenten, denn die vorhandenen Definitionen sind kontextabhängig und konzentrieren sich auf verschiedene Aspekte der agentenbasierten Technologie [Diak20]. Eine der am häufigsten zitierten Definitionen des Agenten ist die von Wooldridge und Jennings [Wool09], die den Agenten als „ein Computersystem, das in seiner situierten Umgebung zu autonomen Handlungen fähig ist, um seine speziellen Ziele zu erreichen“ beschreibt. Zudem kann ein Agent als eigenständiges wissensbasiertes System aufgefasst werden, da er in der Lage ist, in einer dynamischen Umgebung wahrzunehmen, zu schlussfolgern, sich anzupassen, zu lernen, zu kooperieren und zu delegieren, um spezielle Probleme zu bewältigen [Liu08].

Ein MAS ist im Wesentlichen ein System, das als eine Reihe autonomer Agenten strukturiert ist, die in der Lage sind, ihr Verhalten flexibel an sich ändernde Betriebsbedingungen anzu-

passen [Weyn10]. In diesen Systemen besitzt kein einzelner Agent das gesamte Wissen, das zur Lösung eines Problems erforderlich ist [Orti04], denn dieses wird durch den Austausch von Nachrichten miteinander kommuniziert [Berr16]. Ein wirksames agentenbasiertes System sollte nicht nur über starke Kommunikationsfähigkeiten und Problemlösungsfunktionen verfügen, sondern auch anpassungs- und erweiterungsfähig sein [Jia04]. Das Problem der Argumentation wird in den meisten MAS mithilfe von KI-Techniken, wie regelbasierter Inferenz, klassischer Planung, verschiedenen logischen Formalismen und Constraint-Satisfaction Techniken, gelöst.

Die Hauptvorteile von MAS sind nach Jennings [Jenn95]:

- *Offenheit*: Ermöglichung der Verbindung und Interoperation mehrerer bestehender Systeme, z. B. Expertensysteme, Entscheidungsunterstützungssysteme oder bestehende Netzwerkprotokolle.
- *Komplexität*: Lösung von Problemen, die für einen einzelnen zentralisierten Agenten zu groß sind, z. B. aufgrund von Ressourcenbeschränkungen oder aus Gründen der Robustheit.
- *Verteilung von Daten, Kontrolle, Fachwissen oder Ressourcen*: Verbesserung der Skalierbarkeit, Lösungen für inhärent verteilte Probleme, z. B. Telekommunikationssteuerung oder Workflow-Management sowie Lösungen, bei denen das Fachwissen verteilt bzw. heterogenes Denken gefordert ist.

MAS sollen die Entwicklung von Anwendungen ermöglichen, bei denen Agenten und Menschen gemeinsam Dienste für andere Menschen oder Agenten, in einer Umgebung mit vollständiger Integration, erbringen können [Pala20]. MAS unterscheiden sich im Vergleich zu OOP und Expertensystemen durch die erweiterte Nutzung von Zielen, die kontextspezifische Kommunikation und die Erforderlichkeit einer Kooperations- und Wissensebene, während Systeme wie OOP und Expertensysteme (höchstens) auf der Symbol- und Wissensebene arbeiten [Nwan99b].

2.2.1 Agentenorientierte Methoden

Seit den 1990er-Jahren haben sich MAS zu einem der aktivsten Forschungs- und Entwicklungsbereiche im Software-Engineering entwickelt. Angesichts dessen sind auch einige Methoden für die Entwicklung agentenorientierter Systeme erarbeitet worden. Agentenorientierte Methoden können in die Hauptklassen der allgemeinen und domänenspezifischen Methoden eingeteilt

werden [JIA09]. Die VDI-Richtlinie 2653-2 [VDI2653-2] unterscheidet zudem drei Ansätze, welche den Methoden zur Entwicklung von Agenten zugrunde liegen:

- *Wissensbasierte Ansätze (WB)*: Die frühesten Ansätze, um agentenorientierte Systeme zu spezifizieren, sind inspiriert durch die Modellierung von wissensbasierten Systemen. Eine Methode in diesem Bereich ist *MAS-CommonKADS*.
- *Agentenorientierte Ansätze (AO)*: Die Methode *Gaia* basiert auf der Sichtweise eines MAS als eine Rechenorganisation, die aus verschiedenen interagierenden Rollen besteht. Zudem werden die MAS unter dem Gesichtspunkt der Organisation entwickelt und schrittweise verfeinert.
- *Erweiterungen von objektorientierten Ansätzen (OO)*: Der Ansatz basiert auf der Anpassung und/oder Erweiterung bereits etablierter und allgemein anerkannter, objektorientierter Methoden, um Konzepte und Modellierungsmittel der Agentenorientierung. Hierzu zählt die *MaSE-Methode* (Multiagent Systems Engineering), welche sich stark an der „vereinheitlichten Modellierungssprache“ (engl. Unified Modelling Language (UML)) und dem Unified Process orientiert.

MAS-CommonKADS

Schreiber [SCHR94] entwickelt mit CommonKADS (Common Knowledge Acquisition Documentation and Structuring) eine Vorgehensweise zur Entwicklung von wissensbasierten Systemen [VDI5610-2]. Ausgangspunkt der Methode sind eine Reihe von Modellvorlagen, die während der Entwicklung gefüllt und verfeinert werden können (Abbildung 2.6) [HOOG94]. Zu den Modellen, die Teil der CommonKADS Methode sind, gehören das Organisationsmodell, das Aufgabenmodell, das Agentenmodell, das Kompetenzmodell, das Kommunikationsmodell und das Entwurfsmodell.

- *Organisationsmodell*: Das Organisationsmodell identifiziert und analysiert spezifische Organisations- und Anwendungskontexte, in denen das MAS implementiert werden soll und erforscht die Auswirkungen der Implementierung [IGLE96].
- *Aufgabenmodell*: Das Aufgabenmodell liefert unabhängig von einem Agenten eine hierarchische Beschreibung der Aufgaben, die für die Realisierung einer Funktion in der Organisation erforderlich sind [HOOG94].
- *Agentenmodell*: Im Agentenmodell werden alle relevanten Fähigkeiten und Eigenschaften von Agenten beschrieben, um die Aufgaben zu erfüllen. Der Agent kann ein Mensch, eine Computersoftware oder eine andere Entität sein, welche die Aufgabe erfüllen kann.

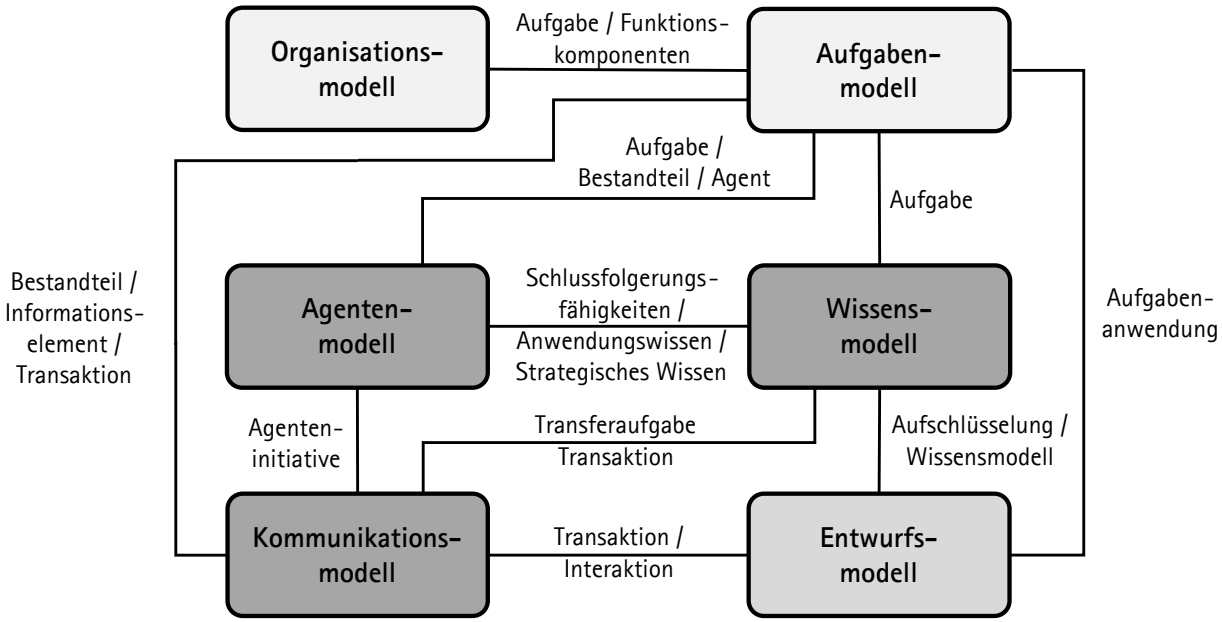


Abbildung 2.6: Beziehungen zwischen den CommonKADS-Modellen (in Anlehnung an [HOOG94])

- *Kompetenzmodell*: Ein zentrales Modell in der CommonKADS-Methodik ist das Kompetenzmodell, welches das Problemlösungsverhalten eines Agenten, in Bezug auf das Wissen, das zur Durchführung einer bestimmten Aufgabe angewendet wird, modelliert [SCHR94]. Das Kompetenzmodell unterscheidet zwischen Anwendungswissen und Problemlösungswissen. Es ist in drei Ebenen unterteilt: Domänenebene, welche deklaratives Wissen über die Domäne enthält, Inferenzebene, welche eine Bibliothek generischer Inferenzstrukturen darstellt und Aufgabenebene, in welcher die Inferenzen angeordnet werden [IGLE96].
- *Kommunikationsmodell*: Innerhalb des Kommunikationsmodells werden die verschiedenen Interaktionen zwischen den verschiedenen Agenten spezifiziert. Unter anderem wird angegeben, welche Art von Information zwischen den Agenten ausgetauscht wird und welcher Agent die Interaktion initiiert [STUD98].
- *Entwurfsmodell*: Mögliche Strategien und relevante technische Faktoren für die Implementierung in eine computerbasierte Umgebung werden mithilfe des Entwurfsmodells behandelt. Hierbei werden u. a. die Plattform sowie relevante mathematische Konzepte und Algorithmen für die Implementierung definiert. Zudem hält es die wichtigen Entscheidungen fest, die beim Entwurf einer Anwendung getroffen werden [HOOG94].

Iglesias et al. [IGLE96] haben die CommonKADS-Methode für wissensbasierte Systeme im Hinblick auf die Verwendung mit MAS erweitert. In der Methode *MAS-CommonKADS* wird

ein neues Modell, das Koordinationsmodell, vorgeschlagen. Es ist ein alternatives Modell zum Kommunikationsmodell für die Modellierung der Interaktion zwischen Agenten. Das Kommunikationsmodell könnte weiterhin für die Mensch-Computer-Interaktion verwendet werden [IGLE96]. Zusammenfassend lässt sich sagen, dass die Agenten- und Kommunikationsmodelle die Rollen und Fähigkeiten von Agenten und die Transaktionen, die während eines bestimmten Prozesses auftreten, darstellen. Sie sind ein wertvoller Bestandteil der gesamten CommonKADS-Suite und können verwendet werden, um die Organisation von erworbenem Wissen, die Reorganisation eines Geschäftsprozesses oder den Entwurf eines wissensbasierten Systems bzw. MAS zu unterstützen [KING01].

Gaia und ROADMAP

Gaia (Generic Architecture for Information Availability) [WOOL00] ist die erste allgemeine Methodik für die agentenorientierte Analyse und den Entwurf. Sie ist auf ein breites Spektrum von MAS anwendbar und sie behandelt sowohl die Makroebene (Gesellschaft), als auch die Mikroebene (Agenten) von Systemen. Gaia umfasst alle Aspekte, die für die Beschreibung von Agentengesellschaften wichtig sind [JIA09]. In der Analysephase werden das Rollenmodell und das Interaktionsmodell erstellt, die das System als eine Reihe von interagierenden abstrakten Rollen darstellen (Abbildung 2.7). Diese beiden Modelle werden dann als Input für die Entwurfsphase verwendet, in der ein Agentenmodell, ein Servicemodell und ein Bekanntschaftsmodell definiert werden, um eine vollständige Entwurfsspezifikation des MAS für die anschließende Implementierungsphase zu erstellen, welche nicht von Gaia behandelt wird [CERN04, MORA03]. Ziel der Analysephase ist es, durch eine abstrakte, implementierungsabhängige Sichtweise ein grundlegendes Verständnis der Struktur des zu realisierenden Systems zu gewinnen [WEIS05].

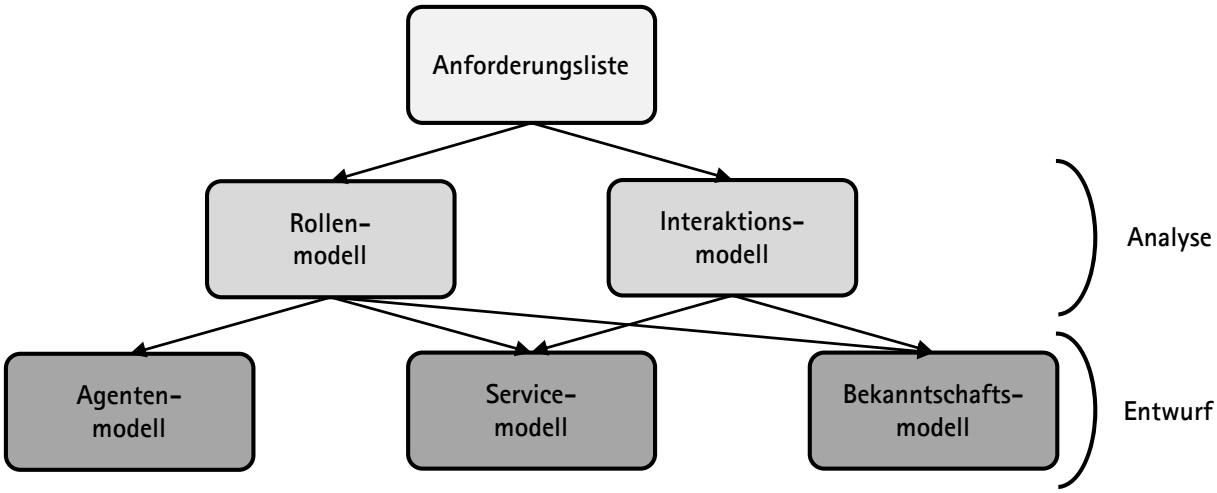


Abbildung 2.7: Beziehungen zwischen Gaia-Modellen (in Anlehnung an [WOOL00])

Die beiden in der Analysephase gewonnenen Modelle sind zwar feinkörnig, aber von relativ abstrakter Natur. In der Entwurfsphase werden die abstrakten Konstrukte der Analysephase, d. h. die Rollen und Protokolle, welche in den Rollen- und Interaktionsmodellen repräsentiert werden, in konkrete Konstrukte, d. h. in Agententypen, die zur Laufzeit instanziiert werden, abgebildet [CERN04]. Hierzu werden aus diesen abstrakten Modellen drei weitere Modelle: das Agentenmodell, das die Typen von Agenten spezifiziert, die das eigentliche System bilden; das Servicemodell, das die Dienste spezifiziert, die von diesen Agententypen implementiert werden und das Bekanntschaftsmodell, das die Kommunikationsbeziehungen zwischen den Agententypen darstellt [WEIS05]. Im Unterschied zur üblichen Zielsetzung beim Entwurf zielt Gaia damit in dieser Phase nicht unmittelbar auf das Erreichen von implementierungsnahen Modellen ab, sondern liefert nur einen abstrakten Entwurf auf hoher Ebene [DELO01].

Die Methode ROADMAP (Role Oriented Analysis and Design for Multi-Agent Programming) wurde als Erweiterung von Gaia vorgeschlagen [JUAN02]. Im Fokus dieser Methode ist die Analyse und der Entwurf von offenen agentenorientierten Systemen und damit von Systemen, für die Gaia nur sehr bedingt geeignet ist [JUAN04]. Besondere Merkmale von ROADMAP sind: (1) die Integration der Identifikation von Systemanforderungen, (2) die Erweiterung der Modelle um ein Anwendungsfallmodell zur Charakterisierung von System-Mensch-Interaktionen, ein Umweltmodell zur Charakterisierung der Systemumgebung und ein Wissensmodell zur Beschreibung des erforderlichen systeminternen Wissens und (3) die Darstellung des Rollenmodells als Rollenhierarchie, wobei mittels Aggregation aus atomaren Rollen zusammengesetzte Rollen gebildet werden [CERN04, WEIS05].

MaSE

Das Aufkommen von MAS hat viele Disziplinen in dem Bemühen zusammengeführt, verteilte, intelligente und robuste Anwendungen zu entwickeln. Deloach et al. [DELO01] haben eine Methodik für den gesamten Lebenszyklus entwickelt, um heterogene MAS zu analysieren, zu entwerfen und zu entwickeln. Hier stellt MaSE eine weitere Abstraktion des objektorientierten Paradigmas dar, in dem Agenten eine Spezialisierung von Objekten sind. Die MaSE-Methode ist unabhängig von einer bestimmten Agentenarchitektur, Programmiersprache oder einem Kommunikationsframework [JIA09]. Die MaSE-Methode wird in eine Analyse- und eine Entwurfsphase unterteilt (Abbildung 2.8).

Die Analysephase zielt darauf ab, die Systemziele aus funktionalen Anforderungen zu definieren und anschließend Rollen abzuleiten, die diese Ziele erfüllen. Der Prozess gliedert sich in drei Schritte: Ziele erfassen, Anwendungsfälle erarbeiten und Rollen verfeinern [DELO01, DELO05]:

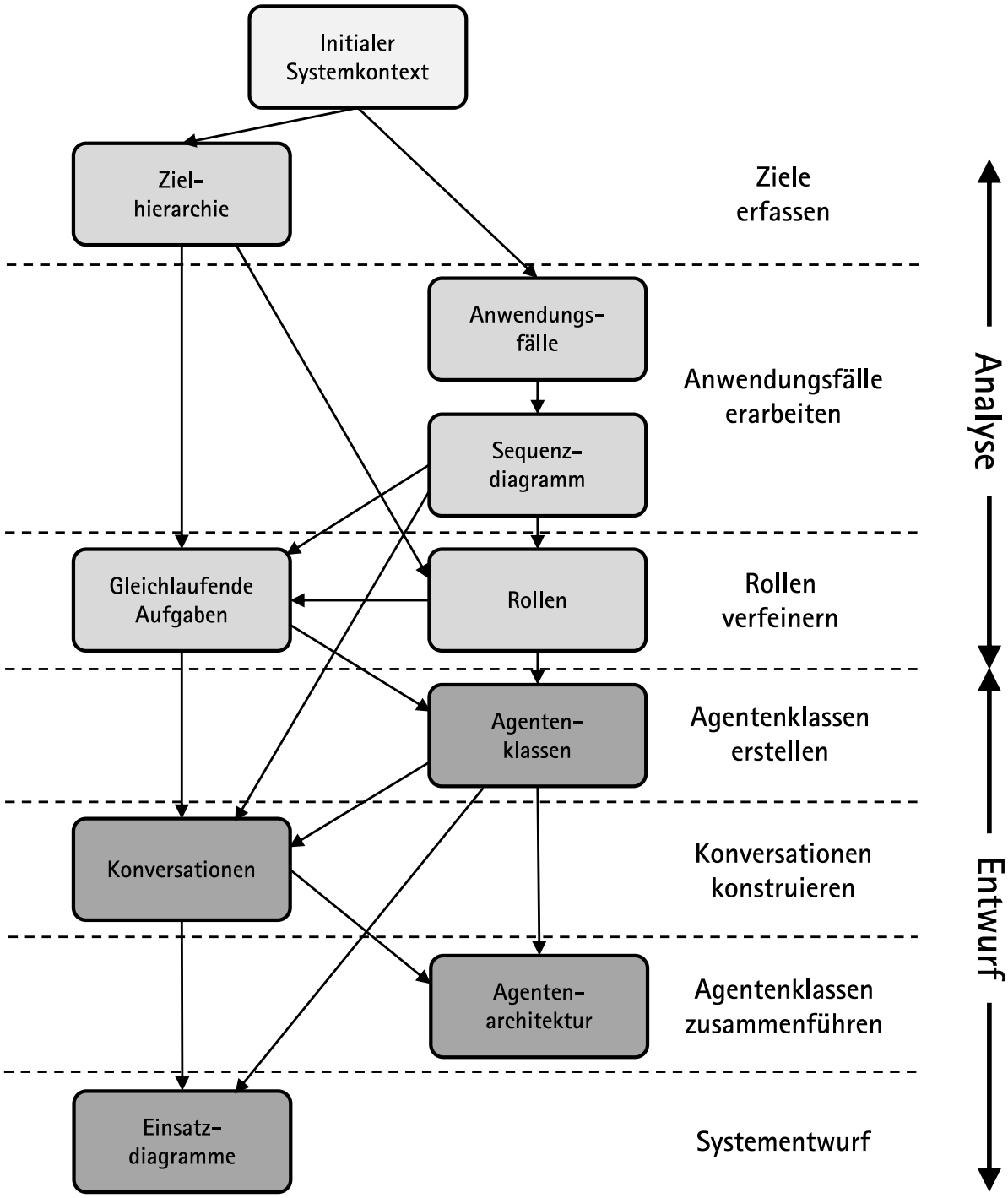


Abbildung 2.8: Phasen der MaSE-Methode (in Anlehnung an [DELO01])

- *Ziele erfassen*: Im ersten Schritt werden die Ziele aus den Anforderungen extrahiert und in das Gesamtsystemziel und die Teilziele aufgeteilt. Die Erfassung der Ziele unterstützt eine stabile Modellierung, Funktionen, Prozessen oder Informationsstrukturen hingegen, ändern sich oft im Laufe der Zeit.
- *Anwendungsfälle erarbeiten*: Im zweiten Schritt werden die Ziele in Anwendungsfälle übersetzt und die Nutzungsszenarien im Detail beschrieben. Anwendungsfälle definieren grundlegende Szenarien, die ein System ausführen können sollte. Die Sequenzdiagramme erfassen die Anwendungsfälle als eine Reihe von Ereignissen zwischen den Rollen, aus denen das System besteht.
- *Rollen verfeinern*: In diesem Schritt werden Rollen zur Umsetzung der Ziele abgeleitet, da Rollen typischerweise zielorientiert sind und sich gut auf Agenten abbilden lassen.

Der Zweck der Entwurfsphase ist es, die Ziele und Rollen in ein MAS zu transformieren, in dem Agenten definiert werden und ihre Kommunikation untereinander beschrieben wird. Die MaSE-Entwurfsphase besteht aus vier Schritten: Agentenklassen erstellen, Konversationen konstruieren, Agentenklassen zusammenführen und Systementwurf [DeLo01, DeLo05]:

- *Agentenklassen erstellen*: In diesem Schritt ordnet die Ingenieurin bzw. der Ingenieur den einzelnen Agenten Rollen zu. Eine Agentenklasse ist eine Schablone für einen Agententyp im MAS, analog zur Objektklasse in der objektorientierten Programmierung.
- *Konversationen konstruieren*: Konversationen modellieren die Kommunikation zwischen zwei Agenten, die als Koordinationsprotokolle definiert sind. Dabei werden die ausgetauschten Nachrichten als Performatives definiert, die sowohl den Zweck der Nachricht als auch eine Reihe von Parametern enthalten, die den Inhalt darstellen.
- *Agentenklassen zusammenführen*: In diesem Schritt wird die interne Architektur der Agenten definiert, die insbesondere die Inferenzmaschine und die Wissensbasis umfasst.
- *Systementwurf*: Der letzte Schritt beinhaltet die Auswahl der aktuellen Konfiguration des Systems, wie z. B. die Anzahl der Agenten und Agententypen oder die Plattform, auf der das MAS eingesetzt wird.

2.2.2 Architekturen für die Interaktion der Agenten

Die Architektur eines MAS ist die abstrakte Definition der Kommunikation und der Interaktion zwischen den Agenten. Die Architekturen reichen von rein logikbasierten, über reaktive oder verhaltensorientierte Architekturen, die in einer einfachen Anregungs-Reaktions-Weise

funktionieren, bspw. die auf der Subsumptionsarchitektur von Brooks [Broo87] basierenden Architekturen, hin zu eher planenden bzw. deliberativen Architekturen, die über ihre Handlungen nachdenken, wie z. B. Architekturen, die auf dem Belief-Desire-Intention-Modell (BDI) aufbauen [Geor92]. Diese Architekturen lassen sich auch untereinander zu sogenannten Hybriden oder Schichtarchitekturen kombinieren, sodass sich die Agentenarchitekturen in vier Hauptgruppen einteilen lassen [Bell07, Niu12]:

- *logikbasierte Architektur*: Die logikbasierte Struktur verwendet eine symbolische Umgebung und verändert diese durch logische Verfahren. Dadurch ist diese für Menschen nachvollziehbar. Allerdings lässt sich die reale Welt nur schwer symbolisch darstellen [Bell07].
- *Reaktive Architektur (Subsumptionsarchitektur)*: Reaktive Architekturen implementieren die Entscheidungsfindung als direkte Zuordnung einer Situation zu einer Handlung und basieren auf einem durch Sensordaten ausgelösten Anregungs-Reaktions-Mechanismus. Die reaktive Struktur besitzt mehrere Schichten, wobei die unteren Schichten über weniger Kontrolle verfügen [Bell07]. Reaktive Architekturen betrachten nur die aktuelle Situation und haben keine Möglichkeit Verhalten zu initiieren oder längerfristige Ziele zu berücksichtigen. Im Extremfall handeln rein reaktive Agenten erst dann, wenn sich etwas in der Umgebung ändert [Buss04]. Zudem kann es zu Konflikten kommen, wenn mehrere Reize auf den Sensor für die Wahrnehmung eines Agenten einwirken, woraufhin Brooks [Broo87] eine Subsumptionsarchitektur entworfen hat, die dem Sensor Unterfunktionen, sogenannte Behaviours (dt. Verhaltensweisen), zuweist. Diese agieren synchron und werden nur dann beendet, wenn sie nicht von anderen Behaviours unterbrochen werden [Plap22a].
- *Deliberative Agentenarchitektur (BDI)*: Deliberative Agentenarchitekturen stellen explizit Ziele dar und bilden Pläne darüber, wie sich der Agent in der Zukunft verhalten will, um seine Ziele zu erreichen. Die wohl bekannteste deliberative Agentenarchitektur ist die BDI-Architektur von Rao und Georgeff [Geor92]. BDI-basierte Agenten sind durch Überzeugungen, Ziele und Pläne gekennzeichnet. Die Überzeugungen (engl. beliefs) repräsentieren das Wissen des Agenten über sich selbst und die Umgebung. Ziele (engl. desires) stellen Zustände der Welt dar, die der Agent herbeiführen möchte. Absichten oder Pläne (engl. intentions) sind die Mittel, mit denen der Agent handeln kann, um seine Ziele zu erreichen. BDI-Agenten eignen sich gut für unvorhersehbare Szenarien, die eine dynamische Entscheidungsfindung erfordern, da sie in der Lage sind, den besten Plan zur Erreichung eines Ziels, unter Berücksichtigung ihrer aktuellen Überzeugungen,

zu wählen. Hierbei führen die Agenten Aktionen aus, sobald durch äußere Einflüsse die eigenen Überzeugungen oder Ziele verändert werden. In den meisten BDI-basierten Ansätzen wird der Prozess des wiederholten Auswählens und Ausführens von Plänen als Argumentationszyklus des Agenten bezeichnet [ALZE20, BELL07, BORD07].

- *Schichtarchitekturen (hybrid)*: Schichtarchitekturen ermöglichen sowohl reaktives als auch deliberatives Agentenverhalten [BELL07]. Ein bekanntes Modell ist die InteRRaP-Agenten-Struktur von Fischer et al. [FISC94]. Durch die hierarchische Struktur soll ein effizientes Lösen von Aufgaben erreicht werden. Ein Agent ist hierbei in verschiedene Komponenten und eine Wissensbasis mit mehreren Schichten aufgeteilt. Zu jeder Komponente gehört eine spezifische Wissensbasis. Eine Komponente kann jeweils mit der nächsthöheren oder niedrigeren Komponente kommunizieren und so Aufgaben austauschen. Fehlt einer Komponente das nötige Wissen zur Lösung der Aufgaben, so gibt es diese an die nächsthöhere Schicht ab und kann sich mit einer neuen Aufgabe beschäftigen. So kann der Agent simultan an mehreren Aufgaben arbeiten [FISC94]. Der größte Nachteil ist, dass die Architektur von allen Schichten abhängt und nicht fehlertolerant ist, d. h. wenn eine Schicht ausfällt, fällt das gesamte System aus [BELL07].

Um einen standardisierten Aufbau und insbesondere eine standardisierte Kommunikation zwischen MAS zu ermöglichen, hat die *Foundation for Intelligent Physical Agents* (FIPA) eine Reihe von anwendungsunabhängigen Spezifikationen veröffentlicht, die fast alle Themen der Agententechnologie abdecken, einschließlich Agentenlebenszyklusmanagement, Agentenkommunikation, Ontologie, Agentenmobilität und Sicherheit [HAO05]. Die FIPA wurde im Dezember 1995 als gemeinnützige Organisation zur Erstellung von Spezifikationen für offene Agentenschnittstellen gegründet. Ihr Hauptzweck ist es, einen Konsens zwischen verschiedenen Organisationen zu schaffen, die sich mit der Agententechnologie befassen und so den Prozess der Standardisierung von Agenten einzuleiten [O'BR98].

Der FIPA-Ansatz zur Entwicklung von MAS basiert auf einem minimalen Rahmen für das Management von Agenten in einer offenen Umgebung [LIU04]. Die FIPA hat drei obligatorische Rollen für eine Agentenplattform identifiziert [BELL07]. Das Agentenmanagementsystem (AMS) übt die Kontrolle über den Zugang und die Nutzung der Agentenplattform (AP) aus. Es ist verantwortlich für die Führung eines Verzeichnisses der ansässigen Agenten und für die Verwaltung ihres Lebenszyklus. Der Nachrichtentransportdienst ist ein Dienst, der von einer AP bereitgestellt wird, um Nachrichten mittels FIPA-Agentenkommunikationssprache (engl. agent communication language (ACL)) zwischen Agenten auf einer bestimmten AP und zwischen Agenten auf verschiedenen APs zu transportieren. Der Verzeichnisdienst (engl. directory

facilitator (DF)) ist eine optionale Komponente eines AP, der anderen Agenten „Gelbe-Seiten-Dienste“ zur Verfügung stellt. Das Agentenmanagement kann als eine Anwendungsdomäne für intelligente Agenten betrachtet werden. Eine Agentenmanagement-Ontologie (Abbildung 2.9) wird definiert, um den Inhalt der kommunikativen Akte des Agentenmanagements auszudrücken, z. B. eine Agentenbeschreibung [O'Br98]. Die Architektur definiert, auf einer abstrakten Ebene, wie zwei Agenten einander finden und miteinander kommunizieren können. Dies erfolgt, indem sie sich registrieren und Nachrichten, welche die grundlegende Form der Kommunikation zwischen Agenten sind, austauschen [Bell07].

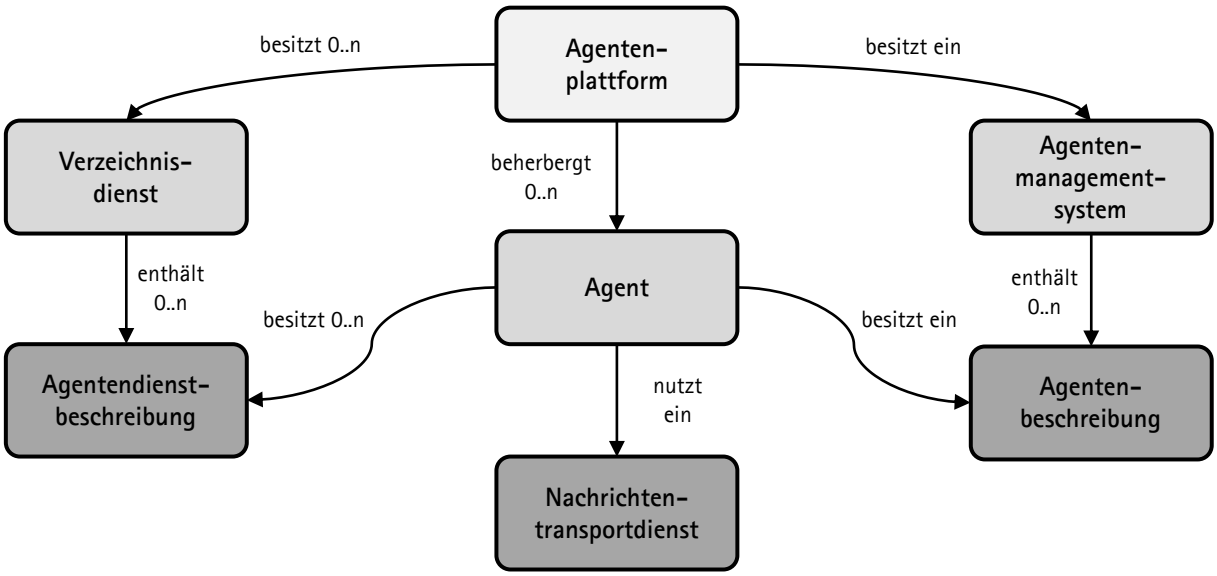


Abbildung 2.9: Beschreibung der Agenten-Management-Ontologie nach FIPA (in Anlehnung an [Bell07])

Des Weiteren kann die interne Struktur eines MAS bzw. einer Organisation folgendermaßen unterteilt werden [Dorr18]:

- *Flach* (homogen): Dies ist die einfachste Organisationsstruktur, bei der alle Agenten als gleichberechtigt angesehen werden. Sie gleichen sich in lokalen Zielen, internen Zuständen, Inferenzmechanismen und möglichen Aktionen. Der einzige Unterschied zwischen ihnen ist der Raum, in dem sie agieren.
- *Hierarchisch*: In der hierarchischen Organisation haben die Agenten baumartige Beziehungen.
- *Holonisch*: In der holonischen Organisation sind die Agenten in mehreren Gruppen organisiert, die als Holons bezeichnet werden. Die Gruppenzuteilung erfolgt anhand bestimmter Merkmale, z. B. der Heterogenität oder der Wahrnehmungsfähigkeit der Agenten im Holon.

- *Koalition*: Bei der Koalitionsorganisation werden Agenten vorübergehend anhand der Grundlage ihres Ziels gruppiert. Diese Organisation ist geeignet, wenn eine Ansammlung von Agenten mit ähnlichen Zielen im MAS existiert, deren Zusammenarbeit sie beim Erreichen ihres Ziels verbindet.
- *Team*: Bei der Teamorganisation bilden die Agenten eine Gruppe (Team) und definieren ein Gruppenziel, das sich von ihrem eigenen Ziel unterscheidet.
- *Matrix*: In der Matrixorganisation wird jeder Agent von mindestens zwei Hauptagenten, den sogenannten Managern, verwaltet bzw. geleitet.
- *Kongregation*: In der Kongregationsorganisation schließen sich Agenten an einem Standort zu einer Kongregation zusammen, um die Anforderungen zu erfüllen, die sie allein nicht erfüllen können.

2.2.3 Kommunikation und Kollaboration

In einem MAS müssen die Agenten untereinander Informationen austauschen, verhandeln und kooperieren [Wool09]. Daher ist es notwendig Interaktionstechniken zu definieren, die eine Kommunikation und Kooperation zwischen Agenten, mit den Anwendenden und Systemressourcen ermöglichen [Alze20]. Für eine solche Kommunikation des Wissens werden Konventionen auf drei Ebenen benötigt: das Format der Repräsentationssprache, das Kommunikationsprotokoll der Agenten und die Spezifikation des Inhalts des geteilten Wissens [Grub95]. Sobald Agenten kommunizieren können, kann die Abstraktionsebene auf die Koordinationsebene oder soziale Interaktion angehoben werden, auf der Verhandlungen möglich sind [Nwan99b]. Die Agenten kommunizieren über eine gegenseitig verständliche Kommunikationssprache und tauschen Informationen aus, um Probleme gemeinsam zu lösen [Sun01].

Eine Kollaboration ist ein kontrollierter Prozess, der in der Kollaborationsumgebung stattfindet und bei dem eine Gruppe von Akteuren gemeinsam auf eine Reihe von Zielen hinarbeitet. Bei der organisatorischen Strukturierung wird die zuvor festgelegte Struktur der Gesellschaft, d. h. die Rollen der verschiedenen Akteure und ihre Beziehungen zueinander, für die Koordination genutzt [Nwan99b], sodass das MAS in einer kohärenten Weise handelt [Opre04]. Um effektiv verhandeln zu können, müssen die Beteiligten von den Überzeugungen, den Wünschen und Absichten der anderen Beteiligten Kenntnis haben [Orti04]. Typische Verhandlungsmechanismen basieren auf der Spieltheorie, auf einer Form von Planung oder auf von Menschen inspirierten Verhandlungen [Nwan99b]. Die wichtigsten Schritte einer Verhandlung sind der Austausch von Informationen, die separate Bewertung der Informationen aus eigenen Per-

spektiven und die Auswahl einer endgültigen Einigung durch gegenseitige Auswahl [OPRE04]. Ein integratives Verhandlungsmodell ist durch drei Schlüsselinformationen gekennzeichnet: das Verhandlungsprotokoll, die Liste der Themen, über die verhandelt wird und das von den Agenten verwendete Argumentationsmodell bzw. die Verhandlungsstrategie [GANZ07]. In manchen Situationen sind sehr einfache Entscheidungsregeln, wie Zuordnungsregeln oder vertragsnetzartige Verhandlungsverfahren, erfolgreich. In anderen Situationen scheint es sinnvoller, zentralisierte Planungsschemata zu verwenden [MÖN06].

Eine wichtige Koordinationstechnik für die Aufgaben- und Ressourcenverteilung zwischen Agenten und die Festlegung der Organisationsstruktur ist das Vertragsnetzprotokoll (engl. contract-net protocol) von Davis und Smith [DAVI83], in welchem ein Agent als Initiator eine Aufgabe von einem oder mehreren anderen Agenten, den Teilnehmenden, ausführen lässt und darüber hinaus eine Funktion optimieren möchte, welche die Aufgabe charakterisiert (Abbildung 2.10) [BELL07]. Die Einfachheit des Systems macht es zu einem der am weitesten verbreiteten Koordinationsmechanismen mit vielen Varianten in der Literatur [NWAN99b].

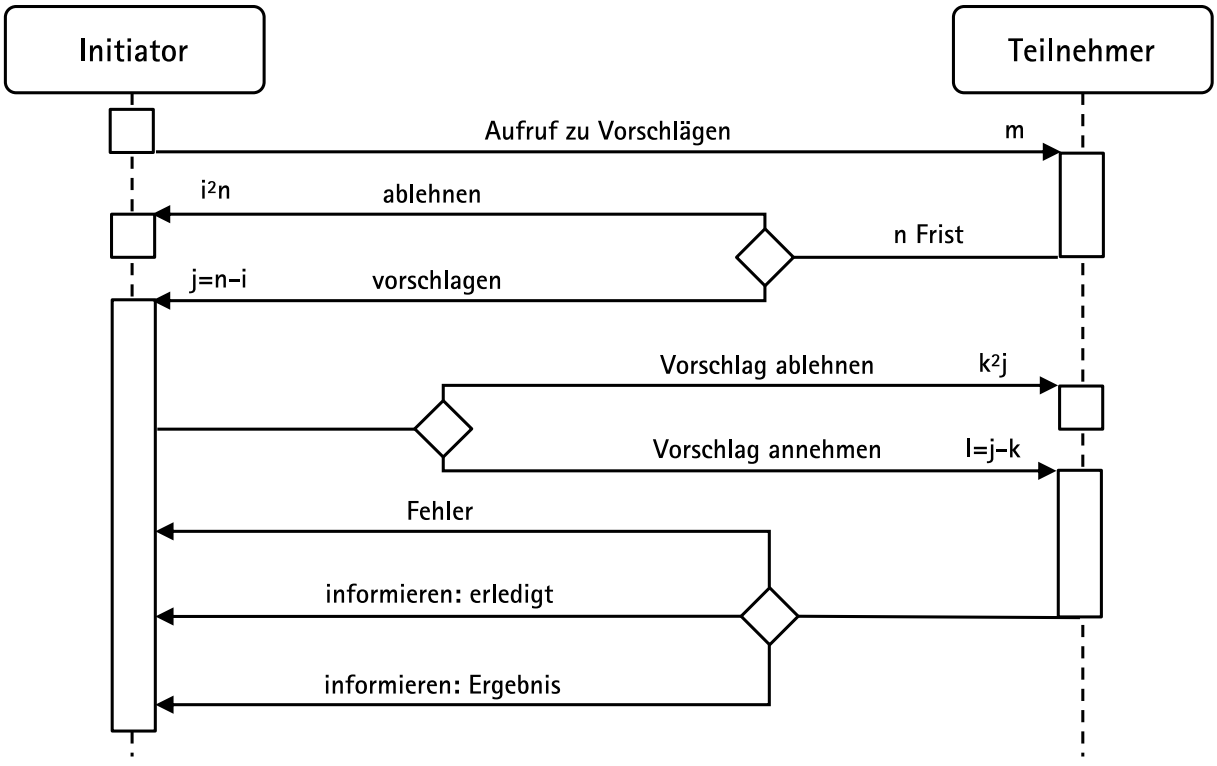


Abbildung 2.10: Das FIPA Vertragsnetz-Interaktions-Protkoll (in Anlehnung an [BELL07])

Um den Agenten einen Nachrichtenaustausch innerhalb der BDI-Architektur zu ermöglichen, kann auf die Theorie der Sprechakte (engl. speech-act) zurückgegriffen werden [SEAR69]. Die Sprechakttheorie geht von dem Grundsatz aus, dass Sprache eine Handlung ist bzw. eine Handlung auslösen kann. Der Unterschied von Sprechakten zu anderen („nicht-sprachlichen“)

Handlungen ist, dass der Bereich eines Sprechakts, welcher eine Handlung des Akteurs in der Umgebung herbeiführen möchte, typischerweise auf den mentalen Zustand der Empfangenden beschränkt ist [BORD07]. Beispiele hierfür können die Veränderung von Überzeugungen, Zielen oder Absichten durch die Nutzung von Sprechakten in Nachrichten sein.

Sprechakte werden im Allgemeinen nach ihrer illokutionären Kraft (engl. illocutionary force) klassifiziert, welche die „Art" der Äußerung angibt [BORD07]. Die Äußerung „Das Bauteil ist herstellbar", ist bspw. im Allgemeinen eine *informierende* (engl. inform) oder *sagende* (engl. tell) Handlung. Die perlokutionäre Kraft gibt an, welche Handlung der Sprecher mit der Äußerung zu erreichen versucht. Bei einer Äußerung wie „stelle das Bauteil her" ist die perlokutionäre Kraft der Zustand, den der Sprecher mit der Äußerung herbeizuführen hofft, obwohl die tatsächliche Wirkung einer Äußerung außerhalb der Kontrolle des Sprechers liegt. Einige der am häufigsten verwendeten Handlungen sind *informieren*, *anfordern*, *zustimmen*, *nicht verstehen* und *ablehnen*.

In der natürlichen Sprache sind die illokutionäre Kraft und die perlokutionäre Kraft implizit im Sprechakt und seinem Kontext enthalten. Bei der Anpassung der Theorie an die Agenten-Kommunikation werden die illokutionären Kräfte jedoch explizit formuliert, um die Verarbeitung des Kommunikationsakts zu vereinfachen [BORD07]. Diesen Aspekt machen sich auch die bekannten Agenten-Kommunikationssprachen KQML (engl. Knowledge Query and Manipulation Language) und FIPA ACL zunutze, welche ebenfalls auf der Sprechakttheorie basieren. Die verschiedenen Arten von Sprechakten im Kontext der Agenten-Kommunikation werden allgemein als *Performatives* bezeichnet [NWAN99a]. Die Hauptmerkmale dieser Sprachen stellen die Möglichkeiten dar, verschiedene Inhaltssprachen zu verwenden und die Konversationen durch vordefinierte Interaktionsprotokolle zu verwalten [BELL07].

Neben der Bereitstellung einer Kommunikation zwischen den Agenten über standardisierte Sprachen, ist die Speicherung und Bereitstellung von Wissen ein entscheidender Faktor für die Funktionsfähigkeit eines MAS. Das Problem ist, dass beim Entwurf eines MAS zur Definition der Bereichsontologie der Zweck oder die Aufgabe bekannt sein muss, für die es verwendet werden soll [NWAN99a]. Daher müssen die Agenten auch die gleiche Ontologie bzw. das gleiche Vokabular für gemeinsame Konzepte verwenden. Hierfür ist ein langfristiges Ziel, Bibliotheken von wiederverwendbaren Wissenskomponenten und wissensbasierten Diensten zu ermöglichen, die über Netzwerke aufgerufen werden können [GRUB95]. Ein bekannter organisatorischer Ansatz zur Strukturierung der Zusammenarbeit ist die Verhandlung über ein *Blackboard* (dt. Schwarzes Brett) [SHEN97]. Bei diesem Mechanismus koordinieren die Agenten ihre Aktionen, indem sie Wissen veröffentlichen und darauf zugreifen können [ORTI04].

2.2.4 Werkzeuge und Rahmenwerke

MAS bieten einen Bottom-up-Ansatz für die Analyse komplexer Systeme und sind ein aktives Forschungsfeld mit dem Ziel, mithilfe von Agenten verschiedene Phänomene zu modellieren und zu simulieren, die mit herkömmlichen analytischen Methoden nicht oder nur schwer abbildbar sind [PAL20]. Für diese Aufgabe werden häufig spezialisierter Werkzeuge und Rahmenwerke für die Programmierung von agentenbasierten Anwendungen eingesetzt. Im Folgenden werden vier Werkzeuge und Rahmenwerke vorgestellt, welche die Programmierung von MAS auf unterschiedliche Weise unterstützen. *ZEUS* ist das älteste der vorgestellten Werkzeuge und bietet eine grafische Umgebung für den Aufbau verteilter Agentensysteme sowie den Ansatz eines generalisierten Templates für Agenten. *Jason*, welches in Java programmiert ist, bietet darüber hinaus einen vollwertigen Interpreter für die Sprache *AgentSpeak*, welche auf der BDI-Architektur aufbaut. *JADE* ist ebenfalls in Java implementiert und die am weitesten verbreitete Umgebung zur Implementierung von Agentensystemen, mittels FIPA-konformer Middleware. Zu den neuesten Vertretern der MAS-Rahmenwerke zählt *SPADE*, welches in Python programmiert ist und auf der XMPP-Technologie (eXtensible Messaging and Presence Protocol) aufbaut. Des Weiteren können BDI-Agenten durch ein Plug-in implementiert werden.

ZEUS

Der agentenbasierte Ansatz, namens ZEUS, führt eine neue Abstraktionsebene der wissensbasierten Zusammenarbeit zwischen autonomen Systemen ein, welche die Interoperabilität, Skalierbarkeit und Rekonfigurierbarkeit verteilter Systeme verbessert [NWAN99b]. ZEUS [NWAN99b] ermöglicht die schnelle Entwicklung von Java-Agenten-Systemen durch die Bereitstellung einer Bibliothek von Agenten-Komponenten, welche eine visuelle Umgebung für die Erfassung von Benutzer-Spezifikationen, eine Agentenmodellierungsumgebung, die einen automatischen Agenten-Code-Generator enthält und eine Sammlung von Klassen, welche die Bausteine der einzelnen Agenten bilden, bereitstellt [BELL01]. Die Idee für ZEUS war es, ein relativ allgemeinen und anpassbaren Werkzeugkasten für den Aufbau von kollaborativen Agenten zu schaffen, welcher von Softwareingenieurinnen und -ingenieuren mit nur grundlegenden Kenntnissen in der Agententechnologie verwendet werden kann, um funktionale MAS zu erstellen [NWAN99b]. Erfahrene Anwendende sollten in der Lage sein, die Bibliothek von Komponenten auf Agentenebene leicht zu erweitern und neue Agenten, mit einer Kombination aus benutzerdefinierten und vom System bereitgestellten Komponenten, zu konfigurieren. Die Hauptannahmen bezüglich des Agentenverhaltens sind, dass die Agenten deliberativ, zielgerichtet und rational sind, immer wahrheitsgemäß im Umgang mit anderen Agenten sowie

vielseitig, sprich, sie können viele Ziele haben, eine Vielzahl von Aufgaben übernehmen und diese kontinuierlich abarbeiten. Als generellen Aufbau innerhalb des agentenbasierten Ansatzes wurde der *Generische ZEUS Agent* definiert (Abbildung 2.11), welcher aus den folgenden Komponenten besteht:

- *Briefkasten*, welcher die Kommunikation zwischen dem Agenten und anderen Agenten abwickelt.
- *Nachrichtenverwaltung*, die eingehende Nachrichten aus dem Briefkasten verarbeitet und sie an die entsprechenden Komponenten des Agenten weiterleitet.
- *Koordinationsstelle*, die Entscheidungen bezüglich der Ziele des Agenten trifft, z. B. wie die Ziele verfolgt werden und wann sie aufgegeben werden. Die Koordinationsstelle ist auch dafür verantwortlich, die Interaktionen des Agenten mit anderen Agenten zu koordinieren, indem sie die ihr bekannten Koordinationsprotokolle und -strategien verwendet, bspw. die verschiedenen Auktionsprotokolle oder das Vertragsnetzprotokoll.
- *Bekanntschaftsdatenbank*, welche die Beziehungen des Agenten zu anderen Agenten in der Gesellschaft und seine Überzeugungen über die Fähigkeiten dieser Agenten beschreibt. Die *Koordinationsstelle* verwendet die in dieser Datenbank enthaltenen Informationen, wenn sie Vereinbarungen zur Zusammenarbeit mit anderen Agenten trifft.
- *Planung und Disposition*, welche die Aufgaben des Agenten auf der Grundlage der, von der *Koordinationsstelle*, getroffenen Entscheidungen und der dem Agenten zur Verfügung stehenden Ressourcen und Aufgabenspezifikationen plant.
- *Ressourcendatenbank*, welche eine Liste von Ressourcen verwaltet, die dem Agenten gehören und ihm zur Verfügung stehen. Die Ressourcendatenbank unterstützt auch eine direkte Schnittstelle zu externen Systemen, die es ermöglicht, sich dynamisch mit proprietären Datenbanken zu verknüpfen und diese zu nutzen.
- *Ontologiedatenbank*, in der die logische Definition jedes Faktentyps gespeichert wird, die zulässigen Attribute und der Bereich der zulässigen Werte für jedes Attribut sowie alle Einschränkungen zwischen Attributwerten und alle Beziehungen zwischen den Attributen des Fakts und anderen Fakten.
- *Aufgaben-/Plandatenbank*, welche logische Beschreibungen der Planung eines dem Agenten bekannten Operators oder einer Aufgabe enthält.
- *Ausführungsüberwachung*, welche die interne Uhr des Agenten aufrechterhält und Aufgaben startet, stoppt und überwacht, die von der *Planung und Disposition* zur Ausführung

oder Beendigung eingeplant sind. Außerdem informiert sie den Planer über erfolgreiche und außergewöhnliche Beendigungsbedingungen der von ihr überwachten Aufgaben. Für die Verwaltung von Aufgaben verfügt die *Ausführungsüberwachung* auch über eine direkte Schnittstelle zu externen Systemen. Es wird davon ausgegangen, dass die Domänenrealisierungen von Aufgaben externe Programme sind.

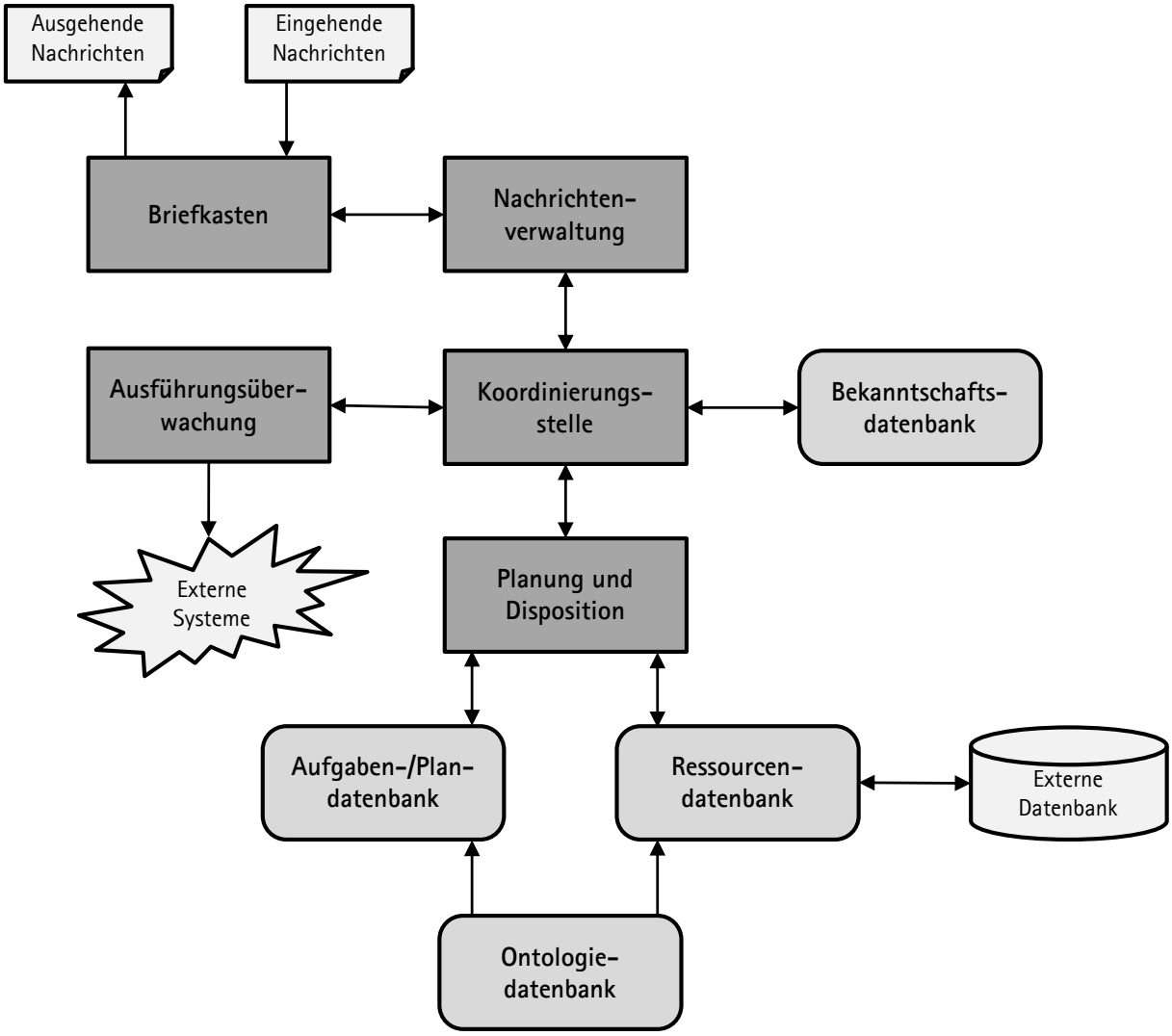


Abbildung 2.11: Architektur des Generischen ZEUS Agenten (in Anlehnung an [NWAN99b])

Jason und AgentSpeak

Jason ist eine Plattform für die Entwicklung von Agentensystemen, welche AgentSpeak als eine der einflussreichsten abstrakten Sprachen auf der Grundlage der BDI-Architektur nutzt. Jason ist der erste vollwertige Interpreter für eine verbesserte Version von AgentSpeak, die auch sprachbasierte Kommunikation zwischen Agenten ermöglicht [PAL20]. Einer der interessantesten Aspekte von AgentSpeak ist, dass es von einem Modell des menschlichen Verhaltens inspiriert wurde. Dieses Modell wird als BDI-Modell bezeichnet (Abbildung 2.12) [BORD07].

Der Agent beginnt mit einem delegierten Ziel und zieht die möglichen Optionen in Betracht, die mit diesem delegierten Ziel vereinbar sind, sodass die von ihm gewählten Optionen daraufhin Absichten darstellen, denen der Agent verpflichtet ist.

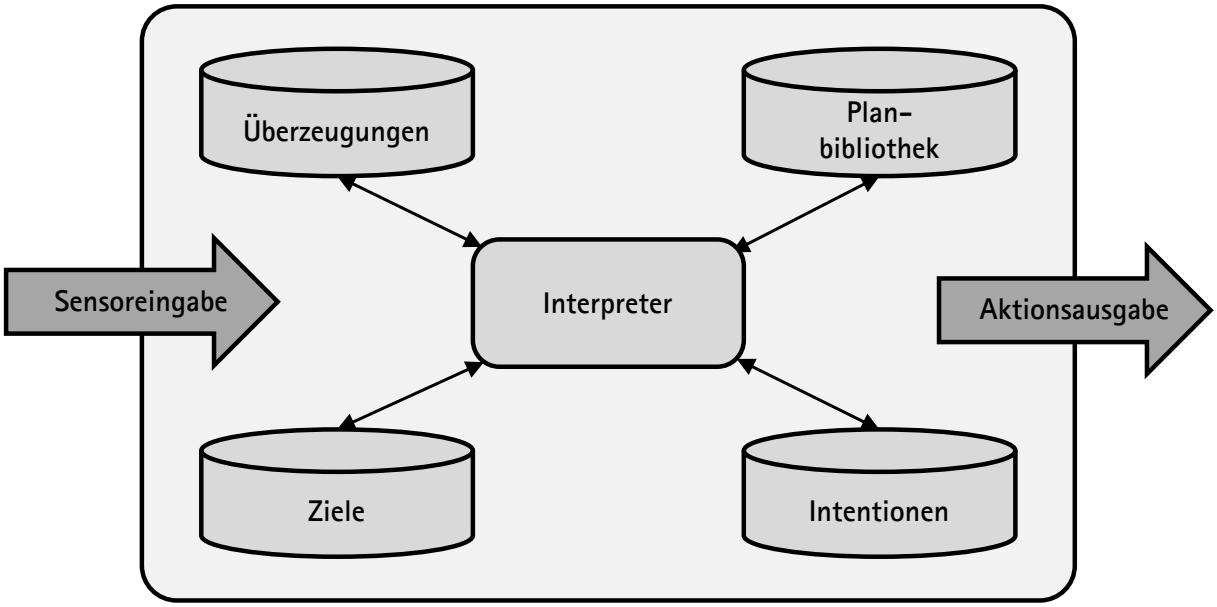


Abbildung 2.12: Das prozedurale Schlussfolgerungssystem (in Anlehnung an [Bord07])

Um die Überzeugungen, Ziele und Pläne innerhalb des Agenten zu hinterlegen, werden diese mittels der Programmiersprache *AgentSpeak* programmiert und als ASL-Dateien abgelegt. Die Grundidee der Programmierung mit AgentSpeak ist, das Problemlösungswissen in Form von Plänen zu definieren, um Reaktionen auf Ereignisse zu charakterisieren. Auf diese Weise können Agenten erstellt werden, die nicht nur systematisch versuchen, Ziele zu erreichen, indem sie das ihnen bereitgestellte Wissen nutzen, sondern die auch auf ihre sich verändernde Umgebung reagieren können [Bord07]. AgentSpeak nutzt hierfür eine eigene Notation, welche in Tabelle 2.1 dargestellt ist. Hierbei werden Überzeugungen mittels Attributen dargestellt, welche einem Objekt zugewiesen und bei Bedarf durch Zusatzinformationen erweitert werden.

Des Weiteren gibt es Ziele, die ein Zustand des Systems darstellen, den der Agent anstrebt. Hierbei sind zwei Arten von Zielen zu unterscheiden: das *Leistungsziel* (engl. achievement goal) und das *Testziel* (engl. test goal). Ein Leistungsziel hat die Notation $\pm!g(t)$, welche besagt, dass der Agent einen Zustand erreichen will, in dem g(t) eine wahre Überzeugung ist. Weiterhin kann der Agent mit einem Testziel, geschrieben als $\pm?g(t)$, prüfen, ob die Formel g(t) eine wahre Überzeugung ist oder nicht [Rao96].

Durch die Definition von Plänen hat ein Agent die Mittel und die Vorgehensweise, mit denen er einen Zweck erfüllen kann. Ein Plan besteht aus einem Kopf und einem Körper. Der Kopf eines Plans besteht aus einem auslösenden Ereignis (engl. triggering event) und einem

	Notation	Beispiel
Überzeugungen	Attribut(Objekt)[Zusatzinformation(Wert)].	herstellbar(Feature1)[Quelle(Fräsagent)].
Zielerfüllung	!Attribut.	!Zählen.
Testziele	?Attribut(Parameter).	?Empfangen(Nachricht).
Pläne	@Bezeichnung **Trigger-Ereignis** : *Bedingung* <- Inhalt **Trigger-Ereignis:** +Überzeugung -Überzeugung +!Zielerfüllung -!Zielerfüllung +?Testziel -?Testziel **Inhalt:** • Aktionen • Zielerfüllungen • Testziele • Mentale Notizen • Interne Aktionen	@Plan1 **+!Zählen** : *Aktuelle_Zahl(X) & maximale_Zahl(Y) & X < Y* <- +Neue_Zahl = X + 1; +-Aktuelle_Zahl(Neue_Zahl); !Zählen.

Tabelle 2.1: Notation von AgentSpeak für ASL-Dateien (in Anlehnung an [Bord07])

Kontext, getrennt durch ein „:“, wobei das auslösende Ereignis angibt, aus welchem Grund der Plan ausgelöst wurde, z. B. durch das Hinzufügen oder Löschen einer Überzeugung oder eines Ziels. Die Addition wird durch den Operator „+“und die Löschung durch den Operator „-“gekennzeichnet. Der Kontext eines Plans gibt die Überzeugungen an, die in der Menge der Basisüberzeugungen des Agenten enthalten sein sollten, um festzustellen, ob ein bestimmter Plan unter den verschiedenen alternativen Plänen wahrscheinlich erfolgreich ist, um das Ereignis zu bewältigen, z. B. ein Ziel zu erreichen, wenn die neuesten Informationen des Agenten zu seiner Umgebung berücksichtigt werden [Bord07]. Der Körper eines Plans ist eine Abfolge von Zielen oder Aktionen und wird vom Kopf des Plans durch einen Pfeil („←“) getrennt. Er spezifiziert die Ziele, die der Agent erreichen oder testen soll und die Aktionen, die der Agent ausführt [Rao96].

Für das Versenden und Empfangen von Nachrichten, innerhalb der AgentSpeak-Programmierung, wird ebenfalls auf die KQML-Performatives zurückgegriffen. Dabei hat jede Nachricht, die vom Agenten empfangen wird, folgende Struktur: <Empfänger, Performative, Betreff(Inhalt)>. Der Empfänger wird hierbei mittels JID (Jabber Identifier) angegeben, welcher den Namen des Agenten sowie die Adresse des verwendeten XMPP-Servers enthält, z. B. sender@xmpp-server.de. Zudem kann der Empfänger auch eine Liste von Agentennamen enthalten. Das Performative bzw. die illokutionäre Kraft bezeichnet explizit die Absicht des Absenders. So wird etwa das KQML-Performative *tell* mit der Absicht verwendet, die Überzeugungen des Empfängers zu ändern, während *achieve* mit der Absicht verwendet wird, die Ziele des Empfängers anzupassen [Bord07]. Der Inhalt der Nachricht variiert je nach Performative und enthält zusätzliche Informationen oder auslösende Ereignisse, mit denen der Empfänger eine Absicht durchführen kann. Folgende Performatives decken die wichtigsten Sprechakte für

die Kommunikation zwischen einem Empfänger und einem Sender ab [Bord07]:

- *tell*: Sender beabsichtigt, dass Empfänger glaubt (dass Sender glaubt), dass die Aussage im Inhalt der Nachricht wahr ist. (Hinzufügen bzw. Ändern einer Überzeugung)
- *untell*: Sender beabsichtigt, dass Empfänger nicht glaubt (dass Sender glaubt), dass die Aussage im Inhalt der Nachricht wahr ist. (Löschen einer Überzeugung)
- *achieve*: Sender fordert Empfänger auf, zu versuchen, einen Zustand zu erreichen, in dem die Wendung im Inhalt der Nachricht wahr ist, d. h. Sender delegiert ein Ziel an Empfänger. (Hinzufügen bzw. Ändern eines Ziels)
- *unachieve*: Sender fordert Empfänger auf, das Ziel, einen Zustand zu erreichen, bei dem der Inhalt der Nachricht wahr ist, aufzugeben. (Löschen eines Ziels)

JADE

Das in den vergangenen zwei Jahrzehnten am weitesten verbreitete ganzheitliche MAS-Rahmenwerk ist JADE (Java Agent Development Framework) [Berg20], welches FIPA-konform ist und ein Netzwerk mittels Agentencontainern, unter Verwendung der objektorientierten Sprache Java, aufbaut [Bell07]. Nachrichtenbasierte asynchrone Kommunikation ist die grundlegende Form der Kommunikation zwischen Agenten in JADE. Ein Agent, der kommunizieren möchte, muss eine Nachricht an ein bestimmtes Ziel oder an eine Reihe von Zielen senden. Darüber hinaus unterstützt JADE die Entwicklung von MAS durch das vordefinierte, programmierbare, erweiterbare Agentenmodell und eine Reihe von Management- und Testwerkzeugen [Bell07]. Zudem bietet JADE eine Reihe von grafischen Werkzeugen innerhalb einer Benutzeroberfläche, um Agenten im MAS zu überwachen, zu steuern oder zu debuggen. Das Laufzeitmodell von JADE-Agenten geht davon aus, dass jeder Agent ein oder mehrere Behaviours einsetzt, um relevante Aufgaben auszuführen. Das Behaviour ist eine Abstraktion, die das Framework verwendet, um Prozeduren, denen die Agenten folgen, zu verwalten und Aufgaben auszuführen. JADE unterstützt verschiedene Arten von Behaviours. Die am häufigsten verwendeten Behaviours sind einmalige Behaviours und zyklische Behaviours, die wiederholt ausgeführt werden, bis sie explizit deaktiviert werden.

SPADE

Zu den neueren Vertretern von MAS-Rahmenwerken gehört die Plattform SPADE 3.0 (Smart Python Agent Development Environment), welche in Python programmiert ist und mit XMPP ein offenes Protokoll für Instant Messaging und Präsenzbenachrichtigung nutzt, um eine verteilte Kommunikation zwischen Menschen, Agenten und Drittsoftware zu ermöglichen [Pala20].

SPADE zeichnet sich durch eine vollständig offene, skalierbare und erweiterbare Entwicklungs- und Ausführungsumgebung, eine transparente Integration von Menschen und Agenten, eine moderne und funktionsreiche Programmiersprache wie Python, die heutzutage in den meisten Bereichen und insbesondere im Bereich der KI führend ist [CASS20] und eine Reihe von Entwicklungsmechanismen, welche die Implementierung von MAS-Anwendungen erleichtern, aus. Des Weiteren schlägt SPADE das asynchrone Programmierparadigma vor, um den Code der Agenten intern zu implementieren. Es bietet nicht nur eine natürliche Human-in-the-Loop-Integration, sondern auch ein offenes, dezentralisiertes, föderiertes Protokoll, das erweiterbar ist und einige interessante Funktionen bietet, wie z. B. den Mechanismus zur Benachrichtigung über die Anwesenheit [PALA20].

Ein Eckpfeiler von SPADE ist die Verwendung eines Kommunikationsmechanismus, der auf dem XMPP-Standard für Instant Messaging basiert, also dem gleichen, der auch in einem Chat-Programm verwendet werden kann. So können Menschen mit Software-Agenten interagieren, wie sie es auch mit anderen Menschen tun würden, indem sie sich mit XMPP-Servern verbinden und Chat-Nachrichten austauschen [PALA22a]. Hierfür wird jedem Agenten eine JID mit einem Passwort zugewiesen, welches es dem Agenten ermöglicht, sich mit dem Server zu verbinden und aufgrund der JID mit anderen Agenten in eine Konversation zu treten. Durch das Aufsetzen eines eigenen XMPP-Servers können die Nachrichten verschlüsselt werden, sodass auch die Sicherheit der Kommunikation gewährleistet werden kann. Zudem ist es möglich durch das Freischalten von indirekten Registrierungen automatisiert, neue Agenten anzulegen und je nach Problemlösungsaufgabe zu verwalten.

Des Weiteren wurde SPADE auf der Grundlage eines asynchronen Programmiermodells entwickelt, um die Leistung und Reaktionsfähigkeit der entwickelten Anwendungen zu erhöhen [PALA22a]. SPADE nutzt hierfür Behaviours, um den Arbeitsteil der Agenten zu programmieren. Es wird zwischen *OneShotBehaviour*, *CyclicBehaviour*, *PeriodicBehaviour*, *TimeoutBehaviour* und *FSMBehaviour* (Finite State Machine Behaviour) unterschieden. Diese haben folgende Eigenschaften [PALA20]:

- *OneShotBehaviour*: Durchläuft bei Aufruf nur einen Durchlauf.
- *CyclicBehaviour*: Läuft in Dauerschleife, bis es gestoppt wird.
- *PeriodicBehaviour*: Läuft in Dauerschleife, mit definiertem Zeitabstand und Start, bis es gestoppt wird.
- *TimeoutBehaviour*: Durchläuft bei zeitlich definiertem Aufruf nur einen Durchlauf.
- *FSMBehaviour*: Ermöglicht komplexe Behaviours durch definierbare Zustände innerhalb

des Agenten, welche aktiviert werden können. Dabei kann nur ein Zustand gleichzeitig aktiv sein.

Zusätzlich zu den klassischen Behaviours, kann ein BDI-Behaviourtyp genutzt werden, welcher um eine im AgentSpeak-Format geschriebene Datei (ASL-Datei) erweitert wird, welche den Code des Verhaltens enthält. Wenn der Agent das Verhalten startet, initiiert er ein zyklisches Ausführungsmuster, in dem SPADE seinen BDI-Zyklus Schritt für Schritt durchläuft. Diese zyklische Ausführung erfolgt auf unbestimmte Zeit und gleichzeitig mit allen anderen Behaviours, die der Agent möglicherweise erstellt. Obwohl das Verhalten automatisch von SPADE ausgeführt wird, hat der Agent immer noch eine gewisse Kontrolle über seine Ausführung, da er sie jederzeit unterbrechen oder fortsetzen kann [PALA22b].

2.3 Multi-Agentensysteme in der Produktentwicklung

In der einschlägigen Literatur zu MAS in der Produktentwicklung gibt es mehrere Ansätze, welche die Konstruierenden bei der Bewertung ihres Entwurfs unterstützen und zu diesem Zweck weitere Aspekte des Produktlebenszyklus einbeziehen [PLAP21]. Der Fokus variiert hierbei zwischen menschenzentrierten Ansätzen, die sich auf die Unterstützung des Konstruierenden konzentrieren und prozesszentrierten Ansätzen, die auf einen reibungslosen Fertigungsprozess abzielen [PLAP22c]. Generell können drei Forschungsrichtungen zu MAS in der Produktentwicklung identifiziert werden [PLAP21]:

1. Bei der ersten Forschungsrichtung können MAS aus agentenbasierten Funktionen aufgebaut werden, wobei sich z. B. CAD-Modelle autonom an neue Situationen anpassen, indem sie als intelligente Features dargestellt werden [FOUG18]. Des Weiteren lassen sich Produkte und Komponenten als Agenten darstellen, um z. B. verschiedene Produktfamilien [MOON09, NIU12, OSTR12, BEND15] und Konzepte [CAMP99, ORSB09] abzuleiten oder multikriterielle Optimierungen durchzuführen [MOUE09].
2. MAS können zur Unterstützung dezentraler Ingenieursarbeit eingesetzt werden, z. B. indem eine geografisch verteilte Gruppe von Anwendenden in die Lage versetzt wird, über das Internet synchron und kollaborativ zu arbeiten [DAVI93, PHOH98, DANE01, FANG03, LIU04, DEEN05, CHIR06, HAO06, NAHM06, BAOL09, WANG09]. Hierzu kann das MAS auch mit einem CAD-System [CHU09, LIU08] oder einer PLM-Umgebung (engl. product lifecycle management) [MAHD10] gekoppelt werden, sodass das Konstruktionswissen verteilt zur Verfügung gestellt wird [SHEN97].

3. MAS werden als Assistenzsysteme für Konstruierende eingesetzt, um sie bei der Entscheidungsfindung für DfX zu unterstützen [DAI02]. Dabei wird in der Regel von außen auf den Entwurf eingewirkt, indem unterschiedliches Vorwissen und Fachwissen zur Bewertung eines Entwurfs herangezogen wird [HAO05, MADH05, LI07, HABH09, KRAT11, GEMB20]. Um die Konstruierenden direkt bei der Modellierung ihres Entwurfs zu unterstützen, werden MAS zunehmend in Kombination mit CAD-Systemen eingesetzt [MEDA06, WANG12]. Für die Betrachtung des gesamten Produktentwicklungsprozesses [JIAN08] wird spezifisches Domänenwissen aus der Arbeitsplanung [TAN96, FENG05, XINH07], spanenden [SUN01, JIA04, MÖN06, MAHE07] und additiven Fertigung [PAPA20] sowie aus der letzten Produktlebensphase, z. B. dem Recycling [DOST16, DIAK20], herangezogen.

Um die bisherigen Arbeiten in den drei o. g. Forschungsrichtungen aufzuzeigen, werden im Folgenden fünf Ansätze aus der Literatur vorgestellt, welche verschiedene Herausforderungen bei der Entwurfsbewertung aufweisen, wie z. B. die Schnittstelle zum CAD-Programm, die Verwaltung von Domänenwissen, die Unterstützung der Konstruierenden sowie die Berücksichtigung der Produktgestaltung nachgelagerten Prozessen.

Fertigungsplanung anhand eines CAD-Vorentwurfs

Der Ansatz von Feng [FENG05] beschreibt Agenten für die Integration von konzeptioneller Konstruktion und Prozessplanung. Diese Agenten werden in einer computergestützten, kollaborativen Umgebung eingesetzt. Der Hauptzweck der Entwicklung der MAS-Plattform ist die Unterstützung des Produktvorentwurfs, die Optimierung der Produktform und -struktur sowie die Reduzierung der Herstellungskosten in der frühen Entwurfsphase. Die Agenten auf der Plattform haben Zugang zu einer Wissensbasis, die Entwurfs- und Planungsregeln enthält. Diese Regeln werden aus einer Analyse der Konstruktionsfaktoren abgeleitet, welche die Prozess- und Ressourcenplanung beeinflussen, z. B. Produktmaterial, Form, Formkomplexität, Features, Abmessungen, Toleranzen, Oberflächenbeschaffenheit, Produktionsvolumen und Produktionsrate. Feng [FENG05] beschreibt den Entwurf und die Implementierung eines Prototyps als Multi-Agentenplattform, in der gleichzeitig Vorentwürfe und Prozessplanungen konzeptionell beschrieben werden. Die Agenten und die Plattform ermöglichen den Informationsaustausch zwischen den Agenten, auf der Grundlage eines zuvor entwickelten integrierten Objektmodells für Konstruktions- und Fertigungsprozesse.

Die Abbildung 2.13 zeigt Gruppen von Agenten, die mit der Plattform verbunden sind und die Integration von Vorentwurf und Prozessplanung unterstützen. Die Agentenplattform ist

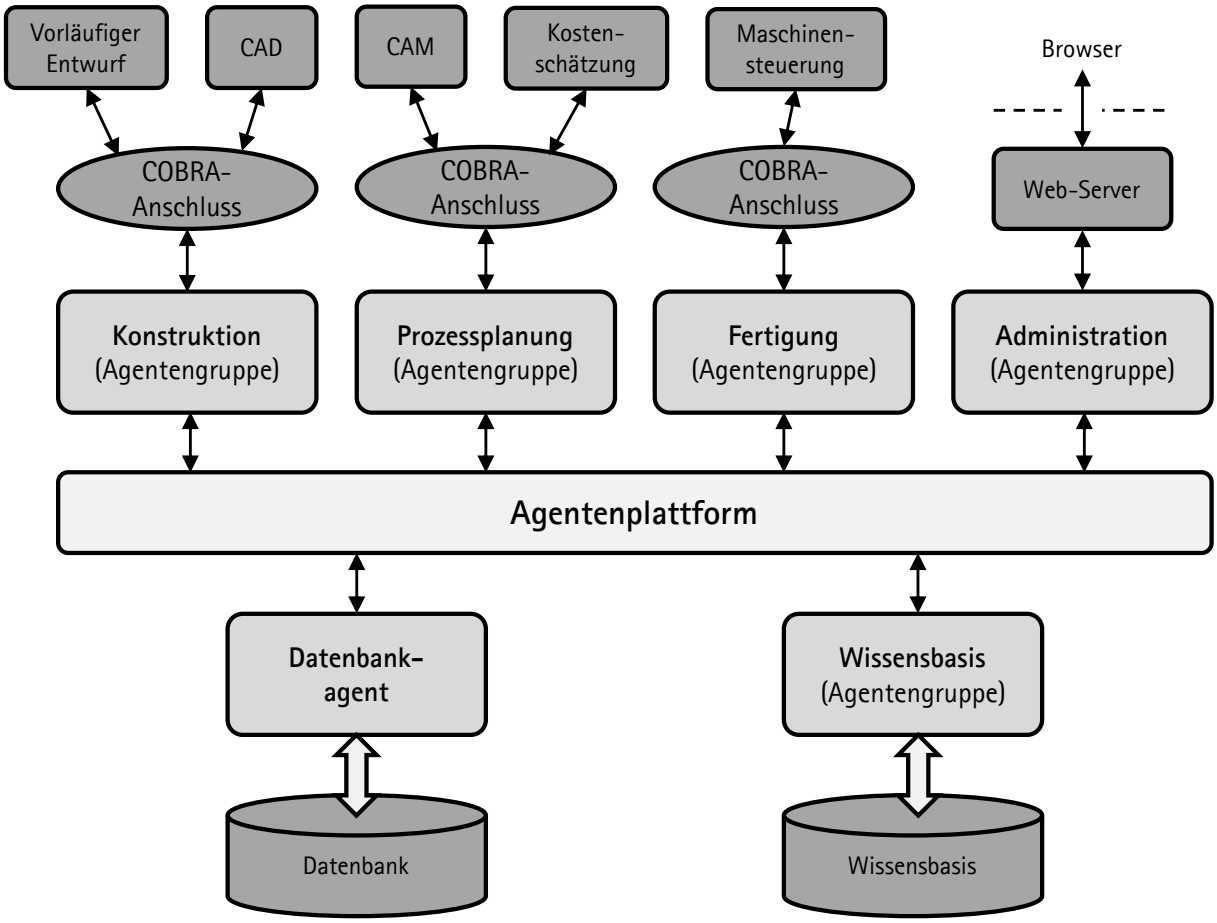


Abbildung 2.13: Prototyp einer Multi-Agenten-Plattform nach [FENG05]

im Grunde ein Kommunikationsmechanismus, über den Entwurfssoftwaresysteme, Prozessplanungssoftwaresysteme, Datenbanken und Wissensdatenbanken integriert werden können. Die Gruppe der Entwurfsagenten besteht aus Vorentwurfs-Agenten, welche die Funktions-, Verhaltens- und Ausführungsentwurf durchführen und Detailentwurfs-Agenten, welche für die geometrische Modellierung von Teilen und Produktbaugruppen zuständig sind [FENG05].

Ein vorläufiger Entwurf der Abtriebsgehäusekomponente eines Planetengetriebes wird als Beispiel verwendet, um die Implementierung der Wissensbasis und der Agentenkommunikation zu testen. Hierfür wurde ein 3D-Volumenmodell des Abtriebsgehäuses mit dem CAD-Programm Pro/Engineer erstellt und die Konstruktionsdaten wie Geometrie, Toleranzen, Oberflächenrauheit und Material mithilfe der Pro/Toolkit Software aus dem CAD-System abgerufen. Auf der Grundlage der Fertigungsregeln in der entwickelten Wissensbasis erzeugt der Prozessplanungsagent vorläufige Fertigungsprozesse [FENG05].

Agentenbasierte Herstellbarkeitsbewertung

Medani und Ratchev [MEDA06] berichten über eine verteilte Methodik zur Bewertung der frühen Herstellbarkeit von Konstruktionen, unter Verwendung kollaborativer autonomer Agenten. Zur Unterstützung des Entscheidungsfindungsprozesses werden ein mit STEP AP224 konformes Produktdatenmodell und entsprechende Prozess- und Anlagendatenmodelle vorgeschlagen. Der Ansatz basiert auf der Anwendung des Produktdatenmodells, unter Verwendung von XML (Extensible Markup Language) für den Austausch von Anfragen und Informationen zwischen Konstruktions-, Fertigungs- und Anlagenplanungsagenten. Der Entscheidungsprozess wird durch mehrstufige Herstellbarkeitsdomänen und Inferenzmodelle unterstützt. Die erste Ebene stellt einen Prozess dar, welcher das Fertigungswissen beschreibt und unabhängig von der Werkzeugmaschine sowie der für die Prozessausführung verwendeten Bearbeitungseinrichtung ist. Die zweite Abstraktionsebene ist die Anlagenebene, die dazu dient, die Verteilung der Fertigungsfähigkeiten, in jeder Anlage des erweiterten Unternehmens, zu beschreiben. Auf der Ebene der Anlage wird jede Maschine, im Hinblick auf die von ihr unterstützten Prozesse sowie ihre Eignung zur Bearbeitung spezifischer Fertigungsanforderungen, definiert. Ein Prototyp eines webfähigen Systems für die schnelle Bewertung der Herstellbarkeit von Produkten, im erweiterten Unternehmen wurde unter Verwendung verteilter Multi-Agentenobjekte, entwickelt.

Der Gesamtansatz ist in Abbildung 2.14 dargestellt. Die Entscheidungsfindung basiert auf mehreren verteilten, kooperierenden Agenten, welche die Entwurfs-, Planungs- und Anlagenzuweisungsaktivitäten unterstützen. Die vorgeschlagene Kommunikationsarchitektur des Prototyp-Systems basiert auf einer verteilten MAS-Architektur mit definierter Schnittstellensprache. Die Agenten-Kommunikation basiert auf einer Client/Server-Architektur. Jeder Agent verfügt über eine Datenbank auf der Serverseite, in der alle für die Argumentation benötigten Daten gespeichert werden, um Aufzeichnungen über die Interaktionen zu dokumentieren und den Agenten zu aktualisieren. Dank dieser Architektur kann die Client-Seite überall ausgeführt werden, wo ein Internetzugang vorhanden ist. Der Client kann die Schnittstellen anzeigen und die für die Schlussfolgerungen benötigten Daten aus der Datenbank herunterladen. Die Funktionalität der agentenbasierten Entscheidungsfindungsumgebung wird anhand eines Beispiels mit einer vereinfachten Komponente als Frästeil beschrieben.

Multi-Agentensystem für die verteilte Fertigung

Im Hinblick auf die Herstellbarkeit von Komponenten haben Sun et al. [SUN01] ein System entwickelt, das Agenten verwendet, um die Merkmale eines Unigraphics-Modells zu identifi-

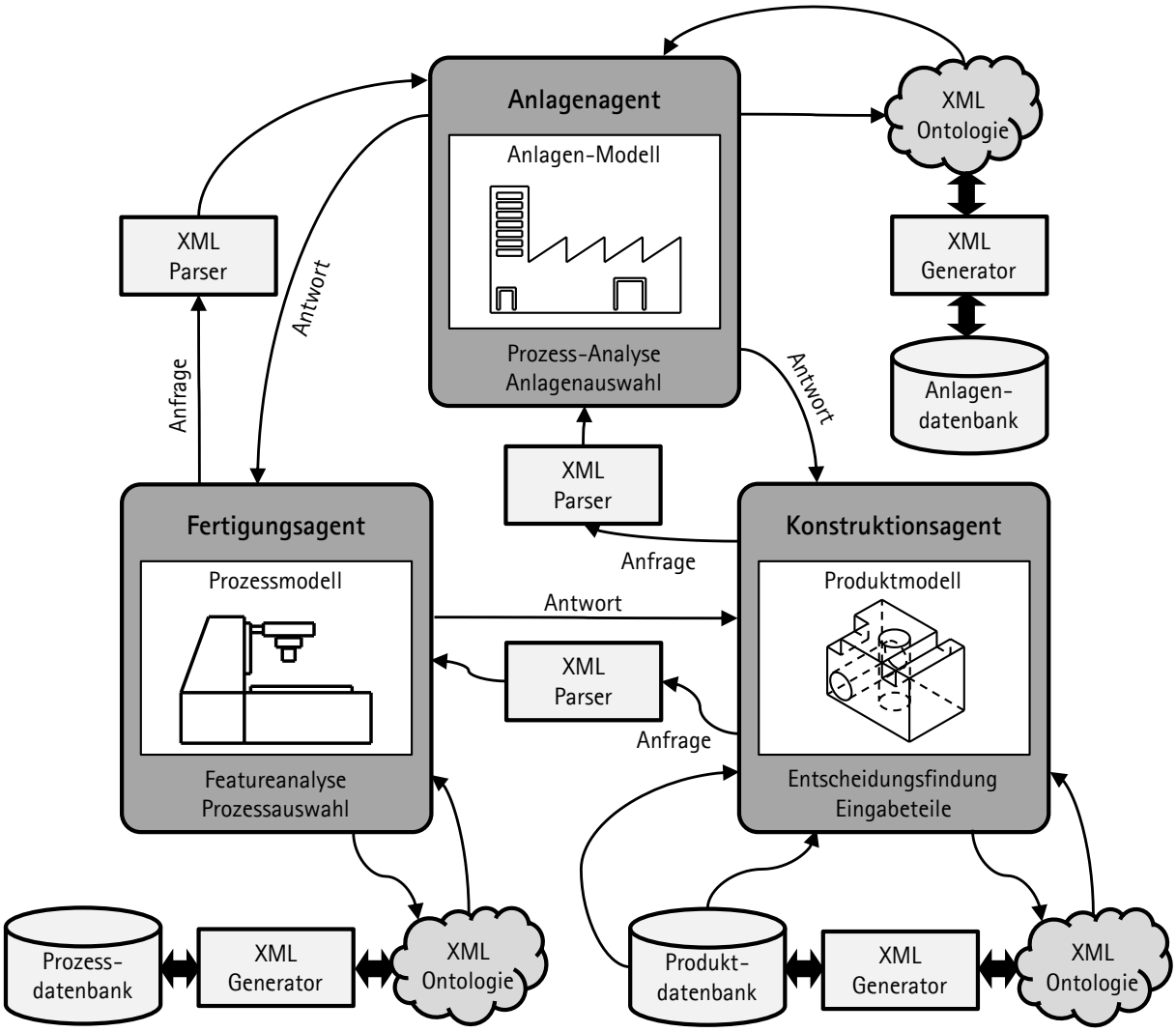


Abbildung 2.14: Rahmenwerk für die Entscheidungsfindung nach [MEDA06]

zieren und z. B. die Zugänglichkeit von Werkzeugen zu bewerten. Dieses System wurde zuerst von Jia et al. [JIA04] im Sinne einer modularen Architektur erweitert, sodass die Agenten leicht wiederverwendet und aktualisiert werden können. Anschließend erweiterten Mahesh et al. [MAHE07] das System als WebMAS, welches speziell für die verteilte Produktentwicklung, die Bewertung der Herstellbarkeit, die Prozessplanung, die Terminierung und die Produktionsüberwachung in Echtzeit entwickelt wurde und die Kommunikation der Agenten über das Internet ermöglicht. Das System besteht aus einem zentralen Produktionsverwaltungsagenten (MMA) und anderen spezifischen funktionalen Agenten, die verschiedene Funktionen für unterschiedliche Produktionsstufen ausführen. Da jeder Agent über einzigartige Fähigkeiten und eine eigene Wissensbasis verfügt, wird die Kommunikation und Koordination von dem MMA verwaltet, welcher als zentrales Datenverwaltungsmodul fungiert. Störungen oder Anomalien eines Agenten haben keine Auswirkungen auf die Funktionen der anderen Agenten. Darüber hinaus macht die Struktur der verteilten Aufgabenbearbeitung und der zentralen Verwaltung

das System flexibel. Jedes Mal, wenn ein lokaler oder globaler Konflikt auftritt, wird der MMA benachrichtigt. Abbildung 2.15 zeigt den vorgeschlagenen WebMAS-Rahmen.

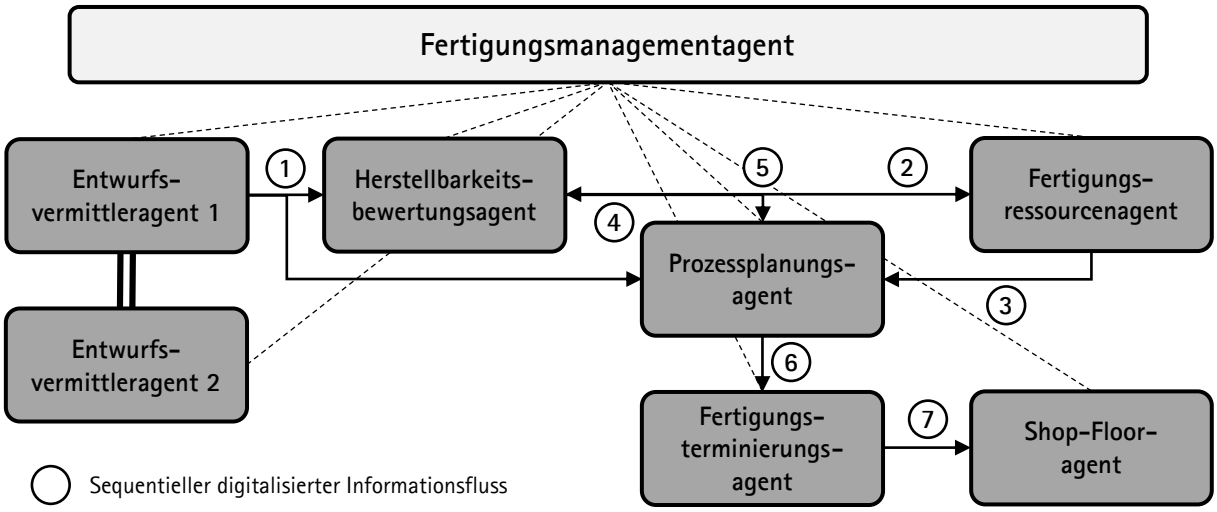


Abbildung 2.15: Rahmenwerk des MAS nach [MAHE07]

Ziel der vorgeschlagenen Architektur ist es, örtlich getrennt arbeitende Konstruierende bei der Übermittlung von Konstruktionsdaten zu unterstützen und zuverlässige Ergebnisse in Bezug auf die Fertigung zu erhalten. Zwei wichtige Komponenten, die beim Entwurf des MAS berücksichtigt wurden, sind der allgemeine Systemrahmen und die Kommunikationsontologie. Alle Agenten kommunizieren über das Internet mit Nachrichten in der KQML-Sprache, um Daten untereinander zu übertragen. Obwohl diese Agenten verstreut sind, wird diese modularisierte Struktur mithilfe des JATLite-Rahmenwerks (Java Agent Template, Lite) verpackt und verbunden. Auf diese Weise bilden alle separaten Komponenten eine enge Einheit und behalten dennoch die wünschenswerten Eigenschaften von Agenten mit integrierten Strukturen bei. Die Wissensbasis eines Agenten ist ein Wissensspeicher und wird mit der Programmiersprache Java als Wenn-Dann-Regeln implementiert [JIA04]. Als Fallstudie des WebMAS wird ein prismatisches Teil verwendet, um die Minimierung der Rüstzeiten mithilfe eines regelbasierten Ansatzes zu testen und den Informationsfluss in der vorgeschlagenen Architektur zu demonstrieren.

Konstruktionsunterstützung in der Entwurfsphase

Kratzer et al. [KRAT11] stellen einen Ansatz zur Unterstützung von Konstruierenden bei der Auslegung und Gestaltung von Produkten nach geltenden Normen und Richtlinien vor, durch welchen eine proaktive, geometrische und semantische Konsistenzprüfung von Produktmodellen innerhalb eines CAD-Systems ermöglicht wird. Bei diesem System handelt es sich um ein agentenbasiertes Konstruktionsunterstützungssystem namens ProKon (Proactive Support of Design Engineering Processes with Agent-based Systems). Hier prüft das System permanent,

ob Änderungen an der Anforderungsliste oder am CAD-Modell durch die Ingenieurin oder den Ingenieur erfolgen, welche anschließend mit einer Referenz verglichen werden, um Inkonsistenzen aufzuspüren. Nachdem eine Änderung festgestellt wurde, müssen andere Objekte, die mit dem geänderten Objekt verbunden sind, informiert werden, da auch deren Konsistenz betroffen sein könnte.

Die Funktionalität des ProKon-Systems realisieren neun Agententypen, die sich in drei Kategorien einteilen lassen (siehe Abbildung 2.16) und mit der Anwendungsspezifität eine zentrale Eigenschaft besitzen. Der umgesetzte Prototyp basiert auf der Programmiersprache Java. Als erstes Anwendungsbeispiel für den Prototyp dient eine Welle-Nabe-Verbindung. Das System ist hier in der Lage, mit dem implementierten Wissen, Geometrie- und Materialänderungen sowie Änderungen bezüglich der aufgebrachten Lasten im CAD-Modell zu erkennen und auf Inkonsistenzen zu prüfen.

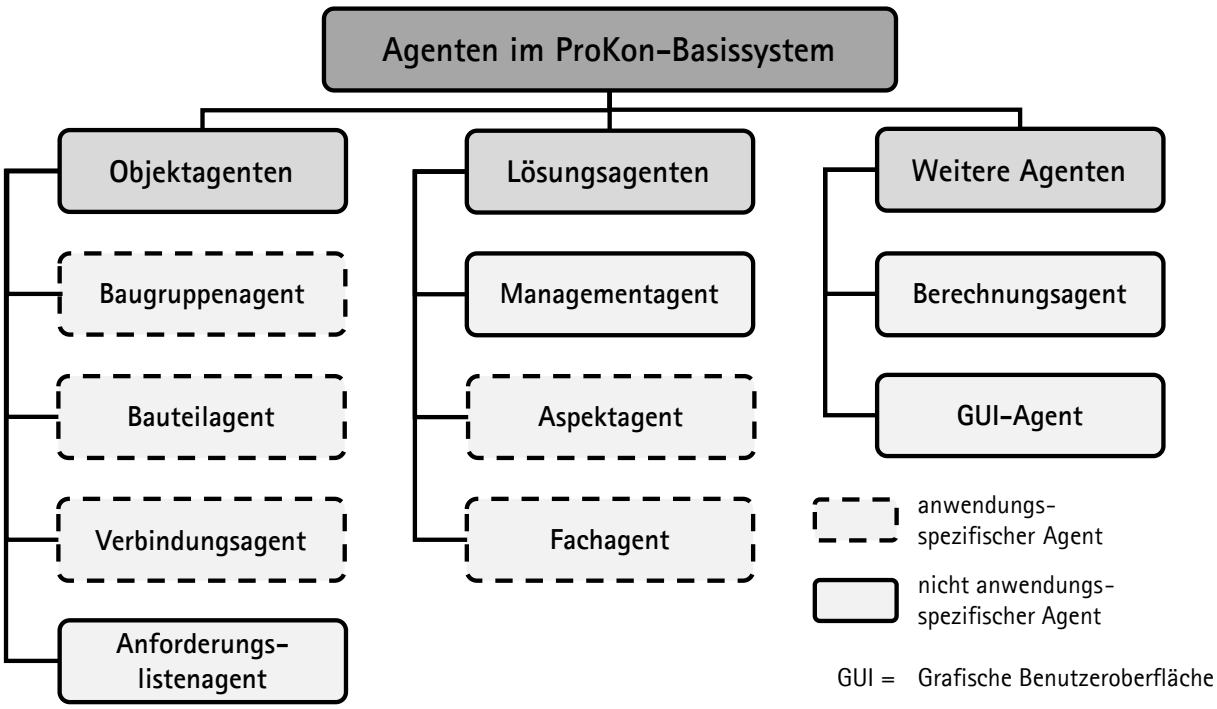


Abbildung 2.16: Beschreibung der Agenten im ProKon-Basissystem nach [KRAT14]

Agentenbasiertes Rahmenwerk für die featurebasierte Konstruktion

Wang et al. [WANG12] haben einen agentenbasierten kollaborativen Entwurfsrahmen für Flugzeugstrukturteile vorgestellt, um die Herstellbarkeitsanalyse und Kostenbewertung, während des Konstruktionsprozesses, zu unterstützen. Hierfür haben sie Agenten mit unterschiedlichen Rollen entwickelt, z. B. für den detaillierten Entwurf, die Featureerkennung, die Bewertung der Herstellbarkeit und die Kostenabschätzung. Zur Verwaltung der Agenten wird eine föde-

rierte MAS-Architektur eingesetzt. Das MAS umfasst einen Kontrollbereich, welcher aus je einem Agenten für die Aufgabenplanung, Aufgabenverteilung und Datenverwaltung besteht und einen Ausführungsbereich aufweist, in welchem der Teilentwurf von den Agenten bearbeitet wird (Abbildung 2.17).

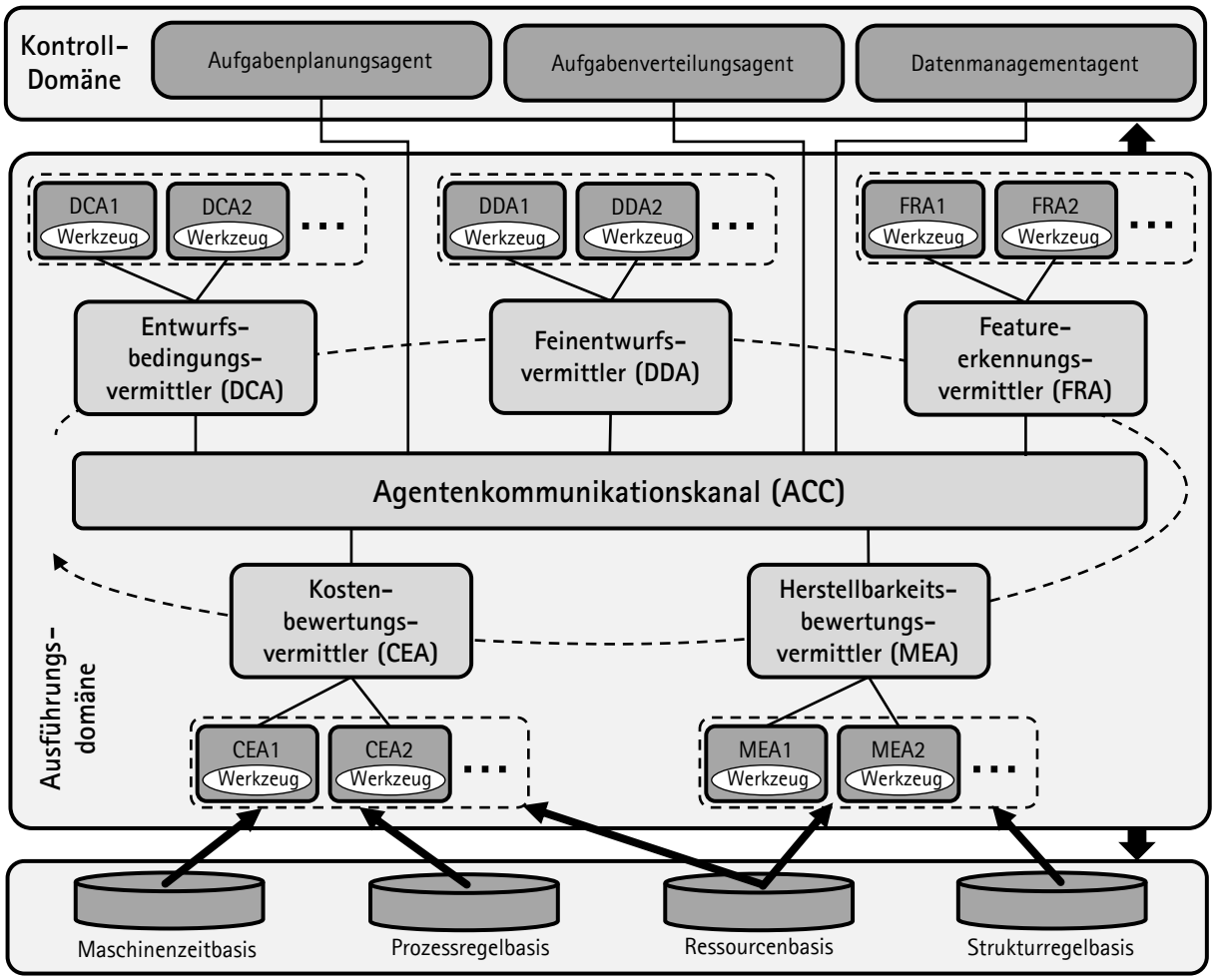


Abbildung 2.17: Rahmenwerk für die kollaborative Entwicklung von Agenten nach [WANG12]

Durch die Kommunikation und die Verhandlungen zwischen den interdisziplinären Agenten erfolgt eine effiziente Rückkopplung von Informationen aus nachgelagerten Prozessen, wie z. B. die Bewertung der Herstellbarkeit und die Kostenschätzung für den vorgelagerten Entwurf, um das Entwurfsergebnis zu optimieren. Die Kommunikation zwischen den Agentengruppen erfolgt über Schnittstellen, die Vermittler genannt werden. Unter der Koordination und Aufsicht von Agenten in der Kontrolldomäne, wird der Entwurf von Flugzeugstrukturteilen, durch Agenten mit verschiedenen Funktionen, in der Ausführungsdomäne vervollständigt. Die Agenten in der Ausführungsdomäne werden automatisch erstellt, organisiert und aufgelöst, indem sie als eine intelligente Reaktion auf Konstruktionsaufgaben fungieren.

Die Bewertung der Herstellbarkeit wird auf der Grundlage der verfügbaren Fertigungsressourcen durchgeführt, sodass das optimierte Ergebnis praxisnah ist. Zudem entwickelten Wang et al. [WANG12] eine dynamische Featureerkennung auf dem Prinzip des zugeordneten Adjazenzgraphen (engl. attributed adjacency graph (AAG)) zur lokalen Featureerkennung. Auf der Featureebene wird der Algorithmus zur Bewertung der Herstellbarkeit für jedes einzelne Feature durchgeführt, um dessen Herstellbarkeit zu prüfen. Das Analyseergebnis wird als Feedback ausgegeben.

3 Problemanalyse

Basierend auf dem vorgestellten Stand der Technik und Forschung wird in diesem Kapitel spezifiziert, wie das methodische Vorgehen und die Entwicklungsumgebung für die Entwicklung von MAS ausgestaltet sein müssen, um digitale Entwurfsbewertungen zu ermöglichen. Hierzu wird betrachtet, wie die Entwicklungsumgebung eine Analyse und Synthese von Entwürfen durchführen kann. Zudem werden die Herausforderungen und die Spezifikation eines MAS im Kontext der Domäne der Produktgestaltung formuliert, um ein Rahmenwerk und ein methodisches Vorgehen für die Entwicklung von MAS zu entwerfen, welches die Konstruierenden bei der Bewertung ihrer Gestaltentwürfe, z. B. im Hinblick auf die Herstellbarkeit, unterstützt. Nach der Analyse der wichtigsten Herausforderungen werden verschiedene agentenorientierte Methoden miteinander verglichen. Zudem werden Vor- und Nachteile bestehender MAS in der Produktentwicklung, anhand definierter Bewertungskriterien, aufgezeigt und Forschungslücken abgeleitet.

3.1 Spezifikation des Multi-Agentensystems

Während des Gestaltungsprozesses werden von den Konstruierenden nicht nur die Gestalt- und Werkstoffeigenschaften festgelegt, sondern es werden indirekt auch die Eigenschaften des Produkts, wie z. B. die Fertigbarkeit, Montierbarkeit und Recyclingfähigkeit, beeinflusst. Durch die steigenden Anforderungen an die Produktentwicklung und die kürzeren Markteinführungszeiten müssen die Konstruierenden immer mehr Aspekte des DfX beherrschen und berücksichtigen. Daher ist das Ziel dieser Arbeit die Unterstützung der Konstruierenden mittels automatisierter und digitaler Entwurfsbewertung, insbesondere im Hinblick auf DfX. Dies spiegelt auch die dritte Forschungsrichtung wider, in welcher MAS als Assistenzsysteme für Konstruierende eingesetzt werden. Hierbei werden domänenspezifische Expertinnen und Experten durch die Agenten repräsentiert und eine direkte Entwurfsbewertung für die Konstruierenden bereitgestellt. Da die Konstruierenden in einem stark vernetzten Umfeld arbeiten und der Gestaltungsprozess

ständig zwischen Synthese- und Analysetätigkeiten wechselt (kreativ-korrektives Verfahren), kann die erste Herausforderung wie folgt formuliert werden.

Herausforderung 1: *Das Rahmenwerk muss vollständig in den Arbeitsablauf der Konstruierenden integriert sein.*

Da die Konstruierenden in der Regel die Gestaltentwürfe mittels eines CAD-Systems erstellen und die erstellten 3D-Modelle häufig für die Produktherstellung weitergenutzt werden, sollte das Rahmenwerk eine Schnittstelle zum CAD-System aufweisen. Darüber hinaus ist es möglich, Informationen direkt aus den CAD-Dateien auszulesen und anhand dieser eine automatisierte Anpassung des CAD-Modells vorzunehmen, um den Arbeitsaufwand für die Konstruierenden zu reduzieren und Fehler in der späteren Produktion des Produkts zu vermeiden. Ferner ist zudem spezifisches Domänenwissen erforderlich, welches häufig dezentral und in verschiedenen Bereichen innerhalb des Unternehmens vorliegt. Dieser Umstand führt zur zweiten Herausforderung:

Herausforderung 2: *Das Rahmenwerk muss das Wissen verschiedener Expertinnen bzw. Experten aus funktionsübergreifenden Disziplinen abbilden.*

Da die Entwurfsbewertung sehr wissensintensiv ist, muss das Wissen aus dem Produktlebenszyklus gesammelt und formalisiert werden. Hierbei kann auf die gemachten Erfahrungen bei der Erstellung von wissensbasierten Systemen zurückgegriffen werden. Zudem können Gestaltungsrichtlinien genutzt werden, mit welchen eine Bewertung des Entwurfs, im Hinblick auf DfX, vorgenommen werden kann. Werden mehrere DfX-Perspektiven, z. B. für die Fertigung oder die Nachhaltigkeit, zusammen verwendet, können ihre Ziele oft in Konflikt geraten, sodass diese Systeme bei der Auswahl der robustesten Lösung zusammenarbeiten müssen. Somit wird als dritte Herausforderung formuliert:

Herausforderung 3: *Das Rahmenwerk muss die Verhandlung bei kollaborativen Entwurfsproblemen unterstützen.*

Um Konflikte zu beheben und optimale Kompromisse zu finden, müssen Informationen ausgetauscht, Kenntnisse über die Perspektiven und Absichten anderer Expertinnen oder Experten gewonnen und Änderungen auf der Grundlage der ausgetauschten Informationen vorgenommen werden. Um die Verhandlungen im Nachhinein nachvollziehen zu können, sollten die Standpunkte der beteiligten Verhandlungspartner dokumentiert werden, sodass folgende vierte Herausforderung formuliert wird:

Herausforderung 4: *Das Rahmenwerk muss den Entscheidungsfindungsprozess dokumentieren.*

Hierbei sollten neben der endgültigen Entscheidung auch deren Begründung, das Vorgehen bei der Schlussfolgerung und die möglichen Alternativen dokumentiert werden, um die Akzeptanz für die Entscheidungen zu erhöhen. Eine weitere Möglichkeit zur Erhöhung der Akzeptanz von Unterstützungssystemen stellt die Konsultation von Konstruierenden bei unsicheren Entscheidungen dar. Dementsprechend wird als fünfte Herausforderung formuliert:

Herausforderung 5: *Das Rahmenwerk muss die Konstruierenden direkt in den Bewertungsprozess einbeziehen (Human-in-the-Loop).*

Des Weiteren kann hierdurch die Komplexität des Gesamtsystems reduziert werden, da die Konstruierenden zum einen über erweitertes Wissen, wie z. B. den Anwendungskontext verfügen und zum anderen die Aussagegüte des Gesamtsystems durch ihre Kompetenz erhöhen können. Um neben der Unterstützung der Anwendenden auch die Entwicklung dieser Assistenzsysteme zu vereinfachen, wird als sechste Herausforderung formuliert:

Herausforderung 6: *Das Rahmenwerk muss durch standardisierte Templates modular und dezentral erweiterbar sein.*

Um eine Vielzahl von Entwürfen zu bewerten und eine einfache Wiederverwendung und Wartung zu unterstützen, sollten die Kommunikation sowie weitere Schnittstellen zur Wahrnehmung und Aktion definiert werden, sodass diese lediglich im Hinblick auf das spezifische Domänenwissen und die entsprechenden Schlussfolgerungsmechanismen angepasst werden müssen.

MAS können verwendet werden, um diese Herausforderungen in einem System abzubilden, da jeder Agent eine Expertin oder einen Experten aus verschiedenen Disziplinen repräsentiert, die Agenten miteinander kommunizieren und verhandeln können und die Anwendenden in den Entscheidungsprozess einbezogen werden. Darüber hinaus können MAS mit Schnittstellen zur Umgebung, z. B. einem CAD-System, erweitert und die getroffenen Entscheidungen sowie deren Alternativen dokumentiert werden. Für die Spezifikation des Rahmenwerks werden die sequenziellen Aktivitäten der digitalen Entwurfsbewertung mit den Werkzeugen für MAS überlagert und daraus Aufgaben für das zu entwickelnde Rahmenwerk abgeleitet, welches die beschriebenen Herausforderungen vollständig erfüllt. Deswegen wird der Einsatz des MAS für die Produktgestaltung in einem Anwendungsfalldiagramm (engl. use-case diagram) in der Abbildung 3.1 dargestellt. Hierbei wird auf die Anwendung des MAS durch die Konstruierenden und die Aufgaben der Agenten zur Entwurfsbewertung folgendermaßen eingegangen:

1. Die Konstruierenden stellen zuerst die Konstruktion als CAD-Modell zur Verfügung.
2. Das MAS, in Form der Agenten, liest die Features aus dem CAD-Modell mittels Featureerkennung automatisiert aus.

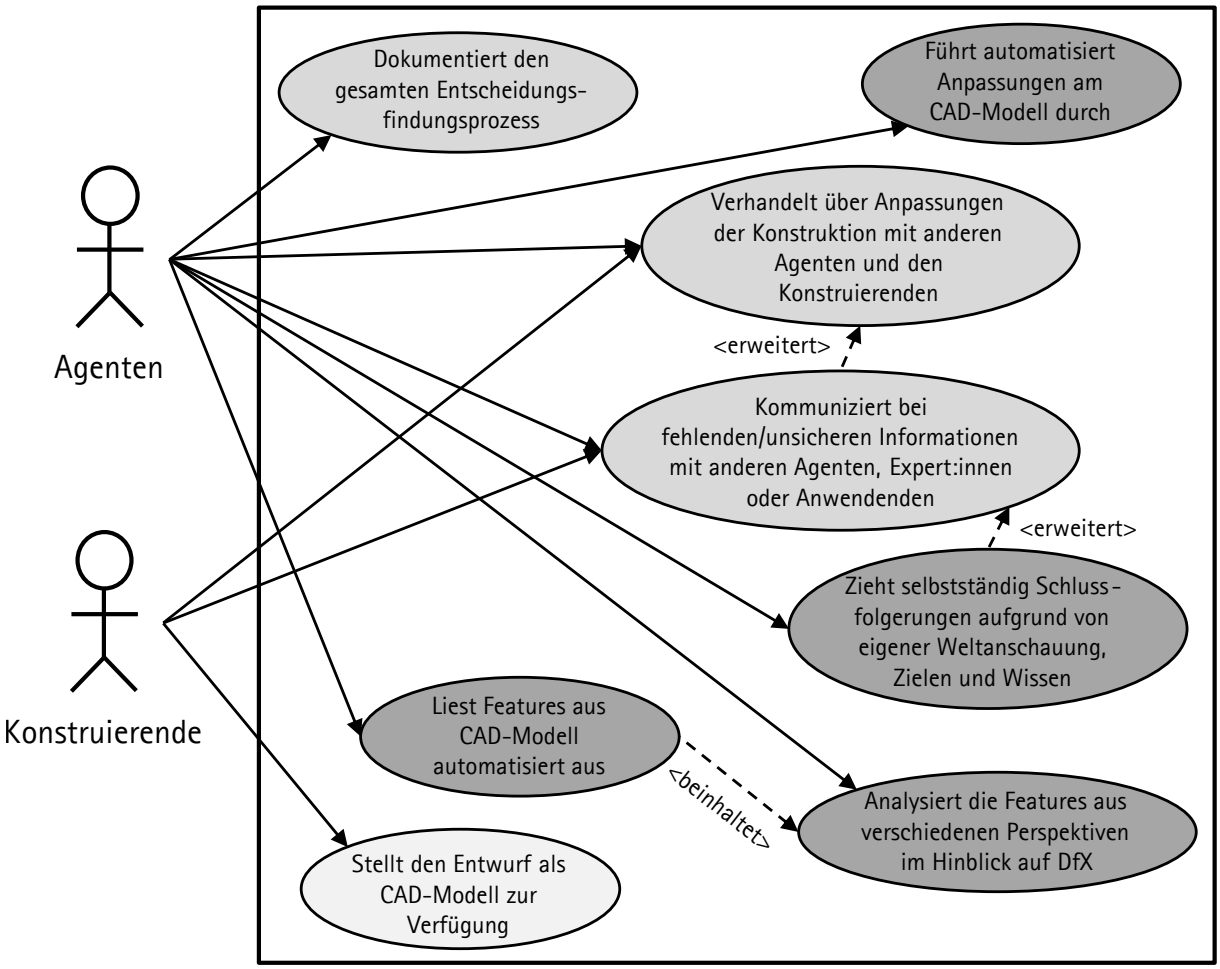


Abbildung 3.1: Spezifikation des MAS zur Entwurfsbewertung der Produktgestalt

3. Anschließend analysieren die Agenten die Features aus den domänenspezifischen Perspektiven unter dem Aspekt des DfX.
4. Aufgrund einer eigenen Weltanschauung, eigenen Zielen und spezifischem Domänenwissen ziehen die Agenten selbstständig Schlussfolgerungen und definieren Aktionen.
5. Fehlen den Agenten Informationen oder können die Agenten nur unsichere Schlussfolgerungen ziehen, kommunizieren sie mit anderen Agenten oder den Anwendenden.
6. Wird aufgrund der Analyse und anschließenden Schlussfolgerung eine Anpassung des CAD-Modells erforderlich, koordinieren die Agenten eine Verhandlung mit anderen Agenten oder den Anwendenden, um eine konsistente Lösung zu finden.
7. Sobald die Schlussfolgerung durchgeführt und mögliche Konflikte gelöst sind, führen die Agenten automatische Anpassungen am CAD-Modell der Konstruktion durch.

8. Der letzte Schritt umfasst die nachvollziehbare Dokumentation des gesamten Entscheidungsfindungsprozesses, inklusive Analysen, Schlussfolgerungen und gegebenenfalls Verhandlungen sowie die Bereitstellung des analysierten und angepassten CAD-Modells.

Um die Konstruierenden bei der Generierung und Bewertung von Lösungen zu unterstützen und sie in den Entscheidungsprozess einzubinden, vereint das MAS hierbei die analytische Anwendung, bei welcher der Entwurf auf Gestaltungsrichtlinien, Normen und Fertigungsprozesse geprüft und die synthetische Anwendung, bei der z. B. ein 80 %-Entwurf automatisch für die Produktion im CAD-Programm vervollständigt wird. Hierfür ist, neben der Formalisierung des domänenspezifischen Wissens und der Analyse sowie der Synthese von 3D-Modellen, auch ein strukturiertes und methodisches Vorgehen beim Software-Engineering notwendig. Hierzu muss die Funktion und Wartbarkeit des MAS sichergestellt werden, um den Zielkonflikt zwischen Modularisierung, durch dezentrale Lösungsräume und Standardisierung, durch wiederverwendbare Komponenten, zu lösen.

3.2 Problemanalyse des methodischen Vorgehens

Für die Erfüllung der beschriebenen Herausforderungen, bezüglich der Spezifikation eines MAS für die Entwurfsbewertung, ist eine Betrachtung des methodischen Vorgehens notwendig, um einerseits die Entwickelnden bei der Erstellung von maßgeschneiderten MAS zu unterstützen und andererseits bereits während der Programmierung von MAS deren Vorteile, wie z. B. die dezentrale Modellierung von Agenten, zu nutzen. Wie bereits in Kapitel 2.2.1 beschrieben, gibt es für die Entwicklung von MAS wissensbasierte, agentenorientierte und objektorientierte Ansätze, welche von den Methoden *MAS-CommonKADS*, *Gaia*, *ROADMAP* und *MaSE* genutzt werden. Tabelle 3.1 stellt diese vier Methoden gegenüber und ordnet ihre Modelle der Analyse, dem Entwurf und deren Unterkategorien zu.

Die Methoden *ROADMAP* und insbesondere *MaSE* setzen ihren Fokus auf die Analysephase, um eine detaillierte Spezifikation der Aufgaben und Rollen zu erstellen, wohingegen *Gaia* bereits eine ausformulierte Anforderungsliste voraussetzt und *MAS-CommonKADS* keine Einteilung in Rollen vorsieht. Die Beschreibung der Rollen stellt einen interessanten Ansatz für die Abbildung der verschiedenen Sichtweisen bei einer Entwurfsbewertung dar, sodass eine ganzheitliche Bewertung der Produktgestalt unterstützt wird. Eine detaillierte Auseinandersetzung mit den Modellen der Analysephase vereinfacht die spätere Implementierung, da bereits bestehende Abläufe und Systeme analysiert und somit eine höhere Akzeptanz bei den Anwendenden erreicht werden kann. Alle untersuchten Methoden haben ihren Ursprung in der IT-Domäne,

		MAS-CommonKADS	Gaia	ROADMAP	MaSE
Analyse	**Kontext**	Organisations-Modell		Umgebungs-Modell	Initialer Systemkontext und Zielerfassung
	Anwendungsfall/ Aufgaben	Aufgaben-Modell	Anforderungsliste	Anwendungsfall-Modell	Anwendungsfälle und Parallele Aufgaben
	Rollen		Rollen-Modell	Rollen-Modell	Rollen
	Interaktion		Interaktions-Modell	Interaktions-Modell	Sequenz-Diagramm
Entwurf	**Wissen**	Kompetenz-Modell		Wissens-Modell	
	Dienste		Dienst-Modell	Dienst-Modell	
	Kommunikation	Koordinations-Modell		Protokoll-Modell	Konversationen
	Agent	Agenten-Modell	Agenten-Modell	Agenten-Modell	Agenten-Klassen
	Bekanntschaft		Bekanntschafts-Modell	Bekannschafts-Modell	Agenten-Architektur
	Implementierung	Entwurfs-Modell			Einsatzdiagramme

Tabelle 3.1: Vergleich bestehender Entwicklungsmethoden und deren Modellen

sodass die Intentionen der Modelle grundsätzlich auf die Domäne der Produktgestaltung übertragen werden können. Es muss jedoch eine Verfeinerung, insbesondere in Hinblick auf die Unterstützung bei der Entwurfsbewertung, vorgenommen werden, z. B. indem das MAS eine Schnittstelle zu einem CAD-System aufweist. Zudem sollten die durchzuführenden Aufgaben und die Rollen definiert werden, welche Einfluss auf die Entscheidungsfindung haben.

In der Entwurfsphase werden die Modelle für Wissen, Dienste und Kommunikation von *MAS-CommonKADS*, *Gaia* und *MaSE* nur teilweise berücksichtigt, jedoch stellt vorrangig das Wissen eine Notwendigkeit für die wissensintensive Bewertung von Gestaltergebnissen dar. Insbesondere die Erfassung und Formalisierung von dezentralem Domänenwissen stellt eine Herausforderung dar, welche beim methodischen Vorgehen berücksichtigt werden muss. Durch die Definition von Diensten können Abläufe innerhalb des MAS standardisiert werden, sodass eine einfachere Bereitstellung von Templates möglich ist. Dieser Aspekt lässt sich auch auf die Kommunikation übertragen. *MAS-CommonKADS* ermöglicht zwar die Entwicklung von mehreren wissensbasierten Systemen, allerdings fehlt für umfangreichere Systeme die Verwaltung der Beziehungen zwischen den Agenten. Die Methode *ROADMAP* bietet für die Entwurfsphase am meisten Modelle und liefert so eine strukturierte Definition der Agenten und deren Bekanntschaften. Allerdings lässt diese Methode gänzlich offen, wie die Implementierung erfolgen soll. Somit ist diese Methode zwar ganzheitlich anzuwenden, für eine konkrete Hilfestellung zur Implementierung fehlen allerdings wichtige Hinweise, welche es gerade unerfahrenen Entwickelnden schwer macht, die Agententechnologie in der Praxis anzuwenden.

Dieser Vergleich zeigt, dass bereits einige Modelle für die Entwicklung von MAS bestehen, auf welchen aufgebaut werden kann. Allerdings bietet keine Methode eine ganzheitliche Vorgehensweise zur Entwicklung von MAS zur Entwurfsbewertung der Produktgestalt, sodass dies eine erste Forschungslücke für die Nutzung von MAS in der Domäne der Produktgestaltung darstellt. Neben dem methodischen Vorgehen stellt die Entwicklungsumgebung für die einfache Adaption auf spezifische Bewertungsaufgaben, im Hinblick auf DfX, einen weiteren entscheidenden Aspekt für den Einsatz der Agententechnologie in der Praxis dar.

3.3 Problemanalyse der Entwicklungsumgebungen

Die Bewertung der bestehenden Ansätze von MAS in der Produktentwicklung erfolgt anhand eines qualitativen Vergleichs und im Kontext der zuvor gestellten problemspezifischen Herausforderungen für die Entwurfsbewertung der Produktgestalt. Die Bewertungskriterien werden nach dem Forschungsansatz aus Kapitel 1.3 in drei Oberkategorien eingeteilt: Anwendungsdomäne, Wissensbasis und Konstruktionswissenschaft. Hierbei repräsentiert die Anwendungsdomäne die *Produktgestaltung*, in welcher das Konstruktionsartefakt mithilfe einer CAD-Schnittstelle aus verschiedenen Sichtweisen bewertet wird. Das *Gestaltungswissen* stellt die Wissensbasis dar, welche die Verteilung des Wissens und deren Repräsentation betrachtet. Weiter spiegelt das *Multi-Agentensystem* die Konstruktionswissenschaft wider. Hierbei spielen das methodische Vorgehen, die Architektur sowie die Koordination und Kommunikation innerhalb des MAS eine wichtige Rolle, um die Konstruierenden bei der Entscheidungsfindung zu unterstützen.

3.3.1 Produktgestaltung

Ausgehend von der Herausforderung 1 werden Bewertungskriterien für den Systemkontext der Entwicklungsumgebung konkretisiert, um die Konstruierenden bei der Bewertung ihrer Gestaltentwürfe zu unterstützen.

Konstruktionsartefakt: Während des Entwurfsprozesses, insbesondere beim Gestalten, legen die Konstruierenden die Eigenschaften des späteren Produkts fest. Hierzu zählen die makrogeometrischen Gestalteigenschaften, wie z. B. Form und Abmessungen bei Einzelteilen sowie Anordnung und Anzahl von Bauteilen, innerhalb von Baugruppen oder Features. Diese Eigenschaften werden normalerweise durch CAD-Modelle repräsentiert. Wird der Detaillierungsgrad erhöht, werden zudem mikrogeometrische Gestalteigenschaften betrachtet, wie z. B. Welligkeit,

Oberflächenrauheit, Maß-, Form- und Lagetoleranzen. Diese Informationen werden entweder in technischen Zeichnungen dokumentiert oder mittels *Model-based Definition* (MBD) direkt im CAD-Modell hinterlegt. Um Aussagen über die Herstellbarkeit zu treffen sind, neben den Gestalteigenschaften, auch die Werkstoffeigenschaften von großer Bedeutung.

Schnittstelle CAD: Zur Abbildung der Gestaltmodelle werden in der Regel CAD-Systeme eingesetzt, welche die geometrischen Merkmale und Abhängigkeiten eines Produktes in einem 3D-Modell darstellen. Um diese Informationen aus den CAD-Modellen auszulesen und für eine Weiterverarbeitung z. B. durch MAS zu nutzen, muss eine Schnittstelle zum CAD-System bereitgestellt werden. Die einfachste Form dieser Schnittstelle stellt die Featureextraktion dar, mithilfe derer auf die programminternen CAD-Features des Modells zurückgegriffen werden kann. Dies setzt jedoch voraus, dass die CAD-Modelle in einem spezifischen CAD-Programm erzeugt und nach einer vorgegebenen Modellierungsstrategie, in Bezug auf die Zuordnung und Reihenfolge der erstellten Features, vorgegangen wird. Im Gegensatz hierzu ermöglicht die Featureerkennung, nach der Analyse eines standardisierten CAD-Modells, die Ableitung von Features, unabhängig von der Vorgehensweise bei der Modellierung durch die Konstruierenden. Darüber hinaus bietet die Featuremodellierung die Möglichkeit, automatisiert Veränderungen am CAD-Modell durch Synthese vorzunehmen. Hierbei kann zwischen der Veränderung von Parametern in einem vorgegebenen Lösungsraum, z. B. bei der Bewertung der Normkonformität von Gestaltelementen und dem Hinzufügen bzw. Löschen von Features, z. B. bei fehlenden Radien für Werkzeugausläufe, unterschieden werden. Dieser Aspekt ermöglicht die direkte Unterstützung der Konstruierenden, indem z. B. ein Vorentwurf im Hinblick auf die Herstellbarkeit überprüft und angepasst werden kann.

Sichtweisen: Um den Aspekt der Entwurfsbewertung aufzugreifen, muss die Produktgestaltung im Hinblick auf DfX aus verschiedenen Perspektiven analysiert und bewertet werden. Hierzu werden in der Literatur meistens drei Sichtweisen eingenommen. Erstens die Konstruktionssicht, welche die Schnittstelle zu den Konstruierenden einnimmt und die Funktionssicherheit des späteren Produkts darstellt. Zweitens die Fertigungssicht, welche überprüft, inwieweit das Produkt durch die zur Verfügung stehenden Fertigungsverfahren hergestellt werden kann und drittens die Prozessplanungssicht, welche überprüft, in welcher Reihenfolge das Bauteil bearbeitet wird und welche Fertigungsmittel hierfür notwendig sind.

3.3.2 Gestaltungswissen

Das Gestaltungswissen umfasst die zweite Herausforderung, da hierbei zum einen die Handhabung von dezentralem Wissen eine Rolle spielt und zum anderen die Repräsentation des Wissens Einfluss auf die Schlussfolgerungsmechanismen hat.

Wissensverteilung: Um Wissen für die Entwurfsbewertung und für die Entscheidungsfindung handhabbar zu machen, muss dieses an dem Ort zur Verfügung gestellt werden, an dem es gebraucht wird, auch wenn die Wissensakquisition an einem anderen Ort stattfindet. Für diese Verteilung des Wissens gibt es zwei Ansätze. Zum einen wird das Wissen zentral in einer Wissensbasis gespeichert, auf welche alle Agenten zugreifen können und daher das gleiche Verständnis hierfür haben. Zum anderen kann das Wissen dezentral verteilt sein, z. B. innerhalb der verschiedenen Agenten und mittels Kommunikation demjenigen bereitgestellt werden, der es für eine Schlussfolgerung benötigt.

Wissensrepräsentation: Neben der Art der Wissensverteilung kann auch die Form der Wissensrepräsentation unterschieden werden. Erstens kann das Wissen in Form von Regeln dokumentiert werden, z. B. in der Form von Wenn-Dann-Sonst-Anweisungen. Zweitens besteht die Möglichkeit, das Wissen implizit in einem Modellkontext abzubilden, z. B. durch die Darstellung von Zuweisungen und Ressourcenverbräuchen oder durch die Bedingungen (Constraints) zwischen Domänen. Die dritte Form der Wissensrepräsentation basiert auf dem Prinzip des Erfahrungswissens. Das Wissen über bereits durchgeführte Lösungen in Form von Fällen wird in einer Fallbasis dokumentiert. Aufgrund der verschiedenen Arten der Wissensrepräsentation können auch unterschiedliche Schlussfolgerungsmechanismen eingesetzt werden.

3.3.3 Multi-Agentensystem

Die Herausforderungen 3 bis 6 werden in dieser Bewertungsgrundlage betrachtet. Im Fokus steht hierbei der Aufbau der MAS und die Unterstützung der Konstruierenden bei der Entwurfsbewertung.

Methoden: Dieses Bewertungskriterium greift die drei Ansätze für das methodische Vorgehen bei der Entwicklung von MAS auf, um den Einsatz dieser wissensbasierten, agentenorientierten und objektorientierten Methoden in den bestehenden Ansätzen zu analysieren.

Architektur: Neben der Vorgehensweise bei der Entwicklung von MAS spielt auch der Aufbau bzw. die Architektur der Agenten eine wichtige Rolle, da hierbei deren Verhalten festgelegt

wird. Wie bereits in Kapitel 2.2.2 beschrieben, kann hierbei zwischen logikbasierter, reaktiver und der BDI-Architektur unterschieden werden.

Koordination & Kommunikation: Für die Funktionsweise des MAS ist neben der Architektur der Agenten auch deren Zusammenspiel von Bedeutung, denn dies hat Einfluss auf die Koordination der Aufgaben innerhalb des MAS und infolgedessen auch auf deren Kommunikationswege. Für den Vergleich der bestehenden Ansätze wird zwischen der flachen, hierarchischen und gruppierten Architektur unterschieden.

Entscheidungsunterstützung: Im letzten Vergleichskriterium geht es um die Entscheidungsunterstützung für die Konstruierenden. Hierbei ist zu beachten, inwieweit das MAS Verhandlungen durchführt, um ein globales Optimum bzw. einen bestmöglichen Kompromiss zu finden. Des Weiteren sollten die Konstruierenden als Teil des Prozesses angesehen werden, sodass diese bei unsicheren Situationen kontaktiert werden oder zusätzliche Informationen bzw. Wissen zur Verfügung stellen können. Der Entscheidungsfindungsprozess sollte dokumentiert werden, sodass zu einem späteren Zeitpunkt nachvollzogen werden kann, wie die Entscheidung zustande gekommen ist.

3.3.4 Qualitativer Vergleich der bestehenden Ansätze

Basierend auf der vorgestellten Bewertungsgrundlage erfolgt nun der Vergleich der bestehenden Ansätze zu MAS in der Produktentwicklung aus Kapitel 2.3. Hierbei wird zum einen unterschieden, ob das Kriterium für die Entwicklungsumgebungen zutrifft und wenn dies der Fall ist, wie stark dieses Kriterium ausgeprägt ist. Hier wird wiederum zwischen „Nur im Ansatz umgesetzt“, „Teilweise umgesetzt“ und „Vollständig umgesetzt“ unterschieden. In Tabelle 3.2 werden die verwandten Arbeiten zu MAS in der Produktentwicklung verglichen.

Im Hinblick auf das *Konstruktionsartefakt* liegt ein starker Fokus auf den Makro-Gestalteigenschaften, da hierbei häufig die Geometrie der zu analysierenden Bauteile betrachtet wird. Nur Feng [Feng05] und Medani und Ratchev [Meda06] beziehen Toleranzen und Oberflächenangaben in ihre Bewertung mit ein, um die Aussagekraft ihrer Systeme, im Hinblick auf die Fertigbarkeit der Bauteile, zu erhöhen. Hierfür nutzen diese Systeme die Featureextraktion als *Schnittstelle zum CAD-System*. Allerdings nutzt nur Wang et al. [Wang12] die Featureerkennung mittels AAG, um unabhängig von der Modellierungsstrategie der Konstruierenden zu sein. Die Featuremodellierung ist nur teilweise umgesetzt. Kratzer et al. [Krat11] nimmt die Anpassung in einem definierten Lösungsraum vor, was eher mit einer Konfiguration gleichzusetzen ist. Eine Kombination aus Analyse- und Synthesefähigkeiten ist bei den bestehenden

			Feng [Feng05]	Medani [Meda06]	Mahesh [Mahe07]	Kratzer [Krat11]	Wang [Wang12]
Produktgestaltung	Konstruktionsartefakt	Makro-Gestalteigenschaften	●	●	●	●	●
		Mikro-Gestalteigenschaften	●	●			
		Werkstoffeigenschaften	●	●		●	
	Schnittstelle CAD	Featureextraktion	●	●	◑	●	
		Featureerkennung (Analyse)					●
		Featuremodellierung (Synthese)				◑	◔
	Sichtweisen	Konstruktionssicht		◔		●	◔
		Fertigungssicht	●	◑	●		●
		Prozessplanungssicht	●	●	●		
Gestaltungswissen	Wissensverteilung	Zentral	●				
		Dezentral		●	●	●	●
	Wissensrepräsentation	Regelbasiert	●	●	●	●	●
		Modellbasiert				◑	
		Fallbasiert					
Multi-Agentensystem	Methoden	Wissensbasiert (WB)				●	
		Objekt-orientiert (OO)					
		Agenten-orientiert (AO)					
	Architektur	Logikbasiert					●
		Reaktiv	●	●	●	●	
		Deliberativ (BDI)					
	Koordination & Kommunikation	Flach		●			
		Hierarchisch			●	●	
		Gruppiert	●				●
	Entscheidungs-unterstützung	Verhandlung			◑		◔
		Human-in-the-Loop	◑	◑	◑	◔	
		Dokumentation Entscheidungsfindungsprozess	◔	◔	●	◑	●

Legende: ◔ Nur im Ansatz umgesetzt ◑ Teilweise umgesetzt ● Vollständig umgesetzt

Tabelle 3.2: Vergleich der verwandten Arbeiten zu Multi-Agentensystemen in der Produktentwicklung

Ansätzen nicht zu ermitteln, sodass dies zu einer weiteren Forschungslücke führt. Beim Vergleich der *Sichtweisen* ist zu beobachten, dass häufig die 3D-Modelle genutzt werden, um diese aus Sicht der Fertigung und Prozessplanung zu analysieren, allerdings wird eine ganzheitliche Bewertung aus allen Sichtweisen nur im Ansatz umgesetzt.

Das Bewertungskriterium der *Wissensverteilung* wird in der Mehrheit durch eine dezentrale Wissensverteilung abgebildet, welche die Besonderheit von MAS widerspiegelt. Für die *Wissensrepräsentation* wird hauptsächlich auf Regeln zurückgegriffen. Diese bieten für Agenten den Vorteil, dass sie einfach umzusetzen sind und die Regelbasis innerhalb der Agenten nur einen begrenzten Umfang einnimmt. Um die Systeme, im Hinblick auf die Wartung und Erweiterbarkeit, robust zu gestalten, sollte eine Abbildung des Wissens durch Modelle und Fallbasen geprüft werden, insbesondere, wenn das System mit Lernfähigkeiten ausgestattet werden soll.

Für die Umsetzung der *Methoden* betrachten lediglich Kratzer et al. [KRAT11] ein methodisches Vorgehen für die Entwicklung von MAS zur Konstruktionsunterstützung, indem sie u. a. die CommonKADS-Methode adaptieren. In Bezug auf die objekt- bzw. agentenorientierten Methoden wurde keine Anwendung gefunden. Dieser Aspekt unterstreicht die Notwendigkeit eines methodischen Vorgehens für die Entwurfsbewertung mittels MAS. Des Weiteren wurde für das Kriterium der *Architektur* festgestellt, dass die meisten MAS mit reaktiven Agenten arbeiten. Erstaunlicherweise nutzt keine der verwandten Arbeiten die BDI-Architektur, obwohl diese einen höheren Grad an Autonomie der Agenten zulässt. Für die *Koordination & Kommunikation* ergibt sich ein gemischtes Bild, da alle Kriterien bedient werden, wobei eine zentralisierte Koordination in hierarchischer Form oder als Gruppen bevorzugt wird. Im Hinblick auf die *Entscheidungsunterstützung* findet die Nutzung von Verhandlungsstrategien zwischen den Agenten und eine direkte Integration der Konstruierenden in den Entscheidungsfindungsprozess, unter dem Aspekt *Human-in-the-Loop*, kaum statt. Die Dokumentation des Entscheidungsfindungsprozesses, insbesondere mit Teilentscheidungen zur besseren Nachvollziehbarkeit, wird zunehmend berücksichtigt, wobei eine ganzheitliche Entscheidungsunterstützung von keinem der bestehenden Ansätze bereitgestellt wird.

Auf Basis des Vergleichs lässt sich ableiten, dass die Methoden und Architekturen für MAS aus informationstechnischer Sicht eine breite Grundlage besitzen, diese jedoch in der Produktentwicklung nur unzureichend Anwendung finden. In der Regel werden MAS für die analytische Anwendung eingesetzt, die sich oft auf die Analyse von CAD-Modellen und die anschließende Empfehlung von Hinweisen und Vorschlägen beschränkt. Die synthetische Anwendung ist oft KBS vorbehalten, die eine Anpassung des CAD-Modells vornehmen, dafür aber in einem geschlossenen Lösungsraum operieren.

4 Methodenentwicklung

Ein Schlüsselfaktor für die einfache und breite Anwendung der Multi-Agententechnologie, in der Domäne der Produktgestaltung, ist die Definition eines Entwicklungsprozesses für die Bewertung der Produktgestalt. Insbesondere die Spezifikation und Implementierung des MAS stellen hierbei die entscheidenden Phasen für ein methodisches Vorgehen dar. Diese Phasen werden in der Literatur häufig als Analyse- und Entwurfsphase deklariert. Für die Entwicklung von MAS, im Bereich der Gestaltung, gibt es keine adäquate Vorgehensweise, um zum einen auf die domänenspezifischen Eigenschaften und Systeme einzugehen und zum anderen eine konkrete Hilfestellung, z. B. in Form von Templates für die Implementierung, zu geben. Daher wird im Folgenden ein methodisches Vorgehen für die Entwicklung von MAS zur Entwurfsbewertung vorgestellt.

4.1 Entwicklungsprozess für Multi-Agentensysteme zur Entwurfsbewertung

Für die Definition einer Software-Engineering-Methode wird in der Regel nicht das „Rad neu erfunden“, sondern es wird ein Verfeinerungszyklus durchlaufen, bei dem neue Aspekte und Perspektiven der Systeme und Sprachen hinzugefügt und die erfolgreichen Bestandteile früherer Methoden integriert werden [IGLE96]. Das vorzustellende methodische Vorgehen basiert einerseits auf der *MaSE*-Methode, da diese eine detaillierte Unterstützung bei der Analysephase liefert und diese Schritte für die Domäne der IT und der Produktgestaltung ähnlich sind. In beiden Domänen muss der Systemkontext berücksichtigt und die Aufgaben des MAS spezifiziert werden. Des Weiteren unterstützt die frühzeitige Aufteilung des Systems in Rollen die Repräsentation von verschiedenen Sichtweisen bei der Entwurfsbewertung. Andererseits liefert die Methode *ROADMAP* eine ganze Reihe an Modellen für die Entwurfsphase, um insbesondere das Wissen zu verwalten und bereits programmierte Dienste wiederzuverwenden. Da sich diese Modelle der Methoden bereits in der IT-Domäne bewährt haben, werden diese auf die

Domäne der Produktgestaltung übertragen und domänenspezifisch angepasst. Eine konkrete Beschreibung der Ausgestaltung des Gesamtsystems, insbesondere dessen Implementierung, fehlt allerdings bei den bestehenden Methoden, sodass diese für die Produktgestaltung ergänzt wird. Die Phasen und Modelle der MaSE4D-Methode (Multiagent Systems Engineering for Engineering Design) werden in Abbildung 4.1 dargestellt. Durch die entgegengesetzten Dreiecke wird symbolisiert, dass zuerst ausgehend von dem initialen Systemkontext eine Detaillierung bis zum Wissensmodell erfolgt und anschließend nach dem Prinzip des *Bottom-up-Ansatzes* das MAS Schritt für Schritt zusammengesetzt wird. Dies spiegelt auch die Einteilung in die vier Ebenen *Operativ*, *Agenten*, *Management* und *User* wider.

Die Phasen für die Entwicklung eines MAS nach der MaSE4D-Methode werden im Folgenden beschrieben:

- *Anwendungsfälle erarbeiten*: Definition des Anwendungsfalls, der Nutzung des MAS und Ableitung der auszuführenden Aufgaben.
- *Rollen verfeinern*: Aufteilung des Systems nach geografischer Distanz, logischer Ausführung von Funktionen oder Wissensverteilung in verschiedene Rollen und Beschreibung der Interaktionen zwischen den Rollen.
- *Wissen sammeln und verwalten*: Akquise und Formalisierung von domänenspezifischem Wissen.
- *Aktionsvorrat entwickeln*: Definition von standardisierten Kommunikationsprotokollen und ausführbaren Funktionen, die durch die Agenten wiederverwendet werden können.
- *Agenten erstellen*: Aufbau der Agenten mit domänenspezifischem Wissen, Kommunikationsprotokollen und Funktionen, sodass die Agenten autonom mit ihrer Umwelt interagieren können.
- *Agenten verwalten*: Koordination der Agenten und Definition einer einheitlichen Ontologie.
- *Gesamtsystem aufbauen*: Implementierung einer Interaktion mit dem Anwendenden und Integration des MAS in den Systemkontext.

Dieses methodische Vorgehen wird in den folgenden Kapiteln für die Analyse- und Entwurfs-Phase detailliert beschrieben und mit Modellen und Templates für die einfache Anwendung erweitert. Hierdurch verliert es zwar seine Allgemeingültigkeit, da die Templates nicht mehr systemneutral sind, es wird aber deutlich einfacher ein MAS im Gestaltungskontext aufzubauen oder zu erweitern. Insbesondere unter dem Aspekt der Modularität können hierdurch

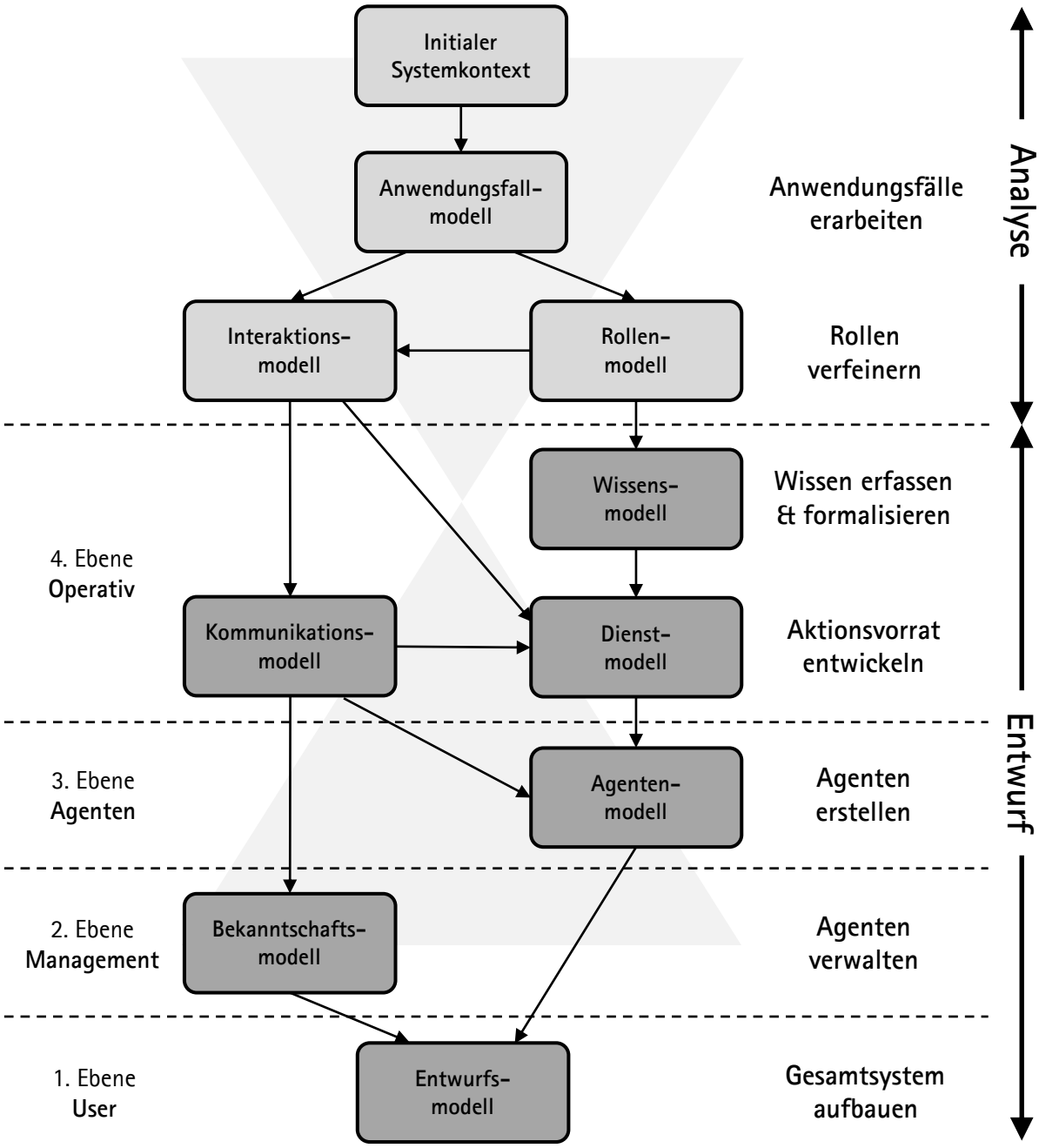


Abbildung 4.1: Phasen der MaSE4D-Methode

Kommunikationsprotokolle oder Dienste wiederverwendet werden.

4.2 Analyse der Anwendungsdomäne

In der Analysephase wird im ersten Schritt der *initiale Systemkontext* für das MAS definiert, z. B. in welches bestehende Arbeitsumfeld das MAS integriert werden soll. Anschließend werden die *Anwendungsfälle* mit den Aufgaben beschrieben, welche durch das MAS ausgeführt werden sollen. Bis zu diesem Zeitpunkt wird das System als Ganzes betrachtet. Nachfolgend wird das System in verschiedene *Rollen* unterteilt, denen wiederum Aufgaben zugewiesen werden können. Die *Interaktion* innerhalb des Systems wird definiert, wobei auch die Interaktionen zwischen den Rollen zu berücksichtigen sind.

Anwendungsfälle erarbeiten

Der erste Schritt bei der Erarbeitung von Anwendungsfällen, welche die grundlegenden Szenarien definieren, die ein System ausführen können sollte, besteht darin, Informationen aus dem initialen Systemkontext zu extrahieren. Hierzu zählen insbesondere die Aufgaben, die das System erfüllen muss, um ein gewünschtes Ziel zu erreichen. Diese Anwendungsfälle können in einem UML-Anwendungsfalldiagramm dargestellt werden. In Abbildung 4.2 ist beispielhaft ein Anwendungsfalldiagramm für ein MAS zur Bewertung der Normkonformität eines Drehteils dargestellt. Hierbei stellen die Konstruierenden und die Agenten die *Akteure* dar. Aufgaben können Kommunikationspfade zu einem oder beiden Akteuren aufweisen, z. B. wenn diese miteinander kommunizieren oder Informationen von einem Akteur zum anderen übergeben werden. Des Weiteren können Aufgaben voneinander abhängig sein, so müssen z. B. erst die Gestaltelemente auf Normkonformität geprüft werden, bevor hierzu Feedback gegeben werden kann.

Rollen verfeinern

Ausgehend von den Anwendungsfällen werden die Rollen verfeinert. Hierzu werden die Aufgaben bzw. Probleme des Gesamtsystems in Teilaspekte aufgeteilt, um diese zum einen handhabbarer zu machen und die Komplexität zu reduzieren und zum anderen um verschiedene Sichtweisen auf das Problem einnehmen zu können. Diese Teilaufgaben können wiederum ver-

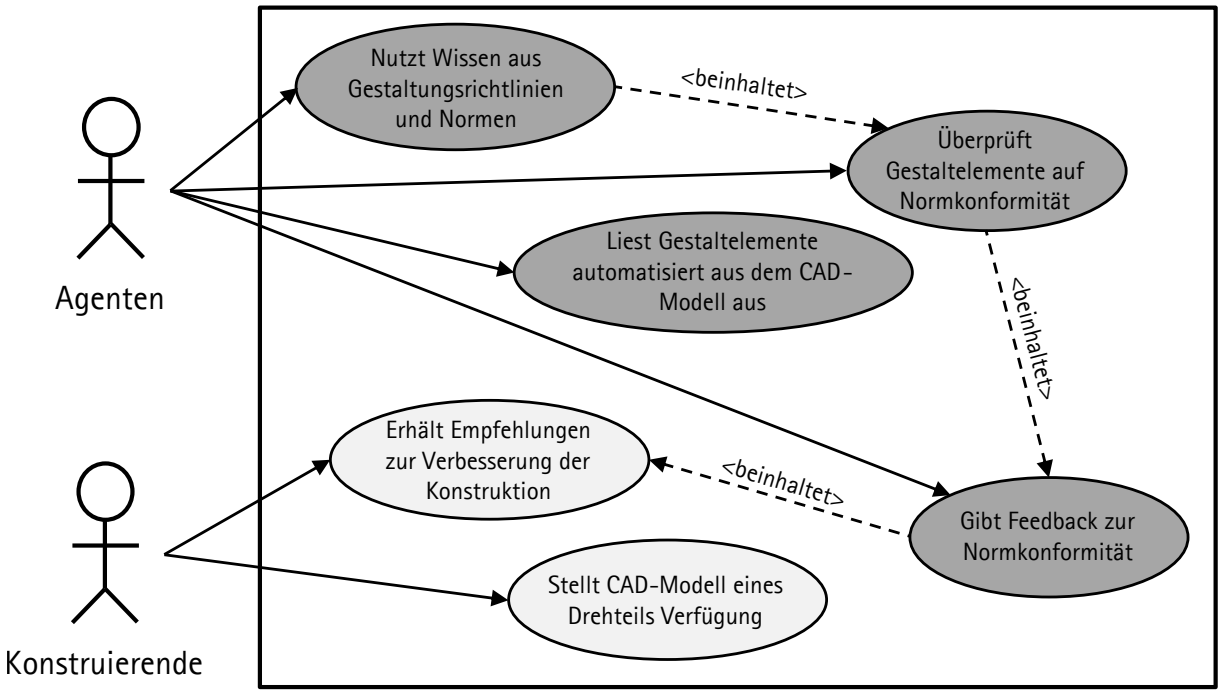


Abbildung 4.2: Anwendungsfalldiagramm für die Prüfung der Normkonformität eines Drehteils

schiedenen Rollen zugeordnet werden. Die Rollen können hierbei nach folgenden Richtlinien aufgeschlüsselt werden [IGLE96]:

- *Geografische Verteilung*: Hierbei werden die Rollen nach dem geografischen Ort aufgeteilt, z. B. wenn die Fertigung auf verschiedene Produktionsstandorte verteilt ist.
- *Logische Verteilung*: Hierbei führt eine Rolle eine oder mehrere Funktionen in der Anwendung aus, um ein „persönliches“ Ziel zu erreichen, z. B. in dem sie an einer Auktion teilnimmt.
- *Wissensverteilung*: Im Falle des Wissenserwerbs durch verschiedene Expertinnen und Experten oder beim Vorhandensein verschiedener Wissensdomänen kann für jeden Bereich eine Rolle definiert werden.

Hierbei können die Richtlinien auch miteinander kombiniert werden. Rollen bilden zudem die Grundlage für die Definition von Agenten und stellen die verschiedenen Sichtweisen während der Entwurfsphase dar. Hierdurch kann sichergestellt werden, dass alle Stakeholder und deren Belange im MAS repräsentiert werden.

Um die Rollen miteinander in Verbindung zu setzen, können diese in einem Liaisongraph dargestellt werden. Der Liaisongraph[1] besteht aus Knoten und Kanten, wobei die Knoten

[1]Ein Liaisongraph bzw. Verbindungsdiagramm ist ursprünglich eine Baugruppendarstellung, welche eine Baugruppe in Form eines Netzwerks darstellt, in dem die Knoten die Teile darstellen und die Linien zwischen

die Rollen und die Kanten zwischen den Knoten die Beziehungen zwischen den verschiedenen Rollen darstellen. Hierbei können die Rollen die verschiedenen Sichtweisen von Expertinnen und Experten repräsentieren oder Komponenten einer Baugruppe und deren Schnittstellen. Zudem kann durch die Verwendung von verschiedenen Farben oder Knotengeometrien eine Hierarchisierung der Rollen vorgenommen werden.

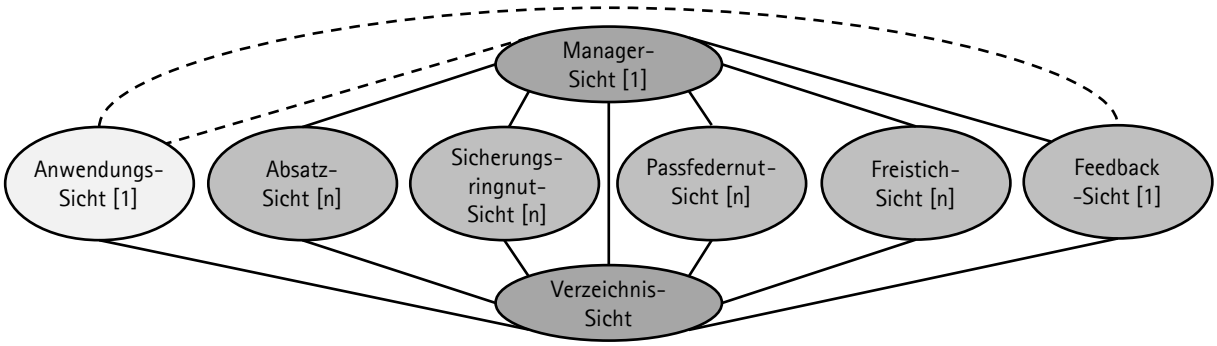


Abbildung 4.3: Liaisongraph mit Rollen und Interaktionen

In Abbildung 4.3 wird das Beispiel der Normkonformitätsprüfung bei Drehteilen aufgegriffen und die verschiedenen Sichtweisen als Rollen im Liaisongraph dargestellt. Zudem sind die *Anwendungs-Sicht*, die *Manager-Sicht* und die *Verzeichnis-Sicht* hervorgehoben, da erstere die Interaktion zu den Anwendenden repräsentiert, zweitere die Koordination der anderen Rollen orchestriert und die letztere die Verwaltung sicherstellt. Die Rollen stellen hierbei verschiedene Wissenscontainer dar, welche zu einer bestimmten Domäne Funktionen enthalten, um diese Domäne im gesamten MAS zu repräsentieren. Die Rollen stehen über Abhängigkeiten zwischen den Domänen miteinander in Verbindung. So gibt z. B. die Feedback-Sicht die Erkenntnisse aus der Normkonformitätsprüfung an die Anwendungs-Sicht weiter.

4.3 Entwurf des Multi-Agentensystems

Für den Entwurf eines MAS bzw. die Entwicklung der Agenten stellen das benötigte *Wissen*, ausführbare *Dienste* und die standardisierte *Kommunikation* eine wichtige Hilfestellung dar. Im Wissensmodell wird das spezifische Domänenwissen hinterlegt. Unter einem Dienst wird eine Funktion des Agenten verstanden, welche von jedem Agenten ausgeführt werden kann, unabhängig von seinem spezifischen Wissen. Durch die Definition von standardisierten Kommunikationsprotokollen, sind die Agenten dazu in der Lage Nachrichten auszutauschen. Die bereits in der Analysephase definierten Rollen werden nun als *Agenten* abgebildet, wobei

den Knoten eine bestimmte benutzerdefinierte Beziehung zwischen den montierten Teilen, die so genannten „Verbindungen“, welche im Allgemeinen einen physischen Kontakt zwischen den Teilen einschließen [AlGe13]

ein Agent auch mehrere Rollen einnehmen kann. Die *Bekanntschaft* definiert die Beziehungen zwischen den Agenten. Zu guter Letzt stellt der *Entwurf* die finale Implementierung und Programmierung des MAS sowie die Integration in den Kontext dar.

Wissen erfassen und formalisieren

In dieser Phase wird das benötigte domänenspezifische Wissen erfasst und formalisiert, was einen Kernaspekt der wissensbasierten Systeme darstellt. Da die Entwurfsbewertung eine sehr wissensintensive Tätigkeit ist, werden die Agenten innerhalb des MAS ebenfalls als wissensbasierte Systeme angesehen. Um diese mit dem erforderlichen Wissen auszustatten, muss dieses in einem ersten Schritt erfasst werden. Hervorzuheben ist hierbei, dass durch den dezentralen Aufbau des MAS die Erfassung und Formalisierung des spezifischen Domänenwissens ebenfalls dezentral erfolgen kann.

Für die Wissenserfassung und Formalisierung in der Domäne der Produktgestaltung kann einerseits auf *aufgabenunabhängiges* Wissen zurückgegriffen werden, wie z. B. aus Normen, welches meist in formalisierter Form zur Verfügung steht und in der Regel keinen häufigen Änderungsintervallen unterliegt. Andererseits kann *aufgabenspezifisches* Wissen aus Datenbanken oder Gestaltungsrichtlinien genutzt werden, um den Agenten das nötige Wissen zur Verfügung zu stellen, sodass diese autonom Schlussfolgerungen ziehen können. Dieses Wissen liegt meist in unstrukturierter Form vor, sodass hierfür die Schritte der Wissenserfassung und Wissensformalisierung durchlaufen werden müssen.

Um das Beispiel der Bewertung der Normkonformität von Gestaltelementen aufzugreifen, zeigt die Abbildung 4.4 zum einen die Parameter, welche für den Vergleich zu den Werten aus der Norm herangezogen werden und zum anderen, wie der Übergang der Passfedernut nicht ausgestaltet werden sollte. Diese Begebenheit der Passfedernut kann als folgende Gestaltungsrichtlinie formalisiert werden:
Gestaltungsrichtlinie. *Passfedernuten bei Umlaufbiegung nicht bis an die Übergänge heranführen, damit die Kerbwirkungen aus beiden Querschnittsveränderungen wegen erhöhter Dauerbruchgefahr nicht in einer Ebene zusammenfallen [*Witt*13].*

Um diese Gestaltungsrichtlinie im Schritt der Wissensverpackung als Programmcode ausführen zu können, muss weiteres Wissen über die vorhandenen Absätze auf der Welle bekannt sein. Die Tabelle 4.1 zeigt eine Auswahl an zur Verfügung stehenden Werkzeugen, welche für die Herstellung der Passfedernuten genutzt werden können. Durch den Zugriff auf Datenbanken

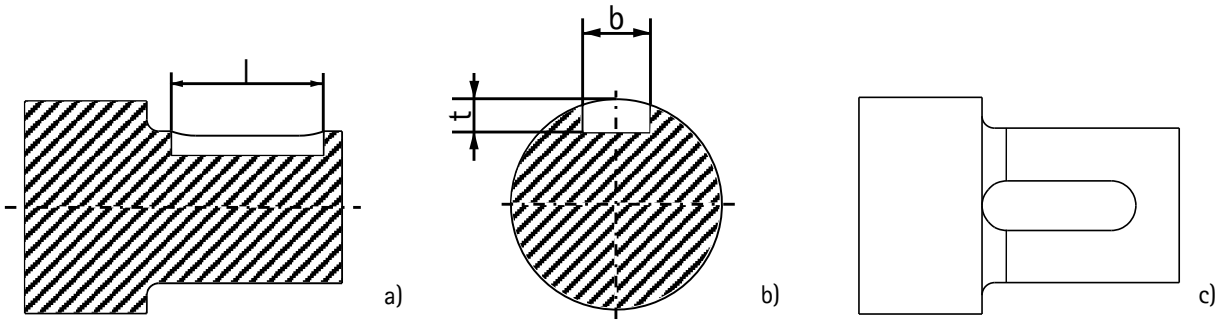
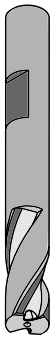


Abbildung 4.4: Gestaltung der Passfedernut: a) Länge, b) Breite und Tiefe, c) Übergang

kann sichergestellt werden, dass die Überprüfung mit den tatsächlich im Werkzeugmagazin vorhandenen Werkzeugen durchgeführt wird.



Name	Durch-messer [mm]	Schneid-länge [mm]	Frei-stellung [mm]	Schaft-länge [mm]
Langlochfräser_2	2	4	9	48
Langlochfräser_3	3	5	11	49
Langlochfräser_4	4	7	13	51
Langlochfräser_5	5	8	14	52
Langlochfräser_6	6	8	14	52
Langlochfräser_8	8	11	17	61
Langlochfräser_10	10	13	21	63

Tabelle 4.1: Werkzeugtabelle mit verschiedenen Langlochfräsern

Nachdem das Wissen formalisiert wurde, kann es nun so verpackt werden, dass es in der Wissensbasis des Agenten hinterlegt werden kann, sodass dieser auf der Grundlage des Wissens Schlussfolgerungen ziehen kann. Je nach Art des vorliegenden Wissens bieten sich verschiedene Schlussfolgerungsmechanismen an. Stellt das Wissen z. B. einen Durchlauf von Prüfkriterien dar, so können diese in Form von Regeln abgebildet werden. Lässt sich stattdessen das Wissen in einem Modell als Abbild der Realität abbilden, können modellbasierte Ansätze zum Auflösen einer spezifischen Aufgabe nützlich sein. Darüber hinaus liegt Erfahrungswissen häufig in der Form von bereits erfolgreich durchgeführten Fällen vor, welche bei einem neuen Problem erstmals mittels fallbasierten Schließens analysiert werden können, um hierbei aufgrund von ähnlichen Fällen auf ähnliche Lösungen zu schließen.

Aktionsvorrat entwickeln

Der Aktionsvorrat gibt dem Agenten die Möglichkeit auf Veränderungen in der Umwelt oder seiner Überzeugungen reagieren zu können, indem vorab Aktionen für antizipierte auslösende Ereignisse definiert werden, um stabile Zustände bei der Ausführung der Agenten zu errei-

chen. In der Realität sind die Expertinnen und Experten in ständigem Kontakt, sie besprechen sich und tauschen Informationen aus. Dieses Verhalten wird bei den Agenten des MAS ebenfalls angestrebt. Damit stellt die Kommunikation eine der grundlegenden Eigenschaften eines Agenten dar [Buss04], wofür die Funktionen des Sendens und Empfangens von Nachrichten sowie ein gemeinsames Verständnis über die Intention der Nachrichten über Performatives entscheidend sind. Ausgehend vom Liaisongraph stellt jede Kante eine Beziehung zwischen zwei Komponenten dar, die in einem MAS durch Kommunikation realisiert wird. Eine Möglichkeit, die Kommunikation zwischen Agenten zu spezifizieren, stellen UML-Sequenzdiagramme dar, welche den Austausch von Nachrichten zwischen Objekten oder in diesem Fall Agenten darstellen. Abbildung 4.5 zeigt ein Sequenzdiagramm, welches die Kommunikation zwischen einem Passfedernut-Agenten und mehreren Maschinen-Agenten abbildet. In diesem Beispiel führt der Passfedernut-Agent eine Auktion nach Vorlage des Vertragsnetzprotokolls durch, bei welcher die verschiedenen Maschinen-Agenten aufgrund ihrer Kapazität und Fähigkeiten, wie z. B. der Möglichkeit für angetriebene Werkzeuge, ein Angebot für die Bearbeitung eines Features abgeben. Nachdem der Passfedernut-Agent alle Angebote verglichen hat, informiert er die Maschinen-Agenten über die Annahme bzw. Ablehnung des Angebots. Nun können im nächsten Schritt die Funktionen für das Senden und Empfangen programmiert werden, welche zusätzlich um weitere Informationen wie z. B. den Status des Vertragsnetzprotokolls erweitert werden. Somit wissen die Agenten, wie sie auf bestimmte Nachrichten reagieren müssen, um mit anderen Agenten zu kollaborieren oder zu verhandeln.

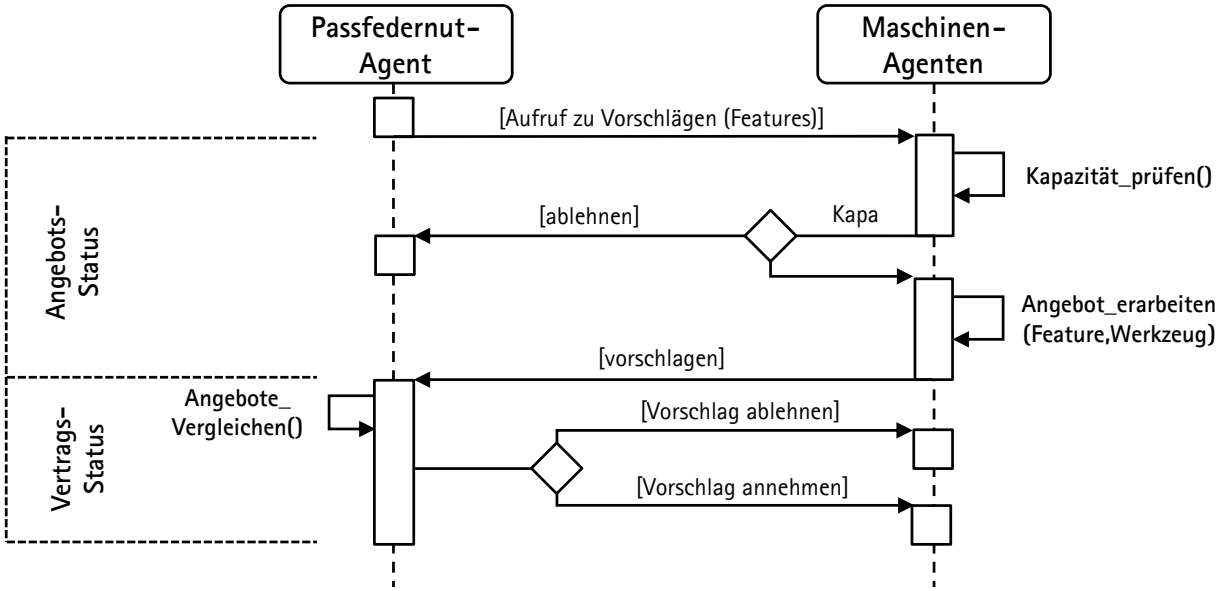


Abbildung 4.5: Vertragsnetzprotokoll für den Vergleich von Fertigungsmaschinen

Darüber hinaus ermöglicht das Dienstmodell die Definition von gekapselten Funktionen, welche je nach Bedarf den Agenten zur Verfügung gestellt werden, um den Programmieraufwand zu

reduzieren. Hierzu zählen die Verarbeitung von Informationen der Schnittstellen des Agenten, wie die Featureerkennung oder der Zugriff auf relationale Datenbanken und die Übersetzung von Wissen in einen Programmcode. Um das Beispiel der Passfedernuten aufzugreifen, zeigt der Algorithmus 1, wie ein als „geschlossene Tasche“ deklariertes Feature als Passfedernut definiert werden kann, sodass der Agent dieses Feature mit den Werten in der zugehörigen Norm vergleichen kann. Hierzu nutzt der Algorithmus die vorhandene Tabelle der Features als Eingabe. Anschließend wird für jedes Feature geprüft, ob innerhalb von *geschlossenen Taschen* Features mit drei planaren und zwei zylindrischen Flächen vorhanden sind. Falls dies der Fall ist, wird das zugehörige Feature als Passfedernut deklariert.

Algorithmus 1: Features als Passfedernuten ermitteln

```
1  Features ← Featuretabelle                    ▷ Features aus Featuretabelle laden
2  for jedes Feature in Features do
3      z=0;
4      p=0;
5      if Feature.Typ=„geschlossene Tasche“ UND Feature.Flächen.Anzahl=5 then
6          for jede Fläche in Feature do
7              if Fläche.Typ = „planar“ then
8                  p=p+1;
9              if Fläche.Typ = „zylindrisch“ then
10                 z=z+1;
11     if z=2 UND p=3 then
12         Feature.Beschreibung = „Passfedernut“;
```

Diese Form der Programmierung ermöglicht einen modularen Aufbau der Agenten, da die einzelnen Funktionen separat programmiert werden können. Des Weiteren ist es einfach möglich, die gekapselten Funktion zu warten oder zu ändern, ohne dass der gesamte Aktionsvorrat auf Konsistenz geprüft werden muss.

Agenten erstellen

Nachdem das Wissen erfasst und formalisiert, die Kommunikation definiert und die Dienste programmiert wurden, können sie nun in den Agenten integriert werden. Hierbei ist der grundlegende Aufbau der Agenten immer gleich, sodass bei Wartungen oder Änderungen eine einfache Nachvollziehbarkeit gegeben ist. Des Weiteren erhält jeder Agent eine eigene Wissensbasis und den Zugriff auf domänenspezifisches Wissen, sodass dieser selbstständig Schlussfolgerungen ziehen kann.

Für die Implementierung in die Agenten werden in einem ersten Schritt die Abhängigkeiten zwischen den Aktionen überprüft, um *Cluster* von Aktionen zu bilden, welche in einer „Welle“ ausgeführt werden können. Insbesondere die zeitliche Abhängigkeit zwischen Aktionen stellt hierbei eine relevante Information für die Programmierung dar, da zeitlich unabhängige Aktionen gleichzeitig von verschiedenen Agenten durchgeführt werden können (asynchrone Programmierung), sodass die Gesamtlaufzeit des Programms verringert wird. Falls Aktionen auf Ergebnisse aus vorherigen Aktionen angewiesen sind, werden diese als *auslösendes Ereignis* deklariert, sodass die Aktionen ausgeführt werden, sobald die vorherige Aktion das Ergebnis freigegeben hat. Nachdem die *Cluster* an Aktionen definiert wurden, können diese nacheinander in die Agenten einprogrammiert werden. Hierbei fungieren die Aktionen als eigenständige Funktionen, welche in modularer Weise in die Programmierung eingebunden werden. Neben der Einbindung der Aktionen müssen ebenfalls die Schnittstellen und Verweise für das spezifische MAS angepasst werden. Eine detaillierte Beschreibung der Programmierung und des Aufbaus der Agenten erfolgt in Kapitel 5.3.

Zur Veranschaulichung der Agenten können diese mittels Prometheus-Notation von Padgham und Winikoff [PADG04] dargestellt werden. Hierbei werden die auslösenden Ereignisse als Perzeptionen, die Veränderung der Umgebung als Aktionen sowie die übergeordneten Fähigkeiten, auszuführenden Pläne und auszutauschenden Nachrichten, aufgrund von hinterlegten Daten, repräsentiert. Abbildung 4.6 zeigt die Prometheus-Notation für einen Gestaltelement-Agenten, welcher eine Überprüfung auf Normkonformität durchführt und bei Bedarf Anpassungen an den Parametern der Gestaltelemente vornimmt.

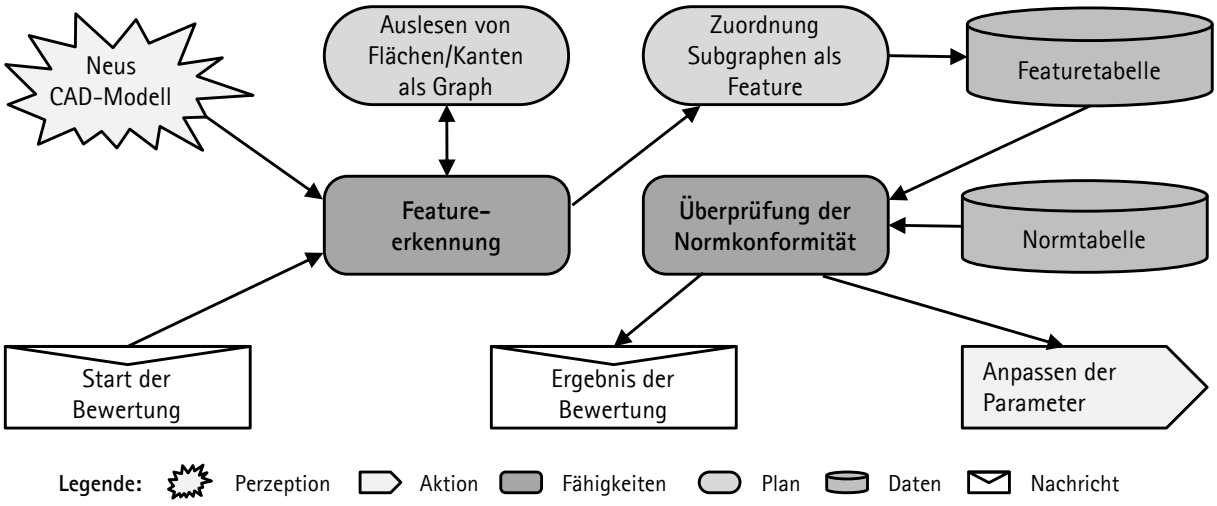


Abbildung 4.6: Prometheus-Notation für Gestaltelement-Agent zur Normkonformitätsprüfung

Agenten verwalten

Sobald die Agenten erstellt wurden, müssen sie nun in Beziehung zueinander gesetzt werden. Dies ist erforderlich, da zwar jeder Agent eine gewisse Autonomie besitzt und selbstständig Entscheidungen treffen kann, aber die Agenten kollaborativ eine gemeinsame Aufgabe bzw. Problemstellung lösen sollen. Je nach Aufgabe können nicht alle Agenten gleichzeitig arbeiten, da u. U. die Durchführung von Aktionen voneinander abhängt. Für den Fall der Entwurfsbewertung müssen z. B. erst alle Features erfasst werden, um anschließend ihre Herstellbarkeit beurteilen zu können. Um diese Abfolge von endlichen Zuständen zu koordinieren, hat sich die Nutzung eines Koordinations- oder Manageragenten bewährt [Mahe07].

Eine weitere bewährte Funktion von MAS ist die Erstellung einer Bekanntschaftsdatenbank bzw. eines Verzeichnisverwalters, welche als Art „Gelbe-Seiten-Service“ fungiert. Dies hat den Vorteil, dass während der Nutzung des MAS, z. B. je nach Auslastung oder Umfang der Aufgabe, Agenten hinzugefügt oder entfernt werden können. Hierfür registrieren sich die Agenten zum Start beim Verwalter und geben an, welche Tätigkeiten sie innerhalb des MAS übernehmen können. Somit hat das MAS immer einen Überblick über die verfügbaren Agenten und kann bei Anfragen der Agenten eine Liste von Agenten mit spezifischen Tätigkeiten zur Verfügung stellen.

Neben der Kollaboration über den Nachrichtenaustausch gibt es auch die Möglichkeit, den Agenten einen gemeinsamen Speicher namens *Blackboard* zur Verfügung zu stellen, auf den verschiedene Agenten Zugriff haben [Plap22a]. In diesem Zusammenhang ist auch die Verwendung einer gemeinsamen Ontologie von entscheidender Bedeutung, welche die Interaktionsdomäne der Agenten abdeckt, sodass die Agenten einander verstehen und Wissen über einen Anwendungsbereich kommunizieren können.

Für die Bewertung der Normkonformität von Gestaltelementen, z. B. einer Getriebewelle, kann die Erweiterung der Featuretabelle als Ontologie verstanden werden, da die Agenten mittels der Features ein gemeinsames Verständnis über die Modellierung des CAD-Modells entwickeln und darauf aufbauend eine Bewertung der Normkonformität vornehmen können, z. B. ob die Gestaltelemente nach der Norm entworfen wurden oder ob hierfür weitere Anpassungen am CAD-Modell erforderlich sind. Durch die Nutzung des Blackboards in Tabelle 4.2 können zum einen, die erkannten Features der Tabelle hinzugefügt werden und zum anderen können alle Agenten des Status sehen, inwieweit Fehler für die Normkonformität gefunden wurden.

ID	Feature	Haupt-fläche	Flächen	Kanten	Absatz	Fehler
1	Passfedernut	[3334]	[3330, 3331, 3332, 3333]	[3374, 3373, 3372, 3377, 3378, 3381, 3380, 3384]	3	Tiefe und Länge der Passfedernut nicht nach Norm DIN 6885-1, Form A
2	Sicherungsringnut	[2901]	[2900, 2902]	[2963, 2964]	2	Innendurchmesser der Nut nicht nach Norm DIN 471
3	Sicherungsringnut	[2904]	[2905, 2903]	[2966, 2967]	6	Innendurchmesser der Nut nicht nach Norm DIN 471
4	Sicherungsringnut	[2907]	[2906, 2908]	[2969, 2970]	3	-
5	Freistich	[3387]	[3385, 3386, 3388, 3389]	[3401, 3400, 3402, 3403]	6	Radius des Freistichs nicht nach Norm DIN 509
6	Freistich	[3392]	[3393, 3394, 3390, 3391]	[3405, 3408, 3407, 3406]	2	Radius des Freistichs nicht nach Norm DIN 509
7	Freistich	[3397]	[3395, 3396, 3398, 3399]	[3410, 3411, 3412, 3413]	3	Tiefe des Freistichs nicht nach Norm DIN 509

Tabelle 4.2: Blackboard für die Dokumentation der Normkonformitätsprüfung

Gesamtsystem aufbauen

Um den Aufbau und die Funktion des Systems sicherzustellen, ist nicht nur der Aufbau der Agenten und ihre Bekanntschaft wichtig, sondern auch der Einsatz des MAS im Gesamtkontext. Ein wichtiger Aspekt autonomer Agenten ist, dass sie sich in einer Umgebung befinden. In MAS wird die Umgebung von mehreren Agenten geteilt, sodass die Aktionen eines Agenten wahrscheinlich mit denen anderer Agenten interferieren. Daher ist ein expliziter Begriff der Umgebung zwar nicht zwingend erforderlich, aber hilfreich bei der Entwicklung eines MAS. Zudem muss für die Bedienung des MAS eine Benutzereingabe bereitgestellt sein und die Ergebnisse sollten nach der Bearbeitung der Aufgabe durch das MAS nachvollziehbar dokumentiert werden. Um die Implementierung des MAS im Gesamtkontext darzustellen, kann das MAS als Einsatzdiagramm strukturiert werden. Dabei können die Entwickelnden verschiedene Konfigurationen von Agenten und Netzwerken festlegen, insbesondere wenn die verschiedenen Agenten örtlich verteilt sind. Abbildung 4.7 zeigt ein Einsatzdiagramm für ein Gesamtsystem zur Bewertung der Normkonformität von Gestaltelementen. Um die Übersicht zu bewahren und die Modularität zu stärken, wird das MAS in vier Ebenen mit unterschiedlichen Abstraktionsstufen unterteilt:

- *User-Ebene*: Die User-Ebene stellt die Schnittstelle zu den Anwendenden dar, welche über eine grafische Benutzeroberfläche (GUI) Eingaben tätigen können. Darüber hinaus kann der Koordinationsagent die Anwendenden in den Entscheidungsprozess einbeziehen.
- *Management-Ebene*: Die Management-Ebene verwaltet das MAS, indem der Koordinationsagent die Informationen der Anwendenden weitergibt. Darüber hinaus werden die Informationen über die Agenten sowie deren Eigenschaften im Verzeichnis archiviert.

- *Agenten-Ebene*: Die Agenten-Ebene enthält die Agenten bzw. die verschiedenen Sichtweisen aus dem Liaisongraph, welche durch Kommunikation die Problemstellung gemeinsam lösen.
- *Operative Ebene*: Die operative Ebene enthält das spezifische Wissen der Agenten.

Durch die Unterteilung des Systems in Ebenen können Änderungen und Wartungsarbeiten separat durchgeführt werden, ohne dass das gesamte System angepasst werden muss. Ausgehend von der Agenten-Ebene kann das spezifische Domänenwissen verknüpft und die Schnittstellen zum Koordinationsagenten bzw. die Agenteninformationen in der Management-Ebene abgelegt werden. Die Schnittstelle zu den Anwendenden erfolgt zum einen über den Manageragenten und zum anderen direkt über den Feedbackagenten, sodass Nachrichten auf direktem Weg ausgetauscht werden. Auch der Austausch der GUI ist ohne großen Aufwand möglich.

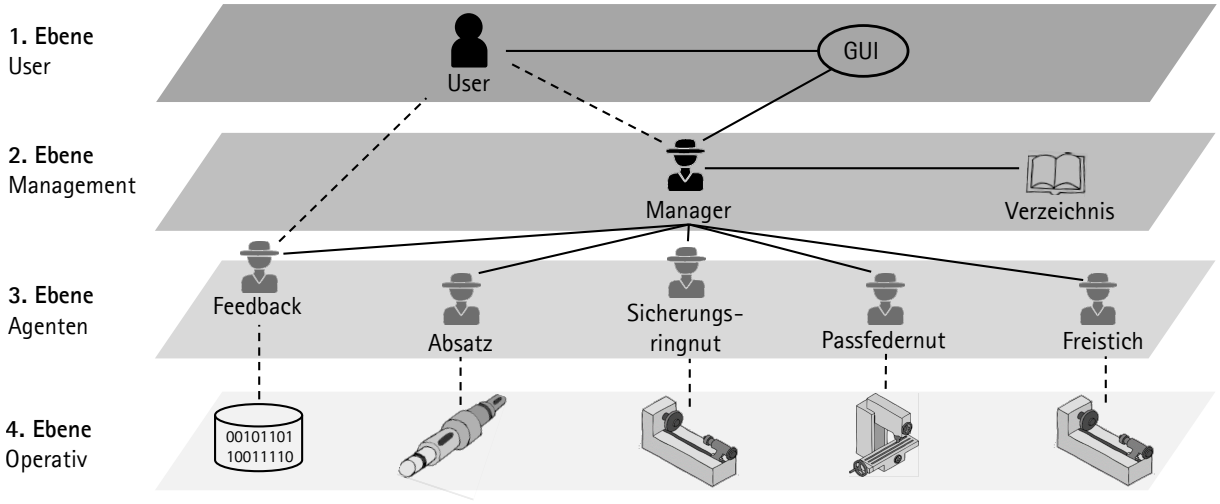


Abbildung 4.7: Einsatzdiagramm für das Gesamtsystem des MAS

In Bezug auf die Entwurfsprüfung der Produktgestalt unterstützt die Methode MaSE4D bei der Analyse des Einsatzkontextes für das MAS, da die Aufgaben und Schnittstellen zu den Anwendenden definiert und die verschiedenen Expertinnen und Experten aus den unterschiedlichen Unternehmensbereichen, in Form von Rollen, repräsentiert werden. Des Weiteren liefert die Methode für die Entwurfsphase konkrete Hilfestellungen und Modelle, um das dezentral verteilte Wissen zu erfassen und zu formalisieren. Durch den Aufbau eines Austauschs zwischen den Agenten über Nachrichten und durch die Entwicklung eines modularen Aktionsvorrats, kann der autonomen Entscheidungsfindung der Agenten Rechnung getragen werden. Aufgrund des strukturierten und methodischen Vorgehens können auch komplexe Problemstellungen im Bereich der Produktgestaltung als dezentrale Agenten abgebildet und durch ihre Kommunikations- und Kollaborationsfähigkeiten zu einem Gesamtsystem zusammengesetzt werden.

5 Entwicklungsumgebung

Neben dem methodischen Vorgehen zur Entwicklung von MAS in der Domäne der Produktgestaltung stellt die Bereitstellung einer Entwicklungsumgebung für die einfache Adaption auf spezifische Bewertungsaufgaben, vor dem Hintergrund des DfX, ein weiteres entscheidendes Kriterium für den Einsatz der Agententechnologie in der Praxis dar. Hierzu müssen zum einen, die Methoden aus der IT-Domäne in die Domäne der Produktgestaltung überführt und zum anderen die ablaufspezifischen Aufgaben während der Entwurfsbewertung der Produktgestalt repräsentiert werden. Diesen Aspekten wird durch den Aufbau einer initialen Software-Architektur Rechnung getragen, sodass eine Implementierung der Entwurfsbewertung in den Arbeitsablauf der Konstruierenden möglich ist. Zudem werden Vorlagen bzw. Templates erarbeitet, um das Programmieren von Agenten mit standardisierten Grundfunktionen zu erleichtern. Nachdem die initiale Entwicklungsumgebung aufgesetzt ist, kann diese mit domänenspezifischen Agenten erweitert werden, um Wissen dezentral zu verwalten, die Kommunikation zwischen den Agenten problemspezifisch auszugestalten und die Autonomie der Agenten durch Perzeptions- und Handlungsfähigkeiten zu ermöglichen.

5.1 Architektur der Entwicklungsumgebung

Aufgrund der Herausforderungen aus Kapitel 3.1 sollte das MAS zur Entwurfsbewertung direkt in den Arbeitslauf der Konstruierenden integriert werden, sodass eine Unterstützung während des Gestaltprozesses im CAD-Programm erfolgen kann. Hierfür muss eine Software-Architektur aufgebaut werden (Abbildung 5.1), welche diese Anforderungen erfüllt und einfach an die Bedürfnisse der Konstruierenden angepasst werden kann. Zur Programmierung des MAS wird auf die Programmiersprache Python zurückgegriffen, die zu den am weitesten verbreiteten Programmiersprachen für Anwendungen der KI gehört [Cass20, Pala20]. Damit wird auch der Kritik an JADE Rechnung getragen, denn die Wahl von Java als einzige Programmiersprache für die Entwicklung von JADE-Agenten wird in vielen Situationen, z. B. bei der agentenorien-

tierten Programmierung, als unangemessen angesehen [BERG20]. Zudem enthält Python eine Vielzahl an Bibliotheken, welche als Open Source zur Verfügung stehen, sodass deren Quelltext öffentlich und von Dritten eingesehen, geändert und genutzt werden kann. Die Programmierung des MAS erfolgt in der Programmierumgebung *Visual Studio 2022*, da diese Umgebung übersichtlich aufgebaut ist und ein Zugriff auf Git-Repositories möglich ist. Für den Aufbau des MAS wird auf die MAS-Plattform SPADE zurückgegriffen, da sie eine FIPA-konforme Kommunikation unter Verwendung von FIPA-Performatives ermöglicht. Mit der Verwendung von SPADE werden die Agenten als Container bzw. Klassen programmiert, welche durch Behaviours gefüllt werden, sodass der Agent auf Veränderungen in seiner Umgebung reagieren kann. Die Behaviours eines Agenten bilden die Grundlage für Reaktivität und Proaktivität. Hauptaufgabe der Behaviours ist, Agenteninteraktionen zu implementieren. Durch die Erweiterung der SPADE-Plattform, um das Plug-in SPADE-BDI, können Agenten programmiert werden, welche auf der BDI-Architektur basieren. Hierfür werden die Agenten mit ASL-Dateien, welche die Programmiersprache AgentSpeak nutzen, erweitert, wobei die ASL-Dateien die initialen Überzeugungen und Ziele der Agenten sowie einen Aktionsvorrat aus Plänen enthalten, welche den Handlungsspielraum der Agenten repräsentieren.

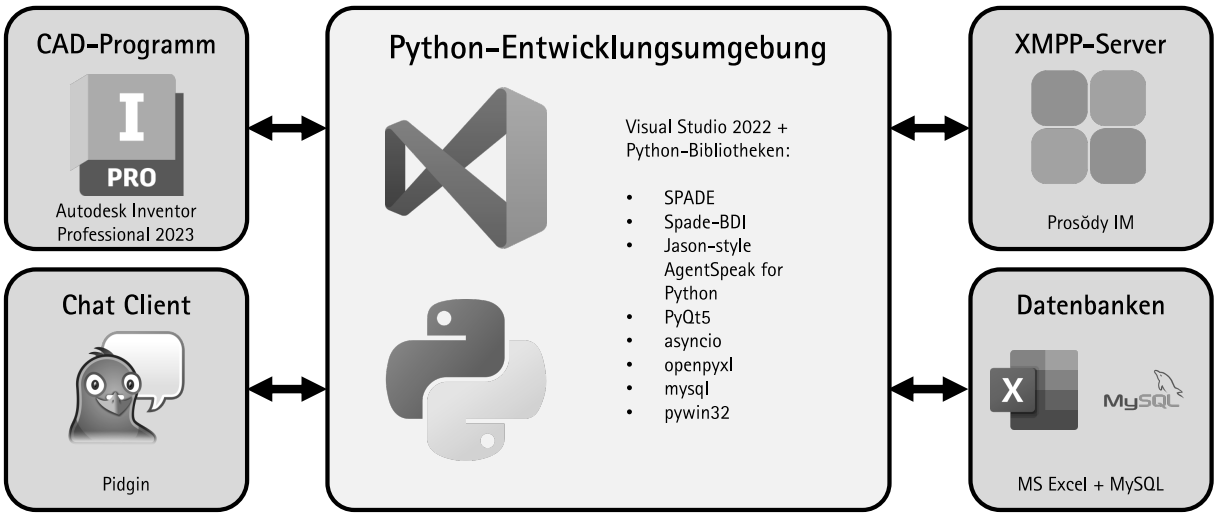


Abbildung 5.1: Software-Architektur der Entwicklungsumgebung

Neben dem Kern der Software-Architektur, der Python-Entwicklungsumgebung, wird die gesamte Entwicklungsumgebung um weitere Peripherie erweitert. Hierzu zählt zum einen der Aufbau eines Kommunikationsservers, welcher den Nachrichtenaustausch mittels XMPP unterstützt. In diesem Fall wird ein XMPP-Server mittels *Prosŏdy IM* und dem Betriebssystem Ubuntu aufgesetzt, welcher den Nachrichten-Austausch über das Internet ermöglicht. Die Kommunikation über XMPP, welches ein offenes Protokoll für Instant Messaging und Präsenzbenachrichtigungen darstellt, ermöglicht den Anwendenden, auch unabhängig vom Standort

über Messenger wie *Spark* oder *Pidgin*, direkt mit den Agenten zu kommunizieren. Um die Einbindung von bestehenden Daten, z. B. von Tabellen aus zur Verfügung stehenden Werkzeugen oder Maschinen zu ermöglichen, werden Schnittstellen zu *Microsoft Excel*, welches ein bekanntes Werkzeug für Konstruierende darstellt und relationalen Datenbanken, wie z. B. *MySQL*, bereitgestellt. Eine Erweiterung um Graphdatenbanken, wie z. B. *Neo4j*, für die Speicherung der AAG, wäre zudem denkbar.

Eine Besonderheit dieser Software-Architektur stellt die Schnittstelle zum CAD-Programm *Autodesk Inventor Professional 2023* dar, sodass die CAD-Modelle automatisiert ausgelesen und nach durchgeführter Bewertung angepasst werden können. Das CAD-Programm Inventor wird genutzt, da dieses System prinzipiell alle Methoden und Werkzeuge für ein wissensbasiertes CAD besitzt [GEMB17] und über die API auf einen Großteil der Funktionen zugegriffen werden kann. Eine detaillierte Beschreibung zur Schnittstelle zwischen der Python-Entwicklungsumgebung und dem CAD-Programm Inventor erfolgt in Kapitel 5.3.3.

5.2 Initialisierung des Multi-Agentensystems

Zur Unterstützung der Programmierenden werden *Templates* genutzt, um Teile des Codes schnell wiederzuverwenden, ohne den Code erneut zu schreiben [OLSS20]. Im Hinblick auf die Standardisierung des Agenten wird die BDI-Architektur genutzt, um ein Rahmenwerk zur Verfügung zu stellen, welches auf eine Vielzahl von Domänen angewendet werden kann. Hierbei ist das klassische Berechnungsmodell eines BDI-Agenten ereignisgesteuert, d. h. ein BDI-Agent reagiert auf Ereignisse, z. B. Veränderungen in der Umgebung oder in seinen eigenen Überzeugungen, nimmt dann einen Plan als eine seiner Absichten an und entscheidet schließlich aus der Menge der aktiven Absichten über seine nächste Aktion. Darüber hinaus bietet ein Agent eine Reihe von Diensten[1] an, die von einem anderen Agenten angefordert werden können.

Um aus dem Template ein domänenspezifisches Assistenzsystem aufzubauen, muss dieses mit spezifischen Fähigkeiten ausgestattet werden, wie z. B. der Ermittlung von fertigungsspezifischen Features aus dem CAD-Modell, der Speicherung von Domänenwissen, wie z. B. relevanten Normen oder Gestaltungsrichtlinien und der Definition des Schlussfolgerungsverhaltens auf Basis von Überzeugungen, Plänen und Intentionen. Dabei definiert die vorgeschlagene Architektur die Agenten so, dass jeder Agent verschiedene Arten von Argumentationsprozessen, z. B. prozedural-basierte und BDI-Prozesse, als Verhalten integrieren und kombinieren kann. Diese Definition stärkt das globale Verhalten des Agenten in bemerkenswerter Weise, indem

[1] Ein Dienst ist eine logische Hülle für eine Reihe von Aktionen, die einem Agenten gehören [MÖN06].

sie ihm erlaubt, verschiedene, mehr oder weniger deliberative Techniken in seinem Entscheidungsfindungsprozess auf transparente und flexible Weise zu kombinieren. Der Programmcode 5.1 stellt ein Skelett für die Python-Programmierung dar, in welches verschiedene Behaviours und Aktionen integriert werden können. Im Kopf des Templates werden die verwendeten und notwendigen Bibliotheken definiert. Diese können je nach Schnittstelle zur Umgebung bzw. Wissensdatenbanken noch erweitert werden. Des Weiteren werden die verwendeten Agenten festgelegt, indem ihre JID und ihr Passwort sowie die zugehörige ASL-Datei definiert wird. Durch die Definition der Agenten als Klassen erfolgt eine Kapselung der Funktionen der Agenten, sodass diese einfach kopiert oder durch andere Programme aufgerufen werden können. Innerhalb der Agentenklasse können zum einen Veränderungen an den Überzeugungen oder Zielen vorgenommen und zum anderen können sich wiederholende Dienste definiert werden, wie z. B. die Featureerkennung oder die Registrierung bei einer Verzeichnisverwaltung (Gelbe-Seiten-Service). Im unteren Bereich des Programmcodes wird das MAS gestartet und der Agent ausgeführt, solange dieser aktiv und „am Leben“ ist.

Zur Programmierung von Plänen, die Aktionen beinhalten, welche ausgeführt werden, sobald ein *auslösendes Ereignis* eintritt, besitzt jeder Agent eine ASL-Datei. In diesen Dateien werden die initialen Überzeugungen und Ziele definiert und während der Laufzeit angepasst. In dem Programmcode 5.2, welcher ein ASL-Template darstellt, wird eine grobe Struktur zur Nutzung von AgentSpeak gezeigt. Hierbei werden zum einen initiale Überzeugungen und Ziele definiert (Zeile 1), welche beim Starten des Agenten ausgeführt werden sollen und zum anderen kann der Start ebenfalls als auslösendes Ereignis für Pläne verwendet werden (Zeile 3). Durch die Nutzung von Plänen als Programmierobjekte können Abhängigkeiten und sich daraus ergebende Aktionen definiert werden. So wird z. B. der Plan in Zeile 6 durch das Empfangen einer Nachricht ausgelöst und daraufhin in Zeile 7 eine Antwort versendet. Innerhalb von Plänen können auch Wenn-Dann-Funktionen, sowie Schleifen programmiert werden (Zeile 11 und 12). Eine Besonderheit von AgentSpeak stellt die Abfrage und das Setzen von Überzeugungen dar (Zeile 14 und 15), da hierdurch die *Überzeugungsbasis* des Agenten verändert werden kann. Des Weiteren besteht die Möglichkeit, den Agenten direkt aus der ASL-Datei zu beenden (Zeile 18–20).

Verhaltensbasierte Architekturen für MAS eignen sich für Umgebungen mit dynamischen und signifikanten Veränderungen, in denen typische Anforderungen eine schnelle und konstante Reaktion, ein hohes Maß an Anpassungsfähigkeit und die Aufrechterhaltung eines Zustands der Umgebung sind, die es den Agenten ermöglichen, aus der Vergangenheit zu lernen. Im Vergleich zu reaktiven Architekturen verteilen die BDI-Behaviours hier also nicht nur Inputs

```python
# Notwendige Bibliotheken
import time
import asyncio
from spade import quit_spade
from spade_bdi.bdi import BDIAgent
from spade.behaviour import OneShotBehaviour, CyclicBehaviour

# Globale Variablen
server = "EnterDomain"  # Name der Server-Domäne
jid_Agent, pw_Agent = "Agent@"+server, "PW" # Benutzername + Passwort
asl_Agent = "EnterPathOfASL" # Pfad der ASL-Datei des Agenten

# Definition des Agenten als Klasse
class AgentClass(BDIAgent):
    def __init__(self, jid: str, password: str, asl: str, i: int,
    actions=None, *args, **kwargs):
        super().__init__(jid, password, asl, actions, *args, **kwargs)
        self.bdi.set_belief("DF","yellowpage@"+server)

    # Definition von Aktionen für den Zugriff aus der ASL
    def add_custom_actions(self, actions):
        @actions.add_function(".EnterNameOfFunction" ,())
        def EnterFunctionName(EnterArguments): # Funtion(Übergabewerte)
            return str(values) # Rückgabe von Werten

# Startbereich
async def main():
    print("Start MAS...")
    AgentVar = AgentClass(jid_Agent, pw_Agent, asl_Agent)
    time.sleep(1)
    AgentVar.start() # Starte Agenten
    while AgentVar.bdi.is_killed() == False:
        time.sleep(1)
    print("MAS finished...")
    quit_spade()

if __name__ == "__main__":
    asyncio.run(main())
```

Programmcode 5.1: Python-Template für Entwurfsbewertungsagenten

```
1  !start.
2
3  +!start <-
4      .print("gestartet.").
5
6  +!planname(Variable)[source(Sender)] <- // Plananfang
7      .send(<JID>, <Performative>, <Inhalt>(<Nachricht>)); // Versenden
8      .print("versendet."); // Ausgabe-Funktion
9      .funktion(Variablen); // Funktion eines Agenten aufrufen
10
11     if (Kondition){ // Wenn-Dann-Funktion
12         .funktion(); }.
13
14     ?ueberzeugung(Variablen); // Erhalte Überzeugung
15     ueberzeugung(Variablen); // Setze Überzeugung
16     !plan(Variablen); // Auslösendes Ereignis
17
18 +!kill <- // Beende Agent
19     .print("beendet");
20     .killagent(R).
```

Programmcode 5.2: Template für ASL-Datei für Entwurfsbewertungsagenten

und Outputs, sondern auch den internen Zustand und den Entscheidungsprozess des Agenten. Einige dieser Behaviours können deliberativ sein und einen Zustand und eine Weltrepräsentation aufrechterhalten. Unter anderem aus diesen Gründen werden verhaltensbasierte Architekturen derzeit am häufigsten für die Entwicklung von MAS eingesetzt [PALA22b].

5.3 Aufbau eines Entwurfsbewertungsagenten

Um ein MAS wie in der Spezifikation beschrieben aufzubauen, ist es zielführend, Agenten zu entwerfen, die standardisierte Schnittstellen haben, sodass eine flexible Kombination von Agenten möglich ist. Nwana et al. [NWAN99b] beschreiben mit ihrem *Generischen ZEUS Agenten* eine Struktur, welche die allgemeinen Funktionalitäten eines Agenten enthält, wie z. B. die Verarbeitung von Nachrichten, die Ausführung von Aktionen, die Nutzung von Informationen aus Datenbanken sowie die Koordination und Planung von Aufgaben. Die Bereitstellung dieser Funktionalitäten für die Anforderungen der Produktgestaltungsdomäne, wie z. B. die Verarbeitung von Geometrieinformationen und der Abgleich mit Normen und Gestaltungsrichtlinien, fehlt jedoch. Ziel der Entwicklung eines Entwurfsbewertungsagenten ist es, die grundlegenden Funktionalitäten auf Basis validierter Ansätze abzubilden und um die Anforderungen aus der Spezifikation zu erweitern, sodass eine Vorlage für die Konstruierenden bereitgestellt werden kann und diese sich auf die Anwendung für den konkreten Anwendungsfall in der Produkt-

gestaltung konzentrieren können. Für das Forschungsdesign bei der Entwicklung des Agenten sind die spezifischen Eigenschaften und Anforderungen von MAS und KBS sowie deren Kombination zu berücksichtigen, da in MAS die Kommunikation für die Problemlösung essenziell und in KBS die Formalisierung von Expertenwissen ein zentraler Aspekt ist. Darüber hinaus ist der Einsatzbereich und die Aufgabe des Agenten zu berücksichtigen, da dieser direkt in eine CAD-Entwicklungsumgebung eingebettet wird und die Konstruierenden bei der Bewertung des Gestaltungsentwurfs unterstützen soll. In Anlehnung an den *Generischen ZEUS Agenten* wird ein Rahmenwerk für die Entwicklung eines Agenten im Kontext der CAD-Modellierung in Abbildung 5.2 vorgestellt, welcher in die vier Module *Wahrnehmungs- und Aktionsmodul*, *Kommunikationsmodul*, *Wissensmodul* und *Entscheidungsmodul* unterteilt ist, sodass die wichtigsten Funktionalitäten getrennt voneinander entwickelt und angepasst werden können [PLAP22c].

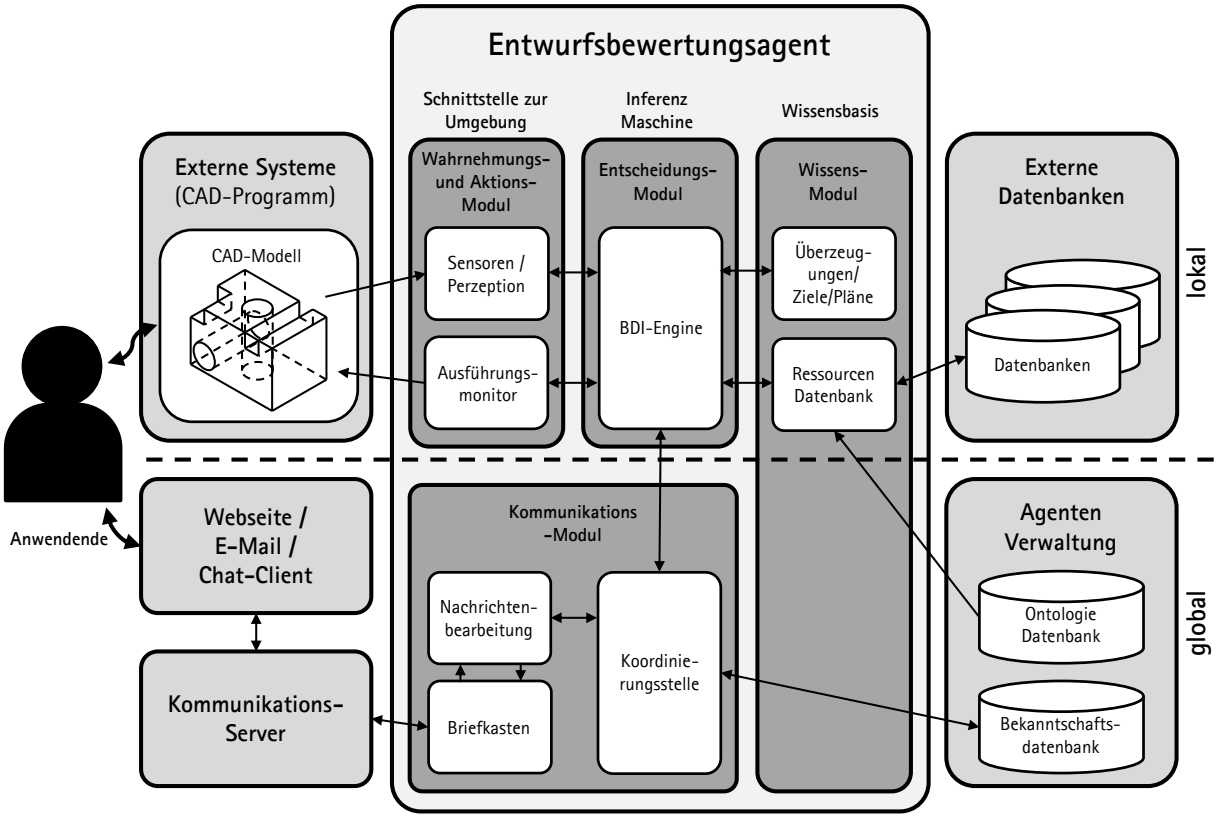


Abbildung 5.2: Aufbau eines Entwurfsbewertungsagenten (in Anlehnung an [PLAP22c])

Das *Wahrnehmungs- und Aktionsmodul* enthält die Fähigkeit zur Featureerkennung, sodass Gestaltelemente aufgrund der ausgelesenen Flächen und Kanten des CAD-Modells zugeordnet werden können. Dieser Weg funktioniert auch in umgekehrter Richtung, weswegen eine Veränderung des CAD-Modells durch die Anpassung von Parametern oder das Hinzufügen bzw. Entfernen von Features erfolgen kann. Das *Kommunikationsmodul* stellt die Kommunikation

mit anderen Agenten, den Anwendenden und anderen menschlichen Expertinnen oder Experten sicher, indem es Nachrichten sendet, empfängt und auswertet. Das *Wissensmodul* enthält das Wissen und die Überzeugungen des Agenten, damit der Agent seine Überzeugungen, Handlungspläne für bestimmte Ereignisse und Absichten zur Veränderung der Umgebung verwalten kann. Darüber hinaus greift der Agent entweder auf Wissen aus externen Datenbanken zurück oder nutzt gespeichertes Wissen in Form von Regeln, Constraints und Fallbasen. Des Weiteren kann durch die Verwendung von Ontologien, die als eine Wissensrepräsentation von erfassten Entitäten, Ideen und Ereignissen, mit ihren Eigenschaften und Beziehungen, angesehen werden können, Wissen über die Umgebung mit dem Agenten geteilt werden. Aufgrund des spezifischen Einsatzgebietes des Agenten in der Produktgestaltung können Vorkehrungen für ein gemeinsames Domänenverständnis und die Verwaltung der Ontologie getroffen werden, um z. B. Features von mechanischen Komponenten und deren Fertigungseigenschaften zu speichern. Das Problemlösungsverhalten, das ein zentraler Aspekt des *Entscheidungsmoduls* ist, wird im Fall des Entwurfsbewertungsagenten durch eine BDI-Architektur repräsentiert. Die deliberative Agentenarchitektur erlaubt es dem Agenten, explizite Ziele zu verfolgen, die ständig mit aktuellen Überzeugungen verglichen werden, um geeignete Pläne für die Umsetzung auszuwählen und diese als Absichten für den Agenten zur Ausführung bereitzustellen. Der Entwurfsbewertungsagent kann in zwei Ebenen unterteilt werden, eine globale und eine lokale Ebene, da einerseits die Kommunikation und Kollaboration mit anderen Agenten innerhalb des MAS globale Aspekte abgedeckt wird und andererseits die Wahrnehmung, Aktion und Schlussfolgerung auf lokaler Ebene erfolgt, da diese Aktivitäten vom Agenten selbstständig durchgeführt werden. Um die wissensbasierte Natur des Agenten aufzugreifen, kann dieser weiter unterteilt werden in eine *Inferenzmaschine*, eine *Wissensbasis* und eine *Schnittstelle zur Umgebung* [PLAP22c]. Eine detaillierte Beschreibung der einzelnen Komponenten des Entwurfsbewertungsagenten und deren Interaktionen wird im Folgenden erläutert:

- Der *Briefkasten* verwaltet die Kommunikation zwischen dem Agenten und anderen Agenten bzw. den Anwendenden durch Senden und Empfangen von Nachrichten.
- Die *Nachrichtenbearbeitung* verarbeitet ein- und ausgehende Nachrichten, indem sie die Metadaten ausliest oder hinzufügt und die Nachrichten für die weitere Verarbeitung durch andere Module des Agenten vorbereitet.
- Die *Koordinierungsstelle* repräsentiert den Agenten nach außen gegenüber anderen Agenten oder den Anwendenden und ist somit auch für die Koordination von Interaktionen als Initiator oder Teilnehmer zuständig, wobei bekannte FIPA-Koordinationsprotokolle und

-strategien, wie verschiedene Auktionsprotokolle oder das Vertragsnetzprotokoll, verwendet werden.

- Die *Bekanntschafts-Datenbank* registriert die Agenten im MAS in einer relationalen Datenbank und verwaltet deren Dienste ähnlich einem Gelbe-Seiten-Dienst, sodass die Koordinationsstelle die enthaltenen Informationen nutzen kann, um Kooperationsvereinbarungen mit anderen Agenten zu treffen.
- Die *Ontologie-Datenbank* enthält Ontologien und Klassifikationen von Features und deren Bezug zu Fertigungsprozessen, sodass eine Zuordnung zu den Fähigkeiten von Agenten möglich ist und eine Bewertung und Beurteilung von Komponenten unterstützt wird. Die Verknüpfung von Ontologien untereinander kann durch den Einsatz von Graphdatenbanken unterstützt werden.
- Die *Ressourcen-Datenbank* enthält explizites Wissen des Agenten in Form von Regeln, Constraints und Fallbasen, wie z. B. Normen oder Konstruktionsrichtlinien und ermöglicht dem Agenten den Zugriff auf externe Ressourcen, wie z. B. vorhandene Fertigungsmaschinen und -werkzeuge.
- Die *Überzeugungen, Ziele und Pläne* enthalten die initialen Überzeugungen, sowie gespeicherten Pläne und Absichten des Agenten und dienen somit als Aktionsvorrat, sodass der Agent auf sich ändernde Umgebungsbedingungen reagieren kann.
- Die *Sensoren und Perzeption* nehmen die Umwelt oder externe Systeme wahr, z. B. werden CAD-Modelle im CAD-Programm Inventor mittels Featureerkennung analysiert, wobei Bereiche von Flächen des B-Rep-Modells identifiziert werden, die eine gemeinsame Funktion haben oder einen Fertigungsschritt verbinden.
- Der *Ausführungsmonitor* nimmt Änderungen in der Umgebung oder in externen Systemen vor, z. B. an einem CAD-Modell, indem er entweder Features hinzufügt, entfernt oder Parameter anpasst, sodass die Korrektur des CAD-Modells automatisiert werden kann.
- Die *BDI-Engine* stellt die autonome Handlungsfähigkeit des Agenten sicher, indem sie die Überzeugungen und Ziele des Agenten ständig aktualisiert und mit der Umwelt vergleicht. Bei wahrgenommenen oder kommunizierten Veränderungen kann der Agent schnell reagieren, indem er mögliche Aktionen mit Plänen abgleicht und als mögliche Intentionen zur Ausführung speichert oder an den Ausführungsmonitor weitergibt.

Innerhalb des BDI-Agenten sind Überzeugungen, Ziele und Absichten die drei primären mentalen Einstellungen, welche die Informations-, Motivations- bzw. Entscheidungskomponenten

eines Agenten erfassen. Die BDI-Engine (Abbildung 5.3) besteht aus vier lokalen Datenbanken, welche den Arbeitsspeicher des Agenten darstellen und bei jedem Start des Agenten neu befüllt werden. Die vier Datenbanken werden unterteilt in die *Überzeugungs-Basis*, welche die Überzeugungen des Agenten enthält, die *Ereignisse*, welche die Veränderungen in der Umgebung bzw. innerhalb des Agenten dokumentieren, die *Plan-Bibliothek*, welche den Aktionsvorrat des Agenten abbildet und die *Absichten*, welche den Arbeitsvorrat des Agenten darstellen. Zudem gibt es fünf Funktionen, welche durch die Verknüpfung der Datenbanken die Schlussfolgerung innerhalb der BDI-Engine sicherstellen. Hierzu zählen die Überzeugungs-Revisions-Funktion (brf), die Ereignis-Auswahl-Funktion (S_E), die Planauswahl-Funktion (S_P), die Absicht-Auswahl-Funktion (S_I) und die Funktion für das Ausführen der Absichten.

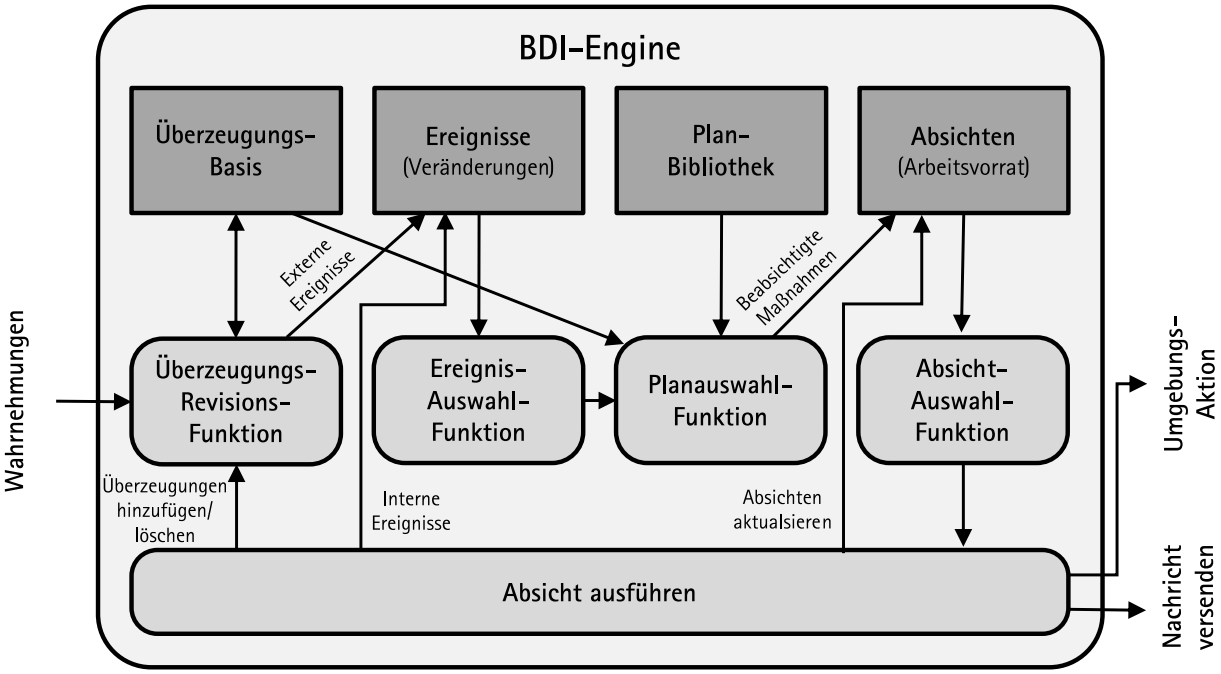


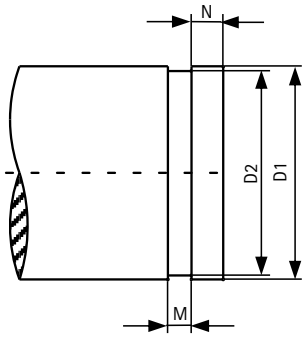
Abbildung 5.3: Architektur der BDI-Engine

5.3.1 Dezentrale Wissensverwaltung

Jeder Entwurfsbewertungsagent stellt selbst ein wissensbasiertes System dar, daher wird das benötigte spezifische Domänenwissen innerhalb des *Wissensmoduls* gespeichert und der Inferenz- bzw. BDI-Engine zur Verfügung gestellt. Für die Modellierung von Gestaltungswissen kann zwischen zwei Arten von Wissen unterschieden werden, dem Fachwissen, welches deklarativer Natur ist und unabhängig von seiner Verwendung den Inhalt bzw. die Struktur von Fakten beschreibt [Mesc12] und dem Kontrollwissen, welches das Wissen über den Prozess bzw. die Problemlösung enthält [Gemb19]. Diese Aufteilung wird ebenfalls für den Entwurfs-

bewertungsagenten aufgegriffen, indem das Fachwissen in der Ressourcen-Datenbank und das Kontrollwissen in den Überzeugungen, Zielen und Plänen gespeichert wird.

Das Fachwissen kann weiter in *aufgabenunabhängiges* und *aufgabenabhängiges* Wissen aufgegliedert werden. Ersteres beschreibt das Wissen, das in Fakten dokumentiert wird, z. B. Daten für Maschinen und Werkzeuge oder Normen. Dieses Wissen wird normalerweise in Form von Tabellen oder relationalen Datenbanken zur Verfügung gestellt. Als Beispiel hierfür kann das Wissen über Sicherungsringnuten aus der DIN 471 [DIN471] angeführt werden, welches abhängig vom Wellendurchmesser die Abmessungen für die Nut angibt (Tabelle 5.1). Innerhalb des Entwurfsbewertungsagenten kann über standardisierte Abfragen auf Daten und Informationen, wie z. B. Excel-Tabellen oder SQL-Datenbanken über die Python-Bibliotheken *openpyxl* und *mysql* zugegriffen werden, sodass diese Informationen vom Agenten weiterverwendet werden können.



Wellen-durchmesser	Nut		
D1	D2	M	N
3	2,8	0,5	0,3
4	3,8	0,5	0,3
5	4,8	0,7	0,3
6	5,7	0,8	0,5
7	6,7	0,9	0,5
8	7,6	0,9	0,6
9	8,6	1,1	0,6
10	9,6	1,1	0,6

Tabelle 5.1: Normtabelle für Sicherungsringnuten (in Anlehnung an [DIN471])

Als *aufgabenabhängiges* Wissen kann z. B. die Formalisierung von Gestaltungsrichtlinien angesehen werden, bei welchen die allgemeinen Regeln oder Formeln auf einen konkreten Anwendungsfall bezogen werden. Hierzu wird das aufgabenabhängige Wissen als Python-Funktion programmiert und kann somit in verschiedene Agenten integriert werden. Das Beispiel der Sicherungsringnut aufgreifend, kann das Vorgehen zum Abgleich der Normkonformität, für den Innendurchmesser von Sicherungsringnuten nach dem Algorithmus 2, programmiert werden.

Auch das Kontrollwissen kann in zwei Arten aufgeteilt werden. Das *Inferenzwissen* beschreibt die Anwendung von Fachwissen in einem einzelnen Schlussfolgerungsschritt [GEMB19]. Dies wird im Entwurfsbewertungsagenten in Form von Aktionen abgebildet. Das *Aufgabenwissen* verknüpft einzelne Schlussfolgerungen zu einem Plan. Dieser Plan repräsentiert das Vorgehen für die Problemlösung. Des Weiteren können mehrere Pläne innerhalb einer Plan-Bibliothek gespeichert werden und einen Aktionsvorrat für den Agenten bieten. Für die Analyse der

Algorithmus 2: Gestaltungsrichtlinie für Innendurchmesser für Sicherungsringnuten

```
1 for jede Reihe in Normtabelle in umgekehrter Reihenfolge do
2     if Zelle(Reihe=Reihe,Spalte=1).Wert ≤ D1_IST then
3         if Zelle(Reihe=Reihe,Spalte=2).Wert = D2_IST then
4             Schreibe(„Innendurchmesser nach Norm DIN 471 erstellt.“);
5         else
6             Schreibe(„Innendurchmesser ist nicht nach Norm DIN 471 konstruiert.“);
7             Gebe D2_Soll zurück;
```

normgerechten Gestaltung des Innendurchmessers einer Sicherungsringnut müssen folgende Aktionen innerhalb eines Plans durchlaufen werden:

1. Einlesen der Normtabelle nach DIN 471
2. Übergabe des Ist-Durchmessers der Welle und der Sicherungsringnut
3. Abgleich der Ist-Werte mit den Soll-Werten der Norm
4. Rückgabe des Ergebnisses des Vergleichs

Der Programmcode 5.3 zeigt die Programmierung des Plans zur Analyse der Normkonformität der Sicherungsringnut als ASL-Datei. Hierbei gibt das initiale Leistungsziel an, dass dieses beim Starten des Agenten ausgeführt werden soll. Somit wird der Plan mit dem Kopf +!start ausgeführt, da das auslösende Ereignis erfüllt ist. Zudem ist für diesen Plan kein weiterer Kontext notwendig, da keine weiteren Überzeugungen vorhanden sein müssen. Der Körper des Plans beschreibt nun die Vorgehensweise zur Prüfung, indem verschiedene Aktionen ausgeführt und dokumentiert werden.

```
1  !start. // Initiales Ziel beim Starten des Agenten
2
3  +!start <-  // Abarbeiten des Plans beim Start des Agenten
4      .print("Lese Norm DIN471 aus ...");
5      .norm_auslesen(N);       // Aktion norm_auslesen aufrufen
6      .print("Norm ausgelesen.");
7      .print("Gebe Ist-Werte ein ...");
8      .ist_werte_eingeben(I); // Aktion ist_werte_eingeben aufrufen
9      .print("Vergleiche Soll- und Ist-Werte ...");
10     .soll_ist_vergleich(N,I,R); // Aktion soll_ist_vergleich aufrufen
11     .print(R);  // Ergebnis des Vergleichs ausgeben
12     .beende_agent(B).   // Aktion zum Beenden des Agenten aufrufen
```

Programmcode 5.3: ASL-Datei für die Analyse der Normkonformität einer Sicherungsringnut

5.3.2 Kommunikation und Agentenmanagement

Die Kommunikation der Agenten untereinander oder mit den Anwendenden erfolgt im *Kommunikationsmodul* durch Senden und Empfangen von Nachrichten. Neben dem Inhalt können auch performative FIPA-Attribute wie Abfragen oder Informationen durch Metadaten angegeben werden. Durch diese Kommunikation zwischen den Agenten ist es möglich, Abstimmungen zwischen den Agenten zu ermöglichen, um Konsistenzen zwischen den Überzeugungen der Agenten sicherzustellen. Somit kann das Wissen dezentral ermittelt, gespeichert und bei Bedarf innerhalb des MAS geteilt werden. Beispielsweise kann in der Produktgestaltung eine Überprüfung des Innendurchmessers einer Sicherungsringnut durchgeführt werden. Bei fehlerhafter Nut wird die Genehmigung zur Anpassung des CAD-Modells von einem anderen Agenten oder dem Nutzer eingeholt. Hierfür führt der Programmcode 5.4 die Überprüfung nach der bereits beschriebenen Vorgehensweise in Kapitel 5.3.1 durch und sendet bei fehlerhaftem Innendurchmesser die Nachricht $< E, tell, anfrage([ID|offen]) >$ an den Empfänger E, um ihm mitzuteilen, dass er den Innendurchmesser auf den normgerechten Wert anpassen möchte. Hierfür gibt er als Inhalt eine Anfrage an, welcher er eine individuelle ID zuweist und den Status auf „offen“ setzt.

```
1 +pruefung(ID) <-    // Ausführen des Plans bei neuer Überzeugung
2     .norm_auslesen(N);  // Aktion norm_auslesen aufrufen
3     .ist_werte_eingeben(I); // Aktion ist_werte_eingeben aufrufen
4     .soll_ist_vergleich(N,I,R); // Aktion soll_ist_vergleich aufrufen
5     if (R == "falsch") {;
6         .print("Sicherungsringnut fehlerhaft. Anpassung anfragen.");
7         .send(E, tell, anfrage([ID|offen]);}.   // Sende Anfrage an E
```

Programmcode 5.4: ASL-Datei für das Senden von Nachrichten

Nachdem der Sender die Nachricht an den Empfänger übermittelt hat, landet diese im Briefkasten des Empfängers. Die Nachrichtenbearbeitung analysiert die Nachricht, indem der Sender, das Performative und der Inhalt ausgelesen werden. Entspricht der Inhalt einem *auslösenden Ereignis*, wird der entsprechende Plan aufgerufen und ausgeführt. Im Falle des beschriebenen Beispiels enthält der Empfänger-Agent den Plan mit dem Kopf $+anfrage([ID|offen])[source(S)]$ (Programmcode 5.5). Da dies dem Inhalt der Nachricht mit dem Status „offen“ entspricht, wird der Körper des Plans ausgeführt. Die Variablen ID für die Anfrage und S für den Sender der Nachricht geben Zusatzinformationen an, um die Antwort der passenden Anfrage zuordnen zu können. Nachdem der Empfänger die Anpassung der Nut

geprüft hat, sendet er entweder eine Nachricht mit der Genehmigung oder der Ablehnung der Anfrage.

```
1 +anfrage([ID|offen])[source(S)] <-  // Ausführen bei offener Anfrage
2     .print("Anfrage erhalten");
3     .anpassung_pruefen(ID,P);   // Aktion anpassung_pruefen aufrufen
4     if (P == "genehmigt") {;
5         .send(S, tell, anfrage([ID|genehmigt]); // Anfrage genehmigt
6     } else {
7         .send(S, tell, anfrage([ID|abgelehnt]);}. // Anfrage abgelehnt
```

Programmcode 5.5: ASL-Datei für das Empfangen von Nachrichten

Durch die verschachtelte Nutzung von Aktionen bzw. Python-Funktionen und dem Nachrichtenaustausch kann ein breites Spektrum an Aufgaben abgedeckt werden. Zudem können verschiedene Sichtweisen in den Agenten integriert werden, welche sie über die Nachrichten kundtun können. So können ebenfalls Verhandlungen wie z. B. das Vertragsnetzprotokoll programmiert werden, bei welchem ein Initiator eine Auktion durchführt und verschiedene Teilnehmer Angebote abgeben. Im Kontext der Konstruktion wäre z. B. die Verhandlung von verschiedenen Maschinenagenten denkbar, welche als Angebote ihre Bearbeitungszeit in Abhängigkeit von ihrer Kapazität für ein bestimmtes Feature abgeben und die Maschine mit der geringsten Bearbeitungszeit, bei gleichzeitig frühstem Starttermin, würde die Auktion gewinnen.

Ein weiterer entscheidender Aspekt für die Nutzung von Nachrichten ist die Einbeziehung eines menschlichen Anwendenden, welcher mittels Chat-Client in den Entscheidungsprozess eingebunden werden kann. Durch das Versenden der Nachrichten über das Internet können die Anwendenden auch mobil kontaktiert werden, um schnellstmöglich eine Entscheidung herbeiführen zu können. Dieser Aspekt wird wichtiger, je zeitkritischer eine Entscheidung bzw. die Prüfung des Entwurfs erfolgen muss. Zudem können Informationen über den Kontext des Entwurfs erfragt werden, welche nicht aus dem CAD-Modell ausgelesen werden können.

Zur Verwaltung der Agenten und um eine standardisierte Registrierung sowie Abfrage nach Agententypen zu gewährleisten, wird der Verzeichnisverwalter ebenfalls als Agent mit eigener JID umgesetzt. Der Programmcode 5.6 zeigt den ASL-Plan, der ausgeführt wird, sobald der Agent gestartet wird.

```
1 !start. // Initiales Ziel beim Starten des Agenten
2
3 +!start <-  // Abarbeiten des Plans beim Start des Agenten
4   .print("Gestartet.");
5   .print("Versucht sich zu registrieren.");
6   .meine_info(Info); // Aktion meine_info aufrufen
7   ?verzeichnis(B); // Überzeugung zu Verzeichnis vorhanden?
8   .print("Sende Nachricht an ", B, ".");
9   .send(B, achieve, registriere(Info)). // Nachricht versenden
```

Programmcode 5.6: ASL-Datei die Registierierung im Verzeichnis

Nachdem der Agent gestartet wurde, führt er die Aktion *meine_info* im Python-Programmcode 5.7 aus, um den Verzeichnisverwalter als Überzeugung zu speichern und seine eigene JID inklusive seiner Zuordnung zu übergeben. Dieser Schritt ist insbesondere bei der Erzeugung von neuen Agenten während der Laufzeit notwendig. Nachdem die Überzeugung für den Verzeichnisverwalter aktualisiert ist und somit der Agent einen konkreten Empfänger für den Nachrichtenaustausch hat, registriert sich der Agent.

```
1  from spade_bdi.bdi import BDIAgent
2
3  class ModelClass(BDIAgent):
4      def add_custom_actions(self, actions):
5          @actions.add_function(".meine_info" ,())
6          def meine_info(): # Übergabewerte eingeben
7              df = "yellowpage@"+self.jid[1]
8              self.bdi.set_belief("verzeichnis", df)
9              jid = self.jid[0]+"@"+self.jid[1] # JID ermitteln
10             erg = str([jid , "Model"]) # Zuordnung Service
11             return erg # Verantwortlichkeit zurückgeben
```

Programmcode 5.7: Python-Funktion für Verzeichnisregistierung

5.3.3 Perzeptions- und Handlungsfähigkeiten

Eine der wichtigsten Eigenschaften eines Agenten ist, dass er dazu in der Lage ist, in einer Umgebung zu handeln. Diese Eigenschaft des Agenten ermöglicht das *Wahrnehmungs- und Aktionsmodul*, indem der Agent seine Umgebung oder externe Systeme wahrnehmen und beeinflussen kann. Im Fall des Entwurfsbewertungsagenten werden Informationen aus dem CAD-Modell ausgelesen und nach dem Durchlaufen eines Schlussfolgerungsprozesses angepasst. Hierzu wird mittels der Python-Bibliothek *pywin32* auf die API des CAD-Programms Inventor zugegriffen,

sodass zum einen die B-Rep-Struktur des CAD-Modells ausgelesen werden kann und zum anderen die Funktionen, wie z. B. Features oder Parameter von Inventor genutzt werden können, um das CAD-Modell anzupassen. Zur Analyse des CAD-Modells wird auf die B-Rep-Struktur von Volumenmodellen zurückgegriffen und die geometrischen und topologischen Informationen der B-Rep-Struktur werden mittels AAG dargestellt, um Fertigungsfeatures effizient zu erkennen.

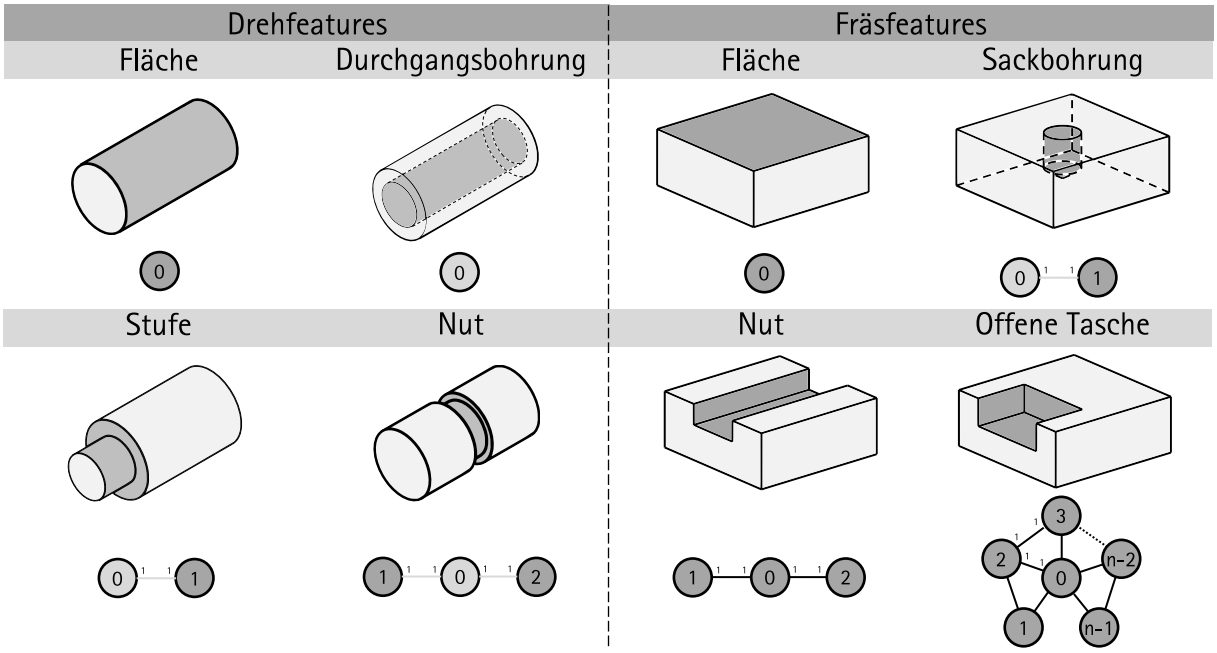
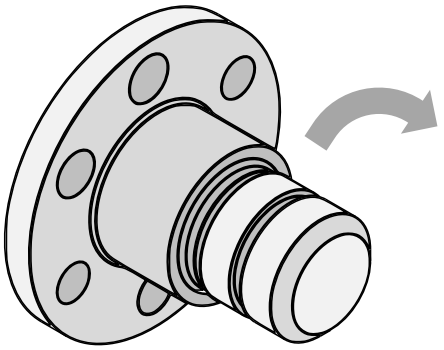


Abbildung 5.4: Ausschnitt für Typen von Dreh- und Fräsfeatures und deren Graphen (in Anlehnung an [ZHU15])

Nachdem der AAG für das CAD-Modell erstellt wurde, wird mittels graphenbasierten Featureerkennungsmechanismus nach möglichen Fertigungsfeatures gesucht. Hierbei wird jedem Feature eine eigene AAG-Darstellung zugeordnet, welche die topologischen und geometrischen Eigenschaften sowie die entsprechende Bearbeitungsmethode und zur Verfügung stehenden Fertigungswerkzeuge enthält. In Abbildung 5.4 werden verschiedene Typen von Features für die Bearbeitung mittels Dreh- und Fräsprozess dargestellt. Hierbei spielt zum einen die Anzahl der Flächen, die Anzahl der Kanten und deren Verbindungstyp sowie die Konvexität (durch eine -1 gekennzeichnet) und Konkavität (durch eine 1 gekennzeichnet) der Kanten eine entscheidende Rolle zur Beschreibung der Features. Allerdings reicht die alleinige Darstellung mittels AAG-Darstellung nicht aus, um das Feature eindeutig einem Fertigungsschritt zuzuordnen, sodass weitere Informationen wie die des Flächen- und Kantentyps oder die Normalvektoren mit erfasst werden müssen. Die aus den CAD-Modellen abgeleiteten topologischen, geometrischen und hierarchischen Informationen bilden die Grundlage für die Darstellung von Features.

Um diese Informationen kompakt darzustellen und die Kommunikation bzw. den Zugriff der Agenten auf die ermittelten Features zu ermöglichen, werden diese als *Blackboard* durch eine Featuretabelle gespeichert. Dies ermöglicht zum einen ein gemeinsames Verständnis der Agenten über das CAD-Modell und zum anderen können die gespeicherten Informationen genutzt werden, um das CAD-Modell nach erfolgtem Schlussfolgerungsprozess anzupassen. Tabelle 5.2 zeigt eine Flanschwelle, welche mittels AAG analysiert wird und deren ermittelte Features in der Featuretabelle abgelegt sind. Hierbei ist zu erkennen, dass das erste Feature als *gedrehte Nut* für einen Sicherungsring mit den zugehörigen Flächen und Kanten zu identifizieren ist. Die *gedrehte Tasche* des zehnten Features kann zudem einem Freistich zugeordnet werden.



Nr.	Feature	Flächen	Kanten
1	Gedrehte Nut	[211, 212, 213]	[231, 233]
2	Durchgangsbohrung	[199]	[ ]
3	Planare Fläche	[170]	[ ]
4	Stufe mit Radius	[205, 206, 207]	[226, 227]
5	Durchgangsbohrung	[200]	[ ]
6	Durchgangsbohrung	[201]	[ ]
7	Durchgangsbohrung	[202]	[ ]
8	Durchgangsbohrung	[203]	[ ]
9	Durchgangsbohrung	[204]	[ ]
10	Freistich	[163, 164, 165, 166, 167]	[193, 194, 195, 197]
11	Zylindrische Fläche	[169]	[ ]
12	Fase	[208]	[ ]
13	Planare Fläche	[210]	[ ]
14	Zylindrische Fläche	[209]	[ ]
15	Zylindrische Fläche	[168]	[ ]

Tabelle 5.2: Feature-Tabelle für eine Flanschwelle

Durch das graphenbasierte Vorgehen kann ein robustes Ermitteln von Features durchgeführt werden, da die einzelnen Features unabhängig im AAG ermittelt werden können. Zudem ist eine Verhandlung zwischen mehreren Agenten während der Featurebestimmung möglich. Des Weiteren ermöglicht das entwickelte System, als Eingabedatei eine STEP-Datei zu verwenden, welche die gängigste Form des Datenaustauschs ist und von vielen CAD-Systemen unterstützt wird. Allerdings gehen hierbei auch Kontextinformationen, z. B. über die prozessuale Vorgehensweise bei der Modellierung, verloren, welche anderweitig für die Schlussfolgerung herangezogen werden müssen, z. B. über die Interaktion mit den Konstruierenden.

Nachdem die Features aus dem CAD-Modell ermittelt und durch die Agenten bzw. BDI-Engine analysiert werden, kann anschließend in einem Syntheseschritt die Anpassung des CAD-Modells erfolgen. Hierzu werden die zu verändernden Features mit ihren zugehörigen Informationen wie z. B. Flächen, Kanten oder Parameterwerten aus der Featuretabelle ausgelesen und an das Aktionsmodul übergeben. Anhand der automatisierten Anpassung des CAD-Modells in Abbildung 5.5 werden die verschiedenen Aktionsmöglichkeiten dargestellt und erläutert. Hierbei wird da-

von ausgegangen, dass das zur Verfügung stehende CAD-Modell mit dem CAD-Programm Inventor und den damit enthaltenen Features erstellt wurde.

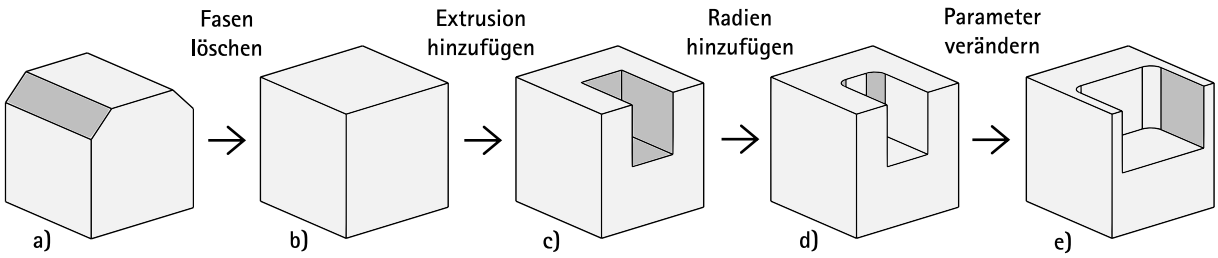


Abbildung 5.5: Automatisierte Anpassung eines CAD-Modells

Im ersten Schritt der Abbildung 5.5 werden die Fasen gelöscht, da sie beispielsweise zu einem erhöhten Fertigungsaufwand führen. Um diesen Schritt auszuführen, werden die zwei Features der Fase aus der Featuretabelle ausgelesen sowie die Flächen-ID der zugehörigen Flächen. Mithilfe dieser Flächen kann über die API auf das CAD-Modell zugegriffen werden und die zugehörigen Flächen, mit welchen Inventor die Fasen erzeugt hat, können ausgelesen werden. Sobald die Flächen in der B-Rep-Struktur ermittelt werden, können diese durch den Inventor-Befehl „Fläche löschen“ gelöscht werden und Inventor schließt automatisch die Geometrie des CAD-Modells zu einem geschlossenen Volumenkörper.

Für das Hinzufügen einer Extrusion muss zuerst festgelegt werden, welches die Ausgangsebene für das Feature sein soll. Dies kann eine bestehende Fläche innerhalb der CAD-Geometrie sein oder eine Arbeitsebene. Auf dieser Fläche wird eine planare Skizze erzeugt, welche anschließend extrudiert werden kann. Diese Vorgehensweise entspricht der Vorgehensweise, welche Konstruierende beim Aufbau eines CAD-Modells nutzen. Ebenso wird beim Hinzufügen der Radien auf das Radius-Feature zurückgegriffen, welches durch die Auswahl der zu verrundenden Kante sowie des Radius charakterisiert wird. Im Programmcode 5.8 wird der Quellcode in der Programmiersprache Python für das Hinzufügen eines Radius in das CAD-Modell dargestellt. In Zeile 1 wird die Funktion für das Hinzufügen des Radius aufgerufen und der Name des CAD-Modells, die zu verrundenden Kanten als Objekt und der Radius als Wert übergeben. Anschließend werden in den Zeilen 2–8 alle Kanten des CAD-Modells durchsucht und wenn diese mit den übergebenen Kanten übereinstimmen, einer sogenannten *Collection* hinzugefügt. In den Zeilen 9 und 10 wird der Featuretyp für einen Radius in Inventor festgelegt, sodass in Zeile 12 die Radien in das CAD-Modell eingepflegt werden können. Abschließend wird das neu hinzugefügte Feature mit seinem Namen zurückgegeben.

Im letzten Schritt wird eine Parameteränderung durchgeführt, wofür das Feature übergeben wird, welches durch die Parameteränderung beeinflusst wird. Hierfür wird die Parametertabelle in Inventor nach dem Parameterwert durchsucht. Um hierbei bereits eine Einschränkung der

```
1  def inv_addfil(ipt_name, edges, radius): # Übergabe der Radius-
       Informationen
2      inv,inv_partdoc = invcall(ipt_name) # Inventor-Datei aufrufen
3      oEdges = inv.TransientObjects.CreateEdgeCollection()
4      for edge in edges:  # Kanten der B-Rep-Struktur durchsuchen
5          for e in inv_partdoc.ComponentDefinition.SurfaceBodies.Item(1).
       Edges:
6              e = win32.CastTo(e, "Edge")
7              if e.TransientKey == edge: # Abgleich Soll- und Ist-Kanten
8                  oEdges.Add(e)
9      fillets = inv_partdoc.ComponentDefinition.Features.FilletFeatures
10     fillets = win32.CastTo(fillets, "FilletFeatures")
11     # Radien hinzufügen
12     fillet1 = fillets.AddSimple(oEdges, radius/10, AllFillets = False)
13     return fillet1.Name
```

Programmcode 5.8: Hinzufügen von Radien im CAD-Modell durch Python

Menge an Parameterwerten vorzunehmen, kann mithilfe der Flächen des zugehörigen Features, wie beim Löschen von Features gezeigt, auf das zugehörige Inventor-Feature geschlossen werden. Über dieses Inventor-Feature ist es möglich, auf die zu dem Feature zugehörigen Parameterwerte zuzugreifen. Wurde der benötigte Parametereintrag in der Parameterliste gefunden, kann dieser nun überschrieben und das CAD-Modell aktualisiert werden.

6 Anwendung des Multi-Agentensystems

Das entwickelte Vorgehen der *MaSE4D* und die Anwendung des Entwurfsbewertungsagenten als Template wird in den folgenden Kapiteln zunächst anhand eines Anwendungsbeispiels für die Bewertung der Herstellbarkeit mittels Fräsprozesses detailliert erläutert und anschließend mit unterschiedlichen Bauteilen validiert.

6.1 Problemstellung und Systemkontext

Die Hauptaufgabe des MAS ist die virtuelle Entwurfsprüfung von Frästeilen, bei welcher 3D-Modelle auf ihre Herstellbarkeit, anhand von Gestaltungsrichtlinien und Werkzeug-, sowie Maschinenverfügbarkeit, geprüft werden. Hierbei ist das Ziel des MAS die Vermeidung von Fehlern in der nachgelagerten Produktion, durch die Überprüfung der Herstellbarkeit und ggf. die Optimierung der Bauteilgestalt. Hierzu soll das System unabhängig von der Vorgehensweise beim Gestalten sein und für jegliche Fräsbauteile genutzt werden können. Der Fräsprozess wird nach der DIN 8580 [DIN8580] in die Hauptgruppe der trennenden Fertigungsverfahren eingeordnet, da hierbei die gewünschte Geometrie eines Körpers durch Verminderung oder lokales Aufheben des Zusammenhalts der Form eines festen Körpers entsteht. Da die Form des Fräswerkzeugs exakt durch Flächen, Winkel und Radien sowie durch die Anzahl und Lage der Schneiden beschrieben werden kann, wird deutlich, dass sich diese Werkzeugmaschinengruppe ideal zur Werkzeugbahnsteuerung anbietet, da sich die Bewegung des Bearbeitungswerkzeuges im Wesentlichen an der Geometrie des Fertigteils orientieren muss [VAJN09].

Durch die Befragung von Konstruierenden und Fertigenden, im Hinblick auf die Herausforderungen, welche sich bei der Gestaltung und Fertigung von Fräsbauteilen ergeben, konnten die in Abbildung 6.1 dargestellten *User Stories*[1] abgeleitet werden. Hierzu haben die Konstruierenden Anforderungen an die automatisierte Bewertung von Fräsentwürfen, die Anpassungen des

[1] User Stories stellen Software-Anforderungen dar, welche nach folgendem Schema aufgebaut sind: „Eine *Rolle* möchte ein *Ziel/Wunsch*, um einen *Nutzen* zu generieren“.

CAD-Modells und die Dokumentation des Analyse- und Syntheseprozesses formuliert. Dies deckt sich generell mit den Anforderungen aus Kapitel 3.1. Allerdings muss die in Kapitel 5 beschriebene Entwicklungsumgebung an die spezifischen Anforderungen und Restriktionen des Fräsprozesses angepasst werden. Diese können mit den Anforderungen der Fertigenden erweitert werden, wie z. B. der Nutzung von bestehende Fräswerkzeugen und -maschinen, der Reduzierung von Werkzeugwechseln und der Vermeidung von Elementen, welche keine Funktion erfüllen und somit die Bearbeitungskosten erhöhen.

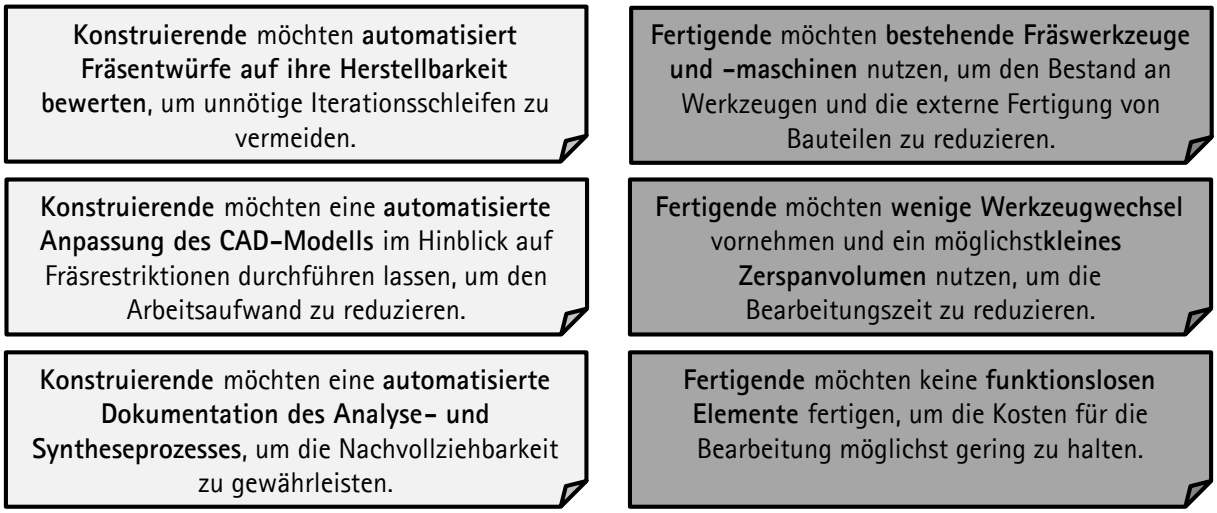


Abbildung 6.1: User Stories für die Prüfung der Herstellbarkeit von Frästeilen

Da der Fräsprozess ein breites Spektrum an herzustellenden Bauteil- und Werkzeuggeometrien sowie Maschinenarten abdeckt, werden für das beschriebene MAS einige Restriktionen vorgenommen. Einerseits werden zylindrische Fräs- und Bohrwerkzeuge genutzt, da sie häufig eingesetzt werden und für die grundlegende Bearbeitung der Bauteile universell geeignet sind. Andererseits liegt der Fokus bei den zu ermittelnden Features auf klassischen Fertigungsfeatures, wie z. B. Fasen, Radien, Bohrungen, Nuten, Taschen und Stufen. Für die Analyse werden die Modelle u. a. auf Hinterschnitte, maximale Nutzungslänge des verwendeten Fräswerkzeugs und fehlende Werkzeugausläufe geprüft.

6.2 Instanziierung der Templates

In diesem Kapitel wird der Einsatz der Entwicklungsumgebung aus Kapitel 5 und die Adaption auf die Bewertung der Herstellbarkeit von Frästeilen beschrieben. Hierzu kann auf der bestehenden Software-Architektur aufgebaut werden, indem der XMPP-Server für den Nachrichtenaustausch genutzt wird. Zudem werden die Agenten standardmäßig mit der Funktion zur Registrierung beim Verzeichnis-Agenten und zur Abfrage des Manager-Agenten ausge-

stattet, sodass diese innerhalb des Netzwerks, auch bei der Erzeugung zur Laufzeit, gefunden werden können. Des Weiteren kann die Python-Funktion zum Auslesen der B-Rep-Struktur aus dem CAD-Programm und dem Aufbau als AAG als Dienst wiederverwendet werden. Somit verfügt das bereitgestellte MAS bereits über die Funktionalitäten der Kommunikation, der Schnittstelle zum CAD-Programm bzw. der Featureerkennung und der Nutzung von Chat-Clients, zur Interaktion mit den Anwendenden.

Für die Adaption auf die spezifische Problemstellung, z. B. im Hinblick auf das verwendete Fertigungsverfahren, müssen folgende Ergänzungen durch die Entwickelnden programmiert werden. Zuerst muss der neu zu entwickelnde Agent mit der Server-Domäne, einem Agentennamen und einem Passwort für die Registrierung beim XMPP-Server versehen werden. Die Registrierung des Agenten auf dem XMPP-Server erfolgt automatisch, sodass Agenten während der Laufzeit erzeugt werden können. Zudem muss der Agent mit einer agentenspezifischen ASL-Datei verknüpft werden, um die Vorteile der BDI-Architektur nutzen zu können. Zur Anpassung der Featureerkennung auf die Problemstellung müssen die Graphentypen hinterlegt werden, wie z. B. bei der Zusammensetzung einer Passfedernut aus drei planaren und zwei zylindrischen Flächen. Hierzu muss u. U. auch die Schnittstelle zur Veränderung der CAD-Modelle angepasst werden, falls eine Löschung von Flächen innerhalb eines Features oder die Variation von Parametern erforderlich ist. Des Weiteren müssen, insbesondere unter dem Gesichtspunkt des betrieblichen Einsatzes, die vorhandenen Werkzeuge und Maschinen definiert werden. Bei neuen Fertigungsverfahren müssen ggf. die Gestaltungsrichtlinien als Python-Funktionen programmiert werden, um anhand dieser das Bauteil bewerten zu können.

Im Hinblick auf die genutzten Wissensbasen stellt dieser Ansatz eine technologieoffene Lösung dar, da die Agenten mit spezifischen Schnittstellen ausgestattet werden können, ohne dass dies Einfluss auf den Rest des MAS hat. Hierzu muss lediglich das dahinterliegende Informationsmodell ausgewertet und an den Agenten übergeben werden, sodass auch auf dynamische Zustandsänderungen reagiert werden kann. Hierzu wäre die dauerhafte Speicherung in Blackboards, welche durch sogenannte Event-Trigger aktualisiert werden, geeignet. Eine weitere Adaption stellt die Kommunikation zu den Anwendenden dar, da hierbei die möglichen Kommunikationspfade vorab definiert werden müssen, um die Antworten der Anwendenden den daraufhin auszuführenden Aktionen der Agenten zuordnen zu können.

6.3 Aufbau des Multi-Agentensystems

Für den Aufbau des MAS für die Entwurfsbewertung von Fräskonstruktionen wird zum einen die Vorgehensweise nach der *MaSE4D*-Methode angewendet und zum anderen die vorgestellten Templates des Entwurfsbewertungsagenten mit spezifischem Domänenwissen erweitert, um jegliche 2,5 D-Frästeilgeometrie analysieren zu können. Nachdem der initiale Systemkontext in der Problemstellung analysiert wird, werden im nächsten Schritt die Aufgaben des MAS mittels Anwendungsfalldiagramm spezifiziert (Abbildung 6.2).

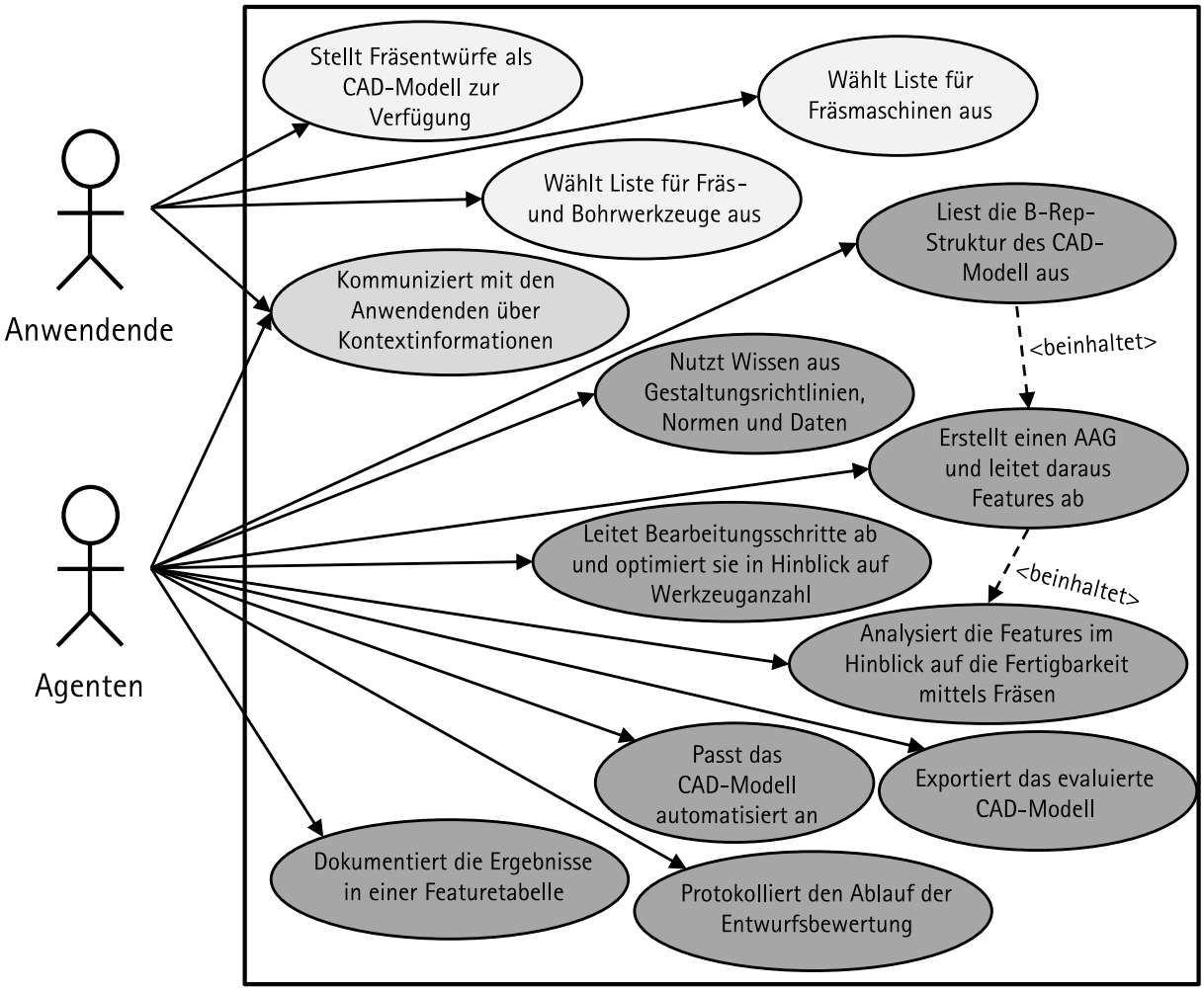


Abbildung 6.2: Anwendungsfalldiagramm für die Prüfung der Herstellbarkeit von Frästeilen

Hierbei stellt die anwendende Person den Entwurf des Fräsbauteils als CAD-Modell zur Verfügung und wählt in diesem Anwendungsbeispiel die Liste für die Fräs- und Bohrwerkzeuge sowie eine Liste für die zur Verfügung stehenden Maschinen aus. Es wäre zudem möglich dieses Wissen direkt in den zugehörigen Agenten zu hinterlegen bzw. mit einem Fertigungsmanagementsystems (engl. manufacturing execution system (MES)) zu koppeln. Anschließend übernimmt das MAS in Form der Agenten, indem es die B-Rep-Struktur des CAD-Modells ausliest und

einen AAG erstellt. Mittels hinterlegtem Wissen können die Features des Fräsbauteils abgeleitet und deren Herstellbarkeit mittels Fräsprozesses analysiert werden. Aus diesen Erkenntnissen werden Bearbeitungsschritte abgeleitet und das CAD-Modell automatisiert angepasst. Als Ergebnis liefert das System die dokumentierten Ergebnisse als Featuretabelle, das Protokoll der Entwurfsbewertung und den Export des evaluierten und angepassten CAD-Modells.

Anhand dieser Aufgaben können im nächsten Schritt die Rollen und deren Interaktion abgeleitet werden (Abbildung 6.3). Zur Koordination des Prozesses der Entwurfsbewertung wird die Rolle der Manager-Sicht eingeführt, welche das System überwacht und das Gesamtproblem auf die untergeordneten Instanzen verteilt. Da ebenfalls Instanzen der Agenten bzw. Rollen während der Laufzeit erzeugt werden sollen, wird zudem die Verzeichnis-Sicht benötigt, damit sich die Agenten bei ihrer Erzeugung bei der Verzeichnis-Rolle registrieren können und so von anderen Akteuren im MAS gefunden werden.

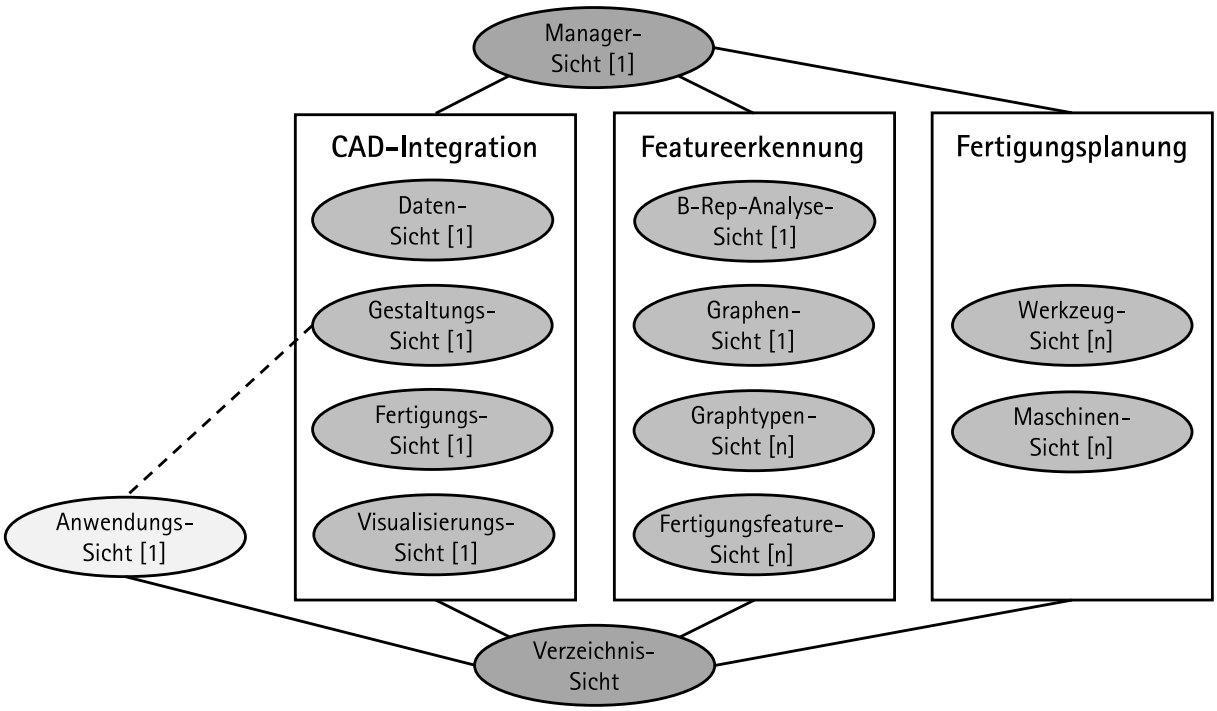


Abbildung 6.3: Liaisongraph mit Rollen und Interaktionen

Die Rollen der Daten-Sicht, Gestaltungs-Sicht, Fertigungs-Sicht und Visualisierungs-Sicht können dem Aufgabenbereich der CAD-Integration zugeordnet werden. Hierbei beinhaltet die Daten-Sicht die Meta-Daten des CAD-Modells, wie z. B. die konstruierende Person, den Werkstoff oder die Bauteilnummer. Die Gestaltungs-Sicht stellt die Schnittstelle zur Anwendungs-Sicht und damit zur anwendenden Person sicher, insbesondere wenn zusätzliche Kontextinformationen benötigt werden. Die Fertigungs-Sicht nimmt Veränderungen des CAD-Modells im Hinblick auf die Herstellbarkeit vor. Zur besseren Nachvollziehbarkeit der zugeordneten Featu-

res werden die CAD-Modelle durch die Visualisierungs-Sicht eingefärbt. Die B-Rep-Analyse-, Graphen-, Graphtypen- und Fertigungsfeature-Sicht sind für die Featureerkennung zuständig. Hierbei liest die B-Rep-Analyse-Sicht die B-Rep-Struktur des CAD-Modells aus und speichert diese in Form einer Matrix aus Flächen und Kanten ab. Diese Matrix wird von der Graphen-Sicht genutzt, um hieraus einen AAG aufzubauen und diesen in verschiedene Subgraphen aufzuteilen. Diese Subgraphen werden von der Graphtypen-Sicht gedeutet und einem Graphtypen zugeordnet. Aufgrund dieser Graphtypen ist es der Fertigungsfeature-Sicht möglich, die spezifischen Fertigungsfeature abzuleiten. Die Maschinen- und Werkzeug-Rollen stellen die Maschinen- und Werkzeugverfügbarkeit fest. Auch bei den Maschinen wäre eine Aufteilung in weitere Instanzen denkbar, wenn es um die Verfügbarkeit geht, somit können sie untereinander aushandeln, wer den Auftrag bearbeitet. Insgesamt stellen die verschiedenen Rollen unterschiedliche Sichtweisen bzw. Teilaufgaben der gesamten Entwurfsbewertung für Fräsbauteile dar, sodass im nächsten Schritt für jede einzelne Rolle das spezifische Domänenwissen erfasst und formalisiert werden kann.

Nachdem die Analysephase abgeschlossen ist, wird in der Entwurfsphase das MAS nach dem Bottom-Up-Ansatz, ausgehend vom Wissensmodell, Schritt für Schritt aufgebaut. Um das Beispiel der Bewertung der Herstellbarkeit der Fräskonstruktion aufzugreifen, zeigt die Abbildung 6.4, dass aufgrund der runden Form der Mantelfläche des Fräswerkzeugs keine scharfe Kante in einer Vertiefung ausgeformt werden kann. Die Übergänge zwischen den Seitenflächen müssen daher stets verrundet sein. Dabei kann der Eckradius nicht kleiner sein als der Radius des Fräswerkzeugs. Um ein Anhalten des Fräskopfs in einer Ecke zu verhindern, sollte der tatsächliche Eckradius stets größer als der Werkzeugradius sein. Diese Begebenheit des Fräsprozesses kann als folgende Gestaltungsrichtlinie formalisiert werden:

Gestaltungsrichtlinie. *Alle Flächen, die als Innenseite einer Tasche gefräst werden, müssen verrundete Kanten zu anderen Seitenflächen einer Tasche aufweisen. Dabei ist der Verrundungsradius größer als der Radius des Hauptkörpers des verwendeten Fräswerkzeugs zu wählen.*

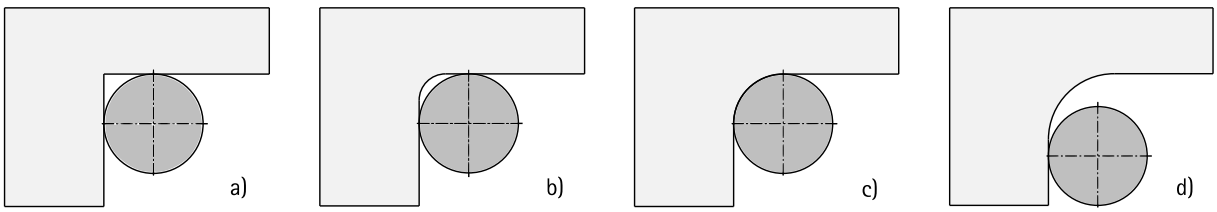


Abbildung 6.4: Ausformungen von Verrundungen mit einem zylindrischen Werkzeug

Um diese Gestaltungsrichtlinie im Schritt der Wissensverpackung als Programmcode ausführen zu können, muss weiteres Wissen über die vorhandenen Werkzeuge und deren Durchmesser

bekannt sein. Da diese sich je nach Maschine unterscheiden können, müssen diese spezifischen Informationen ebenfalls als Datenbank zur Verfügung gestellt werden. Tabelle 6.5 zeigt eine Auswahl an zur Verfügung stehenden Werkzeugen, welche für die Bewertung der Herstellbarkeit genutzt werden können. Die vollständigen Listen für die zur Verfügung stehenden Maschinen und Werkzeuge sind in Anhang A beigefügt. Durch den Zugriff auf Datenbanken kann sichergestellt werden, dass die Überprüfung mit den tatsächlich im Werkzeugmagazin vorhandenen Werkzeugen durchgeführt wird.

Name	Anwendung	Durch-messer [mm]	Schneid-länge [mm]	Frei-stellung [mm]	Länge Schaft [mm]	Radius [mm]	Winkel [°]
Messerkopf _10	Schruppen	10	10	30	50	0,4	45
Messerkopf_20	Schruppen	20	10	35	60	0,8	45
Schaftfräser_F_15	Schlichten	15	40	50	50	0,8	90
Schaftfräser_F_20	Schlichten	20	60	70	50	0,8	90
Schaftfräser_G_15	Schruppen	15	40	50	50	0,8	90
Schaftfräser_G_20	Schruppen	20	60	70	50	0,8	90
Schaftfräser_GF_5	Schruppen & Schlichten	5	30	40	40	0,8	90
Schaftfräser_GF_10	Schruppen & Schlichten	10	40	50	50	0,8	90
Fasenfräser_8	Fasen	8	5	5	40	0	45
Fasenfräser_10	Fasen	10	8	8	40	0	45
Bohrer_5	Bohrungen	5	50	50	40	0	118
Bohrer_8	Bohrungen	8	50	50	40	0	118
Bohrer_10	Bohrungen	10	60	60	40	0	118

Abbildung 6.5: Werkzeugtabelle mit verschiedenen Fräsern und Bohrern

Um dieses Domänenwissen für die Agenten handhabbar zu machen, wird dieses in Aktionen aufbereitet und der Zeitpunkt für deren Aktivierung definiert. Der Aktionsvorrat des MAS besteht hierbei aus allgemeinen Aktionen, welche jeder Agent ausführen können muss, damit eine Kommunikation und Kollaboration unter den Agenten möglich ist. Zu den allgemeinen Aktionen gehört u. a. die Registrierung beim Verzeichnis, welches in diesem Fall als eigenständiger Agent programmiert ist. Hierbei senden die Agenten bei ihrer Erstellung während der Laufzeit eine Nachricht mit ihrer spezifischen JID und ihrer Zugehörigkeit an den Verzeichnis-Agenten. Dieser speichert diese Informationen wiederum in einem *Dictionary*[2], sodass diese Informationen mit anderen Agenten geteilt werden können, wenn diese eine Anfrage an den Verzeichnis-Agenten stellen. Die Aktion der Anfrage und der selbstständigen Beendigung der Agenten, wenn sie ihre Aufgabe erfüllt haben, sind ebenfalls allgemeine Aktionen. Zu den spezifischen Aktionen gehören u. a. die Ermittlung von passenden Maschinen, die Zuordnung der Features zu den Graphentypen oder das Auslesen der B-Rep-Struktur des CAD-Modells. Darüber hinaus ermöglicht das Dienstmodell die Definition von gekapselten Funktionen, welche je nach Bedarf den Agenten zur Verfügung gestellt werden, um den Programmieraufwand zu reduzieren. Hierzu zählt u. a. die Übersetzung von Wissen in einen Programmcode. Die Gestaltungsrichtlinie für die Verrundung von Werkzeugausläufen wird nun im Algorithmus 3

[2]Der Datentyp *Dictionary* (dt. Wörterbuch) ermöglicht die Erstellung einer Zuordnungstabelle, bei welcher einem Schlüssel mehrere Einträge zugeordnet und über diesen Schlüssel auch abgefragt werden können.

abgebildet, sodass der Agent diese automatisiert in das CAD-Modell einpflegen kann. Hierzu nutzt der Algorithmus die vorhandene Werkzeugliste und die Tabelle der Features als Eingabe. Anschließend wird für jedes Feature geprüft, ob innerhalb von *offenen Taschen* oder *geschlossenen Taschen* scharf ausgeformte Kanten vorliegen. Falls dies der Fall ist, wird die zugehörige Kante ausgelesen und mit dem Werkzeugradius verrundet.

Algorithmus 3: Fehlende Werkzeugausläufe ermitteln und hinzufügen

```
1  Werkzeuge ← Werkzeugliste;                          ▷ Werkzeuge aus Datenbank laden
2  Features ← Featuretabelle;                          ▷ Features aus Featuretabelle laden
3  for jedes Feature in Features do
4      if Feature.Typ=„offene Tasche“ ODER Feature.Typ=„geschlossene Tasche“ then
5          for jede Fläche in Feature do
6              if Fläche.Beschreibung ≠ „Grundfläche“ then
7                  for jede Nachbarfläche der Fläche do
8                      if Nachfläche ≠ „Grundfläche“ then
9                          if Nachbarfläche.Typ ≠ „zylindrisch“ then
10                             Kante ← Kante_auslesen(Fläche,Nachbarfläche)
11                             R_Werkzeug ← Radius_ermitteln(Werkzeuge)
12                             Verrundung ← Feature_hinzufügen(Kante,R_Werkzeug)
```

Für die Bewertung der Herstellbarkeit werden einerseits die Subgraphen des AAG ermittelt und andererseits diese mit hinterlegtem Wissen in Form von Zuordnungen von Features zu Graphentypen abgeglichen, um darauf aufbauend domänenspezifische Analysen der Features durchführen zu können. Dieses Wissen liegt z. B. in Form der Tabelle in Tabelle 6.1 vor. Die Zuordnung der Graphtypen zu Fertigungsfeatures muss für jedes zu überprüfende Fertigungsverfahren erzeugt werden. Neben dem Wissen über die Fräsfeatures, spielt das Wissen über vorhandene Werkzeuge und Maschinen eine wichtige Rolle, um den Bestand an Werkzeugen zu reduzieren und Standardisierungspotenziale zu heben. Die verwendeten Werkzeug- und Maschinenlisten sind in Anhang A hinterlegt. Aufgrund der ermittelten Fertigungsfeatures und der hinterlegten Werkzeuglisten kann nun die Herstellbarkeit mittels Fräsprozess überprüft werden. Hierzu wird z. B. die Bearbeitungsrichtung, die Schnitttiefe oder die Radiusgröße von Taschen überprüft. Zudem werden Oberflächenangaben ausgelesen, um entsprechende Schrupp- und Schlichtwerkzeuge auszuwählen. Des Weiteren kann das Wissen aus Gestaltungsrichtlinien hinterlegt werden, um z. B. fehlende Werkzeugausläufe einzufügen.

Nachdem diese Aktionen ermittelt und programmiert wurden, können sie im nächsten Schritt den Agenten zugeordnet werden. Die Agenten werden aus den Rollen des Liaisongraphs abgeleitet, wobei ein Agent mehrere Rollen besitzen kann. Hierbei spielt neben der Verteilung des Wissens auch die Instanziierung der Agenten eine Rolle, z. B. ob die Agenten als einzel-

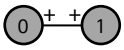
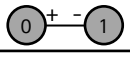
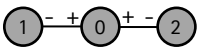
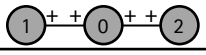
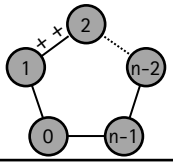
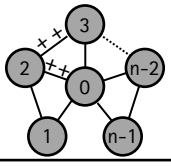
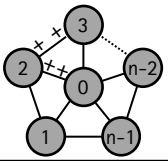
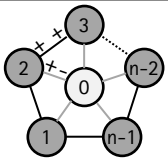
	Graphentyp	Features	Flächentyp	Verbindung
A		Fläche	Ebene Fläche	
		Fase	Ebene Fläche	Achse → X,Y,Z≠1
		Rundung	Zylindrische Fläche	Tangentiale Nachbarflächen
		Durchgangsloch	Zylindrische Fläche	
B		Stufe	2x Ebene Fläche	
		Sackloch	Zyl. & Konus-Fläche	
		Runde Extrusion	Zyl. & Ebene Fläche	
C		Zyl. Stufe	Zyl. & Ebene Fläche	
D		Runde Nut	1x Zyl. Fläche	
		Nut		
E		Nut		
F		Durchgehende Tasche		Normalvektoren in einer Ebene
G		Offene Tasche		Offene Kante
H		Geschlossene Tasche		Nur äußere Kantenzüge
I		Erhebung		Mit innerem Kantenzug

Tabelle 6.1: Graph- und Featuretypen für Fräsbauteile

ne Agenten erzeugt oder die Aufgaben auf mehrere Agenten mit den gleichen Fähigkeiten verteilt werden, um die Vorteile der asynchronen Programmierung zu nutzen. Im Falle der Bewertung der Herstellbarkeit von Fräsbauteilen werden die meisten Sichten durch einen jeweiligen Agenten repräsentiert. Die Ausnahme hiervon stellt der Modell-Agent dar, welcher die Daten-, Gestaltungs-, Fertigungs- und Visualisierung-Sicht in sich vereint. Die Darstellung der Agenten wird in Abbildung 6.6 als Klassendiagramm dargestellt. Hierbei sind zum einen die Schnittstellen der Agenten abgebildet und zum anderen die zugeordneten Aktionen. Für die Programmierung der Agenten wurde auf die Templates des Entwurfsbewertungsagenten zurückgegriffen. Die Python-Programmierung des Feature-Agenten, inklusive der zugehörigen ASL-Datei, ist in Anhang A als Beispiel für die Anwendung der Templates beigefügt. Die ASL-Datei enthält zudem die Cluster von Aktionen in Pläne, sodass diese bei auslösenden Ereignissen gemeinsam durchgeführt werden.

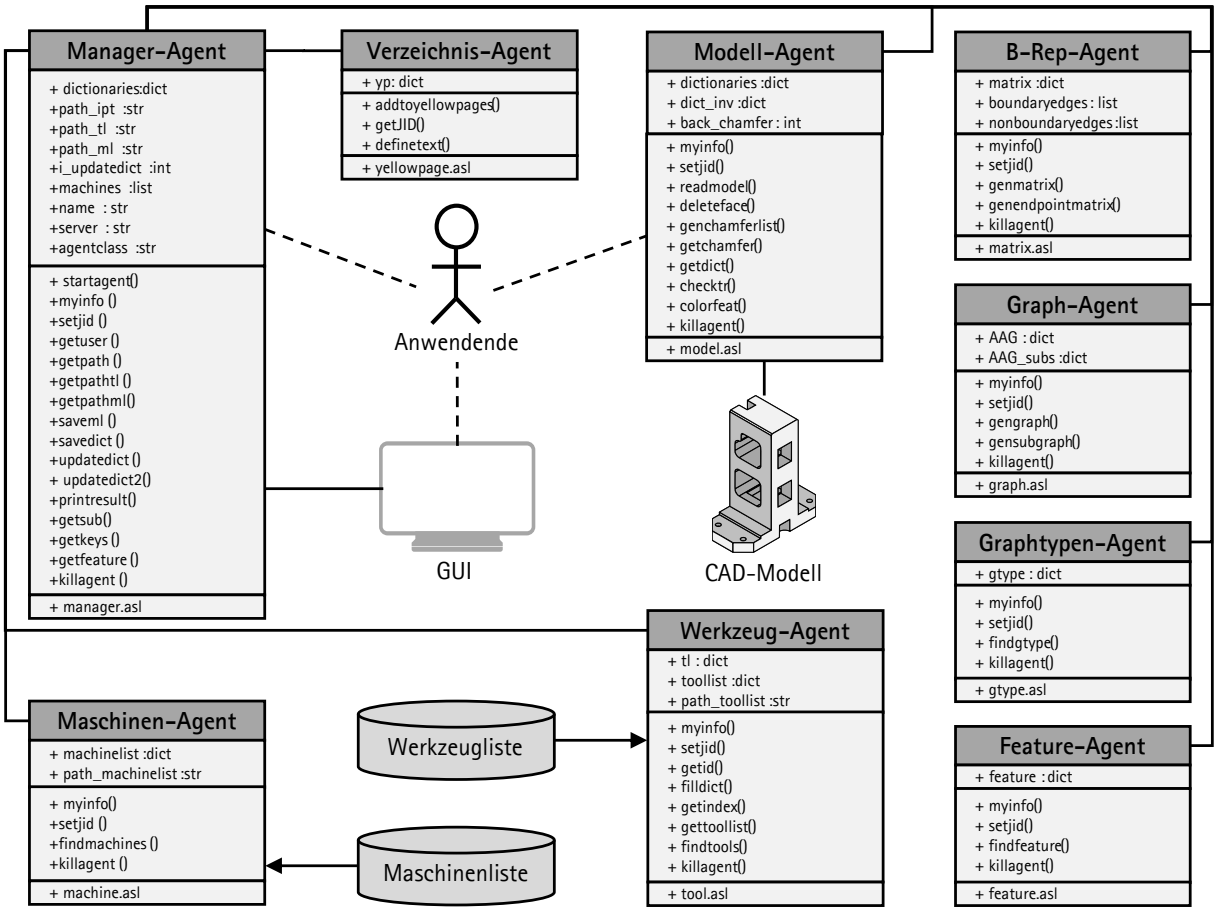


Abbildung 6.6: Klassendiagramm der Agenten für die Bewertung von Fräsentwürfen

Für die Verwaltung der Agenten wird einerseits, wie beschrieben, ein Verzeichnis genutzt, welches zur Laufzeit erzeugt wird, andererseits teilen die Agenten ihre Ergebnisse innerhalb eines Blackboard bzw. einer Featuretabelle, in welcher die ermittelten Features mit dem entsprechenden Graphtypen gesammelt und um weitere Informationen, wie z. B. dem gewählten Werkzeug oder den durchgeführten Entwurfsänderungen, ergänzt werden. Abschließend wird das Gesamtsystem aufgebaut und um ein GUI erweitert, in welcher die Bauteil-, Werkzeug- und Maschinenauswahl durchgeführt wird und die Ergebnisse in einer interaktiven Weise ausgegeben werden. Zudem wird die Schnittstelle zur anwendenden Person mittels Chat-Client eingerichtet.

6.4 Bewertung der Herstellbarkeit

Der vorgestellte Aufbau des MAS für die Entwurfsbewertung der Herstellbarkeit von Fräsbauteilen wird im Folgenden am Beispiel eines Aufspannwinkels präsentiert, um den Ablauf der Bewertung aus Sicht der Ingenieurin oder des Ingenieurs zu demonstrieren. In Abbildung 6.7 ist

der zu analysierende Aufspannwinkel mit den fehlenden Werkzeugausläufen und den unnötigen Fasen dargestellt. An die Aufspannwinkel können beidseits Wechselpaletten positioniert und befestigt werden, um ein rationelles Austauschen von Spannvorrichtungen zu ermöglichen. Zur besseren Übersicht wurden die Aufnahmegewinde für die Wechselplatten unterdrückt.

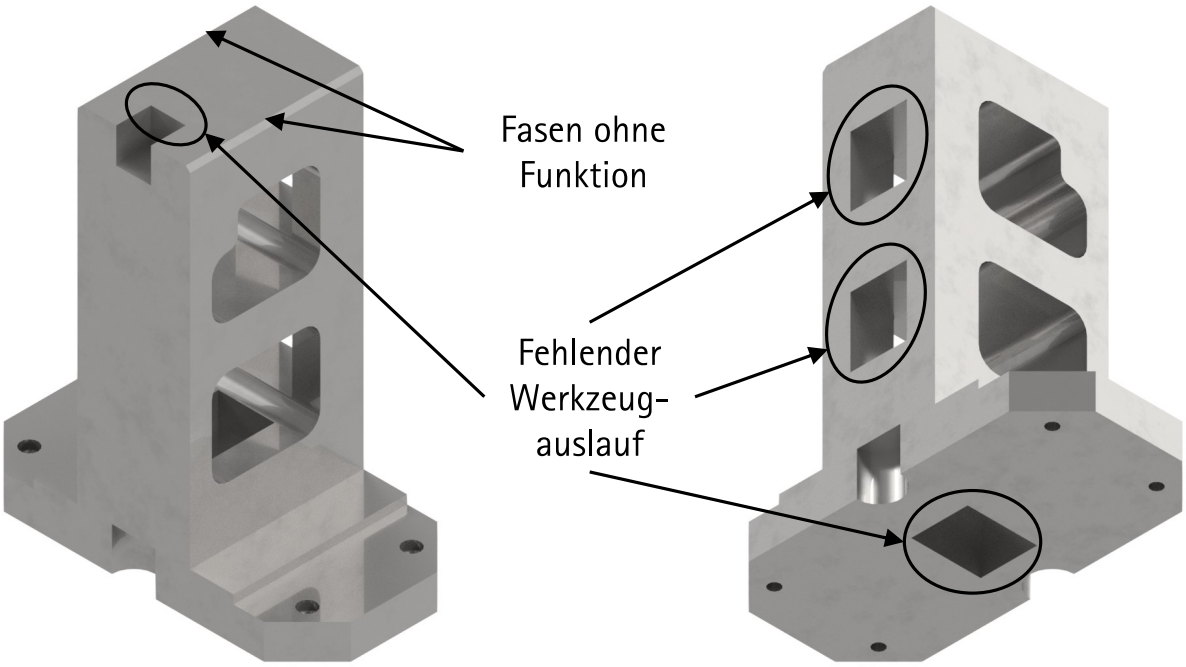


Abbildung 6.7: Aufspannwinkel als Anwendungsbeispiel für die Entwurfsbewertung

Nachdem das MAS von der anwendenden Person gestartet wurde, wird das GUI in Abbildung 6.8 geöffnet. In diesem GUI wählen die Anwendenden den zu überprüfenden Fräsentwurf als CAD-Modell sowie die zu verwendende Werkzeug- und Maschinenliste aus. Neben der Durchführung der kompletten Entwurfsbewertung besteht auch die Möglichkeit nur die Matrix aus Flächen und Kanten sowie den AAG auszugeben. Nach der Auswahl durch die anwendende Person kann das System mit dem Start-Button gestartet werden und der Manager-Agent wird aktiviert. Dieser koordiniert das weitere Vorgehen und fasst die Ergebnisse zusammen.

Der Manager-Agent startet zunächst den Verzeichnis-Agenten, welcher als Adressbuch fungiert. Bei diesem Agenten müssen sich alle Agenten nach dem Start mit ihrer JID und der zuständigen Rolle registrieren und anschließend beim Manager-Agenten als registriert und somit arbeitsbereit melden. Falls ein Agent im Laufe des Bewertungsprozesses mit einer bestimmten Rolle kommunizieren möchte, kann dieser den Verzeichnis-Agenten nach der entsprechenden JID befragen. Ist der Verzeichnis-Agent erfolgreich gestartet, ruft der Manager-Agent den Modell-Agenten auf. Dieser ist für das Auslesen und Manipulieren des CAD-Modells in Inventor zuständig. Hierbei greift der Modell-Agent einerseits auf die iProperties des Bauteils zu, um z. B. den Werkstoff, das Gewicht oder die konstruierende Person auszulesen. Ferner wird die

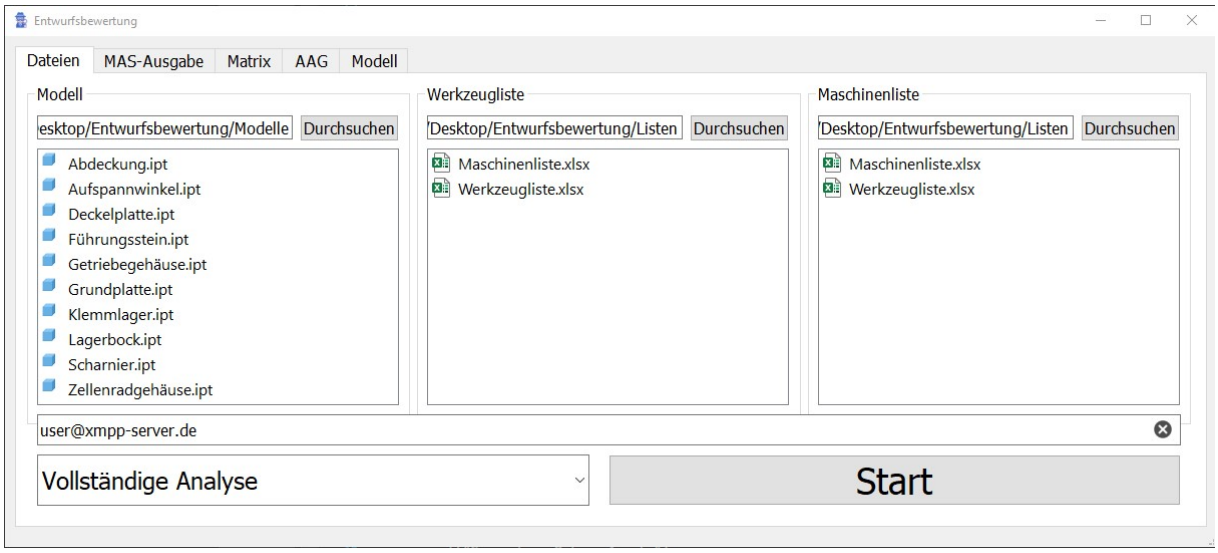


Abbildung 6.8: GUI

B-Rep-Struktur des CAD-Modells ausgelesen, indem die Flächen, Kanten und Schleifen (engl. loops) ermittelt und gesammelt in einem Dictionary abgelegt werden. Der Manager-Agent startet anschließend den Matrix-Agenten, welcher aus den Dictionaries eine Matrix erstellt und an den Manager-Agenten zurückleitet. Weiterhin wird der Maschinen-Agent gestartet, der die Maschinen in der Maschinenliste überprüft und die mögliche Bearbeitung mit den Maschinen bestätigt. Im Anschluss wird der Graph-Agent gestartet und diesem die Dictionaries sowie die Matrix übergeben. Der Graph-Agent bildet damit und mithilfe des Python-Pakets *NetWorkx 3.0* den AAG. Der AAG für den Aufspannwinkel ist in Abbildung 6.9 gezeigt. Hierbei repräsentieren die Knoten die Flächen und die Kanten zwischen den Knoten die Kanten des Modells. Zudem werden die Kantentypen durch unterschiedliche Farben repräsentiert. Nachdem der gesamte AAG erstellt wurde, wird er anschließend an den konvexen Außenkanten getrennt und in seine Subgraphen zerlegt. Die Subgraphen für den Aufspannwinkel sind in Anhang A beigefügt. Die ermittelten Subgraphen werden anschließend vom Graph-Agenten an den Manager-Agenten übergeben.

Der Manager-Agent startet dann so viele Graphtypen-Agenten, wie Subgraphen ermittelt wurden. Somit kann jeder Graphtypen-Agent einen Subgraph entschlüsseln und dem entsprechenden Graphtypen zuordnen. Das Ergebnis wird wieder an den Manager-Agenten gesendet. Dieser startet entsprechend viele Feature-Agenten, welche aus den Subgraphen, Graphtypen und den in Abbildung 6.1 gezeigten Einschränkungen die Features deuten. Die Ergebnisse werden dem Manager-Agenten übergeben und daraus ein erster Entwurf der Featuretabelle erstellt. Die ermittelte Featuretabelle wird anschließend dem Modell-Agenten übermittelt. Dieser stellt nun die Schnittstelle zu den Anwendenden dar. Um die Anwendenden direkt in den

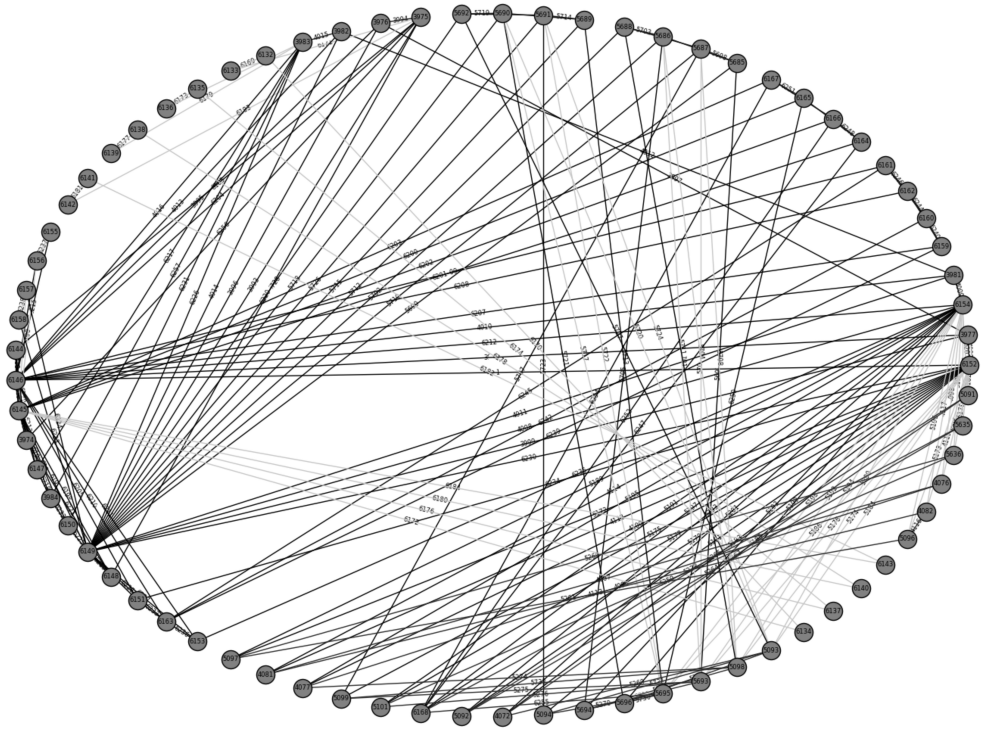


Abbildung 6.9: Zugeordneter Adjazenzgraph (AAG) des Aufspannwinkels

Entscheidungsprozess zu involvieren, können diese direkt in die Kommunikation zwischen den Agenten eingebunden werden, da der Nachrichtenaustausch über XMPP ebenfalls die Nutzung mit Chat-Clients ermöglicht. Hierdurch werden die Anwendenden als Teil der Problemlösung aufgefasst und können für Rückfragen oder weitere Informationen z. B. über die Funktion oder den Kontext des Entwurfs konsultiert werden. In Abbildung 6.10 wird die Interaktion mit der Anwenderin oder dem Anwender mittels XMPP-Nachrichten-Client Pidgin gezeigt, bei welcher der Modell-Agent die Anwenderin oder den Anwender befragt, ob die Fasen eine Funktion erfüllen oder lediglich aus optischen Gründen eingefügt wurden und somit für eine effizientere Fertigung vernachlässigt werden können. Da in der Abfrage gleichzeitig auf den jeweiligen TransientKey der Fasenfläche verwiesen wird und dieser nicht aus der 3D-Ansicht ersichtlich ist, wird ein Makro in Inventor bereitgestellt. Wurden die Features nur aus ästhetischen Gründen eingefügt, werden sie aus dem CAD-Modell entfernt und ein entsprechender Vermerk in der Featuretabelle eingefügt. Die aktualisierte Featuretabelle wird bei Verarbeitung der letzten Antwort der anwendenden Person an den Manager zurückgeschickt.

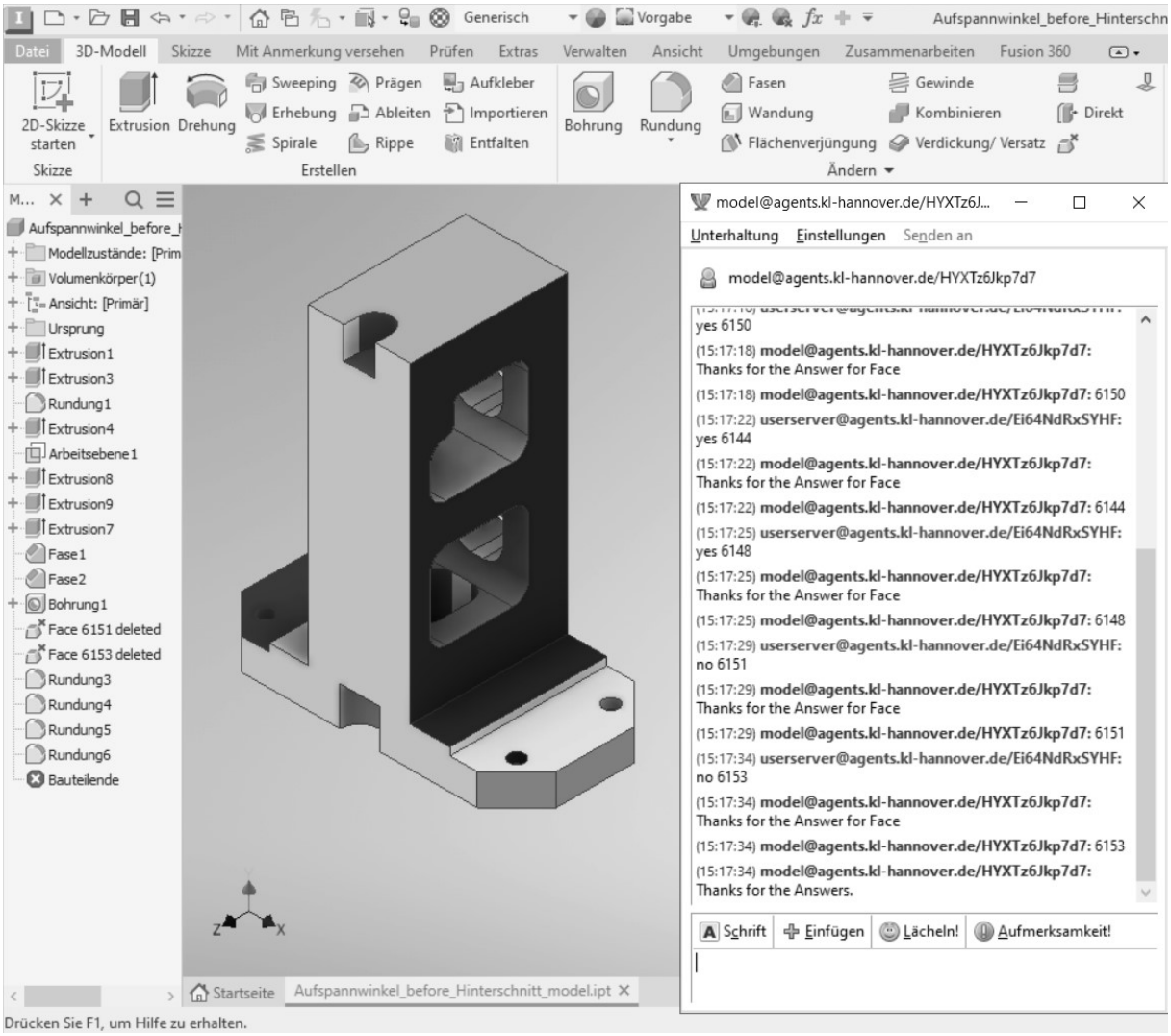


Abbildung 6.10: Interaktion mit Anwendenden und Einbettung in CAD-System

Nach der vollständigen Ermittlung der Fertigungsfeatures wird im nächsten Schritt jedes Feature an einen separaten Werkzeug-Agenten übergeben. Wenn das Feature zuvor als gelöscht markiert wurde, werden die Eingaben leer zurückgegeben. Ist dies nicht der Fall, wird aus den Informationen und der hinterlegten Werkzeugliste eine Auswahl der möglichen Werkzeuge generiert. Diese wird gleichzeitig auf das größte Werkzeug überprüft, um ein möglichst hohes Spanvolumen pro Zeit zu erreichen und somit den Spanprozess zu verkürzen. Dennoch werden die anderen Werkzeuge als Alternativen in der Featuretabelle mitgeführt. Die Änderungen werden dem Manager anschließend mitgeteilt. Anschließend wird dem Modell-Agenten die Featuretabelle übergeben, sodass dieser das Modell auf fehlende Werkzeugausläufe überprüft. Hierfür werden die Kanten in Werkzeugrichtung der Featuretypen wie z. B. Nut, Stufe, durchgehende, offene und geschlossene Tasche betrachtet. Ist der Radius in der Ecke gleich oder größer dem des Werkzeugs, ist eine Herstellung mittels 2,5D-Fräsprozess möglich. Handelt es sich bei der in Richtung des Werkzeugs liegenden Kante um eine konkave Kante, kann diese nicht mit einem zylindrischen Werkzeug gefertigt werden, sodass ein Werkzeugauslauf in Form eines Radius eingefügt werden muss. Daher wird an dieser Kante ein Radius mit der Größe des ausgewählten Werkzeugs eingefügt. Das Hinzufügen der Werkzeugausläufe wird in der Featuretabelle dokumentiert und die vollständige Featuretabelle wieder dem Manager-Agenten übergeben. Im vorletzten Schritt wird der Modell-Agent damit beauftragt, den einzelnen Features eine Farbe zuzuordnen und die Flächen des CAD-Modells dementsprechend einzufärben. Dies erleichtert das Nachvollziehen der Ergebnisse durch die Anwendenden, da jedes Feature mit einem eigenen RGB-Code versehen wird, welcher ebenfalls in der Featuretabelle hinterlegt wird. Zuletzt stellt der Manager-Agent alle Ergebnisse zusammen und gibt diese in einer übersichtlichen Featuretabelle aus (Tabelle 6.2). Zur besseren Übersichtlichkeit wurden hierbei die Zuordnung des Graphtypen, die Abmessungen des Features, die benachbarten Flächen, die alternativen Werkzeuge sowie die RGB-Zuweisung ausgeblendet. Hierbei ist ebenfalls erkennbar, dass das System die fehlenden Werkzeugausläufe automatisiert erkannt und die unnötigen Fasen in Abstimmung mit der anwendenden Person, aus dem CAD-Modell entfernt hat. Des Weiteren stimmen die gewählten Werkzeuge mit der möglichen Bearbeitung durch den Fräsprozess überein. Zudem wird die gesamte Kommunikation zwischen den Agenten protokolliert. Das Protokoll für die Bewertung des Aufspannwinkels ist in Anhang A beigefügt.

Nr.	Feature	Haupt-fläche	Flächen	Achsen	Kanten	Werkzeug	Änderungen
0	Stufe	6152	[3977]	X=1; Y=0; Z=0	[4000]	Mk 50	
1	Stufe	6154	[3981]	X=-1; Y=0; Z=0	[4009]	Mk 50	
2	Offene Tasche	6162	[6160, 6161, 6159, 6499]	X=0; Y=1; Z=0	[6240, 6243, 6244, 6246, 6241]	Mk 40	Rundung3 hinzugefügt für fehlenden Werkzeugauslauf
3	Durchgehende Tasche	-	[6164, 6165, 6166, 6167, 6507, 6509, 6512, 6510]	X=0; Y=-1; Z=0	[6248, 6249, 6251, 6253]	Kein Werkzeug	Rundung4 hinzugefügt für fehlenden Werkzeugauslauf
4	Durchgehende Tasche	-	[5688, 5685, 5686, 5687, 6539, 6541, 6544, 6542]	X=0; Y=0; Z=1	[5709, 5703, 5698, 5700]	Mk 50	Rundung5 hinzugefügt für fehlenden Werkzeugauslauf
5	Durchgehende Tasche	-	[5689, 5690, 5691, 5692, 6576, 6578, 6581, 6579]	X=0; Y=0; Z=-1	[5714, 5716, 5719, 5725]	Mk 50	Rundung6 hinzugefügt für fehlenden Werkzeugauslauf
6	Stufe	3976	[3975]	X=1; Y=0; Z=0	[3994]	Shank_F_25	
7	Stufe	3982	[3983]	X=-1; Y=0; Z=0	[4015]	Shank_F_25	
8	Sackbohrung	6132	[6133]	X=0; Y=-1; Z=0	[6169]	Shank_F_15	
9	Sackbohrung	6135	[6136]	X=0; Y=-1; Z=0	[6173]	Shank_F_15	
10	Sackbohrung	6138	[6139]	X=0; Y=-1; Z=0	[6177]	Shank_F_15	
11	Sackbohrung	6141	[6142]	X=0; Y=-1; Z=0	[6181]	Shank_F_15	
12	Sackbohrung	6155	[6156]	X=0; Y=-1; Z=0	[6237]	Mk 40	
13	Sackbohrung	6158	[6157]	X=0; Y=-1; Z=0	[6238]	Mk 40	
14	Fase	6144	[]	X=0,7; Y=0; Z=0,7	[]	Mk 50	
15	Fläche	6146	[]	X=0; Y=0; Z=1	[]	Mk 50	
16	Fläche	6145	[]	X=0; Y=-1; Z=0	[]	Mk 50	
17	Fläche	3974	[]	X=1; Y=0; Z=0	[]	Mk 50	
18	Fase	6147	[]	X=-0,7; Y=0; Z=0,7	[]	Mk 50	
19	Fläche	3984	[]	X=-1; Y=0; Z=0	[]	Mk 50	
20	Fase	6150	[]	X=-0,7; Y=0; Z=-0,7	[]	Mk 50	
21	Fläche	6149	[]	X=0; Y=0; Z=-1	[]	Mk 50	
22	Fase	6148	[]	X=0,7; Y=0; Z=-0,7	[]	Mk 50	
23	Fase	6151	[]	X=0,7; Y=0,7; Z=0	[]		Nach Abstimmung mit den Anwendenden gelöscht.
24	Fläche	6163	[]	X=0; Y=1; Z=0	[]	Mk 50	
25	Fase	6153	[]	X=-0,7; Y=0,7; Z=0	[]		Nach Abstimmung mit den Anwendenden gelöscht.
26	Durchgehende Tasche	-	[5636, 5096, 5097, 5098, 5099, 4076, 5101, 4077, 4081, 4082, 5693, 5694]	X=-0,99; Y=0; Z=0	[5260, 5259, 5261, 4116, 5274, 5732, 5273, 5275, 4097, 4099, 5276, 4113, 5729]	Kein Werkzeug	
27	Durchgehende Tasche	-	[5696, 5635, 5092, 5091, 5094, 5093, 4072, 6168, 5695]	X=-0,95; Y=0; Z=0	[5730, 5268, 5228, 5264, 5265, 5267, 5270, 5731, 5269, 6255]	Kein Werkzeug	
28	Bohrung	-	[6134]	X=0; Y=-1; Z=0	[]	Mk 10	
29	Bohrung	-	[6137]	X=0; Y=-1; Z=0	[]	Mk 10	
30	Bohrung	-	[6140]	X=0; Y=-1; Z=0	[]	Mk 10	
31	Bohrung	-	[6143]	X=0; Y=-1; Z=0	[]	Mk 10	

Tabelle 6.2: Featuretabelle des Aufspannwinkels

6.5 Validierung des Multi-Agentensystems

Nachdem in Kapitel 6.3 der Aufbau des MAS für die Entwurfsbewertung von Fräsbauteilen und in Kapitel 6.4 der Ablauf der Bewertung anhand des Aufspannwinkels als Anwendungsbeispiel gezeigt wurde, wird in diesem Kapitel das vorgestellte MAS mit zehn verschiedenen Fräsentwürfen validiert. Die Ergebnisse werden anhand von Bewertungskriterien gegenübergestellt. Hierbei unterscheiden sich die Fräsbauteile durch ihre Featureanzahl und -vielfalt sowie durch die Interaktion mit den Anwendenden und den durchgeführten Synthese-Operationen. Die zehn Fräsentwürfe werden in Abbildung 6.3 jeweils auf der linken Seite als zu bewertende CAD-Modelle gezeigt. Auf der rechten Seite die Ergebnisse der angepassten Modelle dargestellt. Für die Validierung wurde die Sichtweise der Konstruierenden eingenommen, bereits bestehende Fräsbauteile wurden auf die Featureerkennung überprüft und Fräsentwürfe wurden für die finale Fertigung angepasst.

Zur Auswertung der durchgeführten Entwurfsbewertungen wird ein *Benchmark*[3] der Fräsbauteile durchgeführt (Abbildung 6.4). Hierbei werden u. a. Kriterien zum Programmablauf wie die *User-Interaktion*, *Agentenanzahl* und die *Gesamtlaufzeit* der Bewertung definiert. Die *User-Interaktion* gibt wieder, ob außen liegende Fasen innerhalb des Modells vorhanden sind und hierzu eine Konsultation der anwendenden Person notwendig ist. Die *Agentenanzahl* setzt sich zum einen aus den festen Agenten, wie z. B. dem Manager-, Modell- oder Verzeichnis-Agenten zusammen und zum anderen aus der variablen Agentenanzahl, in Abhängigkeit der Featureanzahl. Für die *Gesamtlaufzeit* wurde lediglich die Rechenzeit des Computers angenommen, sodass die Antwortzeit der anwendenden Person keinen Einfluss hat und somit eine starke Verzerrung des Vergleichs vermieden wird. Für die Kriterien der Analyse werden die *Featureanzahl*, die *Featuretypen*, die Anzahl der *Graphentypen A-C* und der *Graphentypen D-I*, die *Werkzeuganzahl* sowie die *Fehlenden Werkzeuge* herangezogen. Die *Featureanzahl* spiegelt hierbei die Anzahl der ermittelten Fertigungsfeatures durch das MAS wider, welche sich von der Anzahl der Features im Featurebaum des CAD-Programms unterscheiden können. Um die Vielfalt der ermittelten Features darzustellen, werden diese im Kriterium der *Featuretypen* berücksichtigt. Die Anzahl der *Graphentypen A-C* und *D-I* spiegeln die Kompliziertheit der Features wider, da die Graphentypen A-C maximal aus zwei Flächen mit einer verbindenden Kante bestehen und die Graphentypen D-I u. a. beliebig viele Flächen aufweisen können. Die *Werkzeuganzahl* repräsentiert die unterschiedlichen Werkzeuge, welche für die Fertigung des Frästeils benötigt werden. Hierfür wird die Werkzeugtabelle, welche im Anhang beigefügt ist,

[3] Unter Benchmarking wird die vergleichende Analyse von Ergebnissen mit einem definierten Referenzwert verstanden.

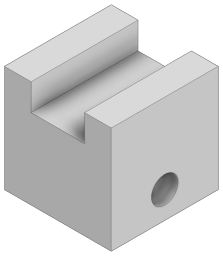
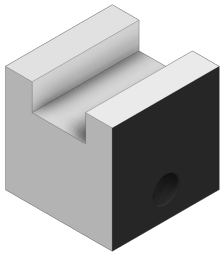
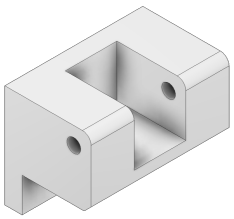
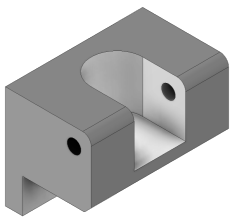
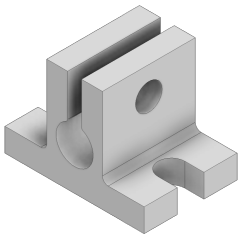
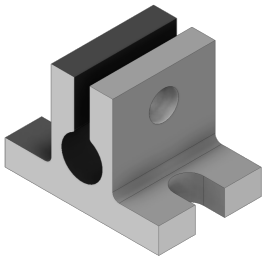
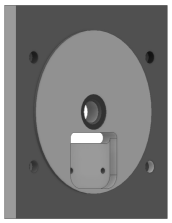
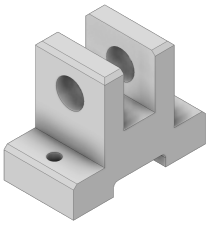
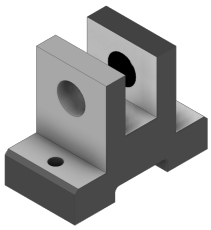
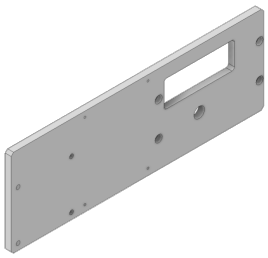
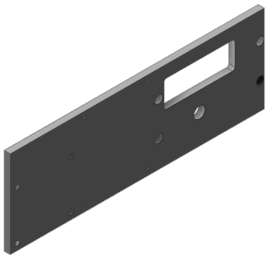
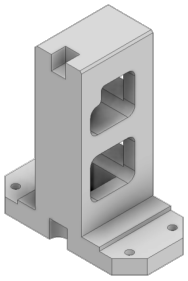
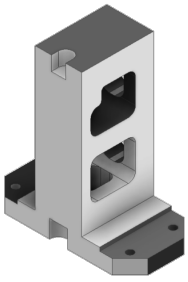
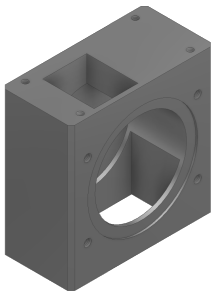
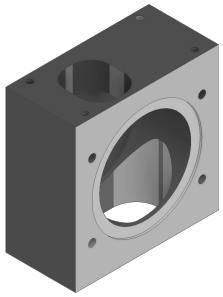
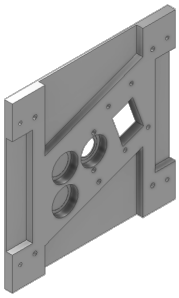
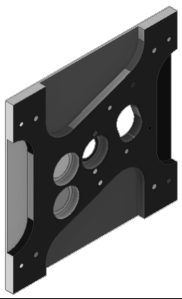
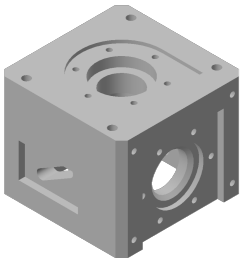
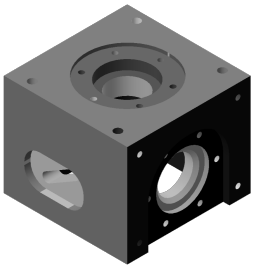
Führungsstein		Scharnier	
			
Klemmlager		**Abdeckung**	
			
Lagerbock		**Grundplatte**	
			
Aufspannwinkel		**Zellenradgehäuse**	
			
Deckelplatte		**Getriebegehäuse**	
			

Tabelle 6.3: Testbauteile

verwendet. Allerdings können nicht alle Features mit den bestehenden Werkzeugen gefertigt werden, sodass das Kriterium *Fehlende Werkzeuge* diesen Sachverhalt wiedergibt und auf eine Erweiterung der Werkzeugtabelle hinweist. Die Kriterien *Werkzeugausläufe* und *Optische Fasen* repräsentieren die Phase der Synthese. Das Kriterium *Werkzeugausläufe* gibt an, ob fehlende Werkzeugausläufe innerhalb des Bauteils vorhanden sind und somit durch das Hinzufügen von Radien behoben werden können. Für die *Optischen Fasen* wird dokumentiert, ob eine Konsultation mit der anwendenden Person stattgefunden hat, welche zum Löschen von Fasen ohne Funktion führte.

		Führungsstein	Scharnier	Klemmlager	Abdeckung	Lagerbock	Grundplatte	Aufspannwinkel	Zellenradgehäuse	Deckelplatte	Getriebegehäuse
Programm-ablauf	User-Interaktion	nein	ja	nein	nein	ja	ja	ja	ja	nein	ja
	Agentenanzahl	33	49	54	69	79	91	103	106	117	190
	Gesamtlaufzeit [s]	31	71	93	127	73	134	312	217	300	492
Analyse	Featureanzahl	9	14	16	21	24	28	32	33	37	61
	Featuretypen	3	7	4	5	6	5	7	7	5	9
	Graphtypen A-C	8	12	11	20	22	27	26	31	32	54
	Graphtypen D-I	1	2	5	1	2	1	6	2	5	7
	Werkzeuganzahl	3	5	3	4	4	5	5	6	7	8
	Fehlende Werkzeuge	nein	nein	ja	ja	nein	ja	ja	ja	ja	ja
Synthese	Werkzeugausläufe	nein	ja	nein	nein	nein	nein	ja	ja	ja	ja
	Optische Fasen	nein	nein	nein	nein	ja	ja	ja	ja	nein	ja

Tabelle 6.4: Benchmark

Anhand der Vergleichstabelle ist zu beobachten, dass die Featureanzahl und die Vielfalt der Featuretypen zwar tendenziell mit dem Umfang der Bauteile zunehmen, allerdings lässt sich daraus nicht ausschließlich auf die Gesamtlaufzeit des MAS schließen. Zusätzlich ist neben der Anzahl der Features auch die Anzahl der komplizierten Features zu berücksichtigen, wie z. B. beim Klemmlager oder dem Aufspannwinkel. Dieser Aspekt zeigt sich insbesondere bei der Abdeckung, welche zwar nur ein Feature mit dem Graphentypen D-I hat, aber dieses Feature aus neun Flächen und 17 Kanten besteht und somit eine aufwendigere Analyse durchzuführen ist.

Das vorgestellte MAS kann eine Entwurfsbewertung der Produktgestalt im Hinblick auf die Herstellbarkeit von CAD-Modellen unterstützen und in diesem Zuge eine Digitalisierung der Entwicklungsbewertung ermöglichen. Darüber hinaus kann das System zur Vereinfachung und Digitalisierung des Freigabeprozesses eingesetzt werden, indem die Reviewerin oder der Reviewer die Featuretabelle zur Überprüfung nutzen kann. Im Vergleich zur manuellen Prüfung kann dadurch Zeit gespart werden, da das System die Bearbeitungsschritte automatisch ausliest, die Herstellbarkeit durch die Werkzeuge überprüft und das CAD-Modell automatisch anpasst. Weiterhin ist eine Überprüfung des Entwurfs hinsichtlich der verfügbaren Werkzeugmaschinen möglich, indem das Teil mit verschiedenen hinterlegten Werkzeuglisten analysiert wird, sodass eine automatische Anpassung der Geometrie, bei einem kurzfristigen Wechsel auf eine andere Werkzeugmaschine, ohne großen Aufwand möglich ist [PLAP22a].

7 Schlussbetrachtung

7.1 Zusammenfassung

Um das Problem des mangelnden Wissensaustauschs, der fehlenden Kommunikation und der daraus resultierenden fehlenden ganzheitlichen Bewertung von Gestaltlösungen zwischen der Fertigungs- und der Entwicklungsabteilung, sowie steigende Komplexität, kürzere Markteinführungszeiten und mangelnde Dokumentation zu überwinden, wurde in dieser Arbeit untersucht, wie ein Assistenzsystem für die Entwurfsbewertung der Produktgestalt für Konstruierende in einer CAD-Entwicklungsumgebung ausgestaltet sein muss, um Entwurfsbewertungen zu operationalisieren und zu digitalisieren.

Nach den Forschungsansätzen DSRC und DSR, muss das zu entwickelnde Assistenzsystem für die Entwurfsbewertung den drei Teilbereichen Anwendungsdomäne, Wissensbasis und Konstruktionswissenschaft folgen. Dafür wurden zunächst der Stand der Technik und Wissenschaft zu wissensbasierten Informations- und Entwicklungssystemen im Kontext der KI erarbeitet. Hierzu wurde die Formalisierung und Verwaltung von Gestaltungswissen im Hinblick auf DfX beschrieben. Durch die Verwendung der Featuretechnologie bei der rechnergestützten Gestaltung ist es zudem möglich, das CAD-Modell in Form von Features wahrzunehmen und diese für die Anpassung des CAD-Modells oder der Fertigung zu nutzen. Als Ansatz zur dezentralen Schlussfolgerung wurden MAS eingeführt. Für die Entwicklung von MAS wurden agentenorientierte Methoden, Architekturen für die Interaktion zwischen den Agenten und Formen der Kommunikation und Kollaboration vorgestellt. Anschließend wurden verschiedene Werkzeuge und Software-Rahmenwerke zur Programmierung von MAS miteinander verglichen. Zudem wurden drei Forschungsrichtungen zu MAS in der Produktentwicklung identifiziert und die Verwendung des MAS als Assistenzsystem für Konstruierende anhand von fünf Ansätzen aus der Literatur beschrieben.

Für den Einsatz von MAS für die Entwurfsbewertung der Produktgestalt wurden daraufhin Herausforderungen abgeleitet und anschließend mit Spezifikationstechniken konkretisiert. Zur

Unterstützung bei der Entwicklung von MAS wurden bestehende Methoden analysiert und Forschungslücken für die Anwendung in der Domäne der Produktgestaltung hergeleitet. Darauf basierend wurde für die Problemanalyse ein qualitativer Vergleich der bestehenden Ansätze für MAS in der Produktentwicklung durchgeführt. Hierbei wurde festgestellt, dass die analytische Anwendung sich oft auf die Analyse von CAD-Modellen und die anschließende Empfehlung von Hinweisen und Vorschlägen beschränkt, wenngleich die synthetische Anwendung häufig KBS vorbehalten ist, welche eine Anpassung des CAD-Modells vornehmen, dafür aber in einem geschlossenen Lösungsraum operieren. Zudem findet eine direkte Integration der Konstruierenden in den Entscheidungsfindungsprozess und die Nutzung von Verhandlungsstrategien zwischen den Agenten kaum statt.

Für die einfache und breite Anwendung der Multi-Agententechnologie in der Domäne der Produktgestaltung ist ein Schlüsselfaktor die Definition eines Entwicklungsprozesses zur Entwurfsbewertung. Hierzu wurden Phasen der MaSE4D-Methode vorgestellt und ausgehend vom initialen Systemkontext bis zur finalen Implementierung und Programmierung als Modelle bereitgestellt, um eine strukturierte Vorgehensweise für die Entwicklung von MAS bei wissensintensiven und entwerfenden Gestaltungstätigkeiten zu liefern.

Die Frage nach einem generalisierten Aufbau einer Software-Architektur wurde durch eine Python-basierte Entwicklungsumgebung beantwortet, welche die grundlegenden Funktionalitäten, wie z. B. die Verarbeitung von Nachrichten, die Ausführung von Aktionen, die Nutzung von Informationen aus Datenbanken sowie die Koordination und Planung von Aufgaben, auf Basis validierter Ansätze, abbildet und diese durch die spezifizierten Anforderungen, wie z. B. Integration in den Arbeitslauf der Konstruierenden, Verhandlung von kollaborativen Entwurfsproblemen, Einbeziehung der Konstruierenden und Dokumentation des Entscheidungsprozesses, erweitert. Zudem sind die Entwurfsbewertungsagenten in der Lage, sowohl eine analytische Bewertung des Entwurfs im Hinblick auf die Herstellbarkeit, als auch eine synthetische Anwendung durch automatische Anpassung des CAD-Modells eines Bauteils vorzunehmen.

Das entwickelte methodische Vorgehen und die Anwendung des Entwurfsbewertungsagenten als Template wurde anhand eines Anwendungsbeispiels für den Aufbau eines MAS und die Bewertung der Herstellbarkeit eines Fräsbauteils detailliert beschrieben. Der vorgestellte Ansatz wurde zudem anhand zehn verschiedener Fräsentwürfe validiert und mittels Benchmarks dokumentiert.

7.2 Kritische Würdigung

Die Entwurfsbewertung der Produktgestalt mittels MAS soll an dieser Stelle abschließend diskutiert werden, bevor im Anschluss ein Ausblick auf einige Aspekte gegeben wird, die einen Ansatz für zukünftige Forschungsarbeiten bieten.

Bei der Implementierung des Assistenzsystems für die Entwurfsbewertung wurden die ermittelten Herausforderungen grundsätzlich gelöst. Die Kombination von KBS und MAS zur analytischen und synthetischen Anwendung unterscheidet den vorgestellten Ansatz von verwandter Literatur, in der lediglich Änderungsvorschläge formuliert werden oder eine Anpassung des CAD-Modells in geschlossenen Lösungsräumen erfolgt. Im Folgenden wird die Umsetzung der einzelnen Herausforderungen aus Kapitel 3.1 diskutiert:

- Das MAS ist in den Arbeitsablauf der Konstruierenden integriert, indem es mittels graphenbasierter Featureerkennung Informationen aus den CAD-Dateien ausliest, Anpassungen des CAD-Modells, z. B. durch Hinzufügen von Werkzeugausläufen, automatisiert durchführt, sodass für Fehler im Hinblick auf die Zugänglichkeit und Verfügbarkeit von Fertigungswerkzeugen und -maschinen eine Korrektur für einfache Dreh- und Frästeile erfolgt.
- Zudem wird das Wissen von verschiedenen Expertinnen und Experten aus funktionsübergreifenden Disziplinen, wie z. B. der Entwicklung, Fertigung und Arbeitsvorbereitung abgebildet, um die Nutzung von Gestaltungsrichtlinien, Normen, Normteilkatalogen und Werkzeugdatenbanken erweitert, sodass eine ganzheitliche Bewertung des Entwurfs im Hinblick auf DfX vorgenommen werden kann.
- Zur Unterstützung der Verhandlung bei kollaborativen Entwurfsproblemen werden Informationen mittels Nachrichten über XMPP und die Nutzung von Sprechakten ausgetauscht und z. B. durch die Nutzung von Vertragnetzprotokollen standardisiert. Zudem bietet die BDI-Architektur Einfluss auf die Überzeugungen und Ziele der Agenten, sodass Änderungen auf der Grundlage der ausgetauschten Informationen vorgenommen werden können, um Konflikte zu beheben und Kompromisse zu finden.
- Die Nachvollziehbarkeit und Akzeptanz von Entscheidungen wird durch die Nutzung von zentralen Wissensaustauschplattformen, wie z. B. Blackboards sowie durch die Dokumentation des Nachrichtenaustauschs und der Änderungen von Überzeugungen und Zielen erhöht.

- Die Entwicklung des Templates für den Entwurfsbewertungsagenten in der Programmiersprache Python und die Nutzung des SPADE-Rahmenwerks ermöglicht einen modularen und dezentralen Aufbau, bei welchem das Template als eine Art *Container* fungiert, indem es standardisierte Schnittstellen enthält und durch verschiedene Python-Bibliotheken, ASL-Dateien und Schlussfolgerungsalgorithmen ausgestattet werden kann.
- Des Weiteren integriert das MAS die Konstruierenden direkt in den Bewertungsprozess, indem sie mittels Chat-Client kontaktiert werden und in diesem Zuge weiteres Wissen z. B. über den Anwendungskontext zur Verfügung stellen können.

Allerdings ist der aktuelle Stand des MAS nicht frei von Limitationen. Im Hinblick auf die Integration in den Arbeitsablauf der Konstruierenden werden lediglich native Inventor-Dateien analysiert und automatisiert angepasst. Freiformflächen können zwar mittels Featureerkennung detektiert werden, sie werden jedoch nicht für die Analyse weiter in Betracht gezogen, da u. a. die Herstellbarkeitsbewertung auf den 2,5 D-Fräsprozess beschränkt ist. Zudem können keine zusammenhängenden Fertigungsfeatures erkannt werden, da hierzu neben der Zugänglichkeit der Werkzeuge auch die Bearbeitungsstrategie Einfluss hat. Des Weiteren erfolgt in der aktuellen Ausbaustufe keine Betrachtung der Nutzungs- oder Recyclingphase des Produktlebenszyklus und das Wissen muss in expliziter Form vorliegen. Im Hinblick auf die Verhandlung bei kollaborativen Entwurfsproblemen müssen diese vorab durch Optimierungsziele oder Prioritäten festgelegt werden. Da die Agenten nur Veränderungen vornehmen, wenn eine Inkonsistenz oder ein Konflikt auftritt, kann es zudem zu lokalen Optima kommen. Die Dokumentation stellt hauptsächlich die Entscheidung des MAS dar, aber nur ansatzweise die Vorgehensweise bzw. die Erkenntnisse des Schlussfolgerungsprozesses der einzelnen Agenten. Darüber hinaus ist das Protokoll des Nachrichtenaustauschs aufgrund der asynchronen Kommunikation durch eine unübersichtliche Darstellungsweise gekennzeichnet. Die Interaktion mit den Konstruierenden erfolgt aufgrund der im Vorfeld definierten Fragen, sodass bereits bei der Programmierung die zusätzlich benötigten Kontextinformationen berücksichtigt werden müssen. Hierzu werden entweder Ja-Nein-Fragen oder Fragen mit einer vorgegebenen Antwortauswahl genutzt.

Die Arbeit soll auch ein Plädoyer dafür sein, dass Multi-Agentensysteme vermehrt in der Produktentwicklung eingesetzt werden, um die Konstruierenden aktiv bei der Entwurfserstellung und Entscheidungsfindung zu unterstützen. Insbesondere unter dem Aspekt des demografischen Wandels und Fachkräftemangels muss auch in den Entwicklungsabteilungen ein höherer Grad an Automatisierung angestrebt werden. Zum einen betrifft dies Routinetätigkeiten, welche immer in gleicher Weise durchgeführt werden und zum anderen Kontrollaktivitäten, welche eine Vermeidung von Fehlern bzw. eine Sicherstellung der Konformität und Herstellbarkeit der

Bauteile zum Ziel haben. Werden diese Aufgaben, wie z. B. Entwurfsbewertungen, manuell durchgeführt, verringern sie zwar das Auftreten von Fehlern in nachgelagerten Prozessen, erhöhen aber gleichzeitig den Kapazitätsbedarf in der Entwicklung, sodass sie meist erst zu einem späten Zeitpunkt im Produktentwicklungsprozess durchgeführt werden. Zu diesem Zeitpunkt ist die Bearbeitung der Produktgestaltung möglicherweise aber schon so weit vorangeschritten, dass Änderungen einen großen Einfluss auf den gesamten Produktentwurf haben und die angepasste Konstruktion wiederum auf Konsistenz geprüft werden muss.

7.3 Weitere Forschungsfragen

Die Erweiterung des MAS bietet Potenzial für zukünftige Forschungsarbeiten, in dem das hinterlegte spezifische Domänenwissen für weitere Fertigungsverfahren ausgedehnt wird, z. B. für zerspanende Verfahren mit mehreren Achsen, Gießverfahren oder die Additive Fertigung. Zudem wäre denkbar, eine Schnittstelle zu PLM-Systemen zur Verfügung zu stellen, um hiermit auf die Daten des gesamten Produktentwicklungsprozesses zugreifen zu können und neue Erkenntnisse wiederum abzulegen. Eine Erweiterung der Analyse- und Synthesefähigkeiten des Entwurfsbewertungsagenten könnte zum einen für die Bewertung von zusammengesetzten Features und zum anderen für die Prüfung von Angaben der modellbasierten Definition, wie z. B. Oberflächen- und Härteangaben sowie Form- und Lagetoleranzen, durchgeführt werden. In diesem Zuge kann eine Verwendung von Open Source CAD-Programmen geprüft werden. Des Weiteren ist eine Anwendung des MAS zur automatisierten Revision von CAD-Modellen möglich, sodass z. B. Maßnahmen zur Standardisierung der eingesetzten Fertigungswerkzeuge oder zur Reduktion des Teilebestands durch Agenten geprüft und direkt im CAD-Modell eingepflegt werden können. Hierdurch können kleine und mittlere Unternehmen im Maschinen- und Anlagenbau, welche häufig Produkte mit einer Losgröße 1 produzieren, profitieren, insbesondere wenn sie bestehende CAD-Modelle an die neuen Kundenanforderungen anpassen und hierfür keine Konstruktionsautomatisierung implementiert haben. Zudem wäre es denkbar, dass fehlende Werkzeuge für die Bearbeitung automatisch nachbestellt werden.

Die vorgestellte Methode für den Aufbau von MAS und die generalisierte Software-Architektur können für die Analyse von Baugruppen erweitert werden. So wäre es denkbar, die Produktstruktur der Baugruppe als Vorrang-Graph, ähnlich zu dem AAG, abzubilden und hierüber die Montagereihenfolge bzw. Strukturierung der Baugruppe automatisiert vorzunehmen. Des Weiteren könnten automatisiert Gleichteilestrategien umgesetzt werden, da z. B. die Veränderung von Normteilen einen direkten Einfluss auf bspw. Dreh- und Frästeile hat, welche mit

dem in dieser Arbeit beschriebenen Ansatz automatisiert angepasst werden können. Da für die Anpassung von Baugruppen häufig Kontextinformationen, z. B. für den funktionalen Zusammenhang, benötigt werden, muss eine stärkere Interaktion mit den Anwendenden ermöglicht werden, welche zudem den Assistenzcharakter des Systems stärkt. Hierfür könnten die Agenten mit Fähigkeiten der Verarbeitung natürlicher Sprache (engl. natural language processing (NLP)) ausgestattet werden, um eine Konversation durchführen, welche nicht vorab beschrieben werden müsste. Zur besseren Nachvollziehbarkeit der Interaktion zwischen den Agenten könnte die Kommunikation in einem Graphen dargestellt werden.

Da die betrachteten Gestaltprozesse sehr wissensintensiv sind, scheint die Untersuchung von Lernalgorithmen im Hinblick auf „intelligente“ Agenten vielversprechend. Bei diesen werden mittels datenbasierten KI-Algorithmen Klassifizierungen von Analyseschritten automatisiert durchgeführt. Des Weiteren wäre die Erweiterung um Konfliktlösungsmechanismen denkbar, welche auf Erfahrungen in einer Fallbasis zurückgreifen, um dieses Wissen auf den neuen Konflikt anwenden. Darüber hinaus können Schlussfolgerungen bei Unsicherheit mithilfe von Bayes'schen Entscheidungsnetzen gezogen werden, bei denen die Wissensrepräsentation durch bedingte Wahrscheinlichkeiten gegeben ist. Hierbei könnten die Agenten auch Sensitivitätsanalysen durchführen, um den Einfluss von Gestaltänderungen, z. B. auf die Herstellbarkeit oder die Nutzungsphase des Produkts, bewerten zu können. Zudem wäre die Abbildung einer multikriteriellen Optimierung durch die Agenten denkbar, welche mittels Kommunikation und Kollaboration zwischen den Agenten aufgelöst wird.

Literaturverzeichnis

[Aamo94] Aamodt, Agnar; Plaza, Enric: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. In: *AI Communications* 7 (1994), S. 39–59. – ISSN 09217126.

[Adwe20] Adwernat, Stefan; Wolf, Mario; Gerhard, Detlef: Optimizing the Design Review Process for Cyber-Physical Systems using Virtual Reality. In: *Procedia CIRP* 91 (2020), S. 710–715. – ISSN 22128271.

[AlGe13] AlGeddawy, Tarek; ElMaraghy, Hoda: Reactive design methodology for product family platforms, modularity and parts integration. In: *CIRP Journal of Manufacturing Science and Technology* 6 (2013), 1, S. 34–43. – ISSN 17555817.

[Alze20] Alzetta, Francesco; Giorgini, Paolo; Najjar, Amro; Schumacher, Michael I.; Calvaresi, Davide: *In-Time Explainability in Multi-Agent Systems: Challenges, Opportunities, and Roadmap*. Bd. 12175 LNAI. Springer, 2020, S. 39–53.

[Andr07] de Andrade, Luiz Fernando S.; Forcellini, Fernando A.: *Interface design of a product as a potential agent for a concurrent engineering environment*. Bd. 51. Springer London, 2007, S. 503–510.

[Baol09] Baolu, Gao; Shibo, Xiong; Meili, Cao: Research and Application of a Product Cooperative Design System Based on Multi-Agent. In: *2009 Third International Symposium on Intelligent Information Technology Application* Bd. 2, IEEE, 2009. – ISBN 978–0–7695–3859–4, S. 198–201.

[Bart05] Bartak, Roman: Constraint Propagation and Backtracking-Based Search. In: *Summer School in Constraint Programming* (2005), S. 43.

[Bart10] Barták, Roman; Salido, Miguel A.; Rossi, Francesca: Constraint satisfaction techniques in planning and scheduling. In: *Journal of Intelligent Manufacturing* 21 (2010), 2, S. 5–15. – ISSN 0956–5515.

[Bave16] Bavendiek, Ann-Kathrin; Inkermann, David; Vietor, Thomas: Supporting Collaborative Design by Digital Tools-Potentials and Challenges. 2016. – Forschungsbericht.

[Beie19] Beierle, Christoph; Kern-Isberner, Gabriele: *Methoden wissensbasierter Systeme*. Bd. 6. Springer Fachmedien Wiesbaden, 2019. – ISBN 978–3–658–27083–4.

[Bell01] Bellifemine, Fabio; Poggi, Agostino; Rimassa, Giovanni: *Developing Multi-agent Systems with JADE*. Springer Berlin Heidelberg, 2001, S. 89–103.

[Bell07] Bellifemine, Fabio; Caire, Giovanni; Greenwood, Dominic: *Developing Multi-Agent Systems with JADE*. Wiley, 2007. – 1–286 S. – ISBN 9780470057476.

[Bend15] Bender, Janek; Kehl, Stefan; Müller, Jörg P.: *A Comparison of Agent-Based Coordination Architecture Variants for Automotive Product Change Management*. Springer International Publishing, 2015, S. 249–267.

[Bend21] Bender, Beate (Hrsg.); Gericke, Kilian (Hrsg.): *Pahl/Beitz Konstruktionslehre*. Springer Berlin Heidelberg, 2021. – ISBN 978–3–662–57302–0.

[Berg03] Bergmann, Ralph; Althoff, Klaus-Dieter; Breen, Sean; Göker, Mehmet; Manago, Michel; Traphöner, Ralph; Wess, Stefan: *Developing Industrial Case-Based Reasoning Applications*. Bd. 1612. Springer Berlin Heidelberg, 2003. – ISBN 978–3–540–20737–5.

[Berg20] Bergenti, Federico; Caire, Giovanni; Monica, Stefania; Poggi, Agostino: The first twenty years of agent-based software development with JADE. In: *Autonomous Agents and Multi-Agent Systems* 34 (2020), 10, S. 36. – ISSN 1387–2532.

[Berr16] Berriche, Fatima Z.; Zeddini, Besma; Kadima, Hubert; Riviere, Alain: *Closed-Loop Product Lifecycle Management Based on a Multi-agent System for Decision Making in Collaborative Design*. John wiley & sons, 2016, S. 540–551.

[Bord07] Bordini, Rafael H.; Hübner, Jomi F.; Wooldridge, Michael: *Programming Multi-Agent Systems in AgentSpeak usingJason*. John Wiley & Sons, 2007. – 1–273 S. – ISBN 9780470061848.

[Brim00] Brimble, Richard; Sellini, Florence: *The MOKA Modelling Language*. 2000, S. 49–56.

[Broo87] Brooks, R.: A hardware retargetable distributed layered architecture for mobile robot control. In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation* Bd. 4, Institute of Electrical and Electronics Engineers, 1987, S. 106–110.

[Buss04] Bussmann, Stefan; Jennings, Nicholas R.; Wooldridge, Michael: *Multi-agent Systems for Manufacturing Control*. Springer Berlin Heidelberg, 2004. – ISBN 978–3–642–05890–5.

[Camp99] Campbell, Matthew I.; Cagan, Jonathan; Kotovsky, Kenneth: A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment. In: *Research in Engineering Design* 11 (1999), 10, S. 172–192. – ISSN 0934–9839.

[Cass20] Cass, Stephen: The top programming languages: Our latest rankings put Python on top-again - [Careers]. In: *IEEE Spectrum* 57 (2020), 8, S. 22–22. – ISSN 0018–9235.

[Cern04] Cernuzzi, Luca; Juan, Thomas; Sterling, Leon; Zambonelli, Franco: *The Gaia Methodology*. Kluwer Academic Publishers, 2004, S. 69–88.

[Chap01] Chapman, Craig B.; Pinfold, Martyn: The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure. In: *Advances in Engineering Software* 32 (2001), 12, S. 903–912. – ISSN 09659978.

[Chir06] Chira, O.; Chira, C.; Roche, T.; Tormey, D.; Brennan, A.: An agent-based approach to knowledge management in distributed design. In: *Journal of Intelligent Manufacturing* 17 (2006), 12, S. 737–750. – ISSN 0956–5515.

[Chou15] Choulier, Denis; Fougères, Alain-Jérôme; Ostrosi, Egon: Developing multiagent systems for design activity analysis. In: *Computer-Aided Design* 59 (2015), 2, S. 201–213. – ISSN 00104485.

[Chu09] Chu, Chih-Hsing; Wu, Ping-Han; Hsu, Yu-Chiung: Multi-agent collaborative 3D design with geometric model at different levels of detail. In: *Robotics and Computer-Integrated Manufacturing* 25 (2009), 4, S. 334–347. – ISSN 07365845.

[Dai02] Dai, Kaiyu; Wang, Yinglin; Zhang, Shensheng; Zhang, Jingyi: Research on DFx evaluation system for distributed design based on multi-agent. In: *The 2nd International Workshop on Autonomous Decentralized System, 2002.*, IEEE Comput. Soc, 2002. – ISBN 0–7803–7624–2, S. 200–204.

[Dane01] Danesh, Mohammad R.; Jin, Yan: An Agent-Based Decision Network for Concurrent Engineering Design. In: *Concurrent Engineering* 9 (2001), 3, S. 37–47. – ISSN 1063–293X.

[Davi83] Davis, Randall; Smith, Reid G.: Negotiation as a metaphor for distributed problem solving. In: *Artificial Intelligence* 20 (1983), 1, S. 63–109. – ISSN 00043702.

[Davi93] Davis, J.; Kannapan, S.: AgentX: an environment for coordinating distributed problem solving in product development. In: *[1993] Proceedings Second Workshop on Enabling Technologies - Infrastructure for Collaborative Enterprises*, IEEE Comput. Soc. Press, 1993. – ISBN 0–8186–4082–0, S. 99–103.

[Deen05] Deen, S. M.: An Engineering Approach to Cooperating Agents for Distributed Information Systems. In: *Journal of Intelligent Information Systems* 25 (2005), 7, S. 5–45. – ISSN 0925–9902.

[Delo01] Deloach, Scott A.; Wood, Mark F.; Sparkman, Clint H.: Multiagent Systems Engineering. In: *International Journal of Software Engineering and Knowledge Engineering* 11 (2001), 6, S. 231–258. – ISSN 0218–1940.

[DeLo05] DeLoach, Scott A.: Multiagent systems engineering of organization-based multiagent systems. In: *Proceedings of the fourth international workshop on Software engineering for large-scale multi-agent systems - SELMAS '05*, ACM Press, 2005. – ISBN 1595931163, S. 1.

[Diak20] Diakun, J.; Dostatni, E.: End-of-life design aid in PLM environment using agent technology. In: *Bulletin of the Polish Academy of Sciences: Technical Sciences* 68 (2020), S. 207–214. – ISSN 23001917.

[Doel20] Doellken, M.; Zimmerer, C.; Matthiesen, S.: Challenges faced by design engineers when considering manufacturing in design - an interview study. In: *Proceedings of the Design Society: DESIGN Conference* 1 (2020), 5, S. 837–846. – ISSN 2633–7762.

[Dorr18] Dorri, Ali; Kanhere, Salil S.; Jurdak, Raja: Multi-Agent Systems: A Survey. In: *IEEE Access* 6 (2018), S. 28573–28593. – ISSN 2169–3536.

[Dost16] Dostatni, Ewa; Diakun, Jacek; Grajewski, Damian; Wichniarek, Radosław; Karwasz, Anna: Multi-agent system to support decision-making process in design for recycling. In: *Soft Computing* 20 (2016), 11, S. 4347–4361. – ISSN 1432–7643.

[Dueh20] Duehr, Katharina; Nix, Bernhard; Albers, Albert: Charakterisierung der Entwicklungsaufgabe zur methodischen Potentialfindung in der standortverteilten Produktentwicklung. 2020. – Forschungsbericht.

[Ecke05] Eckert, Claudia; Maier, Anja; McMahon, Chris: *Communication in design*. Springer, 2005, S. 232–261.

[Eign14] Eigner, Martin (Hrsg.); Roubanov, Daniil (Hrsg.); Zafirov, Radoslav (Hrsg.): *Modellbasierte virtuelle Produktentwicklung*. Springer Berlin Heidelberg, 2014. – ISBN 978–3–662–43815–2.

[El-M03] El-Mehalawi, Mohamed; Miller, R A.: A database system of mechanical components based on geometric and topological similarity. Part I: representation. In: *Computer-Aided Design* 35 (2003), 1, S. 83–94. – ISSN 00104485.

[Fang03] Fang, Weidong; Tang, Ming X.; Frazer, John H.: *Supporting Collaborative Product Design in an Agent Based Environment*. Bd. 2718. Springer Berlin Heidelberg, 2003, S. 447–460.

[Feng05] Feng, Shaw C.: Preliminary design and manufacturing planning integration using web-based intelligent agents. In: *Journal of Intelligent Manufacturing* 16 (2005), 10, S. 423–437. – ISSN 0956–5515.

[Fisc94] Fischer, Klaus; Müller, Jörg P.; Pischel, Markus: *A Testbed for the Development of DAI Applications*. 1994, S. 179–190.

[Foug18] Fougères, Alain-Jérôme; Ostrosi, Egon: Corrigendum to "Intelligent agents for feature modelling in computer aided design" [J. Comput. Des. Eng. (2018) 19–40]. In: *Journal of Computational Design and Engineering* 5 (2018), 7, S. 379–379. – ISSN 2288–5048.

[Fran76] Franke, Hans-Joachim: *Untersuchungen zur Algorithmisierbarkeit des Konstruktionsprozesses*. VDI-Verlag, 1976.

[Ganz07] Ganzha, Maria; Paprzycki, Marcin: Implementing rule-based automated price negotiation in an agent system. In: *Journal of Universal Computer Science* 13 (2007), S. 244–266.

[Gemb17] Gembarski, Paul C.; Li, Haibing; Lachmayer, Roland: *KBE-Modeling Techniques in Standard CAD-Systems: Case Study—Autodesk Inventor Professional*. 2017, S. 215–233.

[GEMB19] GEMBARSKI, Paul C.: *Komplexitätsmanagement mittels wissensbasiertem CAD*. TEWISS-Technik und Wissen GmbH, 2019.

[GEMB20] GEMBARSKI, Paul C.: *On the Conception of a Multi-agent Analysis and Optimization Tool for Mechanical Engineering Parts*. Bd. 186. John Wiley & Sons, 2020, S. 93–102.

[GEMB21] GEMBARSKI, Paul C.; PLAPPERT, Stefan; LACHMAYER, Roland: Making design decisions under uncertainties: probabilistic reasoning and robust product design. In: *Journal of Intelligent Information Systems* 57 (2021), 12, S. 563–581. – ISSN 0925–9902.

[GEOR92] GEORGEFF, Michael P.; RAO, A: An abstract architecture for rational agents. In: *Proc. of the Third International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1992, S. 439–449.

[GRAB97] GRABOWSKI, H; GEIGER, K: *Neue Wege zur Produktentwicklung (New Approaches to Product Development)*. 1997.

[GRUB95] GRUBER, Thomas R.: Toward principles for the design of ontologies used for knowledge sharing? In: *International Journal of Human-Computer Studies* 43 (1995), 11, S. 907–928. – ISSN 10715819.

[HABH09] HABHOUBA, Dounia; DESROCHERS, Alain; CHERKAOUI, Soumaya: Agent-based assistance for engineering change management: An implementation prototype. In: *2009 13th International Conference on Computer Supported Cooperative Work in Design*, IEEE, 2009. – ISBN 978–1–4244–3534–0, S. 288–293.

[HAO05] HAO, Qi; SHEN, Weiming; ZHANG, Zhan: An autonomous agent development environment for engineering applications. In: *Advanced Engineering Informatics* 19 (2005), 4, S. 123–134. – ISSN 14740346.

[HAO06] HAO, Qi; SHEN, Weiming; ZHANG, Zhan; PARK, Seong-Whan; LEE, Jai-Kyung: Agent-based collaborative product design engineering: An industrial case study. In: *Computers in Industry* 57 (2006), 1, S. 26–38. – ISSN 01663615.

[HEIN96] HEINRICH, M; JÜNGST, E W.: The Resource-Based Paradigm: Configuring Technical Systems from Modular Components. (1996).

[HEVN04] HEVNER; MARCH; PARK; RAM: Design Science in Information Systems Research. In: *MIS Quarterly* 28 (2004), S. 75. – ISSN 02767783.

[Hoog94] Hoog, Robert D.; Martil, Rob; Wielinga, Bob: The Common KADS model set / ESPRIT Project P5248 KADS-II. 1994. – Forschungsbericht.

[Hopg21] Hopgood, Adrian A.: *Intelligent Systems for Engineers and Scientists*. CRC Press, 11 2021. – ISBN 9781003226277.

[Hopp20a] Hoppe, Lukas V.; Gembarski, Paul C.; Plappert, Stefan; Lachmayer, Roland: Anwendung wissensbasierter Konstruktionssysteme zur Lösungsbewertung von CAD-Konstruktionen. In: *18. Gemeinsames Kolloquium Konstruktionstechnik 2020: Nachhaltige Produktentwicklung : KT 2020*, 2020, S. 199–208.

[Hopp20b] Hoppe, Lukas V.; Plappert, Stefan; Gembarski, Paul C.; Lachmayer, Roland: Development of an Intelligent Tutoring System for Design Education. In: *Proceedings of the 22nd International Conference on Engineering and Product Design Education*, The Design Society, 2020. – ISBN 978–1–912254–10–1.

[Horv20] Horvat, N.; Škec, S.; Martinec, T.; Lukačević, F.; Perišić, M. M.: Identifying the Effect of Reviewers' Expertise on Design Review using Virtual Reality and Desktop Interface. In: *Proceedings of the Design Society: DESIGN Conference* 1 (2020), 5, S. 187–196. – ISSN 2633–7762.

[Hubk84] Hubka, Vladimir: *Theorie Technischer Systeme: Grundlagen einer wissenschaftlichen Konstruktionslehre*. 1984.

[Huet07] Huet, Greg; Culley, Stephen J.; McMahon, Christopher A.; Fortin, Clément; Sellini, Florence: Communication, information and knowledge processes observed during engineering design reviews. In: *DS 42: Proceedings of ICED 2007, the 16th International Conference on Engineering Design, Paris, France, 28.-31.07. 2007*, 2007, S. 527–528.

[Huth20] Huth, Tobias; Vietor, Thomas: Systems Engineering in der Produktentwicklung: Verständnis, Theorie und Praxis aus ingenieurswissenschaftlicher Sicht. In: *Gruppe. Interaktion. Organisation. Zeitschrift für Angewandte Organisationspsychologie (GIO)* 51 (2020), 3, S. 125–130. – ISSN 2366–6145.

[Igle96] Iglesias, Carlos A.; Garijo, Mercedes; González, José C.; Velasco, Juan R.: A methodological proposal for multiagent systems development extending CommonKADS. In: *Proceedings of the 10th Banff knowledge acquisition for knowledge-based systems workshop* Bd. 1, 1996, S. 21–25.

[Jenn95] Jennings, NICHOLAS R.; Wooldridge, MICHAEL: Applying Agent Technology. In: *Applied Artificial Intelligence* 9 (1995), 7, S. 357–369. – ISSN 0883–9514.

[JIA04] JIA, H.Z.; ONG, S.K.; FUH, J.Y.H.; ZHANG, Y.F.; NEE, A.Y.C.: An adaptive and upgradable agent-based system for coordinated product development and manufacture. In: *Robotics and Computer-Integrated Manufacturing* 20 (2004), 4, S. 79–90. – ISSN 07365845.

[JIA09] JIA, Yubo; HUANG, Chengwei; CAI, Hao: A comparison of three agent-oriented software development methodologies: MaSE, Gaia, and Tropos. In: *2009 IEEE Youth Conference on Information, Computing and Telecommunication*, IEEE, 9 2009. – ISBN 978–1–4244–5074–9, S. 106–109.

[JIAN08] JIANJUN, Yi; BAIYANG, Ji; BIN, Yu; LEI, Du; JINXIANG, Dong: Research on the knowledge management architecture of LCED based ontologies and multi-agent system. In: *The International Journal of Advanced Manufacturing Technology* 37 (2008), 5, S. 605–612. – ISSN 0268–3768.

[JIAN10] JIAN, Guo; GAO, James; WANG, Yinglin: A multi-agent based knowledge search framework to support the product development process. In: *International Journal of Computer Integrated Manufacturing* 23 (2010), 3, S. 237–247. – ISSN 0951–192X.

[JOSH88] JOSHI, S.; CHANG, T.C.: Graph-based heuristics for recognition of machined features from a 3D solid model. In: *Computer-Aided Design* 20 (1988), 3, S. 58–66. – ISSN 00104485.

[JUAN02] JUAN, Thomas; PEARCE, Adrian; STERLING, Leon: ROADMAP. In: *Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 1 - AAMAS '02*, ACM Press, 2002. – ISBN 1581134800, S. 3.

[JUAN04] JUAN, Thomas; STERLING, Leon: *The ROADMAP Meta-model for Intelligent Adaptive Multi-agent Systems in Open Environments*. Bd. 2935. 2004, S. 53–68.

[KING01] KINGSTON, John: *Modelling Agents and Communication using CommonKADS*. Springer London, 2001, S. 301–319.

[KLOS90] KLOSE, Johannes: *Konstruktionsinformatik im Maschinenbau*. Verlag Technik, 1990.

[KRAH21] KRAHE, Carmen; KALAIDOV, Maksym; DOELLKEN, Markus; GWOSCH, Thomas; KUHNLE, Andreas; LANZA, Gisela; MATTHIESEN, Sven: AI-Based knowledge extraction for automatic design proposals using design-related patterns. In: *Procedia CIRP* 100 (2021), S. 397–402. – ISSN 22128271.

[KRAT11] KRATZER, Martin; RAUSCHER, Michael; BINZ, Hansgeorg; GOEHNER, Peter: An agent-based system for supporting design engineers in the embodiment design phase. In: *ICED 11 - 18th International Conference on Engineering Design - Impacting Society Through Engineering Design* 10 (2011), S. 178–189.

[KRAT14] KRATZER, Martin: *Anwendungsspezifische Entwicklung eines proaktiven Konstruktionssystems auf Basis von Softwareagenten*. 2014. – 259 S.

[LACH13] LACHMAYER, Roland; MOZGOVA, Iryna; SAUTHOFF, Bastian; GOTTWALD, Philipp: *Product Evolution and Optimization Based on Gentelligent Components and Product Life Cycle Data*. 2013, S. 685–694.

[LACH17] LACHMAYER, Roland; GEMBARSKI, Paul C.; GOTTWALD, Philipp; LIPPERT, R. B.: *The Potential of Product Customization Using Technologies of Additive Manufacturing*. 2017, S. 71–81.

[LAND97] LANDER, S.E.: Issues in multiagent design systems. In: *IEEE Expert* 12 (1997), 3, S. 18–26. – ISSN 0885–9000.

[LEND09] LENDERS, Michael: *Beschleunigung der Produktentwicklung durch Lösungsraum-Management*. Bd. 2009,1. Apprimus-Verl., 2009. – Zugl.: Aachen, Techn. Hochsch., Diss., 2009. – ISBN 978–3–940565–26–6.

[LI07] LI, Yan: Application of Multi-Agent for Collaborative Product Design Engineering. In: *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)* Bd. 2, IEEE, 7 2007. – ISBN 0–7695–2909–7, S. 450–454.

[LIU04] LIU, Hong; TANG, Mingxi; FRAZER, John H.: Supporting dynamic management in a multi-agent collaborative design system. In: *Advances in Engineering Software* 35 (2004), 8, S. 493–502. – ISSN 09659978.

[LIU08] LIU, Quan; CUI, Xingran; HU, Xiuyin: *An Agent-Based Intelligent CAD Platform for Collaborative Design*. Bd. 15. 2008, S. 501–508.

[LUPA09] LUPA, Norman: *Einsatz wissensbasierter Features für die automatische Konfiguration von Produktkomponenten*. Cuvillier Verlag, 2009.

[MADH05] MADHUSUDAN, Therani: An agent-based approach for coordinating product design workflows. In: *Computers in Industry* 56 (2005), 4, S. 235–259. – ISSN 01663615.

[Mahd10] Mahdjoub, Morad; Monticolo, Davy; Gomes, Samuel; Sagot, Jean-Claude: A collaborative Design for Usability approach supported by Virtual Reality and a Multi-Agent System embedded in a PLM environment. In: *Computer-Aided Design* 42 (2010), 5, S. 402–413. – Advanced and Emerging Virtual and Augmented Reality Technologies in Product Design. – ISSN 00104485.

[Mahe07] Mahesh, M.; Ong, S. K.; Nee, A. Y. C.: A web-based multi-agent system for distributed digital manufacturing. In: *International Journal of Computer Integrated Manufacturing* 20 (2007), 1, S. 11–27. – ISSN 0951–192X.

[Matt19] Matthiesen, Sven; Grauberger, Patric; Bremer, Frank; Nowoseltschenko, Konstantin: Product models in embodiment design: an investigation of challenges and opportunities. In: *SN Applied Sciences* 1 (2019), S. 1078. – ISSN 2523–3963.

[Meda06] Medani, O.; Ratchev, S.M.: A STEP AP224 agent-based early manufacturability assessment environment using XML. In: *The International Journal of Advanced Manufacturing Technology* 27 (2006), 2, S. 854–864. – ISSN 0268–3768.

[Mesc12] Mescheder, Bernhard; Sallach, Christian: *Wettbewerbsvorteile durch Wissen*. Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–27895–2.

[Milt08] Milton, Nick: *Knowledge Technologies*. Polimetrica, International Scientific Publisher, 2008. – ISBN 978–88–7699–100–4.

[Moon09] Moon, Seung K.; Simpson, Timothy W.; Kumara, Soundar R. T.: An agent-based recommender system for developing customized families of products. In: *Journal of Intelligent Manufacturing* 20 (2009), 12, S. 649–659. – ISSN 0956–5515.

[Mora03] Moraïtis, Pavlos; Petraki, Eleftheria; Spanoudakis, Nikolaos I.: *Engineering JADE Agents with the Gaia Methodology*. 2003, S. 77–91.

[Moue09] Mouelhi, Ouael; Couturier, Pierre; Redarce, Tanneguy: A hybrid search algorithm in a multi-agent system environment for multicriteria optimization of products design. In: *2009 International Joint Conference on Neural Networks*, IEEE, 6 2009. – ISBN 978–1–4244–3548–7, S. 2160–2167.

[Mön06] Mönch, Lars; Stehli, Marcel: ManufAg: a multi-agent-system framework for production control of complex manufacturing systems. In: *Information Systems and e-Business Management* 4 (2006), 4, S. 159–185. – ISSN 1617–9846.

[MÜL12] MÜLLER, Patrick; PASCH, Florian; DREWINSKI, Roland; BEDENBENDER, Heinz; HAYKA, Haygazun; STARK, Rainer: IFIP AICT 388 - Study on Collaborative Product Development and Digital Engineering Tools. 2012. – Forschungsbericht. – 389–399 S.

[NAHM06] NAHM, Y.-E.; ISHIKAWA, H.: An Internet-based integrated product design environment. Part II: its applications to concurrent engineering design. In: *The International Journal of Advanced Manufacturing Technology* 27 (2006), 1, S. 431–444. – ISSN 0268–3768.

[NIU12] NIU, Qinzhou; HU, Lei: Design of case-based hybrid agent structure for machine tools of intelligent design system. In: *2012 IEEE International Conference on Computer Science and Automation Engineering*, IEEE, 6 2012. – ISBN 978–1–4673–2008–5, S. 59–62.

[NWAN99a] NWANA, Hyacinth S.; NDUMU, Divine T.: A perspective on software agents research. In: *The Knowledge Engineering Review* 14 (1999), 9, S. S0269888999142012. – ISSN 02698889.

[NWAN99b] NWANA, Hyacinth S.; NDUMU, Divine T.; LEE, Lyndon C.; COLLIS, Jaron C.: Zeus: A toolkit for building distributed multiagent systems. In: *Applied Artificial Intelligence* 13 (1999), 1, S. 129–185. – ISSN 0883–9514.

[O'BR98] O'BRIEN, Patrick D.; NICOL, Richard C.: FIPA - towards a standard for software agents. In: *BT Technology Journal* 16 (1998), S. 51–59.

[OLSS20] OLSSON, Mikael: *Templates*. Apress, 2020, S. 173–187.

[OPRE04] OPREA, Mihaela: *Applications of Multi-Agent Systems*. Kluwer Academic Publishers, 2004, S. 239–270.

[ORSB09] ORSBORN, Seth; CAGAN, Jonathan: Multiagent Shape Grammar Implementation: Automatically Generating Form Concepts According to a Preference Function. In: *Journal of Mechanical Design* 131 (2009), 12, S. 1210071–12100710. – ISSN 1050–0472.

[ORTI04] ORTIZ-ARROYO, Daniel; LARSEN, Henrik L.: A Multi-Agent System Framework for Collaborative Decision Making Under Uncertainty. In: *Symposium on Challenges in the Internet and Interdisciplinary Research, International IPSI-2004*, IPSI, 2004. – null ; Conference date: 19-05-2010.

[Ostr12] Ostrosi, E.; Fougères, A.-J.; Ferney, M.; Klein, D.: A fuzzy configuration multi-agent approach for product family modelling in conceptual design. In: *Journal of Intelligent Manufacturing* 23 (2012), 12, S. 2565–2586. – ISSN 0956–5515.

[Padg04] Padgham, Lin; Winikoff, Michael: *Developing Intelligent Agent Systems*. Wiley, 6 2004. – ISBN 9780470861202.

[Pal20] Pal, Constantin-Valentin; Leon, Florin; Paprzycki, Marcin; Ganzha, Maria: A Review of Platforms for the Development of Agent Systems. (2020), 7, S. 1–40.

[Pala20] Palanca, Javier; Terrasa, Andres; Julian, Vicente; Carrascosa, Carlos: SPADE 3: Supporting the New Generation of Multi-Agent Systems. In: *IEEE Access* 8 (2020), S. 182537–182549. – ISSN 2169–3536.

[Pala22a] Palanca, J.; Rincon, J. A.; Julian, V.; Carrascosa, C.; Terrasa, A.: *IoT Artifacts: Incorporating Artifacts into the SPADE Platform*. Bd. 483. Springer Science and Business Media Deutschland GmbH, 2022, S. 69–79.

[Pala22b] Palanca, J.; Rincon, J.A; Carrascosa, C.; Julian, V.; Terrasa, A.: *A Flexible Agent Architecture in SPADE*. 2022, S. 320–331.

[Papa20] Papakostas, Nikolaos; Newell, Anthony; George, Abraham: An Agent-Based Decision Support Platform for Additive Manufacturing Applications. In: *Applied Sciences* 10 (2020), 7, S. 4953. – ISSN 2076–3417.

[Petr12] Petrie, Charles J.: *Automated Configuration Problem Solving*. Springer New York, 2012. – ISBN 978–1–4614–4531–9.

[Phoh98] Phoha, S.; Peluso, E.; Eberbach, E.; Kiraly, A.: Coordination of engineering design agents for high assurance in complex dynamic system design. In: *Proceedings Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No.98EX231)* Bd. 1998-Novem, IEEE Comput. Soc, 1998. – ISBN 0–8186–9221–9, S. 296–303.

[Plap20a] Plappert, Stefan; Gembarski, Paul C.; Lachmayer, Roland: *The Use of Knowledge-Based Engineering Systems and Artificial Intelligence in Product Development: A Snapshot*. Bd. 1051. Springer International Publishing, 2020, S. 62–73.

[Plap20b] Plappert, Stefan; Hoppe, Lukas; Gembarski, Paul C.; Lachmayer, Roland: Application of Knowledge-based Engineering for Teaching Design Knowledge to Design Students. In: *Proceedings of the Design Society: DESIGN Conference* 1 (2020), 5, S. 1795–1804. – ISSN 2633–7762.

[Plap21] Plappert, Stefan; Gembarski, Paul C.; Lachmayer, Roland: *Multi-Agent Systems in Mechanical Engineering: A Review*. Bd. 241. Springer Singapore, 2021, S. 193–203.

[Plap22a] Plappert, Stefan; Becker, Christian; Gembarski, Paul C.; Lachmayer, Roland: Feasibility Evaluation of Milling Designs Using Multi-Agent Systems. In: *Proceedings of the Design Society* 2 (2022), 5, S. 763–772. – ISSN 2732–527X.

[Plap22b] Plappert, Stefan; Becker, Christian; Gembarski, Paul C.; Lachmayer, Roland: Application of Decentralized and Self-Regulating Knowledge Bases for Assembly Design Automation. In: *Proceedings of the 10th International Conference on Mass Customization and Personalization - Community of Europe (MCE-CE 2022)*, University of Novi Sad - Faculty of Technical Sciences, 2022, S. 115–123.

[Plap22c] Plappert, Stefan; Gembarski, Paul C.; Lachmayer, Roland: Development of a knowledge-based and collaborative engineering design agent. In: *Procedia Computer Science* 207 (2022), S. 946–955. – ISSN 18770509.

[Plap22d] Plappert, Stefan; Gembarski, Paul C.; Lachmayer, Roland: *Knowledge-Based Design Evaluation of Rotational CAD-Models with a Multi-Agent System*. Springer International Publishing, 2022, S. 47–56.

[Ponn11] Ponn, Josef; Lindemann, Udo: *Konzeptentwicklung und Gestaltung technischer Produkte*. Springer Berlin Heidelberg, 2011. – ISBN 978–3–642–20579–8.

[Quin15] Quintana-Amate, S.; Bermell-Garcia, Pablo; Tiwari, A.: Transforming expertise into Knowledge-Based Engineering tools: A survey of knowledge sourcing in the context of engineering design. In: *Knowledge-Based Systems* 84 (2015), 8, S. 89–97. – ISSN 09507051.

[Rao96] Rao, Anand S.: *AgentSpeak(L): BDI agents speak out in a logical computable language*. 1996, S. 42–55.

[Reic20] Reichler, Ann-Kathrin; Redeker, Julian; Gabriel, Felix; Falke, Fabio K.; Vietor, Thomas; Dröder, Klaus: Combined Design and Process Planning for Incremental Manufacturing. In: *Procedia CIRP* 93 (2020), S. 927–932. – ISSN 22128271.

[Rocc12] Rocca, Gianfranco L.: Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. In: *Advanced Engineering Informatics* 26 (2012), 4, S. 159–179. – ISSN 14740346.

[Sabi98] Sabin, D.; Weigel, R.: Product configuration frameworks-a survey. In: *IEEE Intelligent Systems* 13 (1998), 7, S. 42–49. – ISSN 1094–7167.

[Saut17] Sauthoff, Bastian: *Generative parametrische Modellierung von Strukturkomponenten für die technische Vererbung*. Garbsen: TEWISS-Technik und Wissen GmbH Verlag, 2017.

[Schr94] Schreiber, Guus; Wielinga, Bob; de Hoog, Robert; Akkermans, Hans; de Velde, Walter: CommonKADS: A comprehensive methodology for KBS development. In: *IEEE Expert* 9 (1994), S. 28–37.

[Sear69] Searle, John R.: *Speech Acts*. Cambridge University Press, 1 1969. – ISBN 9780521096263.

[Shen97] Shen, W.; Barthès, J.-P. A.: *An experimental environment for exchanging engineering design knowledge by cognitive agents*. Springer US, 1997, S. 19–38.

[Shi20] Shi, Yang; Zhang, Yicha; Xia, Kaishu; Harik, Ramy: A Critical Review of Feature Recognition Techniques. In: *Computer-Aided Design and Applications* 17 (2020), 1, S. 861–899. – ISSN 16864360.

[Skar07] Skarka, Wojciech: Application of MOKA methodology in generative model creation using CATIA. In: *Engineering Applications of Artificial Intelligence* 20 (2007), 8, S. 677–690. – Soft Computing Applications. – ISSN 09521976.

[Sonn12] Sonnenberg, Christian; vom Brocke, Jan: *Evaluations in the Science of the Artificial – Reconsidering the Build-Evaluate Pattern in Design Science Research*. Bd. 7286 LNCS. Springer Berlin Heidelberg, 2012, S. 381–397.

[Stok01] Stokes, Melody: *Managing engineering knowledge: MOKA: methodology for knowledge based engineering applications*. Bd. 3. Professional Engineering Publishing London, 2001.

[Stud98] Studer, Rudi; Benjamins, V.Richard; Fensel, Dieter: Knowledge engineering: Principles and methods. In: *Data & Knowledge Engineering* 25 (1998), 3, S. 161–197. – ISSN 0169023X.

[Stum98] Stumptner, Markus; Friedrich, Gerhard E.; Haselböck, Alois: Generative constraint-based configuration of large technical systems. In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 12 (1998), 9, S. 307–320. – ISSN 0890–0604.

[Sun01] Sun, J.; Zhang, Y. F.; Nee, A. Y. C.: A distributed multi-agent environment for product design and manufacturing planning. In: *International Journal of Production Research* 39 (2001), 1, S. 625–645. – ISSN 0020–7543.

[Tan96] Tan, Gek W.; Hayes, C.C.; Shaw, M.: An intelligent-agent framework for concurrent product design and planning. In: *IEEE Transactions on Engineering Management* 43 (1996), S. 297–306. – ISSN 00189391.

[Ugur17] Ugurlu, Sinan; Gerhard, Detlef: Supporting development teams in the early stages of product development through DFX-based knowledge management system and communication platform. In: *Proceedings of the International Conference on Engineering Design, ICED* 6 (2017), S. 121–130. – ISSN 22204342.

[Ullm01] Ullman, David G.: Robust decision-making for engineering design. In: *Journal of Engineering Design* 12 (2001), 3, S. 3–13. – ISSN 0954–4828.

[Ullm09] Ullman, David G.: *The mechanical design process*. 2009. – 467 S. – ISBN 9780073398266.

[Vajn09] Vajna, Sàndor: *CAx für Ingenieure*. Springer Berlin Heidelberg, 2009. – ISBN 978–3–540–36038–4.

[Verh12] Verhagen, Wim J.; Bermell-Garcia, Pablo; van Dijk, Reinier E.; Curran, Richard: A critical review of Knowledge-Based Engineering: An identification of research challenges. In: *Advanced Engineering Informatics* 26 (2012), 1, S. 5–15. – ISSN 14740346.

[Wang09] Wang, Jian X.; Tang, Ming X.; Song, Lin N.; Jiang, Shou Q.: Design and implementation of an agent-based collaborative product design system. In: *Computers in Industry* 60 (2009), 9, S. 520–535. – ISSN 01663615.

[Wang12] Wang, Wei; Li, Yingguang; Li, Hai; Liu, Changqing: An agent-based collaborative design framework for feature-based design of aircraft structural parts. In: *International Journal of Computer Integrated Manufacturing* 25 (2012), 10, S. 888–900. – ISSN 0951–192X.

[Wang17] Wang, Wei M.; Lünnemann, Pascal; Preidel, Maurice; Stark, Rainer: Wissen in Produktentwicklungsprozessen – Ein aktivitätenbasierter Analyseansatz. In: *15. Gemeinsames Kolloquium Konstruktionstechnik 2017* Bd. 15, 2017, S. 183–192.

[WEBE00] WEBER, Christian; WERNER, Horst: Klassifizierung von CAX-Werkzeugen für die Produktentwicklung auf der Basis eines neuartigen Produkt- und Prozessmodells. 2000. – Forschungsbericht.

[WEIS05] WEISS, Gerhard; JAKOB, Ralf: Gaia. In: *Agentenorientierte Softwareentwicklung: Methoden und Tools* (2005), S. 41–79.

[WEYN10] WEYNS, Danny: *Architecture-Based Design of Multi-Agent Systems*. Springer Berlin Heidelberg, 2010. – 1–224 S. – ISBN 978–3–642–01063–7.

[WITT13] WITTEL, Herbert; MUHS, Dieter; JANNASCH, Dieter; VOSSIEK, Joachim: *Roloff/Matek Maschinenelemente*. Springer Fachmedien Wiesbaden, 2013. – ISBN 978–3–658–02326–3.

[WOOL00] WOOLDRIDGE, Michael; JENNINGS, Nicholas R.; KINNY, David: The Gaia Methodology for Agent-Oriented Analysis and Design. 2000. – Forschungsbericht. – 285–312 S.

[WOOL09] WOOLDRIDGE, Michael: *An introduction to multiagent systems*. John Wiley & Sons, 2009. – ISBN ISBN: 978–0–470–51946–2.

[WU96] WU, MC; LIT, CR: Analysis on machined feature recognition techniques based on B-rep. In: *Computer-Aided Design* 28 (1996), 8, S. 603–616. – ISSN 00104485.

[XINH07] XINHUA, Liu; XUTANG, Zhang; WENJIAN, Liu: Integration of CAPP and CAFD based on agent technology. In: *Mechatronics, 2007 IEEE International Conference on*, IEEE, 5 2007. – ISBN 1–4244–1183–1, S. 1–4.

[ZHAN01] ZHANG, Y.; HU, W.; RONG, Y.; YEN, David W.: Graph-based set-up planning and tolerance decomposition for computer-aided fixture design. In: *International Journal of Production Research* 39 (2001), 1, S. 3109–3126. – ISSN 0020–7543.

[ZHU15] ZHU, Jiang; KATO, Masato; TANAKA, Tomohisa; YOSHIOKA, Hayato; SAITO, Yoshio: Graph based automatic process planning system for multi-tasking machine. In: *Journal of Advanced Mechanical Design, Systems, and Manufacturing* 9 (2015), S. JAMDSM0034–JAMDSM0034. – ISSN 1881–3054.

Verwendete Normen und Richtlinien

[DIN471] DIN 471:2011-04: *Sicherungsringe (Halteringe) für Wellen – Regelausführung und schwere Ausführung*. Beuth Verlag, Berlin

[DIN8580] DIN 8580:2022-12: *Fertigungsverfahren – Begriffe, Einteilung*. Beuth Verlag, Berlin

[DIN61160] DIN EN 61160:2006-06: *Entwicklungsbewertung*. Beuth Verlag, Berlin

[VDI2209] VDI 2209:2009-03: *3-D-Produktmodellierung – Technische und organisatorische Voraussetzungen – Verfahren, Werkzeuge und Anwendungen – Wirtschaftlicher Einsatz in der Praxis*. Beuth Verlag, Berlin

[VDI2218] VDI 2218:2003-03: *Informationsverarbeitung in der Produktentwicklung – Feature-Technologie*. Beuth Verlag, Berlin

[VDI2221-1] VDI 2221 Blatt 1:2019-11: *Entwicklung technischer Produkte und Systeme – Modell der Produktentwicklung*. Beuth Verlag, Berlin

[VDI2223] VDI 2223:2004-01: *Methodisches Entwerfen technischer Produkte*. Beuth Verlag, Berlin

[VDI2249] VDI 2249:2003-09: *Informationsverarbeitung in der Produktentwicklung – CAD-Benutzungsfunktionen*. Beuth Verlag, Berlin

[VDI2653-1] VDI/VDE 2653 Blatt 1:2018-05: *Agentensysteme in der Automatisierungstechnik – Grundlagen*. Beuth Verlag, Berlin

[VDI2653-2] VDI/VDE 2653 Blatt 2:2018-02: *Agentensysteme in der Automatisierungstechnik – Entwicklung*. Beuth Verlag, Berlin

[VDI2653-3] VDI/VDE 2653 Blatt 3:2020-03: *Agentensysteme in der Automatisierungstechnik – Anwendung*. Beuth Verlag, Berlin

[VDI5610-1] VDI 5610 Blatt 1:2009-03: *Wissensmanagement im Ingenieurwesen – Grundlagen, Konzepte, Vorgehen*. Beuth Verlag, Berlin

[VDI5610-2] VDI 5610 Blatt 2:2017-05: *Wissensmanagement im Ingenieurwesen – Wissensbasierte Konstruktion (KBE)*. Beuth Verlag, Berlin

Anhang

A Prüfung der Herstellbarkeit

Programmcode des Feature-Agenten in Python

```
1  from sys import path
2  from spade_bdi.bdi import BDIAgent
3  from spade.agent import Agent
4  from spade.behaviour import OneShotBehaviour, CyclicBehaviour
5  from spade.message import Message
6  from spade.template import Template
7  from spade import quit_spade
8
9  import time
10 import asyncio
11 import json
12 import networkx as nx
13
14 from Code.Auslese import feat_AAG_subs, split_faces
15
16 class FeatureClass(BDIAgent):
17     def add_custom_actions(self, actions):
18         @actions.add_function(".myinfo" ,())
19
20         def myinfo():                  # Füge Übergabewerte ein
21             rec = "yellowpage@"+self.jid[1]
22             self.bdi.set_belief("yellowpage", rec)
23             # Gebe JID und Verantwortlichkeiten zurück
24             jid = self.jid[0]+"@"+self.jid[1]
25             erg = str([jid , "Feature"])
26             return erg
27
28         # Definition von Aktionen für den Zugriff aus der ASL
29         @actions.add_function(".setjid" ,(tuple,))
30
```

```
31          def setjid(resjid):
32              responsibility = resjid[0].lower()
33              jid = resjid[1]
34              self.bdi.set_belief(responsibility, jid)
35              return
36
37          @actions.add_function(".findfeature" ,(tuple,))
38          def findfeature(DictsandFeat):
39              feat = eval(DictsandFeat[1])
40              dictionaries = eval(DictsandFeat[0])
41              i = DictsandFeat[2]
42
43              feature = feat_AAG_subs(dictionaries, feat)
44              feature = split_faces(dictionaries, feature)
45              return (i, str(feature))
46
47          @actions.add_function(".killagent" ,())
48          def killagent():
49              self.bdi.kill()
50              return
```

Programmcode 1: Python-Code des Feature-Agenten zur Bewertung der Herstellbarkeit eines Fräsbauteils

Aktionsvorrat des Feature-Agenten als ASL-Datei

```
1  !start.
2
3  +!start <-
4    .print("Started.");
5    .print("Trying to registrate.");
6    .myinfo(Info);
7    ?yellowpage(B);
8    .print("Sending message to", B, ".");
9    .send(B, achieve, register(Info)).
10
11 +!registered(X)[source(Sender)] <-
12   .print("Received a message from", Sender, "with \"", X, "\".");
13   !getmanager.
14
15 +!getmanager <-
16   .print("Requesting Manager-JID.");
17   ?yellowpage(B);
18   .send(B, achieve, request("Manager")).
19
20 +!jidrequest(Jid)[source(Sender)] <-
21   .print("Received a JID from", Sender, ":\"", Jid, " \".");
22   .setjid(Jid, R).
23
24 +manager(Jid)<-
25   .send(Jid, achieve, featureready("Ready to work")).
26
27 +!getfeature(Dictandfeat)[source(Sender)] <-
28   .print("Received Dictionaries and Feature from", Sender, ".");
29   !startanalyse(Dictandfeat).
30
31 +!startanalyse(Dictandfeat) <-
32   .print("Searching for Feature and analyse their Faces.");
33   .findfeature(Dictandfeat, Feature);
34   .print("Feature found.");
35   ?manager(Jid);
36   .print("Sending Feature to", Jid, ".");
37   .send(Jid, achieve, backfeature(Feature));
38   .killagent(R).
```

Programmcode 2: Aktionsvorrat des Feature-Agenten zur Bewertung der Herstellbarkeit eines Fräsbauteils als ASL-Datei

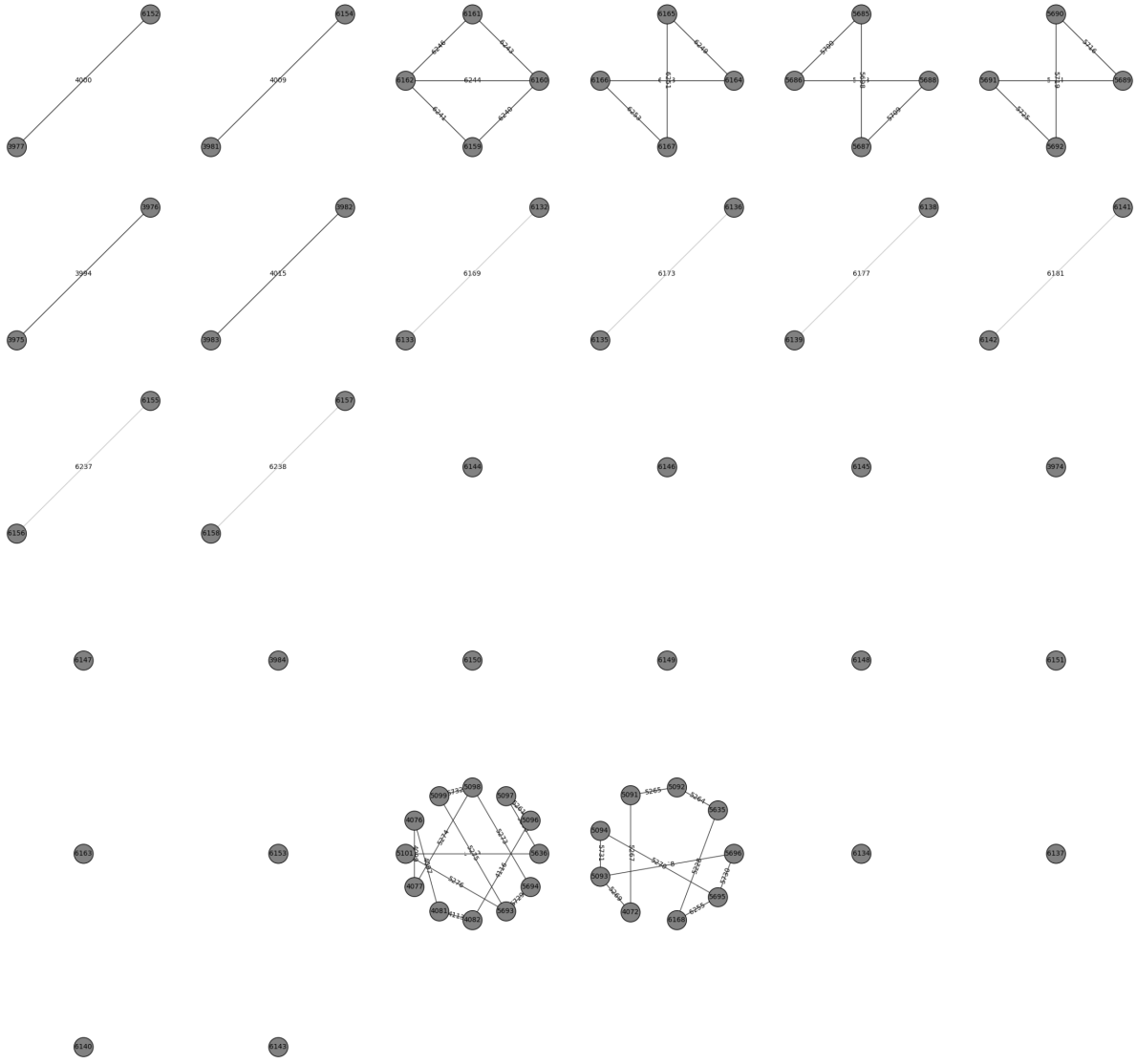


Abbildung 1: Feature-Graphen des Aufspannwinkels

Protokoll der Entwurfsbewertung des Aufspannwinkels

```
1  Start MAS...
2  manager-agent: Started.
3  manager-agent: Generating Yellowpage-Agent.
4  yellowpage-agent: Started
5  manager-agent: Trying to registrate.
6  manager-agent: Sending message to yellowpage-agent.
7  yellowpage-agent: Got registration request from  manager-agent.
8  manager-agent: Received a message from yellowpage-agent: with "
       Registration confirmed".
9  manager-agent: Sending User-Registration to yellowpage-agent.
10 manager-agent: Generating Model-Agent.
11 yellowpage-agent: Got Userregistration request from  manager-agent.
12 manager-agent: Received a message from yellowpage-agent with "User-
       Registration confirmed".
13 model-agent: Started.
14 model-agent: Trying to registrate.
15 model-agent: Sending message to yellowpage-agent.
16 yellowpage-agent: Got registration request from model-agent.
17 model-agent: Received a message from yellowpage-agent with "Registration
        confirmed".
18 model-agent: Requesting Manager-JID.
19 yellowpage-agent: Got a JID request from model-agent to find a Manager.
20 model-agent: Received a JID from yellowpage-agent: "[Manager, manager-
       agent]".
21 manager-agent: Received a message from model-agent with "Ready to work".
22 manager-agent: Sending path with work instruction to model-agent.
23 model-agent: Received path of model from manager-agent.
24 model-agent: Starting analysing the model.
25 model-agent: Translated model to Dictionary.
26 model-agent: Sending Dictionary to manager-agent.
27 manager-agent: Received Dictionaries from model-agent.
28 manager-agent: Generating Machine-Agent.
29 manager-agent: Generating Matrix-Agent.
30 machine-agent: Started.
31 machine-agent: Trying to registrate.
32 machine-agent: Sending message to yellowpage-agent.
33 yellowpage-agent: Got registration request from machine-agent.
34 matrix-agent: Started.
35 machine-agent: Received a message from yellowpage-agent: with "
       Registration confirmed".
36 matrix-agent: Trying to registrate.
37 machine-agent: Requesting Manager-JID.
38 matrix-agent: Sending message to yellowpage-agent.
39 yellowpage-agent: Got a JID request from machine-agent to find a Manager
       .
```

```
40 yellowpage-agent: Got registration request from matrix-agent.
41 machine-agent: Received a JID from yellowpage-agent: "[Manager, manager-
       agent]".
42 matrix-agent: Received a message from yellowpage-agent with "
       Registration confirmed".
43 manager-agent: Received a message from machine-agent with "Ready to work
       ".
44 matrix-agent: Requesting Manager-JID.
45 manager-agent: Sending Path and Dictionaries with work instruction to
       machine-agent.
46 yellowpage-agent: Got a JID request from matrix-agent to find a Manager.
47 machine-agent: Received Dictionaries and Path from manager-agent.
48 machine-agent: Searching for possible machines.
49 matrix-agent: Received a JID from yellowpage-agent: "[Manager, manager-
       agent]".
50 machine-agent: Machinelist found.
51 machine-agent: Sending Machinelist to manager-agent.
52 manager-agent: Received a message from matrix-agent with "Ready to work"
       .
53 manager-agent: Sending path and dictionaries with work instruction to
       matrix-agent.
54 manager-agent: Received Subgraphs from machine-agent.
55 matrix-agent: Received Path and Dictionaries from manager-agent.
56 matrix-agent: Generating Matrix.
57 matrix-agent: Matrix generated.
58 matrix-agent: Sending Matrix to manager-agent.
59 manager-agent: Received Matrix from matrix-agent.
60 manager-agent: Generating Graph-Agent.
61 graph-agent: Started.
62 graph-agent: Trying to registrate.
63 graph-agent: Sending message to yellowpage-agent.
64 yellowpage-agent: Got registration request from graph-agent.
65 graph-agent: Received a message from yellowpage-agent with "Registration
        confirmed".
66 graph-agent: Requesting Manager-JID.
67 yellowpage-agent: Got a JID request from graph-agent to find a Manager.
68 graph-agent: Received a JID from yellowpage-agent: "[Manager, manager-
       agent]".
69 manager-agent: Received a message from graph-agent with "Ready to work".
70 manager-agent: Sending Path and Matrix with work instruction to graph-
       agent.
71 graph-agent: Received Path and Matrix from manager-agent.
72 graph-agent: Generating Graph.
73 graph-agent: Graph generated.
74 graph-agent: Searching for subgraphs.
75 graph-agent: Sending Subgraphs to manager-agent.
76 manager-agent: Received Subgraphs from graph-agent.
```

```
77 manager-agent: Generating GType-Agents.
78 ...
79 gtype-agent: Started.
80 gtype-agent: Trying to registrate.
81 gtype-agent: Sending message to yellowpage-agent.
82 yellowpage-agent: Got registration request from gtype-agent.
83 gtype-agent: Received a message from yellowpage-agent with "Registration
       confirmed".
84 gtype-agent: Requesting Manager-JID.
85 yellowpage-agent: Got a JID request from gtype-agent to find a Manager.
86 gtype-agent: Received a JID from yellowpage-agent: "[Manager, manager-
     agent]".
87 manager-agent: Received a message from gtype-agent with "Ready to work".
88 manager-agent: Sending Dictionaries and Index = 1 Subgraphs with work
     instruction to gtype-agent.
89 gtype-agent: Received Dictionaries and Subgraph from manager-agent.
90 gtype-agent: Searching for Graphtype.
91 gtype-agent: Graphtype found.
92 gtype-agent: Sending Graphtype to manager-agent.
93 manager-agent: Received Graphtype from gtype-agent.
94 ...
95 manager-agent: Generating Feature-Agents.
96 feature-agent: Started.
97 feature-agent: Trying to registrate.
98 feature-agent: Sending message to yellowpage-agent.
99 yellowpage-agent: Got registration request from feature-agent.
100 feature-agent: Received a message from yellowpage-agent with "
     Registration confirmed".
101 feature-agent: Requesting Manager-JID.
102 yellowpage-agent: Got a JID request from feature-agent to find a Manager
     .
103 feature-agent: Received a JID from yellowpage-agent: "[Manager, manager-
     agent]".
104 manager-agent: Received a message from feature-agent with "Ready to work
     ".
105 feature-agent: Received Dictionaries and Feature from manager-agent.
106 feature-agent: Searching for Feature and analyse their Faces.
107 feature-agent: Feature found.
108 feature-agent: Sending Feature to manager-agent.
109 manager-agent: Received Feature from feature-agent.
110 ...
111 manager-agent: Requesting Model-JID.
112 yellowpage-agent: Got a JID request from manager-agent: to find a Model.
113 manager-agent: Received a JID from yellowpage-agent: "[Model, model-
     agent]".
114 manager-agent: Sending Dictionaries with work instruction to model-agent
     .
```

```
115 model-agent: Received Dictionaries from manager-agent.
116 yellowpage-agent: Got a JID request from model-agent to find a User.
117 model-agent: Received a JID from yellowpage-agent: "[User, userserver-
    agent]".
118 model-agent: Asking if Chamfers are necessary.
119 model-agent: Received the Answer for 6144 saying yes
120 model-agent: Received the Answer for 6147 saying yes
121 model-agent: Received the Answer for 6150 saying yes
122 model-agent: Received the Answer for 6148 saying yes
123 model-agent: Received the Answer for 6151 saying no
124 model-agent: Received the Answer for 6153 saying no
125 model-agent: Finished Asking.
126 manager-agent: Received Dictionary from model-agent.
127 manager-agent: Generating Tool-Agents.
128 ...
129 tool-agent: Started.
130 tool-agent: Trying to registrate.
131 tool-agent: Sending message to yellowpage-agent.
132 yellowpage-agent: Got registration request from tool-agent.
133 tool-agent: Received a message from yellowpage-agent with "Registration
    confirmed".
134 tool-agent: Requesting Manager-JID.
135 yellowpage-agent: Got a JID request from tool-agent to find a Manager.
136 tool-agent: Received a JID from yellowpage-agent: "[Manager, manager-
    agent]".
137 manager-agent: Received a message from tool-agent with "Ready to work".
138 manager-agent: Sending Dictionaries and Index = 1 Feature with work
    instruction to tool-agent.
139 tool-agent: Received Dictionaries and Path from manager-agent.
140 tool-agent: Searching for possible Tools.
141 tool-agent: Tools found.
142 tool-agent: Sending Toollist to manager-agent.
143 manager-agent: Received Tools from tool-agent.
144 ...
145 manager-agent: Sending Dictionaries with work instruction to model-agent
    .
146 model-agent: Received Dictionaries and Toollist from manager-agent.
147 model-agent: Starting checking for toolrunout.
148 model-agent: Finished checking for toolrunout.
149 model-agent: Sending Dictionary to manager-agent.
150 manager-agent: Received Dictionaries from model-agent.
151 manager-agent: Sending Dictionaries with work instruction to model-agent
    .
152 model-agent: Received Dictionaries from manager-agent.
153 model-agent: Starting coloring the model.
154 model-agent: Finished coloring.
155 model-agent: Sending Dictionary to manager-agent.
```

```
156 model-agent: Saving model.
157 model-agent: Finished.
158 manager-agent: Received Dictionaries from model-agent.
159 manager-agent: Collect and output results.
160 Possible machines: DMG MORI DMU 50 3rd Generation, DMG MORI M1, DMG MORI
        CMX 600 V, DMG MORI CMX 1100 V, DMG MORI DMU 65 monoBLOCK, DMG MORI
       DMU 150 monoBLOCK, DMG MORI DMU 80 eVo linear, DMG MORI NMV 3000 DCG,
        HEDELIUS TILTENTA 6-2300
161 Numbers of Agents
162 GType: 32
163 Feature: 32
164 Tool: 32
165 MAS finished...
166 The System needs 342.90612387657166s.
```

Programmcode 3: Protokoll der Entwurfsbewertung des Getriebegehäuses

Maschinenliste

Name	Maschinentyp	Rundtisch	Bauraum			Last [kg]
			Länge [mm]	Breite [mm]	Höhe [mm]	
DMG MORI DMU 50 3rd Generation	5-Achs-Fräsen	Ja	650	520	475	300
DMG MORI M1	Vertikal-Fräsen	Nein	550	550	510	600
DMG MORI CMX 600 V	Vertikal-Fräsen	Nein	600	560	510	600
DMG MORI CMX 1100 V	Vertikal-Fräsen	Nein	1100	560	510	1000
DMG MORI DMP 35	Vertikal-Fräsen	Nein	350	420	380	400
DMG MORI DMU 65 monoBLOCK	5-Achs-Fräsen	Ja	735	650	560	1000
DMG MORI DMU 150 monoBLOCK	5-Achs-Fräsen	Ja	1135	1050	750	2000
DMG MORI DMU 40 eVo linear	5-Achs-Fräsen	Ja	400	400	375	250
DMG MORI DMU 80 eVo linear	5-Achs-Fräsen	Ja	800	650	550	600
DMG MORI NMV 3000 DCG	5-Achs-Fräsen	Nein	18000	4500	2000	150000
DMG MORI DMU 600 Gantry linear	5-Achs-Fräsen	Ja	500	350	510	100
HEDELIUS TILTENTA 6-2300	5-Achs-Fräsen	Ja	2300	600	800	2000

Abbildung 2: Maschinenliste

Werkzeugliste

Name	Verwendung	Durchmesser [mm]	Schnittlänge [mm]	Schaftlänge [mm]	Radius [mm]	Winkel [°]
Mk 10	Schruppen	10	30	50	0,4	45
Mk 20	Schruppen	20	35	60	0,8	45
Mk 30	Schruppen	30	35	60	0,8	45
Mk 40	Schruppen	40	50	70	0,8	45
Mk 50	Schruppen	50	55	80	0,8	45
Shank_F_15	Schlichten	15	50	50	0,8	90
Shank_F_20	Schlichten	20	70	50	0,8	90
Shank_F_25	Schlichten	25	100	50	0,8	90
Shank_R_15	Schruppen	15	50	50	0,8	90
Shank_R_20	Schruppen	20	70	50	0,8	90
Shank_R_25	Schruppen	25	100	50	0,8	90
Shank_RF_5	Schruppen und Schlichten	5	40	40	0,8	90
Shank_RF_10	Schruppen und Schlichten	10	50	50	0,8	90
Shank_RF_15	Schruppen und Schlichten	15	50	50	0,8	90
Shank_RF_20	Schruppen und Schlichten	20	80	50	0,8	90
Shank_RF_25	Schruppen und Schlichten	25	100	50	0,8	90
Cham_C_8	Fasen	8	5	40	0,0	45
Cham_C_10	Fasen	10	8	40	0,0	45
Cham_C_15	Fasen	15	10	50	0,0	45
Cham_C_20	Fasen	20	15	60	0,0	45
Drill_5	Bohren	5	50	40	0,0	118
Drill_6	Bohren	6	50	40	0,0	118
Drill_7	Bohren	7	50	40	0,0	118
Drill_8	Bohren	8	50	40	0,0	118
Drill_9	Bohren	9	50	40	0,0	118
Drill_10	Bohren	10	60	40	0,0	118
Drill_11	Bohren	11	60	40	0,0	118
Drill_12	Bohren	12	60	40	0,0	118
Drill_13	Bohren	13	60	40	0,0	118
Drill_14	Bohren	14	60	40	0,0	118
Drill_15	Bohren	15	80	40	0,0	118
Drill_16	Bohren	16	80	40	0,0	118
Drill_17	**Bohren**	**17**	**80**	**40**	**0,0**	**118**
Drill_18	**Bohren**	**18**	**80**	**40**	**0,0**	**118**
Drill_19	**Bohren**	**19**	**80**	**40**	**0,0**	**118**
Drill_20	**Bohren**	**20**	**80**	**40**	**0,0**	**118**

Abbildung 3: Werkzeugliste

B Eigene Veröffentlichungen und Patente

Tabelle 2: Aufstellung eigener Veröffentlichungen und Patente

	Journalbeiträge (peer-reviewed)
J1	Plappert, Stefan; Gembarski, Paul Christoph; Lachmayer, Roland (2022). **Development of a knowledge-based and collaborative engineering design agent**. In: *Procedia Computer Science* 207, S. 946–955.
J2	Plappert, Stefan; Becker, Christian; Gembarski, Paul Christoph; Lachmayer, Roland (2022). **Feasibility Evaluation of Milling Designs Using Multi-Agent Systems**. In: *Proceedings of the Design Society 2*, 5, S. 763–772.
J3	Gembarski, Paul Christoph; Plappert, Stefan; Lachmayer, Roland (2021). **Making design decisions under uncertainties: probabilistic reasoning and robust product design**. In: *Journal of Intelligent Information Systems* 57, 12, S. 563–581.
J4	Plappert, Stefan; Hoppe, Lukas; Gembarski, Paul Christoph; Lachmayer, Roland (2020). **Application of Knowledge-based Engineering for Teaching Design Knowledge to Design Students**. In: *Proceedings of the Design Society: DESIGN Conference* 1, 5, S. 1795–1804.

	Buchbeiträge (peer-reviewed)
B1	Plappert, Stefan; Ganter, Nicola Viktoria; von Hören, Thimo; Gembarski, Paul Christoph; Lachmayer, Roland (2023). **Decision-Support for Additive Repair Processes with a Multi-Agent System**. In: *Selvaraj, H., Fujimoto, T. (eds) Applied Systemic Studies. Lecture Notes in Networks and Systems, vol 611. Springer, S.3-11.*
B2	Plappert, Stefan; Gembarski, Paul Christoph; Lachmayer, Roland (2021). **Knowledge-Based Design Evaluation of Rotational CAD-Models with a Multi-Agent System**. In: *Advances in Systems Engineering: Proceedings of the 28th International Conference on Systems Engineering, ICSEng 2021*. Springer International Publishing, S. 47-56.
B3	Plappert, Stefan; Wolniak, Philipp; Gembarski, Paul Christoph; Lachmayer, Roland (2021). **Implicit and Explicit Modeling of Uncertainty in Early Design Stages of Product Design: A Comparative Study**. In: *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems: Proceedings of the 8th Changeable, Agile, Reconfigurable and Virtual Production Conference (CARV2021) and the 10th World Mass Customization & Personalization Conference (MCPC2021)*. Springer International Publishing, S. 653-660.

B4 Ganter, Nicola Viktoria, Plappert, Stefan, Gembarski, Paul Christoph, Lachmayer, Roland (2021). **Assessment of Repairability and Process Chain Configuration for Additive Repair**. In: *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems: Proceedings of the 8th Changeable, Agile, Reconfigurable and Virtual Production Conference (CARV2021) and the 10th World Mass Customization & Personalization Conference (MCPC2021)*. Springer International Publishing, S. 261-268.

B5 Plappert, Stefan; Gembarski, Paul Christoph; Lachmayer, Roland (2021). **Multi-Agent Systems in Mechanical Engineering: A Review**. Bd. 241. Springer Singapore, S. 193–203.

B6 Plappert, Stefan; Gembarski, Paul Christoph; Lachmayer, Roland (2020) **Decision-Making with Probabilistic Reasoning in Engineering Design**. In: *Foundations of Intelligent Systems: 25th International Symposium, ISMIS 2020, Graz, Austria, September 23–25, 2020, Proceedings*. Springer International Publishing. S. 56-65.

B7 Plappert, Stefan; Gembarski, Paul Christoph; Lachmayer, Roland (2020). **The Use of Knowledge-Based Engineering Systems and Artificial Intelligence in Product Development: A Snapshot**. Bd. 1051. Springer International Publishing, S. 62–73.

Konferenzbeiträge (peer-reviewed)

K1 Steinnagel, Carl; Bastimar, Cem; Gembarski, Paul Christoph; Plappert, Stefan, Müller, Patrik; Lachmayer, Roland (2023). **Characterization of Additive Manufactured Structures for the Development of Foam-Replacement Cushions**. In: *Innovative Product Development by Additive Manufacturing 2022*. Springer International Publishing, S. 76-96.

K2 Plappert, Stefan; Becker, Christian; Gembarski, Paul Christoph; Lachmayer, Roland (2022). **Application of Decentralized and Self-Regulating Knowledge Bases for Assembly Design Automation**. In: *Proceedings of the 10th International Conference on Mass Customization and Personalization - Community of Europe (MCE-CE 2022)*, University of Novi Sad - Faculty of Technical Sciences, S. 115–123.

K3 Plappert, Stefan, Teves, Simon, Öztürk, Mevali, Gembarski, Paul Christoph (2022). **Constraint solver for a fixture design: results of a student case study**. In: *Proceedings of the 26th ACM International Systems and Software Product Line Conference-Volume B*, S. 237-244.

K4 Hoppe, Lukas Valentin; Gembarski, Paul Christoph; Plappert, Stefan; Lachmayer, Roland (2020). **Anwendung wissensbasierter Konstruktionssysteme zur Lösungsbewertung von CAD-Konstruktionen**. In: *18. Gemeinsames Kolloquium Konstruktionstechnik 2020: Nachhaltige Produktentwicklung: KT 2020*, S. 199–208.

K5 Plappert, Stefan, Gembarski, Paul Christoph, Lachmayer, Roland (2020). **Product configuration with Bayesian network**. In: *Proceedings of the 9thInternational Conference on Mass Customization and Personalization–Community of Europe (MCP-CE 2020)*. Novi Sad: University of Novi Sad.

K6 Hoppe, Lukas Valentin; Plappert, Stefan; Gembarski, Paul Christoph; Lachmayer, Roland (2020). **Development of an Intelligent Tutoring System for Design Education**. In: *Proceedings of the 22nd International Conference on Engineering and Product Design Education*, The Design Society.

	Patente (erteilt)
P1	Kloock-Schreiber, Daniel; Bode, Behrend; da Silva de Siqueira, Renan; Plappert, Stefan; Gembarski, Paul Christoph; Lachmayer, Roland. (2021). **Vorrichtung zum vollautomatischen Zubereiten von Heißgetränken und Verfahren zum vollautomatischen Zubereiten von Heißgetränken**. DE 10 2020 119 007 B3. Deutsches Patent- und Markenamt.
P2	Kloock-Schreiber, Daniel; Bode, Behrend; da Silva de Siqueira, Renan; Plappert, Stefan; Gembarski, Paul Christoph; Lachmayer, Roland. (2021). **Abscheideeinrichtung, Vorrichtung mit einer solchen Abscheideeinrichtung und Verfahren zum Abscheiden von Feststoffen aus einer Flüssigkeit**. DE 10 2020 119 008 B3. Deutsches Patent- und Markenamt.

C Betreute studentische Arbeiten

Tabelle 3: Alphabetische Aufstellung der betreuten studentischen Arbeiten

Name	**Art der Arbeit: *Titel.* Jahr**
Ahrens, Kjell Tharon	Bachelorarbeit: *Aufbau einer Konstruktionsbewertung und Fertigungsplanung von Drehteilen mittels Multi-Agenten System.* 2022
Becker, Christian	Bachelorarbeit: *Aufbau von Templates zur Bewertung von Fräskonstruktionen mittels Multi-Agenten-Systemen.* 2021
Becker, Christian	Studienarbeit: *Automatisierte Analyse und Synthese von Fräskonstruktionen mittels Multi-Agentensystemen.* 2023
Behrens, Dörthe	Bachelorarbeit: *Aufbau einer wissensbasierten Entwicklungsumgebung zur Bewertung von Fräskonstruktionen.* 2022
Führer, Fabian	Bachelorarbeit: *Vorgehen zur Erweiterung eines bestehenden mechatronischen Produktes zu einem Produkt Service System (PSS).* 2021
Gehnen, Cornelius	Bachelorarbeit: *Aufbau eines CAD-Konfigurators mittels Multi-Agentensystemen für einen Hochvoltakkumulator.* 2023
Giesecke, Pascal	Bachelorarbeit: *Absicherung von Konstruktionskonfiguratoren.* 2020
Gode, Maximilian	Studienarbeit: *Aufbau eines Werkzeugs zur Funktionsstruktur-Modellierung.* 2020
Koopmans, Hendrik	Masterarbeit: *Aufbau einer Entwicklungsumgebung zur Bewertung der Montagefähigkeit von Baugruppen.* 2022
Müller, Inga	Masterarbeit: *Entwicklung eines Ansatzes zur Abbildung von unsicheren Anforderungen im Produktentwicklungsprozess.* 2020
Osterkamp, Lena	Bachelorarbeit: *Entwicklung eines Ansatzes zur automatisierten Auswahl von Form- und Lagetoleranzen.* 2020
Rehner, Anna	Bachelorarbeit: *Bewertung von CAD-Modellen mittels fallbasierten Schließen.* 2020
Rippe, Dennis	Bachelorarbeit: *Abbildung von Anforderungen in einer CAD-Entwicklungsumgebung.* 2020
Roffmann, Oliver	Studienarbeit: *Aufbau eines Morphologischen Kastens in einer CAD-Entwicklungsumgebung.* 2020
Roffmann, Oliver	Masterarbeit: *Entwicklung eines Ansatzes zur Implementierung von Set-Based Concurrent Engineering in eine CAD-Entwicklungsumgebung.* 2021

Name	**Art der Arbeit: *Titel.* Jahr**
Schröder, Christian	Studienarbeit: *Entwicklungsumgebung zur automatisierten Konfiguration von Vorrichtungen für die Additive Reparatur.* 2022
Van Agen, Tim	Bachelorarbeit: *Aufbau einer Benutzer-Interaktion für virtuelle Konstruktionsbewertungen.* 2022
Von Hören, Thimo	Bachelorarbeit: *Aufbau eines Multi-Agenten Systems zur automatisierten Bewertung von Konstruktionen für die Additive Reparatur.* 2022

Lebenslauf

	Persönliche Daten
Name	Stefan Plappert
Geburtsdatum	20. Dezember 1993
Geburtsort	Heilbronn

	Akademischer Werdegang
seit 2019	Doktorand an der Fakultät für Maschinenbau der Gottfried Wilhelm Leibniz Universität Hannover
2016 - 2017	Berufsbegleitendes Studium Maschinenbau an der Hamburger Fern-Hochschule und Hochschule Heilbronn Abschluss: Master of Engineering
2012-2015	Duales Studium Maschinenbau an der Dualen Hochschule Baden-Württemberg Abschluss: Bachelor of Engineering

	Schulbildung
2009-2012	Technisches Gymnasium an der Wilhelm-Maybach-Schule Heilbronn Abschluss: Allgemeine Hochschulreife

	Beruflicher Werdegang
seit 2019	Wissenschaftlicher Mitarbeiter am Institut für Produktentwicklung und Gerätebau
2015-2019	Entwicklungsingenieur bei Illig Maschinenbau GmbH & Co. KG Heilbronn