Mobile Roboter und Systeme

Stefanie Krause
Frieder Stolzenburg
Harz University of Applied Sciences
38855 Wernigerode, Germany
mailto:fstolzenburg@hs-harz.de
mailto:skrause@hs-harz.de

Das Material entstand im Rahmen des Projektes AI-Engineering und
dient als Lehr-Lernmaterial für den gleichnamigen Studiengang.
https://ai-engineer.de
Das Projekt wird gefördert im Rahmen der BMFTR-Richtlinie zur
Förderung von Projekten zum Thema „Künstliche Intelligenz in der Hochschulbildung“,
Förderkennzeichen: 16DHBKI010.

1. September 2025

Bundesministerium
für Forschung, Technologie
und Raumfahrt

Hochschule Harz
Hochschule für angewandte
Wissenschaften

Inhaltsverzeichnis

1 Intelligente Agenten

1.1 Agenten und Umgebungen

Es gibt vielfältige Literatur und Definitionen zum Thema Agenten. Brenner et al. [35] beispielsweise sehen Agenten als *Programme, die dem Nutzer bei der Suche nach Informationen unterstützen und Aufgaben in einer vernetzten, digitalen Welt erledigen*. Eine weit allgemeinere Definition findet man bei Russell und Norvig [47]:

Definition 1.1.1 (Agent)**.** Ein *Agent* ist alles, was über Sensoren seine Umgebung wahrnimmt und diese wiederum durch Aktionen mittels seiner Effektoren beeinflusst.

Nach Wooldridge und Jennings [67] ist ein Agent ein *hardware- und/oder softwarebasiertes Computersystem, welches die Eigenschaften von Autonomie, sozialer Kompetenz, Reaktivität und Proaktivität zeigt*. Legt man diese Begriffe weit genug aus, lässt sich von mechanischen Geräten über Computerprogramme bis hin zu Teams von autonomen Robotern alles als Agent klassifizieren. Ein Konsens unter Forschern deutet darauf hin, dass *Autonomie*, die Fähigkeit ohne Menschen oder andere Systeme zu handeln, ein Schlüsselmerkmal eines Agenten ist.

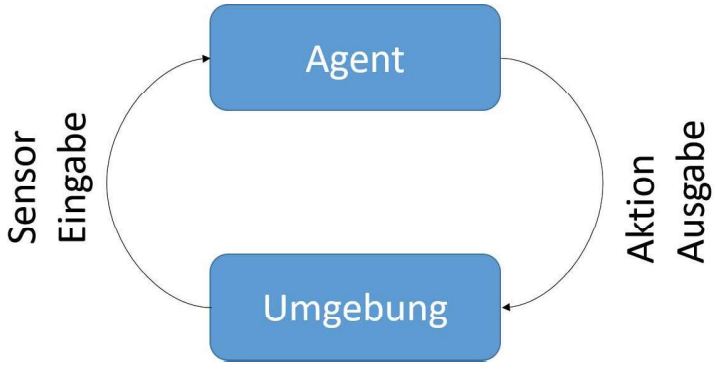


Abbildung 1.1: Der Agent nimmt Sensoreingaben aus der seiner Umgebung auf und produziert als Output Aktionen, welche die Umwelt beeinflussen.
Eigene Darstellung von S. Krause lizenziert unter CC BY 4.0.

In Abbildung 1.1 ist eine allgemeine Ansicht eines Agenten in seiner Umgebung dargestellt.

- Ein Agent besitzt *Sensoren*, mit denen er seine *Umgebung* wahrnehmen kann.
- Die *Wahrnehmung* beschreibt die wahrgenommenen Eingaben des Agenten zu jedem beliebigen Zeitpunkt.
- Die *Wahrnehmungsfolge* eines Agenten ist der vollständige Verlauf von allem, was der Agent je wahrgenommen hat.

- Die *Auswahl einer Aktion* durch den Agenten ist im Allgemeinen abhängig von der gesamten Wahrnehmungsfolge, jedoch nicht von etwas, was noch nicht wahrgenommen wurde. Sobald der Agent die Aufgaben bereits deutlich erkannt hat, sollte er in Aktion treten.
- Die entsprechende Tätigkeit kann durch *Aktoren* erfolgen.
- Die vom Agenten generierte Aktionsausgabe beeinflusst die Umgebung.

Was genau sind Sensoren und Aktoren?

- **Sensoren**:
  - Sensoren sind Geräte oder Komponenten, die physikalische Größen wie Temperatur, Druck, Licht, Bewegung oder Magnetfelder in elektrische Signale umwandeln oder
  - Befehle mit denen Eigenschaften der Umgebung ermittelt werden z. B. Dateiinhalte, Netzwerkpakete
- **Aktoren**:
  - Aktoren sind Antriebselemente, die elektrische Signale und Strom in mechanische Bewegung transformieren oder
  - Computerbefehle mit denen Eigenschaften der Umgebung verändert werden können z. B. Dateien schreiben, Bildschirmausgaben, Netzwerkpakete versenden

Es können zwei verschiedene Arten von Agenten unterschieden werden *Roboter-Agent* und *Software-Agent*.

Übung 1.1.2 (Software-Agent)**.** Beschreiben Sie, was Software-Agenten sind und überlegen Sie sich Beispiele für Software-Agenten.

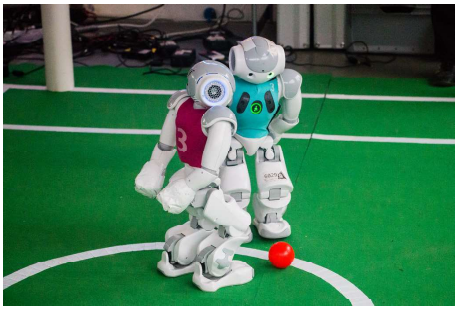
Ein Software-Agent ist ein *Programm*, das seine Umgebung wahrnimmt und in dieser Umgebung agiert. Zum Beispiel die online Werbung ist ein Ergebnis des Software-Agentens, der die Interessen von jedem Benutzer und Inserenten lernt, um den besten Gewinn zu erzielen.

Übung 1.1.3 (Mobilität)**.** Mobilität ist die Fähigkeit, die physikalische Position zu ändern. Was kann das bedeuten?

Mobile Roboter können in verschiedensten Bereichen eingesetzt werden, z. B. in der Logistik, als Dienstleistungs- oder Serviceroboter, in der Industrie, in der Medizin, in der Verteidigung und viele mehr. Beispiele für mobile Roboter wie ein Mähroboter, Fußballroboter, Serviceroboter oder ein Expeditionsroboter sind in den Abbildungen **??**, 1.4 und 1.3 dargestellt. Der Roboter *Curiosity* ist ein autogroßer Mars-Rover, der im Rahmen der NASA-Mission Mars Science Laboratory den Gale-Krater und den Mount Sharp auf dem Mars erkundet.

Übung 1.1.4 (Beispiele mobiler Roboter)**.** Welche weiteren Beispiele mobiler Roboter kennen Sie?

(a) Mähroboter STIGA Autoclip M5. Von Marco Verch https://piwigo.wuestenigel.com lizenziert unter CC BY 2.0.



(b) Nao Robot Soccer: Duel. Von Ann Wuyts https://www.flickr.com/photos/vintagedept/21540115539/ lizenziert unter CC BY 2.0.

Abbildung 1.2: Beispiele mobiler Roboter.

Abbildung 1.3: Serviceroboter Cosero. Von Sven Behnke https://openverse.org/image/03e82d62-aa2e-4bc3-ae40-9340e663dee3?q=service+robot lizenziert unter CC BY-SA 2.0.

Mathematisch ausgedrückt wird das Verhalten eines Agenten durch die **Agentenfunktion** beschrieben, die jede beliebige Wahrnehmungsfolge auf eine Aktion abbildet. Wir können die Agentenfunktion, die einen beliebigen Agenten beschreibt, tabellarisch anordnen. Für die meisten Agenten würde daraus eine sehr große Tabelle entstehen – eigentlich unendlich groß, es sei denn, wir begrenzen die Länge der Wahrnehmungsfolgen, die wir berücksichtigen wollen. Für einen Agenten, mit dem wir experimentieren wollen, können wir diese Tabelle erstellen, indem wir alle möglichen Wahrnehmungsfolgen ausprobieren und dann aufzeichnen, welche Aktionen der Agent als Reaktion darauf ausführt. Um diese Konzepte zu verdeutlichen, betrachten wir ein einfaches Beispiel [48]: die *Staubsaugerwelt* (Abbildung 1.5).

Abbildung 1.4: Curiosity Mars-Rover der NASA.
Von rawpixel `https://www.rawpixel.com/image/441297/free-photo-image-mars-robot-nasa` lizenziert unter CC BY 2.0.

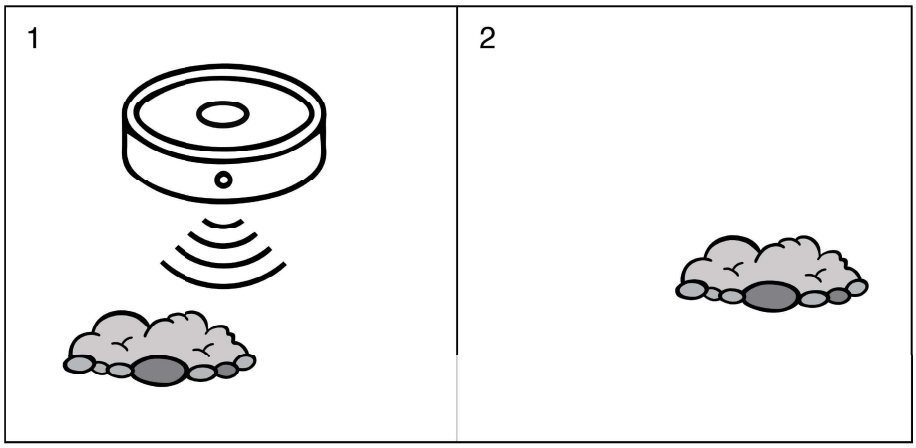


Abbildung 1.5: Eine Staubsaugerwelt mit zwei Positionen.
Eigene Darstellung von S. Krause lizenziert unter CC BY 4.0. angelehnt an [48, Abb 2.2].

Diese künstliche Welt ist so einfach, dass wir alles beschreiben können, was passieren kann:

- Die konkrete Welt hat nur zwei Positionen: die Quadrate 1 und 2.
- Der Staubsauger-Agent nimmt wahr, in welchem Quadrat er sich befindet und ob Schmutz in dem Quadrat liegt.
- Er kann entscheiden, sich nach links oder rechts zu bewegen, den Schmutz aufzusaugen oder nichts zu tun.

Eine sehr einfache Agentenfunktion ist die folgende: Wenn das aktuelle Quadrat schmutzig ist, soll gesaugt werden, andernfalls soll eine Bewegung zum anderen Quadrat erfolgen. Die Abbildung 1.1 zeigt einen Teil der tabellarischen Darstellung dieser Agentenfunktion. Die wichtige Frage lautet dabei:

Wahrnehmungsfolge	Aktion
[1, Sauber]	Rechts
[1, Schmutzig]	Saugen
[2, Sauber]	Links
[2, Sauber]	Saugen
[1, Sauber], [1, Sauber]	Rechts
[1, Sauber], [1, Schmutzig]	Saugen
...	...

Tabelle 1.1: Ausschnitt einer tabellarischen Darstellung einer einfachen Agentenfunktion für die Staubsaugerwelt.
Eigene Darstellung angelehnt an [48, Abb. 2.3].

Was macht einen Agenten intelligent oder dumm?

- In den meisten Anwendungsfällen hat ein Agent keine vollständig **Kontrolle** über seine Umwelt.
- Dies bedeutet das **dieselbe Aktion, die zweimal unter scheinbar identischen Umständen durchgeführt wurde, kann ganz andere Wirkungen haben.**
- Es kann nicht der gewünschte Effekt eintreten, daher müssen Agenten auf die Möglichkeit des **Scheiterns** vorbereitet werden.

Normalerweise verfügt ein Agent über eine Menge an Aktionen, die ihm zur Verfügung stehen, um seine Umwelt zu beeinflussen. Allerdings können nicht alle Aktionen in allen Situationen ausgeführt werden.

Beispiel 1.1.5 (Aktion nicht ausführbar)**.** Zum Beispiel ist die Aktion *Tisch anheben* nur in Situationen anwendbar, in denen das Gewicht des Tisches so gering ist, dass der Agent ihn anheben kann. In ähnlicher Weise scheitert die Aktion *Kauf eines Ferraris*, wenn nicht genügend Geldmittel zur Verfügung stehen, um dies auszuführen.

Aktionen sind also mit Vorbedingungen verknüpft, die festlegen, in welchen möglichen Situationen sie angewendet werden können.

Definition 1.1.6 (Intelligenter Agent)**.** Ein
emphintelligenter Agent ist zu flexiblen autonomen Aktionen in einer Umwelt in der Lage (um seine Ziele zu erreichen). Dabei bedeutet Flexibilität:

- *Reaktivität*: Intelligente Agenten können Änderungen in ihrer Umwelt wahrnehmen und auf diese rechtzeitig reagieren, indem sie ihr Verhalten ändern, um so weiterhin ihre Ziele erreichen zu können. Dies ist in einer dynamischer Umwelt besonders wichtig.
- *Pro-Aktivität*: Intelligente Agenten zeichnen sich durch ziel-orientiertes Verhalten aus. Sie sind nicht ausschließlich durch äußere Ereignisse getriggert, sondern auf eigene Initiative und erkennen günstige Gelegenheiten.

- *Soziale Fähigkeiten*: Soziale Fähigkeiten von Agenten bedeutet, dass sie mit anderen Agenten (möglicherweise Menschen) interagieren können. Dies ist Voraussetzung zur Kooperation.

1.2 Konzept der Rationalität

Entspricht die Folge von Aktionssequenzen eines Agenten unseren Vorstellungen, hat sich der Agent gut verhalten. Dieses Konzept des angestrebten Ergebnisses wird durch ein *Leistungsmaß* erfasst, das jede gegebene Sequenz von Umgebungszuständen auswertet. Es gibt kein feststehendes Leistungsmaß für alle Aufgaben und Agenten. In der Regel muss der Entwickler ein den Umständen angepasstes Maß empfehlen. Das ist allerdings oft nicht einfach.

Beispiel 1.2.1 (Leistungsmaß Staubsauger-Agenten)**.** Betrachten wir den Staubsauger-Agenten aus Abbildung 1.5. Wir könnten die Leistung zu messen, indem wir untersuchen, wie viel Schmutz in einer Achtstundenschicht entfernt wurde. Ein rationaler Agent kann diese Leistungsbewertung maximieren, indem er den Schmutz aufsaugt, ihn dann wieder auf den Boden wirft, ihn dann wieder aufsaugt usw. Eine geeignetere Leistungsbewertung wäre, den Agenten dafür zu belohnen, einen sauberen Boden zu hinterlassen. Beispielsweise könnte je ein Punkt für jedes saubere Quadrat pro Zeitschritt erteilt werden und Strafpunkte für verbrauchte Elektrizität und erzeugten Lärm. Als Faustregel kann formuliert werden:

Es ist sinnvoller, Leistungsbewertungen nicht auf der Grundlage dessen zu gestalten, was man tatsächlich in der Umgebung haben möchte, sondern auf der Grundlage dessen, wie man erwartet, dass sich der Agent verhalten sollte.

Beispielsweise basiert das Konzept des sauberen Bodens auf einer durchschnittlichen Sauberkeit im Laufe der Zeit. Dennoch kann dieselbe durchschnittliche Sauberkeit von zwei ganz unterschiedlichen Agenten erzielt werden, wobei der eine immer einen mittelmäßigen Job leistet, während der andere tatkräftig saugt, aber lange Pausen macht.

Was jeweils *rational* ist, hängt nach [48] von vier Dingen ab:

- die Leistungsbewertung, die das Erfolgskriterium definiert,
- das Vorwissen des Agenten über die Umgebung,
- die Aktionen, die der Agent ausführen kann,
- die bisherigen Wahrnehmungsfolge des Agenten.

Das führt zur Definition eines rationalen Agenten:

Definition 1.2.2 (rationaler Agent)**.** Ein *rationaler Agent* soll für jede mögliche Wahrnehmungsfolge eine Aktion auswählen, von der erwartet werden kann, dass sie seine Leistungsbewertung maximiert, wenn man seine Wahrnehmungsfolge sowie vorhandenes Wissen, über das er verfügt, in Betracht zieht.

Übung 1.2.3 (Rationaler Staubsauger)**.** Diskutieren Sie, ob es sich bei dem Staubsauger Beispiel um einen rationalen Roboter handelt.

Perfektion, Allwissenheit, Lernen und Autonomie

- Rationalität ist nicht vergleichbar mit *Perfektion*. Rationalität maximiert die *erwartete Leistung* wohingegen Perfektion die tatsächliche Leistung maximiert.
- Ein rationaler Agent erreicht auch nicht *Allwissenheit*. Ein allwissender Agent kennt das tatsächliche Ergebnis seiner Aktionen und kann entsprechend Handeln. Allwissenheit ist in der Realität unmöglich.
- Nach ausreichender Erfahrung mit seiner Umgebung kann das Verhalten eines rationalen Agenten letztlich unabhängig von seinem Vorwissen werden. Dadurch kann durch *Lernen* unvollständiges oder fehlerhaftes Vorwissen kompensiert werden.
- Ein rationaler Agent sollte *autonom* sein und sich selbstständig bewegen und interagieren können.

Übung 1.2.4 (Allwissenheit und Perfektion)**.** Erklären Sie in Ihren eigenen Worten, warum Allwissenheit und Perfektion in der Praxis für einen Agenten nicht möglich sind.

Verteilte KI

Definition 1.2.5 (Verteilte Künstliche Intelligenz)**.** *Verteilte Künstliche Intelligenz* (eng. Distributed Artificial Intelligence) ist ein Gebiet, das Systeme untersucht, in denen mehrere autonom agierende Agenten zusammenarbeiten, um ein gegebenes Ziel zu erreichen. Ein spezielles Problem wird in kleinere Teilprobleme aufgeteilt, die als Knoten bezeichnet werden. Diese Knoten verfügen über gemeinsames Wissen. Die Lösungsmethode wird zentral vorgegeben.

Definition 1.2.6 (Multiagenten-Systeme)**.** Man spricht von *Multiagenten-Systemen (MAS)* wenn mehrere Agenten ihr Wissen und ihre Aktionen (Semantik beschreibt dies) koordinieren.

Heutzutage sind beide Begriffe mehr oder weniger synonym. Aber warum sind überhaupt mehrere Agenten nötig? Große Informationssysteme sind verteilt, offen und heterogen. Dies erfordert intelligente, interagierende Agenten, die selbstständig agieren. Der Unterschied zwischen KI und verteilter KI ist in Tabelle 1.2 dargestellt.

Künstliche Intelligenz	Verteilte KI
Agent	System von Agenten
Intelligenz: Eigenschaft *eines Agenten*	Intelligenz: Eigenschaft *mehrerer* Agenten
Kognitive Prozesse *eines* Agenten	Soziale Prozesse *mehrerer* Agenten

Tabelle 1.2: Gegenüberstellung von KI und verteilter KI.

Übung 1.2.7 (Beispiele MAS)**.** Überlegen Sie sich praktische Beispiele, für die mehrere Agenten nötig sind, um eine Aufgabe zu erfüllen.

1.3 Die Natur der Umgebungen

Oft wird die **PEAS**-Beschreibung [48] verwendet, um die benötigten Komponenten zu beschreiben. Die Abkürzung PEAS-Beschreibung steht jeweils für die englischen Begriffe:
Performance (Leistungsbewertung),
Environment (Umgebung),
Actuators (Aktuatoren),
Sensors (Sensoren).

- Die *Leistungsbewertung* verlangt konkrete Ziele, was Agenten erreichen sollten. Es könnten natürlich auch mehrere Ziele angeben werden.
- *Umgebung* bezeichnet die Gruppe, mit denen mögliche Interaktion stattfinden.
- *Aktuatoren* werden unter dem Hardware-Aspekt betrachtet. Das sind bewegliche Bauteile mit denen der Roboter die Form, Position und Orientierung ändern können.
- *Sensoren* sind technische Bauteile, die bestimmte physikalische oder chemische Eigenschaften und/oder die stoffliche Beschaffenheit der Roboterumgebung erfassen können.

Beispiel 1.3.1 (PEAS-Beschreibung Rubik's Cube Solver)**.** Beispielhafte PEAS-Beschreibung eines Rubik's Cube Solvers:

- Leistungsbewertung: Durchschnittliche Anzahl der Schritte zur Lösung; Rechenzeit, bis Lösungsweg gefunden ist
- Umgebung: Zustandsraum des Rubik's Cubes, d. h. Menge aller möglichen Permutationen
- Aktuatoren: Durchführung von Drehungen
- Sensoren: Beobachtung der aktuellen Konfiguration des Würfels

Übung 1.3.2 (PEAS-Beschreibungen)**.** Geben Sie für die folgenden Aktivitäten eine PEAS-Beschreibung der Aufgabenumgebung an und charakterisieren Sie sie in Bezug auf die in Abschnitt aufgelisteten Eigenschaften.

1. Auf dem Meeresgrund nach Titan forschen
2. Im Internet gebrauchte Bücher zu KI kaufen
3. Tennis gegen eine Ballwand spielen
4. Auf einer Auktion für einen Artikel bieten
5. Webcraler zur Sammlung von Emailadressen
6. Einen Wettkampf mit Fußballrobotern spielen

1.4 Eigenschaften von Aufgabenumgebungen

Da die Umgebung sehr umfangreich sein kann werden in [48] die Bereiche nach sieben bestimmten Merkmalen klassifiziert.

- **Vollständig beobachtbar** vs. **Teilweise beobachtbar**
  Sensoren sollten in einer vollständig beobachtbaren Umgebung zu jedem beliebigen Zeitpunkt alle für die Aktionsauswahl relevanten Aspekte erkennen. Im Gegensatz dazu wird eine Umgebung als teilweise beobachtbar betrachtet, wenn die Teile des Umweltzustands nicht in den Sensordaten enthalten sind.

  Beispiel 1.4.1 (teilweise beobachtbar)**.** Ein Agent defekten Sensoren wird als in einer Art von teilweise beobachtbaren Umgebung angesehen.

  Wenn jedoch ein Agent ohne Sensoren ausgestattet sind, ist die befindliche Umgebung keinen von denen gerade genannten, sondern gehört zur Kategorie *nicht beobachtbar*.

- **Einzelagent** vs. **Multiagenten**
  Wie es sich schnell vermuten lässt, hängt es bei Einzel- oder Multi-agenten davon ab, wie viele Agenten sich in der Umgebung befinden. Zum Beispiel ist ein Kreuzworträtsel eine typische Einzelagentenumgebung. Im Vergleich dazu befindet ein Agent sich in einer Zweiagentenumgebung bzw. in einer Multiagentenumgebung beispielsweise beim Schach oder Taxifahrer, weil während jeder Entscheidung der Agent ein anderes Objekt betrachten sowie behandeln muss.

  Es stehen zwei Arten von Multiagentenumgebungen zur Verfügung, *konkurrierende Multiagentenumgebungen* und *(partiell) kooperative Multiagentenumgebungen*. Die Leistungsbewertung ist in einer konkurrierenden Multiagentenumgebungen zu maximieren, sowie die Leistungsbewertung des Gegenspielers zu minimieren.

  Beispiel 1.4.2 (Multiagenten)**.** Ein Beispiel für eine konkurrierenden Multiagentenumgebungen ist Schach. Eine partiell kooperative Multiagentenumgebung ist beispielsweise der Roboterfußball, da alle Agenten eines Teams ein gemeinsames Ziel haben, Tore für die eigene Mannschaft erzielen, aus diesem Grund ist die Leistungsbewertung davon zu maximieren. Aus einer anderer Ansicht kann ein Kampf um Ball zwischen zwei Agenten gegnerischer Teams in der Roboterfußballumgebung auch als konkurrierend betrachtet werden.

- **Deterministisch** vs. **Stochastisch**
  Wenn der nächste Zustand der Umgebung vollständig durch den aktuellen Zustand und die durch den Agenten ausgeführten Aktionen festgelegt wird ist die Umgebung deterministisch. Der Agent muss sich keine Sorgen über Unsicherheiten in einer vollständig beobachtbaren, deterministischen Umgebung machen.Das Gegenstück dazu ist die stochastische Umgebung, die in einer teilweise beobachtbaren Umgebung auftreten kann. Das heißt, dass alle zukünftigen Zustände nicht genau vorhergesagt werden können.

  Weiterhin gibt es Umgebungen, die nicht vollständig beobachtbar oder nicht deterministisch ist. Dieser *unbestimmten* Umgebung ist normalerweise die Eigenschaft stochastisch zugeordnet, welche mit Wahrscheinlichkeiten sowie Zufallsvariablen zu tun haben.

Beispiel 1.4.3 (deterministisch vs. stochastisch)**.** Das Taxifahren ist zweifellos von Natur aus stochastisch, da das Verhalten des Verkehrs nicht genau vorausgesagt werden kann. Darüber hinaus können unerwartete Probleme wie Reifenpannen oder ein plötzlicher Motorversagen auftreten, ohne Vorwarnung. Im Gegensatz dazu ist die Staubsaugerwelt, die wir zuvor beschrieben haben, deterministisch. Es gibt jedoch Varianten davon, die stochastische Elemente enthalten können, wie beispielsweise das zufällige Auftreten von Schmutz oder einen unzuverlässigen Saugmechanismus.

- **Episodisch** vs. **Sequenziell**
  Die Aktion in einer episodischen Umgebung wird nicht von vorherigen Zuständen beeinflusst und ist unabhängig von in früheren Episoden ausgeführten Aktionen, während in einer sequenziellen Umgebung die Aktion auf früheren Ergebnissen basiert.

  Beispiel 1.4.4 (episodisch vs. sequentiell)**.** Ein Packroboter, der ohne Berücksichtigung der Umstände jede Entscheidung treffen kann ist episodisch. Im Vergleich dazu sollte ein Agent beim Schach in der kurzen Zeit die gespielten Züge analysieren, um zu siegen (sequenziell). Aus diesem Grund ist eine sequenzielle Umgebung auch viel komplexer als eine episodische.

- **Statisch** vs. **Dynamisch**
  Wenn die Umgebung sich ändert, weil z. B. andere Agenten dort sein, während der Agent nachdenkt, wird diese als dynamisch definiert. Dauerhaft wird in dynamischen Umgebungen die Frage an den Agenten gestellt, was er tun will. Bevor eine Entscheidung getroffen ist, wird es so erkannt, als wolle er nichts tun. In einer statischen Situation muss der Agent dagegen nicht an die Umgebung bei der Auswahl der nächsten Aktion denken, noch nicht mal an Zeitbeschränkungen.

  Beispiel 1.4.5 (statisch vs. dynamisch)**.** Die medizinische Diagnose ist dynamisch, allerdings ist ein Kreuzworträtsel eine statische Umgebung.

  In welcher Klassifikation befindet sich jedoch Schach? Das strategische Brettspiel gehört zu der *semidynamischen* Umgebung, im Falle, dass das Spiel mit Zeit begrenzt wird. Obwohl in der begrenzten Zeit der Zustand still bleibt, ändert die Leistungsbewertung des Agent sich nach jedem Zug.

- **Diskret** vs. **Stetig**
  Der Unterschied zwischen einer diskreten und stetigen Umgebung liegt an dem Zustand der Umgebung. Eine diskrete Umgebung mit nur endlich vielen oder zumindest abzählbaren, festgelegten und deutlich gegeneinander abgrenzbaren Zuständen.

  Beispiel 1.4.6 (diskret vs. stetig)**.** Stetig ist z. B. das Spielen wie Schach (ohne Zeitbeschränkung) oder Skat [3], da in beiden Umgebung Wahrnehmungen und Aktionen diskret sind. Das Gegenbeispiel sind Taxifahrer und Roboterfußball, die sich in Umgebungen mit stetigem Zustand und stetiger Zeit befinden. Darüber hinaus werden die Positionen und Geschwindigkeiten von anderen Agenten im Laufe der Zeit stetig geändert. Allerdings gibt es keine extremen Änderungen und die Werte befinden sich in einem bestimmten Bereich.

- **Bekannt** vs. **Unbekannt**
  Streng genommen präsentieren die beiden Umgebungstypen bekannt und unbekannt

keine Attribute. Der Zustand bekannt wird einer Umgebung je nach dem Wissensstand des Agenten (oder Designern) über die physikalischen Gesetze der realen oder virtuellen Umgebung gegeben. Sozusagen werden die Ergebnisse aller Aktionen in einer bekannten Umgebung angeboten, und sind die Ergebniswahrscheinlichkeiten aller Aktionen in einer stochastische bekannten Umgebung gegeben. Während der Agent in anderen unbekannten Umgebung selbst lernen muss um eine gute Entscheidung für die nächste Aktion zu treffen.

Insofern könnte man davon ausgehen, dass die folgenden Fälle: *teilweise beobachtbar, Multiagenten, stochastisch, sequenziell, dynamisch, stetig und unbekannt* die Schwierigsten sind. Natürlich ändern sich Eigenschaften jeder Umgebung je nach Designer*in oder der Definition von der Umgebung. Dadurch ist beispielsweise der Agent zum Taxifahren in einer unbekannten Stadt am kompliziertesten, jedoch ändert sich die Situation, wenn ein Großteil der Umgebung auf einer Landkarte erkennbar ist.

Beispiel 1.4.7 (Charakterisierung Rubik's Cube Solver)**.** Eine mögliche Charakterisierung des Rubik's Cube Solvers:

- Man könnte entweder den gesamten Zustand als direkt einsehbar ansehen, oder aber nur einzelne Seiten der Würfels sichtbar machen.
- Es handelt sich um ein Spiel für eine Person.
- Das Ergebnis jeder Aktion ist eindeutig definiert also deterministisch.
- Aktionen beeinflussen Folgezustände, langfristige Planung ist hilfreich, Bewertung erfolgt auf Basis von Sequenzen und nicht von Einzelaktionen.
- Der Zustand ändert sich nur durch Aktionen des Agenten und ist somit statisch. Die Bewertung des Agenten ist aber möglicherweise zeitabhängig.
- Es gibt eine endliche Menge an Zuständen und Aktionen und ist somit diskret.

Bei Multiagenten-Systemen wird oft außerdem zwischen *homogen vs. heterogen* unterschieden:

- Homogenes Agenten-System: alle Agenten haben grundsätzlich die gleichen Fähigkeiten, in Abhängigkeit von der aktuellen Situation können sie jedoch unterschiedliche Rollen einnehmen
- Heterogenes Agenten-System: die Agenten haben unterschiedliche Aufgaben und daher meist auch eine unterschiedliche Architektur und Implementierung

Übung 1.4.8 (Homogen vs. heterogen)**.** Nennen Sie jeweils ein Beispiel für 1. ein homogenes bzw. 2. ein heterogenes Multiagenten-System mit (a) den zugehörigen PEAS-Beschreibungen und (b) Charakterisierungen nach Umwelteigenschaften.

1.5 Struktur von Agenten

Die Aufgabe der künstlichen Intelligenz ist es, das Agentenprogramm zu entwerfen, das die *Agentenfunktion* implementiert – *die Zuordnung von Wahrnehmungen zu Aktionen*. Wir gehen davon aus, dass dieses Programm auf einem Computer mit physischen Sensoren und Aktuatoren ausgeführt wird – was wir auch als Architektur bezeichnen wollen:
$Agent = Architektur + Programm$

Physikalischer Aufbau

Wenn Sensoren eine Wahrnehmung empfangen, läuft das Agentenprogramm. Nach der Entscheidung vom Programm wird diese an Aktuatoren weitergeleitet, welche die gewählte Aktion ausführen. Der Ablauf sieht im Allgemeinen so aus:
$Sensoren \rightarrow Programm \rightarrow Aktuatoren$

Agentenprogramme

Die Agentenprogramme haben alle denselben Aufbau: Sie nehmen die aktuelle Wahrnehmung als Eingabe von den Sensoren entgegen und geben eine Aktion an die Aktuatoren zurück. Sensoren und Aktuatoren spielen jeweils wichtige Rollen bei der Eingabe und Ausgabe. Wir wissen bereits, dass manche Agenten durch die gegenwärtige Aktion beeinflusst werden. In diesen Fällen sollten die Wahrnehmungen im Agenten gespeichert werden, um sie in der Zukunft zu erkennen.

Eine Tabelle wie die Beispieltabelle für die Staubsaugerwelt Tab. 1.1 wird am Anfang der Erstellung eines Agentenprogramms erwartet. Danach könnte der Agent die richtige Aktion in der Tabelle nachschlagen. Der tabellengesteuerte Ansatz führt aber zum Problem der Größe, da die Anzahl des Tabellenindex insgesamt $\sum_{t=1}^{T} |P|^t$ ist. Dabei steht P für die Menge möglicher Wahrnehmungen, und T für die Lebensdauer des Agenten (die Gesamtzahl der Wahrnehmungen, die er entgegennimmt) [48].

Beispiel 1.5.1 (Tabelle Taxifahrer-Agenten)**.** Betrachten wir ein Beispiel eines Taxifahrer-Agenten. Dabei nimmt eine einzelne Kamera des Agenten pro Sekunde 30 Bilder, die 640x480 Pixel und 24 Bit Farbtiefe besitzen, auf. Demnach werden 27 MB pro Sekunde eingegeben. Davon geht man aus, dass in einer Stunde in Betrieb die Tabelle mehr als $10^{250.000.000.000}$ Indexe enthält (fern von jeder Realität). Im Vergleich dazu könnte die Indextabelle für Schach weniger Einträge beinhalten, allerdings ist die Anzahl 10^{150} immer noch riesig.

Aus diesem Grund treten folgende **Probleme** auf [48]:

- Für Agenten ist nicht ausreichend physischer Speicher zur Verfügung.
- Der Agent kann die Tabelle nicht völlig lernen.
- Für die Entwickler ist zu wenig Zeit beim Erstellen der Tabelle.
- Für die Entwickler gibt es keine Anhaltspunkte beim Füllen der Tabelle.

Übung 1.5.2 (Pseudocode Agentenprogramm)**.** Schreiben Sie als Pseudocode das Agentenprogramm für den Tabellen-Agenten.

Die wichtigste Herausforderung der KI ist es, herauszufinden, wie man Programme schreibt, die rationales Verhalten so weit wie möglich aus einem vergleichsweise kleinen Programm statt aus einer riesigen Tabelle erzeugen. Im Folgenden werden grundlegende Agentenprogramme angelehnt an [48] dargestellt.

Einfache Reflexagenten

Während der einfache Reflexagent eine Aktion entscheidet, ist der Wahrnehmungsverlauf (vorherige Wahrnehmungen) nicht in seine Überlegungen einbezogen - nur die aktuelle Umgebung spielt eine Rolle.

Beispiel 1.5.3 (Staubsauger-Agent)**.** Wie beispielsweise ein Staubsauger-Agent kümmert sich der Agent nur darum, ob die Position, in der er sich aktuell befindet, sauber oder nicht ist. Wenn Schmutz entdeckt wird, dann reagiert der Agent darauf. Dies wird als *Bedingungs-Aktions-Regel* bezeichnet, wie z. B.: **if** *Status: schmutzig* **then** *Saugen*.

Die Abbildung 1.6 präsentiert die Interaktion zwischen Agenten und Umgebung. Nach Erkennung der Situation wird eine Aktion durch Bedingungs-Aktions-Regeln ausgewählt und durchgeführt.

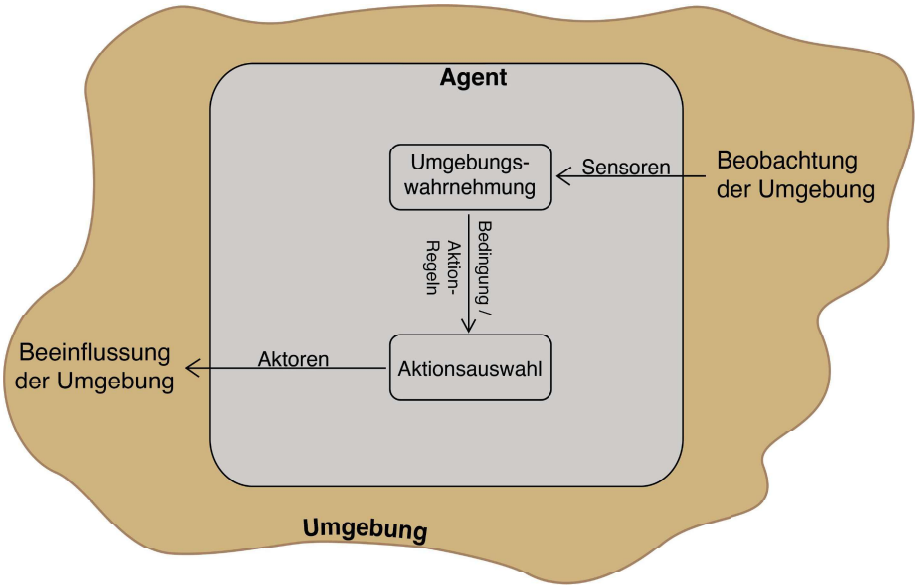


Abbildung 1.6: Schematische Darstellung eines einfachen Reflexagenten.
Eigene Darstellung von S. Krause lizenziert unter CC BY 4.0. angelehnt an [48, Abb. 29].

Wegen der *if-then* Eigenschaft sind einfache Reflexagenten einfach, allerdings ist ihre Intelligenz sehr begrenzt. Sie werden nur in vollständig beobachtbaren Umgebungen angesetzt, um Endlosschleifen zu vermeiden.

Modellbasierte Reflexagenten

Bevor eine Aktion beschlossen ist, sollte ein modellbasierter Reflexagent den Wahrnehmungsverlauf beobachten, um den eigenen *internen Zustand* zu aktualisieren. Dabei werden zwei Informationen benötigt.

1. Zunächst ist es wichtig *wie sich die Welt im Allgemeinen unabhängig vom Agenten weiterentwickelt*, wie übliche Verhaltensweisen.

2. Zweites betrifft die Aktionsfolge, *wie sich die eigenen Aktionen des Agenten auf die Welt auswirken*.

Die Kombination von obigen Kenntnissen strukturiert ein Modell der Welt, welches der Agent verwendet. Dadurch kann der Agent das Wissen, wie die Welt funktioniert, erfassen. Deswegen finden modellbasierte Reflexagenten sich in der teilweise beobachtbaren Umgebung zurecht.

Beispiel 1.5.4 (Taxifahrer als modellbasierter Reflexagent)**.** Betrachten wir das Beispiel eines Taxifahrers als modellbasierten Reflexagenten. Sieht der Agent beim Blick in den Rückspiegel ein schnell annäherndes Auto, so kann er sich dies merken, wenn es wenig später in den toten Winkel blickt. Er weiß, dass ein überholendes Auto im Allgemeinen näher ist, als es noch vor einem Moment war. Zudem benötigen wir Informationen darüber, wie sich die eigenen Aktionen des Agenten auf die Welt auswirken. Zum Beispiel, wenn der Agent das Lenkrad im Uhrzeigersinn dreht, fährt das Auto nach rechts, oder wenn man fünf Minuten lang mit 120 km/h auf der Autobahn nach Norden gefahren ist, befindet man sich normalerweise zehn Kilometer nördlich von dem Punkt, an dem man noch vor fünf Minuten war. Dieses Wissen darüber, wie die Welt funktioniert, wird als Modell der Welt bezeichnet.

Zielbasierte Agenten

Ein zielbasierter Agent beruht sowohl auf dem oben genannten Modell der Welt als auch dem entscheidenden Faktor der *Zielinformation*. Die Zielinformation enthält den erwarteten Endzustand. Die "Gedanken" des Agenten sind relativ logisch, im Vergleich zum einfachen Reflexagenten, der anhand von Ja-/Nein-Fragen reagiert. Statt Bedingung/Aktion-Regeln beeinflusst das Ziel direkt die Entscheidung der Aktion. Außerdem sollte die Frage "*Was passiert, wenn ich Aktion A ausführe*" berücksichtigt werden. Dann werden Aktionsfolgen gesucht, die eine Zielerreichung ermöglichen.

Beispiel 1.5.5 (KI-Anlageberater)**.** Eine Anlageberater-KI, welche unterschiedliche Anlageoptionen auf der Grundlage potenzieller Erträge und Risiken bewertet, ist ein zielorientierter Agent.

Nutzenbasierte Agenten

Ziele allein sind nicht ausreichend, um in den meisten Umgebungen ein hochqualitatives Verhalten zu erzeugen. Eine allgemeinere Leistungsbewertung sollte einen Vergleich verschiedener Zustände der Welt erlauben, wie nützlich sie für den Agenten sind. Deshalb

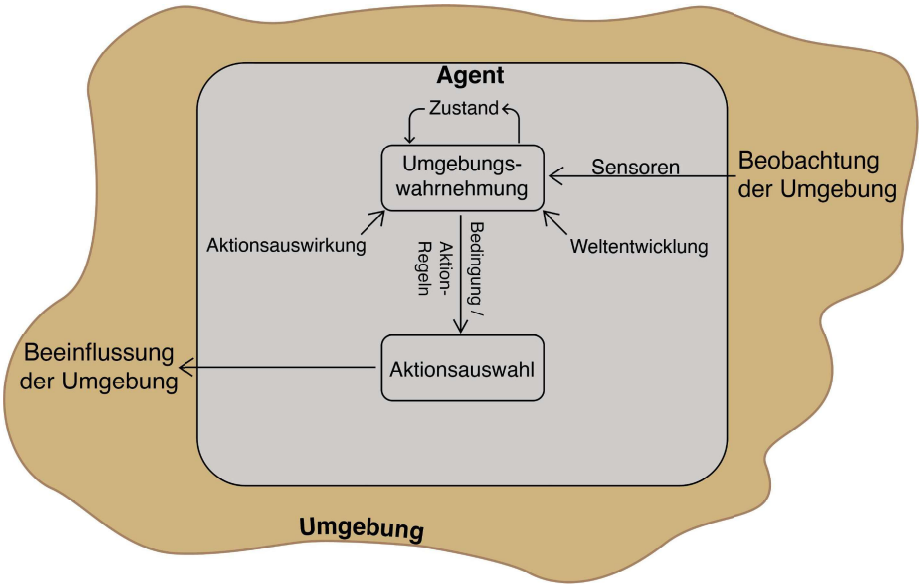


Abbildung 1.7: Schematische Darstellung eines modellbasierten Reflexagenten.
Eigene Darstellung von S. Krause lizenziert unter CC BY 4.0. angelehnt an [48, Abb. 2.11].

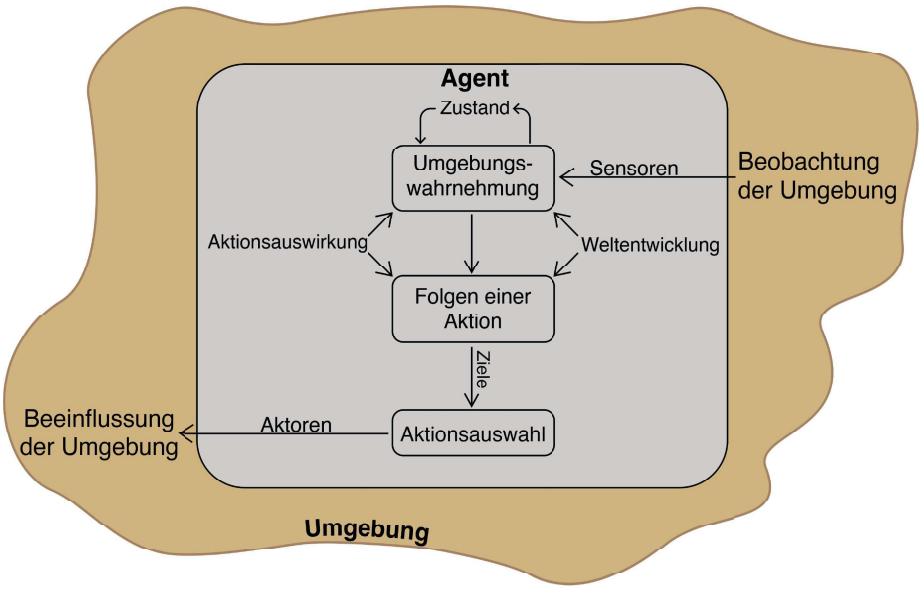


Abbildung 1.8: Schematische Darstellung eines zielbasierten Agenten.
Eigene Darstellung von S. Krause lizenziert unter CC BY 4.0. angelehnt an [48, Abb. 2.13].

ist statt einem einzelnen Entscheidungsfaktor die ausgewählte Aktion bei nutzenbasierten Agenten auf eine Übereinstimmung in der *internen Nutzenfunktion* sowie der *externen Leistungsbewertung* zu überprüfen. Beim Treffen einer Entscheidung stehen verschiedene Alternativen zur Verfügung. Die Nutzenfunktion ist eine interne Leistungsbewertung. Wenn

Ziele in manchen Fällen nicht erreicht werden können, kann ein nutzenbasierter Agent dank eigener Vorteile wie Flexibilität und Lernen trotzdem rational beurteilen. Ersten können Kompromisse in Konfliktsituationen gesucht werden, zweitens ist die Wahrscheinlichkeit eines Erfolges gegen die Wichtigkeit der Ziele abzuschätzen. Es können Optimierungsaufgaben mit nutzenbasierten Agenten gelöst werden.

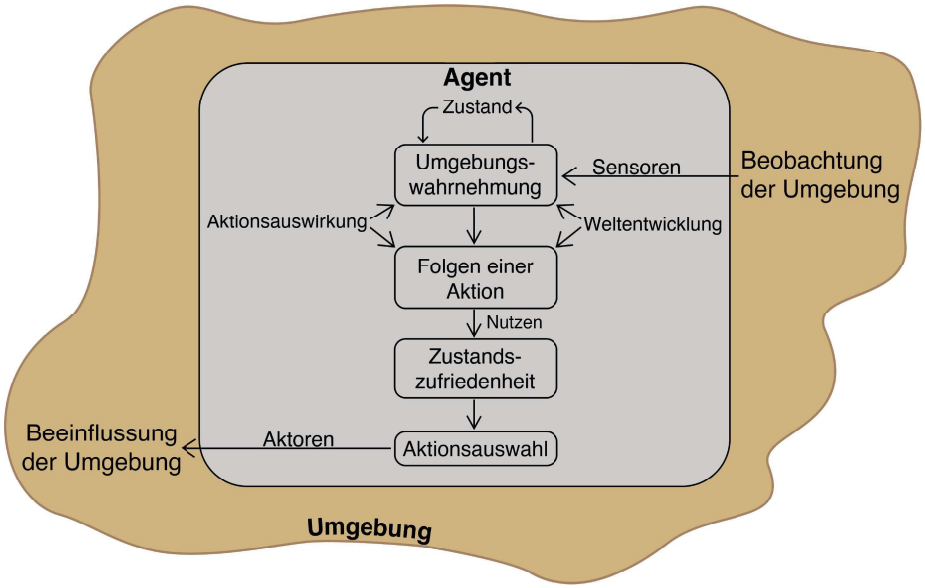


Abbildung 1.9: Schematische Darstellung eines nutzenbasierte Agenten.
Eigene Darstellung von S. Krause lizenziert unter CC BY 4.0. angelehnt an [48, Abb. 2.14].

Übung 1.5.6 (Agentenprogramm)**.** Ein einfacher Thermostat soll einen Ofen einschalten, wenn die Temperatur mindestens 4° unterhalb der Einstellung liegt, und den Ofen ausschalten, wenn die Temperatur mindestens 4° oberhalb der Einstellung liegt. Ist ein Thermostat eine Instanz eines einfachen Reflexagenten, eines nutzenbasierten Agenten oder eines zielbasierten Agenten?

Lernende Agenten

Die Lernfähigkeit soll den Agenten dazu bringen, in unbekannten Umgebungen zurechtzukommen. Es ist sogar erwartbar, sich Agenten besser entwickeln können, als wenn alles vordefiniert ist. In der Abbildung 1.10 sind wesentliche Konzepte lernender Agenten präsentiert.

- Das *Leistungselement* ist für die Auswahl externer Aktionen verantwortlich. Es ist das, was wir zuvor als den ganzen Agenten betrachtet haben: Es nimmt Wahrnehmungen auf und entscheidet, welche Aktionen ausgeführt werden.
- Das *Lernelement*, das dafür verantwortlich ist, Verbesserungen zu erzielen. Es verwendet das Feedback aus der Kritik darüber, welche Ergebnisse der Agent erzielt, und entscheidet, ob das Leistungselement abgeändert werden soll, um in Zukunft bessere Ergebnisse zu erzielen.

- Die *Kritik* teilt dem Lernelement mit, wie gut sich der Agent im Hinblick auf einen festgelegten Leistungsstandard verhält. Die Kritik ist notwendig, weil die Wahrnehmungen selbst keinen Hinweis auf den Erfolg des Agenten bieten.
- Der *Problemgenerator* ist dafür verantwortlich, Aktionen vorzuschlagen, die zu neuen und informativen Erfahrungen führen.

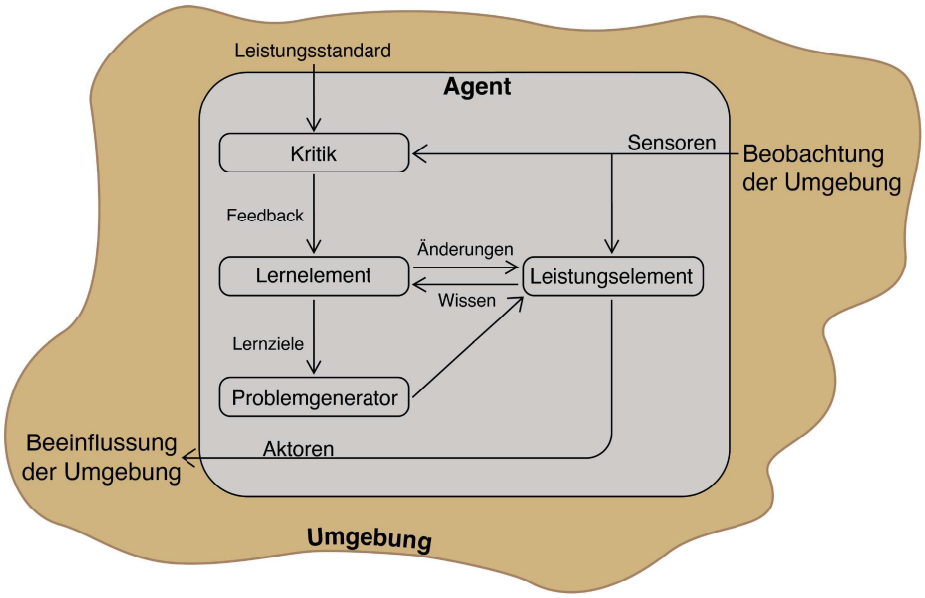


Abbildung 1.10: Schematische Darstellung eines allgemeinen lernenden Agent. Eigene Darstellung von S. Krause lizenziert unter CC BY 4.0. angelehnt an [48, Abb. 2.15].

Beispiel 1.5.7 (Maus in Labyrinth)**.** Stellen wir uns eine Maus in einem Labyrinth vor, die nach einem Weg zum Käse sucht. Wenn sie auf eine Sackgasse stößt, nimmt sie einen anderen Weg. Sie setzt diesen Prozess fort, bis sie das Labyrinth erfolgreich durchquert hat. Wenn sie dasselbe Labyrinth erneut betritt, erinnert sie sich an den richtigen Weg. In einem neuen Labyrinth verwendet sie dieselbe Methodik für die Suche und erinnert sich auch dort beim zweiten Durchlauf an die erfolgreiche Route. Die Maus verfügt somit über alle Elemente eines lernenden Systems, neben Sensoren und Aktoren: eine Nutzenfunktion, wie etwa das Finden von Futter, die eine Bewertung ihres Erfolgs vornimmt, ein Lernelement, das Erfahrungen speichert, beispielsweise erfolgreiche oder gescheiterte Routen durch ein Labyrinth, einen Innovator, der neue Wege vorschlägt, und ein Leistungselement, das die vorgeschlagenen Optionen bewertet.

Übung 1.5.8 (Spam-Filter)**.** Beschreiben Sie, wie ein Spam-Filter funktionieren muss, um ein lernender Agent zu sein?

1.6 Mathematische Beschreibung für Agenten und Umgebungen

Betrachten wir nun eine abstrakte Beschreibung von Agenten nach Wooldridge [66]:

- Wir nehmen dafür zunächst an, dass eine Umgebung sich zu jeder Zeit in irgendeinem diskreten, zeitlich klar abgegrenzten *Zustand* befindet $E = \{e, e', ...\}$.
- Es wird weiter angenommen, dass Agenten eine Menge an *möglichen Aktionen* zur Verfügung steht, mit denen sie den Zustand der Umgebung beeinflussen können: $Ac = \{\alpha, \alpha', ...\}$ ist eine (endliche) Menge an Aktionen.
- Ein *Lauf* r eines Agenten in einer Umgebung ist eine Sequenz von Umgebungszuständen und Aktionen. Die Abfolge, welche mit dem Ursprungszustand der Umwelt beginnt, gefolgt von der ersten Aktion des Agenten, woraufhin die Umwelt ihren Zustand verändert usw. : $r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_0} ... \xrightarrow{\alpha_{u-1}} e_u$.

  Dabei sei
  - R eine Menge aller möglichen endlichen Läufe (über E und Ac)
  - R^{Ac} eine Teilmenge von R, welche Läufe enthält, die mit einer Aktion enden
  - R^E eine Teilmenge von R, welche Läufe enthält, die mit einem Umgebungszustand enden
- Die Wirkung, die ein Agent auf seine Umgebung hat, wird durch die *Zustandsübergangsfunktion* repräsentiert: $\tau : R^{Ac} \rightarrow 2^E$. Eine Zustandstransformationsfunktion bildet einen Lauf, der mit einer Aktion des Agenten endet, auf einer Menge möglicher Umgebungszustände ab (die ein mögliches Ergebnis von Aktionen sein können). Diese Definition lässt also eine nichtdeterministische Umwelt zu, da diese auf eine Aktion des Agenten mit einer Reihe von möglichen Zuständen reagieren kann. Es wird angenommen, dass Umgebungen abhängig von ihrer Vergangenheit sind. Das bedeutet, der nächste Zustand einer Umgebung ist nicht nur abhängig vom aktuellen Zustand und der Aktion. Die Aktionen, welche in der Vergangenheit von dem Agenten ausgeführt wurden, haben ebenfalls Einfluss auf den aktuellen Zustand.
- Formal sagen wir, dass eine *Umgebung* Env ein Tripel ist $Env =< E, e_0, \tau >$, wobei E eine Menge von Umgebungszuständen ist, e_0 der Initialzustand und τ die Zustandstransformationsfunktion.
- Wir modellieren *Agenten* als Funktionen, die Läufe auf Aktionen abbilden:
  $Ag : R^E \rightarrow Ac$. Ein Agent entscheidet darüber, welche Aktion er ausführt, auf Basis der Vergangenheit.

Beispiel 1.6.1 (Staubsaugroboter)**.** Wir versuchen uns an einer mathematischen Beschreibung des Staubsaugeragenten:

- Zustand $E = \{$Position Feld A, Position Feld B, Feld sauber, Feld schmutzig$\}$
- Aktionen $Ac = \{$fahre nach rechts, fahre nach links, saugen$\}$

- Angenommen der Anfangszustand ist in Feld A dann gilt $e_0 =$ Position Feld A
- Angenommen der Agent führt die Aktionen $\alpha_0 =$ fahre nach rechts und $\alpha_1 =$ saugen aus. Dann ist de zugehörige Lauf:
  $r : e_0 =$ Position Feld A $\xrightarrow{\alpha_0 = \text{fahre nach rechts}} e_1 =$ Position Feld B $\xrightarrow{\alpha_1}$ $e_2 \xrightarrow{\alpha_2 = \text{saugen}} e_3 =$ Position Feld B, Feld sauber
- Zustandsübergangsfunktion τ, die auf der Aktion saugen endet, bildet auf den Zustand $e_3 =$ Position Feld B, Feld sauber ab

1.7 Architekturen

Einen guten Überblick über Roboterkontrollarchitekturen bietet [10] welcher maßgeblich als Quelle für diesen Abschnitt diente.

Reaktive Agenten

Bestimmte Arten von Agenten entscheiden ohne Beachtung der Vergangenheit, wie sie sich verhalten. Ihre Entscheidungsfindung basiert allein auf dem aktuellen Zustand. Diese Art von Agenten wird rein reaktiv (engl. purely reactive) genannt.

Beispiel 1.7.1 (Thermostat)**.** Ein Thermostat-Agent ist ein Beispiel für einen solchen rein reaktiven Agenten. Angenommen, der Umgebungszustand kann entweder zu kalt oder die gewünschte/richtige Temperatur sein. Dann kann das Thermostat wie folgt als Agent modelliert werden: $Ag(e) = \begin{cases} \text{heizen}, & \text{wenn e= Temperatur richtig} \\ \text{nicht heizen}, & \text{andernfalls} \end{cases}$

Subsumptions-Architektur

- Wurde für Echtzeitanwendungen entwickelt und ist ein schrittweiser Aufbau eines mehrstufigen Kontrollsystems.
- Die Ideen von Rodney Brooks [8] ist den Roboter mit zunehmender Kompetenz in verschiedenen sogenannten *Schichten* arbeiten zu lassen.
- Die Schichten bestehen aus asynchronen Modulen, die über Kanäle mit geringer Bandbreite kommunizieren.
- Jedes Modul ist eine Instanz einer recht einfachen Berechnungsmaschine.
- Schichten auf höherer Ebene können die Aufgaben der unteren Ebenen übernehmen, indem sie deren Ausgaben unterdrücken.
- Die Ebenen des Steuerungssystems nutzen die Eingaben der Sensoren zur Steuerung der Aktoren.
- Eine *Kompetenzstufe* ist eine informelle Spezifikation einer gewünschten Klasse von Verhaltensweisen eines Roboters für alle Umgebungen, denen er begegnet.

- Eine höhere Kompetenzstufe impliziert eine spezifischere gewünschte Klasse von Verhaltensweisen.
- Jede Kompetenzstufe enhält als Teilmenge jedes frühere Kompetenzniveau.

Folgenden Kompetenzstufen hat Brooks definiert:

0. Kontakt mit Objekten vermeiden (unabhängig davon, ob sich die Objekte bewegen oder stationär sind).
1. Ziellos umherwandern, ohne Dinge zu berühren.
2. Erkunde die Welt, indem du Orte in der Ferne siehst.
3. Erstelle eine Karte der Umgebung und plane Routen anhand dieser Karte.
4. Nimm Veränderungen in der Umgebung wahr.
5. Beschreibe Welt anhand von identifizierbaren Objekte und führen Aufgaben aus, die mit bestimmten Objekten zusammenhängen.
6. Formulierung und Ausführung von Plänen, welche die Welt auf eine gewünschte Weise verändern.
7. Führe Schlussfolgerungen über das Verhalten von Objekten in der Welt und ändere Pläne entsprechend.

Ablauf der Subsumptions-Architektur:

- Der Kerngedanke der Kompetenzebenen besteht darin, dass wir Schichten eines Kontrollsystems aufbauen können, die jeder Kompetenzebene entsprechen, und einfach eine neue Schicht zu einem bestehenden Satz hinzufügen, um die nächsthöhere Stufe der Gesamtkompetenz zu erreichen.
- Wir beginnen bei dem Aufbau eines Robotersteuerungssystems mit der Kompetenzstufe 0.
- Als Nächstes bauen wir eine weitere Steuerungsebene, die wir als Steuerungssystem der ersten Ebene bezeichnen. Sie ist in der Lage, Daten aus dem System der Ebene 0 zu untersuchen und darf auch Daten in die internen Schnittstellen der Ebene 0 einspeisen, wodurch der normale Datenfluss unterdrückt wird.
- Der gleiche Prozess wird wiederholt, um höhere Kompetenzstufen zu erreichen.

Diese Architektur ist vereinfacht in Abbildung 1.11 dargestellt.

Betrachten wir nun die Subsumptions-Architektur als Beispiel einer *verhaltensbasierten Architektur* nach [21]. Grundidee in einer verhaltensbasierten Architektur: Softwaremodule, die Verhaltenselementen des Roboters entsprechen, die sog. Verhaltensbausteine, ständig nebenläufig Sensordaten analysieren zu lassen und ihren jeweiligen Beitrag zur Roboterkontrolle zu sammeln und daraus jeweils die effektiven Kontrollkommandos zu errechnen.

Verhaltensbausteine:

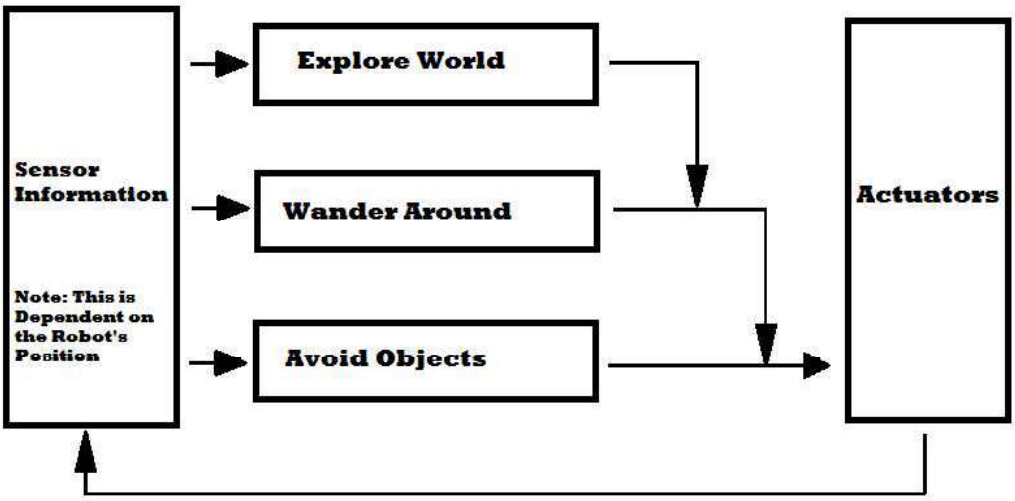


Abbildung 1.11: Abstraktes Diagramm der Subsumptions-Architektur. Von KodoKB `https://openverse.org/image/e2331c11-235c-425f-be75-0fd6787c316f?q=Subsumptions+architectur` lizenziert unter CC0 1.0.

- entsprechen Teilleistungen (Kompetenzstufen)
- entstammen bewusst unterschiedlichen Zeitskalen
- nicht erforderlich, dass jedes Modul in jedem Kontrollzyklus einen Beitrag liefert
- dürfen jeweils einen internen Zustand haben (‚sodass sie nicht direkte Abbildungen von Sensordaten in Kontrollparameter sind,) sondern ihr Gedächtnis verwenden
- rein reaktive, gedächtnislose Verhaltensbausteine sorgen für die Reaktivität der Roboterkontrolle insgesamt, z. B. Hindernisvermeidung

Die Subsumptions-Architektur hat sie einen speziellen Mechanismus definiert, mittels dessen im aktuellen Kontrollzyklus die Systemantwort aus den aktuellen Beiträgen der Verhaltensbausteine ermittelt wird: Die *Subsumption* bedeutet übertragen etwa überschreiben oder überstimmen. Das bedeutet, dass der Output eines Verhaltensbausteins den eines anderen überschreibt. Festzulegen, unter welchen Voraussetzungen ein Verhaltensbaustein einen Anderen subsumiert, ist Teil der Programmierung des Roboterkontrollsystems. Verhaltensbasierte Architekturen erlauben es, relativ einfach und transparent Roboterkontrollsysteme zu bauen, die elementare, gewissermaßen unvermeidliche Verhaltensbausteine in ein funktionierendes und reaktives Gesamtsystem integrieren.

Beispiel 1.7.2 (Sechsbeiniger Roboter)**.** Betrachten wir nun als Beispiel für die Verhaltensbausteine einen sechsbeinigen Roboter, der über unwegsames Gelände laufen und einem Menschen folgen soll. Der Roboter ist mit zwei Schnurrhaaren ausgestattet, die umliegende Hindernisse erkennen können. Stellen wir uns einen Roboter ähnlich zu diesem in Abbildung 1.12. Folgende Schichten sind hier möglich:

Abbildung 1.12: VEX IQ Six-Legged Walking Robot.
Von vexrobotics `https://openverse.org/image/f3f5ab06-d062-4e16-9c32-a459bd0ae9c8?q=VEX` lizenziert unter CC BY-ND 2.0.

- Aufstehen: Kontrolle der Schwungposition des Beins und Anheben
- Einfaches Gehen: Mit der Kombination aus lokaler beinabhängiger Mechanik und einer Maschine, die versucht, die Summe der Positionen aller Beine global zu koordinieren, kann der Roboter laufen.
- Kraftausgleich: Kraftsensoren liefern Informationen über das Bodenprofil
- Beinheben: Schritt über Hindernisse
- Hindernisse vermeiden: Um auf Hindernisse besser reagieren zu können und nicht darauf zu warten das ein Hindernis direkt gegen die Beine rammt können Schnurrhaare verwendet werden um frühzeitig zu reagieren.
- Neigungsstabilisierung: Das der Roboter nicht absackt in besonders unebenen Gelände soll die Neigung der Roboters beachtet werden.
- Streifzüge: Der Roboter geht nur dann, wenn sich in der Nähe etwas bewegt. Der Roboter sitzt also still, bis z. B. eine Person vorbeigeht, und dann bewegt er sich ein wenig vorwärts.
- Gesteuertes Herumstreifen: Der Roboter nimmt die vorherrschende Richtung zur Kenntnis und kann einem sich bewegendem Objekt wie z. B. einem Menschen folgen.

Übung 1.7.3 (Mars-Roboter)**.** Formulieren Sie für den Mars-Roboter aus Abb. 1.4 welche Schichten eine Subsumptionsarchitektur enthalten müsste, wenn die Aufgabe des Roboters ist, an verschiedenen Stellen Gesteinsproben zu entnehmen. Was ändert sich, wenn radioaktive Gesteinsproben aufgespürt werden können?

Vor- und Nachteile der Subsumptions-Architektur:

- **Vorteile**: Die Architektur ist einfach, wirtschaftlich, effizient und robust.
- **Nachteile**: Je mehr Schichten, desto komplizierter ist es zu verstehen, was vor sich geht.

Übung 1.7.4 (Reaktiver Agent mit Subsumptionshierarchie)**.** Geben Sie zu den folgenden Agenten (i) eine PEAS-Beschreibung, (ii) eine Klassifizierung nach Umwelteigenschaften und (iii) eine Realisierung als reaktiver Agent mit Subsumptionshierarchie an.

- Webcrawler, d. h. ein Computerprogramm, das automatisch das World Wide Web durchsucht und Webseiten analysiert (vor allem von Suchmaschinen eingesetzt);
- Fußball-Roboter, der einen Ball verfolgt und ins Tor kickt, wenn er nahe genug am Ball ist;
- Spieler-Agent, der Texas Hold'em Poker gegen andere Agenten spielen kann mit Berücksichtigung des zur Verfügung stehenden Kapitals.

Zur besseren Skalierbarkeit kann *Schwarmintelligenz* helfen [32]. Zum Beispiel könnte ein Schwarm von Robotern in Szenarien wie der gemeinsamen Suche dazu beitragen, einen größeren Bereich abzudecken als ein einzelner Roboter. Das kollektive Verhalten im Schwarm basiert im Wesentlichen auf drei Komponenten:

- Lokale Kommunikation und Interaktion zwischen den Individuen
- Regeln, die für alle Individuen gleich sind
- Entwicklung einer Führung, aus dem Schwarm heraus

Auch wenn ein Anführer nicht explizit festgelegt ist und die Individuen nach denselben Regeln programmiert werden, übernimmt ein bestimmtes Individuum die Rolle des Anführers. Die Rolle eines temporären Anführers ist ein wesentliches Element zur Bewältigung einer festgelegten Aufgabe. Beispiele für solche Führung können in der Natur oft beobachtet werden: In einem Schwarm von Vögeln, die nach Nahrung suchen, übernimmt ein Vogel an der Spitze die Rolle des Anführers.

Beispiel 1.7.5 (Goldsuche)**.** Nehmen wir an, dass ein entfernter Planet Gold enthält. Proben sollen zu einem Raumschiff gebracht werden, das auf dem Planeten landet. Es ist nicht bekannt, wo sich das Gold befindet. Mehrere autonome Fahrzeuge sind verfügbar und können verschiedene Bereiche des Planeten absuchen.

Hybride Roboterkontrollarchitekturen

Die Abbildung 1.13 zeigt die grobe Struktur von hybriden Roboterkontrollarchitekturen:

- Die *Handlungsplanung* arbeitet auf hoher, strategischer Granularitätsstufe in langen Zeitzyklen.
- Die *reaktive Aktionsüberwachung* enthält die Verhaltensbausteine auf operativer Stufe, die in schnellen Zeitzyklen die physische Roboteraktion anstoßen und überwachen.

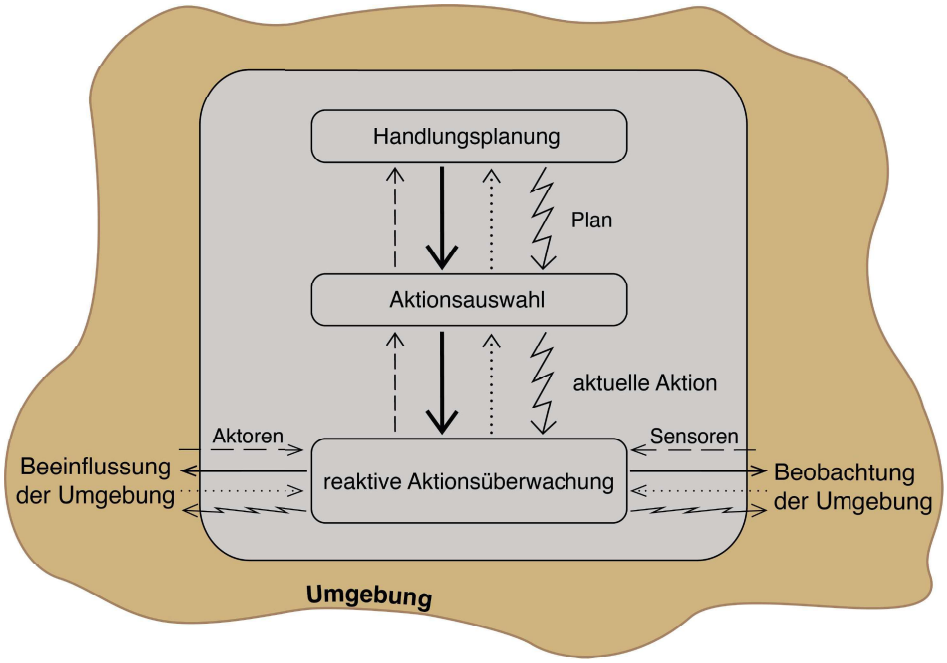


Abbildung 1.13: Schematische Darstellung der drei Schichten Architektur. Die unterschiedlichen Pfeile sollen die verschiedenen Datenaustauschformen und -volumen darstellen.
Eigene Darstellung von S. Krause lizenziert unter CC BY 4.0. angelehnt an [41].

- Dazwischen vermittelt eine taktische Stufe, die jeweils nächste Aktion aus dem aktuellen Plan auszusuchen, gegebenenfalls passend zu instanziieren und auf die Ebene der Verhaltensbausteine zu zerlegen.
- In der Gegenrichtung muss sie die Rückmeldungen von der Aktionsüberwachung interpretieren, entscheiden, ob eine Aktion erfolgreich oder erfolglos abgeschlossen ist, weiterlaufen muss, möglicherweise variiert werden muss, oder ob die Handlungsplanung einen neuen Plan erstellen und dafür mit den aktuellen Umgebungssensordaten versorgt werden muss.
- Die Vielzahl von Pfeilen zwischen denselben Kästen soll andeuten, dass hier nicht nur potenziell große Datenvolumina hin und her fließen, sondern dass die Daten jeweils von unterschiedlichen Arten und Granularität sind.

Übung 1.7.6 (Vorteile hybrider Roboterkontrollarchitekturen)**.** Diskutieren Sie, welche Vorteile hybride Roboterkontrollarchitekturen bieten?

BDI-Agenten

In den letzten 40 Jahren hat das BDI-Agentenmodell, das auf mentalen Einstellungen von Überzeugungen basiert, die Grundlage für einen großen Teil der Forschung über Architekturen für autonome Agenten gebidet [13]. Beginnend mit der philosophischen Arbeit

von Bratman 1987 [7], und Implementierungen wie das *Procedural Reasoning System* von Georgeff und Lansky 1987 [25].

Die BDI-Architektur stützt sich auf praktische Überlegungen im Sinne von Bratmans philosophischer Betonung der intentionalen Haltung [7]. Praktisches Denken ist ein Denken, das auf Handlungen abzielt - der Prozess des herauszufinden was zu tun ist. Dieser Prozess unterscheidet sich vom theoretischen Denken: es wird Wissen abgeleitet oder Schlussfolgerungen erlangt, indem die eigenen Überzeugungen und das eigene vorhandene Wissen genutzt werden. Menschliches praktisches Denken umfasst zwei Aktivitäten: Überlegungen und Schlussfolgern [10].

- Bei der Überlegung wird entschieden, welcher Zustand erreicht werden soll.
- Das Schlussfolgern entscheidet darüber, wie dieser Zustand erreicht werden soll.

  Handeln ist nicht nur Reaktion auf Reize. Agenten gehen nicht nur unmittelbaren Bedürfnissen nach, sondern verfolgen langfristigere Absichten (Ziele), ausgehend vom eigenen Wissen [7].

In der BDI-Architektur besteht der Agent aus drei logischen Komponenten, die als mentale Zustände/ Einstellungen bezeichnet werden:

- Überzeugungen(**B**elief) sind die Informationen, die ein Agent über die Welt hat.
- Wünsche (**D**esire) sind die Motivation des Agenten oder mögliche Optionen zur Ausführung von Handlungen.
- Absichten (**I**ntention) sind die Verpflichtungen des Akteurs gegenüber seinen Wünschen und Überzeugungen.

Intentionen sind eine Schlüsselkomponente in der praktischen Argumentation. Sie beschreiben Zustände, zu deren Herbeiführung sich der Handelnde verpflichtet hat. Sie sind handlungsauslösend. Die Bildung von Absichten ist entscheidend für den Erfolg eines Akteurs. [10]

Vor- und Nachteile der BDI-Architektur nach [10]:

- **Vorteile**: Die Architektur ist klar und intuitiv gestaltet.
- **Nachteile**: Hört der Agent nicht auf zu überdenken, könnte er versuchen, eine Absicht zu erreichen, die nicht erreichbar ist oder nicht mehr gültig ist. Möglich ist das Agent seine Ziele nicht erreicht, weil er nicht genügend Zeit für die Bearbeitung der Aufgabe hat.

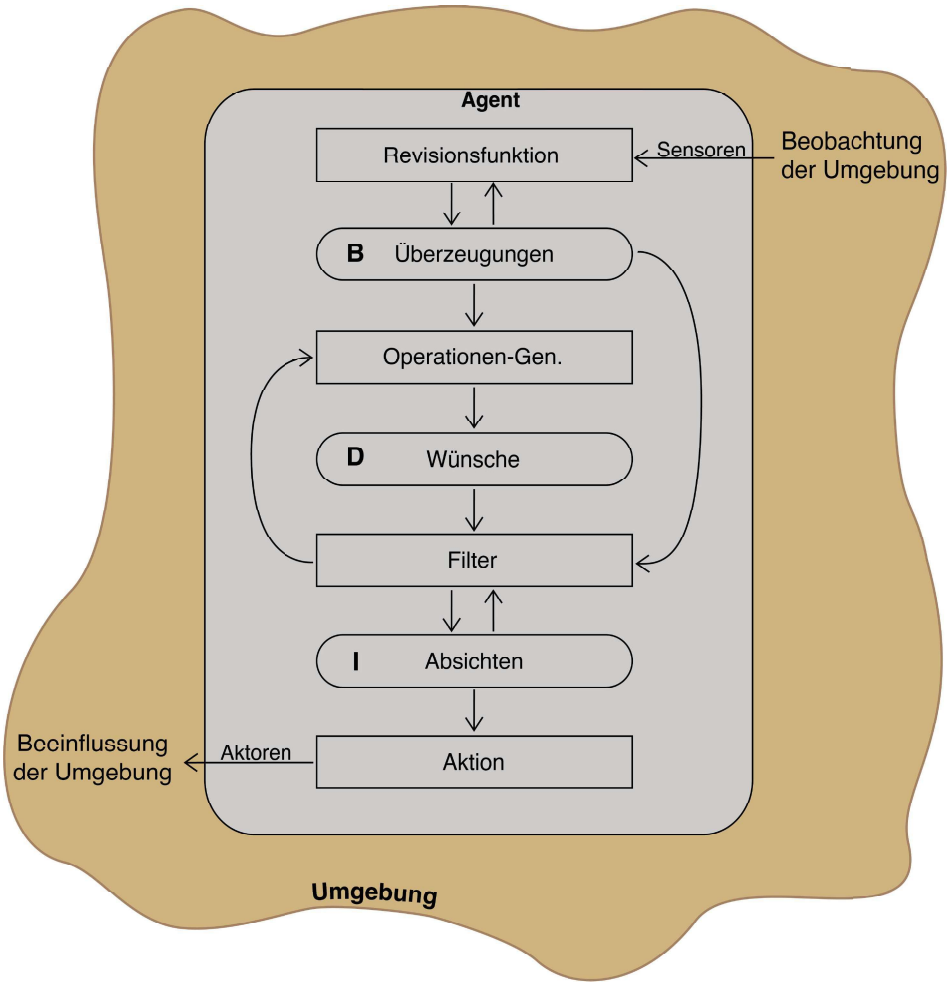


Abbildung 1.14: Schematische Darstellung der BDI-Architektur.
Eigene Darstellung von S. Krause lizenziert unter CC BY 4.0. angelehnt an [41].

Lizenzinformationen

Literaturverzeichnis

[1] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[2] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: a survey. *Artificial Intelligence Research and Development*, pages 363–371, 2008.

[3] Christoph Beierle and Gabriele Kern-Isberner. *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen*, pages 384–410. Springer Fachmedien Wiesbaden, Wiesbaden, 2014.

[4] Mordechai Ben-Ari and Francesco Mondada. *Elements of robotics*. Springer Nature, 2017.

[5] Margrit Betke and Leonid Gurvits. Mobile robot localization using landmarks. *IEEE transactions on robotics and automation*, 13(2):251–263, 1997.

[6] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[7] M Bratman. Intention, plans, and practical reason, harvard un. *Cambridge, MA*, 1987.

[8] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986.

[9] A. Chayeb, N. Ouadah, Z. Tobal, M. Lakrouf, and O. Azouaoui. Hog based multi-object detection for urban navigation. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2962–2967, 2014.

[10] Kim On Chin, Kim Soon Gan, Rayner Alfred, Patricia Anthony, and Dickson Lukose. Agent architecture: An overviews. *Transactions on science and technology*, 1(1):18–35, 2014.

[11] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[12] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[13] Lavindra De Silva, Felipe Rech Meneguzzi, and Brian Logan. Bdi agent architectures: A survey. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI), 2020, Japão.*, 2020.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[15] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[16] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[17] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[18] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[20] Jacob E Goodman and Richard Pollack. Allowable sequences and order types in discrete and computational geometry. In *New trends in discrete and computational geometry*, pages 103–134. Springer, 1993.

[21] Joachim Hertzberg, Kai Lingemann, and Andreas Nüchter. *Mobile Roboter: Eine Einführung aus Sicht der Informatik*. Springer-Verlag, 2012.

[22] Julia Kleeberger, Natalia Prost, and Helena Sternkopf. Machine learning. intelligente maschinen. *Medien in die Schule (Gemeinschaftsprojekt von FSM und Google Deutschland)*, 1. Auflage, 2019.

[23] Stefanie Krause. Klassifizierung von personen und detektion von fahrrädern auf bildern. Master's thesis, 2022.

[24] Rudolf Kruse, Christian Borgelt, Frank Klawonn, Christian Moewes, Georg Ruß, Matthias Steinbrecher, et al. *Computational intelligence*. Springer, 2011.

[25] Amy Lansky. *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop: June 30-July 2, 1986, Timberline, Oregon*. Morgan Kaufmann Publishers Inc., 1987.

[26] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[27] Stephen C Levinson. Frames of reference and molyneux's question: Crosslinguistic evidence. *Language and space*, 109:169, 1996.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[31] Till Miller. *Mastering Reinforcement Learning*. The University of Queensland, 2024.

[32] Sanaz Mostaghim and Sebastian Mai. Kooperation mittels schwarmintelligenz. *Zusammenwirken von natürlicher und künstlicher Intelligenz*, pages 55–69, 2021.

[33] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. Deep reinforcement learning: an overview. In *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 2*, pages 426–440. Springer, 2018.

[34] Justin Munafo, Michael G Wade, Nick Stergiou, and Thomas A Stoffregen. The rim and the ancient mariner: The nautical horizon affects postural sway in older adults. *PloS one*, 11(12):e0166900, 2016.

[35] Richard Murch and Tony Johnson. *Intelligent software agents*. prentice Hall PTR, 1998.

[36] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006.

[37] Peter Norvig and Stuart Russel. *Artificial Intelligence – A modern approach*. Prentice Hall, Englewood Cliffs, NJ, 3rd edition, 2010.

[38] Edgar Osuna, Robert Freund, and Federico Girosit. Training support vector machines: an application to face detection. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 130–136. IEEE, 1997.

[39] Gerhard Paaß, Dirk Hecker, Gerhard Paaß, and Dirk Hecker. Was kann künstliche intelligenz? *Künstliche Intelligenz: Was steckt hinter der Technologie der Zukunft?*, pages 15–43, 2020.

[40] Franz Pauer and Florian Stampfer. Was ist ein skalarprodukt und wozu wird es verwendet? *Schriftenreihe zur Didaktik der Österreichischen Mathematischen Gesellschaft*, 49:100–109, 2016.

[41] R Peter Bonasso, R James Firby, Erann Gat, David Kortenkamp, David P Miller, and Mark G Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3):237–256, 1997.

[42] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[43] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. *An introduction to machine learning*. Springer, 2019.

[44] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[46] Raul Rojas and Alexander Gloye Förster. Holonomic control of a robot with an omnidirectional drive. *KI-Künstliche Intelligenz*, 20(2):12–17, 2006.

[47] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach prentice-hall, inc. *A Simon and Schuster Company, Englewood Cliffs, New Jersey*, 7632, 1995.

[48] Stuart Russell, Peter Norvig, and Frank Kirchner. Künstliche intelligenz: Ein moderner ansatz. informatik, 2012.

[49] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive Computation and Machine Learning series, 2018.

[50] Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia computer science*, 132:377–384, 2018.

[51] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*, volume 200. Springer, 2008.

[52] David Silver. Lectures on reinforcement learning. url: `https://www.davidsilver.uk/teaching/`, 2015.

[53] Thomas Smits and Melvin Wevers. The visual digital turn: Using neural networks to study historical images. *Digital Scholarship in the Humanities*, 35, 01 2018.

[54] Frieder Stolzenburg. Localization, exploration, and navigation based on qualitative angle information. *Spatial Cognition & Computation*, 10(1):28–52, 2010.

[55] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[56] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[57] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[58] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. A deeper look at dataset bias. In *Domain adaptation in computer vision applications*, pages 37–55. Springer, 2017.

[59] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[60] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[61] Markus Vincze, Michael Zillich, and Johann Prankl. Roboter lernen mit gegenständen umzugehen: neue entwicklungen und chancen. *e & i Elektrotechnik und Informationstechnik*, 134(6):304–311, 2017.

[62] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[63] Paul Viola, Michael Jones, et al. Robust real-time object detection. *International journal of computer vision*, 4(34-47):4, 2001.

[64] Wei Wang, Yujing Yang, Xin Wang, Weizheng Wang, and Ji Li. Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering*, 58(4):040901, 2019.

[65] H Durrant Whyte. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *Robotics and Automation Magazine*, 2006.

[66] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[67] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995.

[68] Wai K Yeap and Margaret E Jefferies. On early cognitive mapping. *Spatial cognition and computation*, 2:85–116, 2000.

[69] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

[70] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.