Sonderdruck aus OBJEKTspektrum

Eine Referenzarchitektur für digitale Agenten

Gestaltung von Software autonomer Systeme

Wie bezeichnet man Zwillinge, deren Geburten typischerweise einige Monate auseinander liegen? Digitale Zwillinge! Ein digitales Abbild eines physischen Produkts entsteht heute meist einige Zeit, bevor das physische „Original" hergestellt wird, es kann aber auch erst einige Jahre, nachdem ein Produkt in Betrieb genommen wurde, diesem noch als „Upgrade" hinzugefügt werden. Allein schon daran sieht man, dass der Modebegriff des digitalen Zwillings etwas unglücklich ist. Dieser Artikel konzentriert sich deshalb auf die eine Hälfte eines solchen Zwillingspaars: die digitale Ergänzung eines Gegenstands, um dessen Fähigkeiten bis hin zur Autonomie erweitern zu können – ein sogenannter „digitaler Agent". Der Artikel stellt dazu eine Referenzarchitektur vor, die sich als kritischer Benchmark für die Funktionalität digitaler Agenten jeglicher Art eignet.

Im November 2019 ging ein Video aus Neuseeland durch die Medien [HEISE19], in dem aus der Perspektive der Fahrzeugkameras zu sehen ist, wie ein Tesla Model 3 bei einer Geschwindigkeit von rund 100 km/h innerhalb von Sekundenbruchteilen selbstständig auf die Gegenfahrbahn ausweicht, um eine Entenfamilie nicht zu überfahren.

Was ging da vor? Das Fahrzeug hatte den Auftrag, einen am Navigationssystem eingegebenen Zielort zu erreichen und dabei möglichst „nicht von der Straße zu fallen"[1]. Seine Kameras und die dahinter liegenden neuronalen Netzwerke sind dabei zum Schluss gekommen, dass sich vor ihnen ein Hindernis befindet, und haben dies als „Fußgänger" klassifiziert. Da man Fußgänger um*fahren* und nicht *um*fahren sollte, hat das Auto selbstständig entschieden, ein Ausweichmanöver durchzuführen. Wir wissen zwar nicht, was passiert wäre, wenn in genau dieser Sekunde Gegenverkehr geherrscht hätte, aber wir hoffen doch, dass in diesem Fall anstelle des Ausweichmanövers eine andere Aktion durchgeführt worden wäre …

Dieses Beispiel zeigt, wie heute technische Maschinen mittels Software angereichert werden, um immer autonomer agieren zu können, und dass sich dabei wichtige Fragen stellen, die seriös zu klären sind. Wenn solche Maschinen einen hohen oder sehr hohen Automatisierungsgrad aufweisen, möchte ich die dazu notwendige Software als digitalen Agenten bezeichnen:

> *Ein digitaler Agent ist ein Softwaresystem, welches Aufgaben von (und für) Menschen unterstützt oder sogar vollständig übernimmt.*

Ein digitaler Agent muss also Stimuli und Daten aus seiner Umwelt aufnehmen, verarbeiten und in geeigneter Form wie-

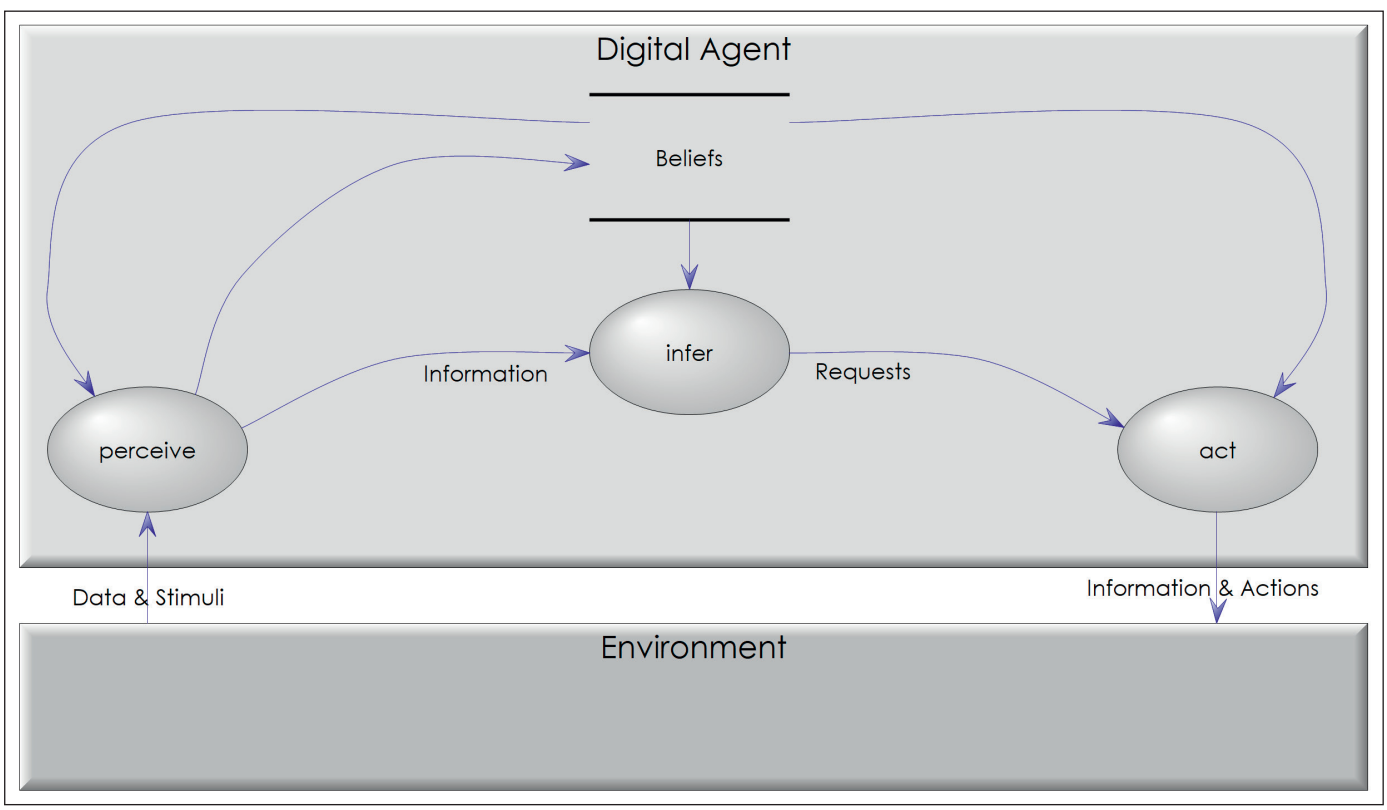

Abb. 1: Die Minimalarchitektur eines digitalen Agenten

[1]) Diese Formulierung stammt von einem Kollegen, der aufgrund seines Fahrstils etwas Mühe hatte, sein Auto auf der Straße zu halten …

der abgeben. Die obige Definition deckt damit fast alle kommerziell eingesetzten Softwaresysteme ab!
Basierend auf den in [Sch15] und [Sch18] eingeführten Ideen möchte ich im Folgenden eine Referenzarchitektur aus einer Reihe von Fähigkeiten entwickeln, die ein digitaler Agent typischerweise in mehr oder weniger ausgeprägter Form aufweist und die aufzeigt, wie diese Fähigkeiten zueinander in Beziehung stehen. Diese Referenzarchitektur lässt sich nutzen, um einen neuen digitalen Agenten zu konzipieren, aber auch um eine bestehende Softwarelösung kritisch zu hinterfragen.
Um die vorgestellte Referenzarchitektur etwas plastischer zu machen, verwende ich in diesem Artikel drei Vertreter von Softwaresystemen, an denen sich diese Fähigkeiten gut illustrieren lassen:

- *Spielcomputer:* Schachcomputer oder Computer wie AlphaGo [DM], die Brettspiele wie Schach oder Go gegen Menschen (und andere Computer) spielen können.
- *MES:* Ein „Manufacturing Execution System“, welches den Maschinenpark eines industriellen Produktionsbetriebs so weit wie möglich steuert.
- *Moderner PKW:* Ein moderner Personenwagen, welcher den Fahrer mit vielen Assistenzsystemen unterstützt.

PIA: Die Grundlage

Wir starten mit Fähigkeiten, die mindestens so alt sind wie die Begriffe „EDV“ oder gar „Elektronengehirn“: EVA für Eingabe/Verarbeitung/Ausgabe oder auch in Englisch IPO für Input/Processing/Output. Eine Software, die diese drei Kernfähigkeiten nicht in irgendeiner Form aufweist, ist kein digitaler Agent. Ich möchte diese drei Kernfähigkeiten (**siehe Abbildung 1**) im Kontext dieser Referenzarchitektur allerdings etwas allgemeiner bezeichnen:

- *Perceive* (wahrnehmen),
- *Infer* (ableiten) und
- *Act* (handeln).

Im Folgenden sind die einzelnen Fähigkeiten nun genauer beschrieben.

P: Perceive (Wahrnehmen)
Die Fähigkeit *Perceive* hat den Zweck, aus Daten (z. B. physikalischen Messgrößen) und Ereignissen im Umfeld des digitalen Agenten Informationen (semantische Bedeutungen) zu erzeugen. Dazu sind verschiedene „Sinne“ üblich: Schalter, Tasten, Sensoren, Mikrofone, Kameras, Berührung usw., aber auch verschiedene „Wahrnehmungsarten“ wie die Unterscheidung zwischen zeitdiskreten Daten (Ereignissen), die im Push-Verfahren entgegengenommen werden, und kontinuierlichen Daten (Messungen), die im Pull-Verfahren gelesen werden. In *Perceive* sind schließlich auch verschiedene Vorverarbeitungen enthalten, wie Scanning, Filterung (z. B. Entprellung), Plausibilisierung, Sensor-Fusion usw.

Beispiele:
- *Spielcomputer:* Ein Spielcomputer muss die Spielzüge seines Gegners wahrnehmen – entweder durch explizite Eingabe oder durch die Überwachung des Spielfelds mittels Sensoren.
- *MES:* Ein MES muss nicht nur Aufträge aus der Produktionsplanung entgegennehmen, sondern auch jederzeit über den aktuellen Zustand seiner Maschinen informiert sein.
- *Moderner PKW:* Ein PKW muss die Impulsfrequenz eines radialen Inkrementalgebers an seinen Rädern messen, um ihre Drehzahl bestimmen zu können. Ein autonom fahrender PKW muss aber auch mittels Ultraschall-, Radar-, gegebenenfalls Lidar-Sensoren sowie Kameras Hindernisse in seinem Umfeld wahrnehmen.

I: Infer (Ableiten)
Die Fähigkeit *Infer* hat den Zweck, aus Informationen unter Anwendung von vorhandenem Vorwissen neue Informationen zu erzeugen (Inferenz). Insbesondere erlaubt dies die Bestimmung von nicht direkt beobachtbaren Zuständen im Umfeld des digitalen Agenten oder Aussagen über dessen Zukunft. Das dabei zur Anwendung gelangende vorhandene Vorwissen kann in ganz unterschiedlichen Formen vorliegen: Formeln, Zustandsmaschinen, Algorithmen, Regeln, Muster usw. Auch für die Verarbeitung dieses Vorwissens gibt es verschiedene Verarbeitungsverfahren: von der Algebra, über die Logik, Statistik, bis hin zur Intuition.

Beispiele:
- *Spielcomputer:* Ein Spielcomputer muss bestimmen können, ob die von seinem Gegner ausgeführten Spielzüge überhaupt regelkonform sind.
- *MES:* Ein MES muss Rohdaten aufbereiten und in eine produktionstechnisch nutzbare Form bringen (aktueller Produktionsfortschritt, Fehlerrate, Maschinenauslastung usw.).
- *Moderner PKW:* Ein PKW muss aus der Drehzahl seiner Räder unter Berücksichtigung der Reifengröße die Geschwindigkeit des Fahrzeugs berechnen und allenfalls Schlupf detektieren. Er muss aber auch aus den Signalen seiner Ultraschall-, Radar-, gegebenenfalls Lidar-Sensoren sowie Kameras Objekte bilden und diese aufgrund ihrer Form, Größe und Bewegungsmuster unter Berücksichtigung vergangener Erfahrungen in verschiedene Typen (Straßenrand, Fußgänger, Gegenverkehr usw.) klassifizieren.

A: Act
Die Fähigkeit *Act* dient dazu, die durch *Infer* abgeleiteten Informationen in Form von Aufträgen zu verwenden, um deren Umfeld zu beeinflussen. Diese Beeinflussung kann durch Ausgabe von Informationen, aber auch durch Betätigung eines Aktuators erfolgen. Zudem kann sie zeitdiskret (z. B. vorübergehende Anzeige einer Information oder kurzfristige Ansteuerung eines Aktuators) oder kontinuierlich (ständige Anzeige einer Information oder Anwendung einer physikalischen Kraft über einen Aktuator) sein. Dies heißt auch, dass diese Fähigkeit unter Umständen komplexe Funktionalitäten wie Regelungen beinhalten kann.

Beispiele:
- *Spielcomputer:* Ein Spielcomputer muss seine eigenen Züge zumindest seinem Gegner mitteilen oder sogar selbst mechanisch Steine setzen können.
- *MES:* Ein MES muss nicht nur Aufträge an die zu steuernden Maschinen übertragen, sondern auch die für diese Aufträge erforderlichen spezifischen Maschinenkonfigurationen.
- *Moderner PKW:* Ein Fahrassistenzsystem muss in der Lage sein, nicht nur den Energiefluss zum Antriebsmotor, sondern auch die Bremsen und die Lenkung des Fahrzeugs beeinflussen können. Das Komfortsystem muss auch zudem die Klappen und Lüfter zur Steuerung des Luft- und Wärmeflusses in die Kabine betätigen.

PIDA: Entscheidungen treffen

Ein wichtiger Schritt in Richtung intelligenter Systeme ist die Aufspaltung der bisher eingeführten Inferenz-Fähigkeit in das Generieren mehrerer möglicher Folgerungen (eine Art Vorstellungskraft, daher *Imagine*) und dem Entscheiden für eine der erzeugten Folgerungen (*Decide*) (**siehe Abbildung 2**). Dies wird immer dann erforderlich, wenn die Herleitung höherer Informationen unterspezifiziert ist und sich dadurch mehrere Alternativen ergeben [Sch15]. Dazu kommt oft

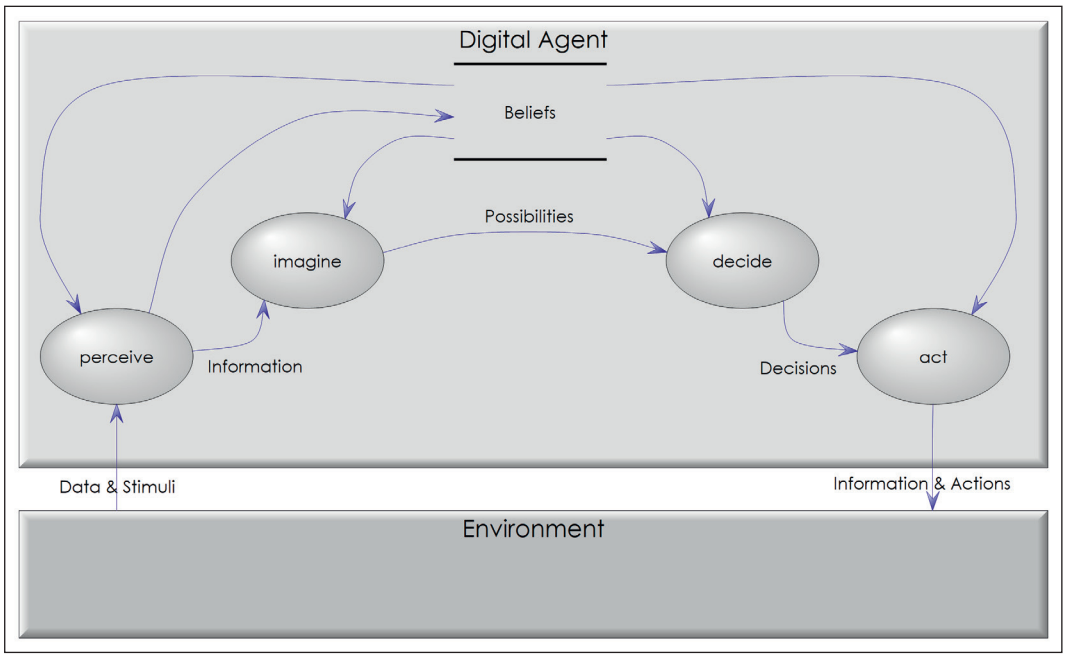

Abb. 2: Die PIDA-Architektur eines digitalen Agenten

eine Variante des „Generate & Test"-Algorithmus zur Anwendung [AI]. Die daraus entstehenden Systeme sind so in der Lage, für bestimmte Problemstellungen eine möglichst optimale Lösung zu finden.

I: Imagine
Die Fähigkeit *Imagine* generiert aus den aus *Perceive* stammenden Informationen aufgrund unvollständiger Informationen oder Wissens mehrere hypothetische Schlüsse. Diese generierten Schlüsse können sich gegenseitig ausschließen, aber auch mit unterschiedlichen qualitativen Eigenschaften oder Wahrscheinlichkeiten versehen sein.

Beispiele:
- *Spielcomputer:* In jeder Spielsituation muss ein Spielcomputer eine Vielzahl möglicher, das heißt regelkonformer Spielzüge von sich selber, aber auch von seinem Gegner als Hypothesen erzeugen können.
- *MES:* Je nach aktueller Verfügbarkeit von Maschinen muss ein MES in der Lage sein, verschiedene alternative Belegungspläne für einen komplexen Fertigungsauftrag zu generieren.
- *Moderner PKW:* Die neuronalen Netze des Rechners für das autonome Fahren können mehrere mögliche Interpretationen eines Kamerabilds, aber auch mehrere möglichen Trajektorien der identifizierten Objekte mit unterschiedlichen Konfidenzen liefern. Ein Navigationssystem muss mehrere mögliche Routen zu einem vorgegebenen Ziel berechnen können.

D: Decide
Decide ist die Fähigkeit, verschiedene Möglichkeiten zu bewerten und die „geeignetste" Option auszuwählen. Dabei werden sowohl die Nützlichkeit einer Alternative (Utility) als auch Risikoüberlegungen berücksichtigt, um den Nutzen zu maximieren und möglichen Schaden zu minimieren.

Beispiele:
- *Spielcomputer:* Aus einer großen Menge möglicher Spielzüge wählt ein Spielcomputer aufgrund verschiedener Heuristiken denjenigen aus, der ihm am wahrscheinlichsten zu einer besseren Konstellation verhilft.
- *MES:* Ein MES muss den Belegungsplan mit der optimalsten Auslastung seines Maschinenparks sowie der termintreusten Ausführung eines komplexen Fertigungsauftrags auswählen.
- *Moderner PKW:* Aufgrund der ermittelten Konfidenz der Objekt- und Trajektorien-Interpretation sowie allfälliger Plausibilitätsprüfungen bestimmt der Bordrechner die wahrscheinlichste Interpretation eines Kamerabilds. Das Navigationssystem schlägt einem Autofahrer diejenige Route vor, die unter Berücksichtigung der aktuellen Verkehrslage sowie der Fahrerpräferenzen die zeitlich oder distanzmäßig kürzeste ist.

PIDAG: Lernen

Bis etwa Ende der 1990er Jahre waren PIDA-basierte IT-Systeme die große Mehrheit und selbst der Hauptfokus der KI-Forschung. Mit den ersten deutlichen Erfolgen von Deep-Learning-Systemen kam um die Jahrtausendwende eine weitere Fähigkeit hinzu: das Lernen aus großen Mengen von Beispielen. Wahrgenommene Informationen können als Beobachtungen für späteren Gebrauch gespeichert werden. Können in diesen spezifischen Beobachtungen generelle Zusammenhänge (Muster) gefunden werden, so können diese viele spezifische Beobachtungen ersetzen.

G: Generalize
Bei der Fähigkeit *Generalize* geht es darum, aus den Korrelationen zwischen spezifischen Beobachtungen allgemeingültige Zusammenhänge zu identifizieren, um damit

- große Mengen dieser spezifischen Beobachtungen durch diese allgemeinen Zusammenhänge ersetzen zu können (Kompression),
- auf der Basis spezifischer Beobachtungen sowie allgemeiner Zusammenhänge neue Beobachtungen mit großer Wahrscheinlichkeit vorauszusagen (Induktion),
- auf der Basis spezifischer Beobachtungen sowie allgemeiner Zusammenhänge spezifische Ursachen mit großer Wahrscheinlichkeit zu vermuten (Abduktion),
- spezifische Beobachtungen gegenüber den bekannten generellen Zusammenhängen zu vergleichen (Plausibilisierung).

Beispiele:
- *Spielcomputer:* Ein Spielcomputer kann durch die Analyse vergangener Spiele typisches Verhalten seiner Gegner erkennen und daraus Heuristiken generieren, mit denen diese Gegner zu bezwingen sind.
- *MES:* Trends und Verschiebungen im Verlauf von Konfigurationsparametern können durch ein MES mit Maschinen- oder Produktausfällen korreliert werden, um daraus typische Muster zur Früherkennung zu generieren.
- *Moderner PKW:* Wenn der Besitzer eines PKW jeden Morgen zwischen 7:55 und 8:10 (ausgenommen am Wochenende) zu seinem Arbeitsplatz in München fährt, so kann das Naviga-

tionssystem schon einmal die optimale Route aufgrund der aktuellen Verkehrslage und des Wetters bestimmen und der Wagen kann im Winter pünktlich auf die geplante Abfahrzeit aufgewärmt werden.

In Kombination mit generativen Mechanismen (siehe *Imagine* oben) lassen sich dadurch kausale Modelle bilden [Pea18], welche in der Lage sind, auch kontrafaktische Fragestellungen zu beantworten (im Sinne von „hätten Kängurus keine Schwänze, würden sie dann umfallen?“, [Lew01]). **Abbildung 3** zeigt den aktuellen Zwischenstand der bisher beschriebenen Architektur.

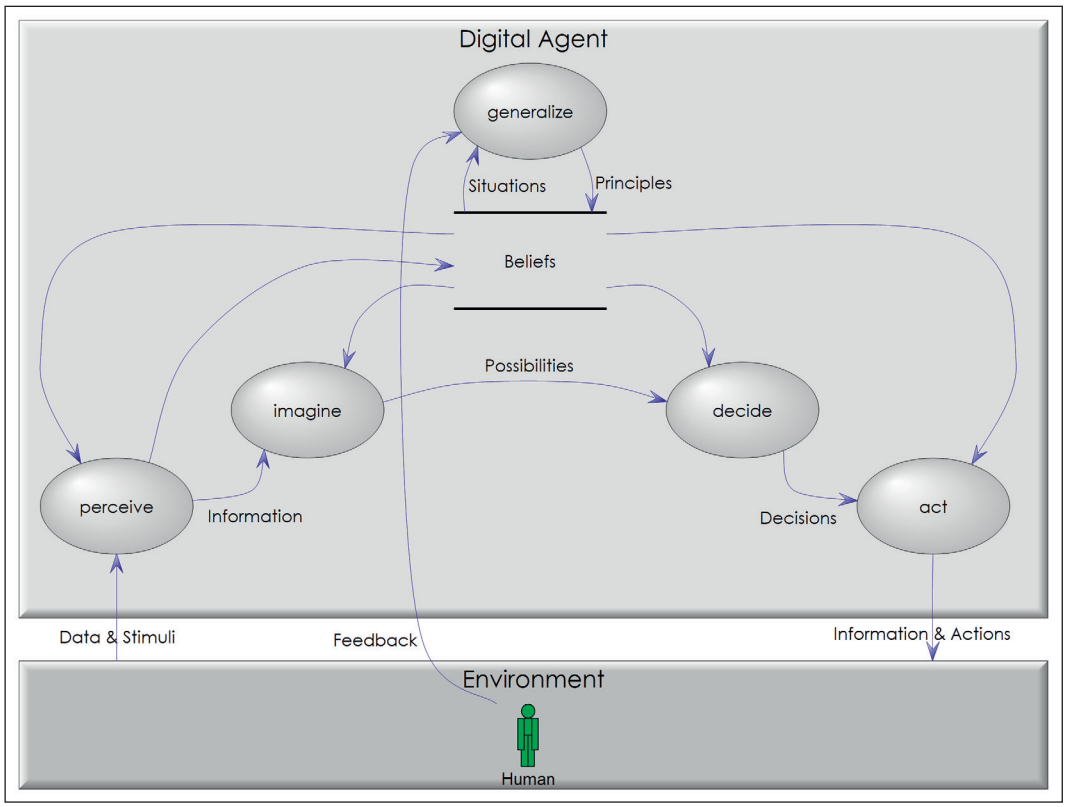

Abb. 3: Die PIDAG-Architektur eines digitalen Agenten

PIDAGE: Grenzen kennen

Auf absehbare Zeit kann keine der bisher genannten Fähigkeiten perfekt bereitgestellt werden (nicht einmal Menschen können das). Daher ist es unabdinglich, dass auch ein Rechner und insbesondere seine Software ihre Grenzen kennt. Solche Systeme müssen erkennen, wenn eine gegebene Situation sie überfordert, und sie müssen bei Bedarf außenstehende Instanzen beiziehen, statt sich einfach „durchzuwursteln“. Diese Funktion übernimmt die Fähigkeit *Enclose*. Im Falle von wirklich autonomen Systemen muss *Enclose* auch sicherstellen, dass der digitale Agent auch Vorgaben von außen (gesetzliche Vorgaben oder soziale Regeln) wirklich einhält.

E: Enclose
Die Fähigkeit *Enclose* hat den Zweck, insbesondere die Fähigkeiten Perceive, *Infer*, *Decide* und *Generalize* zu überwachen, um zu verhindern, dass das Gesamtsystem aufgrund fehlerhafter Ausführung unerwünschte oder gar fatale Konsequenzen verursachen kann. In einer solchen Situation ist, wenn immer möglich, die Kontrolle an eine „höhere“ Instanz zu übergeben, um den Knoten zu lösen – zum Beispiel an den menschlichen Benutzer des Systems.

Beispiele:
- *Spielcomputer:* Ein Spielcomputer sollte erkennen, wenn er sich in eine ausweglose Situation begeben oder keine Chance mehr auf einen Sieg hat, und daher rechtzeitig aufgeben.
- *MES:* Ein MES muss erkennen, wenn ein generierter Plan nicht mehr realisierbar ist, und entweder die Lösung verwerfen und eine neue erzeugen oder den Benutzer um Unterstützung bitten.
- *Moderner PKW:* Ein (teil-)autonom fahrender PKW muss erkennen, wenn er die Straßenränder nicht mehr sicher erkennt, und die Kontrolle rechtzeitig an den menschlichen Fahrer übergeben. Die Steuerung der Klimaanlage muss deren physikalischen Grenzen kennen, um allfällige Beschädigungen oder übermäßigen Energiekonsum zu vermeiden.

PIDAGEM: Der eigene Antrieb

Reine reaktive Systeme zeigen nur eine Reaktion, wenn ein Stimulus wahrgenommen wird. Ein digitaler Agent kann aber auch seine eigenen Ziele verfolgen, das heißt, er kann eine eigene Motivation haben, um über einen längeren Zeitraum mehr oder weniger autonom zu handeln. Ein solches Ziel kann mittel- oder langfristig sein und gegebenenfalls durch einen digitalen Agenten selbstständig unter Berücksichtigung seiner aktuellen Situation in einzelne kürzerfristige Etappenziele zerlegt werden. Diese Aufgabe repräsentiert die Fähigkeit *Motivate*.

M: Motivate
Bei der Fähigkeit *Motivate* geht es darum, aufgrund von Veränderungen im Umfeld des digitalen Agenten (die oft vom Agenten selber verursacht worden sind) (Teil-) Ziele dynamisch zu generieren und zu aktualisieren. Zudem kann zwischen Erreichungszielen (Achievement Goals) und Erhaltungszielen (Maintenance Goals) unterschieden werden. Erstere sind Ziele, welche wieder vergessen werden können, wenn sie einmal erreicht sind. Letztere sind Ziele, welche permanent gültig sind, also Rahmenbedingungen, welche unbedingt einzuhalten sind.

Beispiele:
- *Spielcomputer:* Ein Spielcomputer verfolgt zwar grundsätzlich das Endziel, den Gegner zu besiegen, muss aber seine Planung nach jedem Zug neu überprüfen und gegebenenfalls für die nächsten Züge seine Etappenziele in Richtung Sieg revidieren.
- *MES:* Ein MES kann unter Berücksichtigung des zur Verfügung stehenden Maschinenparks und der vorhandenen Ressourcen die Produktion eines ganzen Fertigungsloses planen, steuern und überwachen. Zudem muss es bei unerwarteten Problemen oder Ausfällen automatisch eine neue Disposition mit neuen Zwischenzielen erstellen.
- *Moderner PKW:* Ein autonom fahrender PKW muss permanent die (teilweise

konträren) Erhaltungsziele „wenn immer möglich auf der Straße bleiben“ und „nach Möglichkeit einem Hindernis ausweichen“ gegeneinander abwägen. Das Navigationssystem eines PKW weiß aus der elektronischen Agenda seines Besitzers in Ravensburg, dass er um 9:00 ein Meeting bei einem Kunden in Lindau am Bodensee hat, und kann aus diesem Endziel die Zwischenziele „8:15 Abfahrt in Ravensburg“, „8:25 in Eschach“ und „8:45 in Kressbronn“ ableiten. Im Falle eines unerwarteten Staus kann es zudem „Meckenbeuren“ als ein neues Zwischenziel einplanen.

PIDAGEMS: Der Selbstschutz

Cyber-technische Systeme sind heute in zunehmendem Maße bösartigen Angriffen ausgesetzt. Daher gilt es, diese Systeme vor solchen Angriffen ausreichend zu schützen. Ein digitaler Agent kann auf verschiedene Arten angegriffen werden:

- Seine Wahrnehmung (*Perceive*) kann beeinflusst oder gar gefälscht werden.
- Seine Regeln (*Imagine*) und Entscheidungsgrundlagen (*Decide*) können unautorisiert gelesen oder gar beeinflusst werden.
- Die Wahrnehmung kann so beeinflusst werden, dass daraus falsche Generalisierungen (*Generalize*) entstehen (auch als „Bias“ bekannt).
- Einem digitalen Agenten können von außen neue Ziele eingepflanzt werden (*Motivate*).
- Seine Begrenzungsmechanismen (*Enclose*) können aufgehoben werden.
- Seine Handlungen (*Act*) können behindert werden.

S: Safeguard
Die Fähigkeit schützt durch geeignete Mittel die „Innereien“ eines digitalen Agenten und stellt so sicher, dass seine Fähigkeiten wie vorgesehen trotz möglicher äußerer Angriffe ausgeführt werden können.

Beispiele:

- *Spielcomputer:* Für einen Spielcomputer sind wahrscheinlich keine besonderen (Selbst-)Schutzmaßnahmen erforderlich (außer es geht um hohe Wetteinsätze).
- *MES:* Verhaltensmuster, Regeln und Parameter über den Betrieb einer industriellen Anlage müssen sowohl vor unautorisiertem Lesezugriff als auch vor Manipulationen von außen geschützt werden.
- *Moderner PKW:* Ein PKW mit elektronischem Remote-Verschließsystem muss gegen unberechtigte Zugriffe abgesichert sein. Ein autonomer PKW muss seine Sensoren funktionstüchtig halten, indem er beispielsweise ein Reinigungssystem für verschmutzte Kameras besitzt und autonom nutzt.

PIDAGEMS-X: Erklärungen

Aus den Erfahrungen rund um die Erfolge von Deep-Learning-Ansätzen der letzten Jahre ist auch die Erkenntnis gewachsen, dass dabei tendenziell automatisch Entscheide durch eine Black Box gefällt werden, die nur sehr schwer nachvollziehbar sind. Daher wird heute vermehrt die

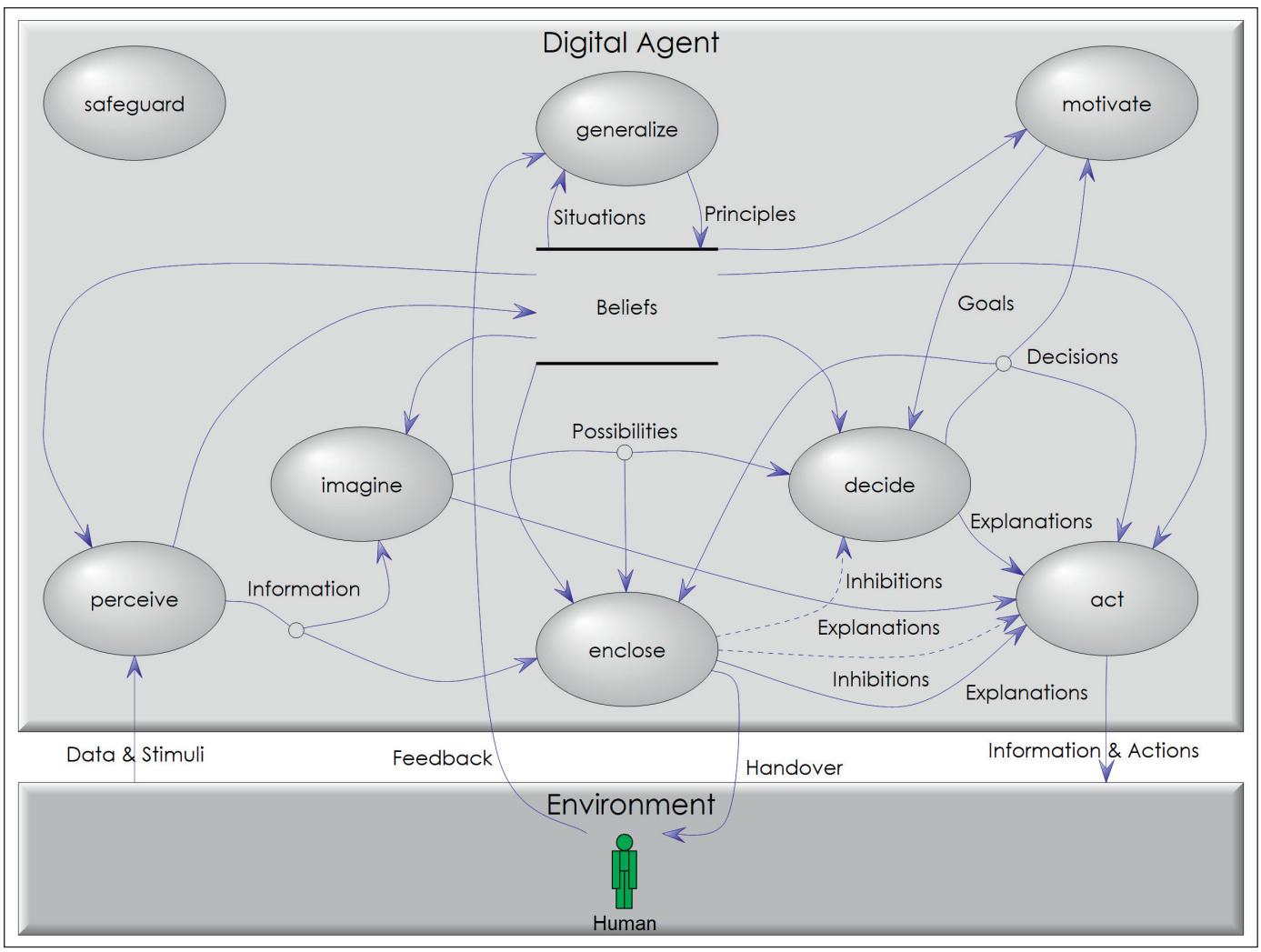

Abb. 4: Die vollständige Referenzarchitektur eines digitalen Agenten

Literatur & Links

[AI] Artificial Intelligence – Foundation of Computational Agents, Generate-and-Test Algorithms, siehe: **https://artint.info/html/ArtInt_77.html**

[Bos14] N. Bostrom, Superintelligence: Paths, Dangers, Strategies, Oxford University Press, 2014

[DM] DeepMind: AlphaGo, siehe: **https://deepmind.com/research/case-studies/alphago-the-story-so-far**

[HEISE19] T. Wittenhorst, Teslas Autopilot nimmt Hunde als Fußgänger wahr – und weicht Enten aus, Heise, 16.11.2019, siehe: **https://www.heise.de/newsticker/meldung/Teslas-Autopilot-nimmt-Hunde-als-Fussgaenger-wahr-und-weicht-Enten-aus-4587773.html**

[LewO1] D. Lewis: Counterfactuals, Wiley-Blackwell, 2001

[Pea18] J. Pearl, D. Mackenzie, The Book of Why, Basic Books, 2018

[Rus19] S. Russel, Human Compatible: Artificial Intelligence and the Problem of Control, Viking, 2019

[Sch15] M. Schacher, BPMN, Geschäftsregeln und der freie Wille – Eine etwas andere Sicht auf Modelle, in: OBJEKTspektrum, 6/2015

[Sch18] M. Schacher, Was ist eigentlich „Digitalisierung"? Ein Modewort macht Karriere, in: OBJEKTspektrum, 06/2018

Erklärfähigkeit (eXplainability) automatisch getroffener Entscheidungen gefordert, was wiederum das Vertrauen in die Entscheidungsfähigkeit des Systems fördert. Diese Erklärfähigkeit kann sowohl auf explizite Anfrage (online oder offline), aber auch unaufgefordert geleistet werden. Dies wird auch mit dem Begriff „Explainable AI" (XAI) bezeichnet.

X: eXplain
Bei der Fähigkeit *eXplain* geht es darum, eine automatisch abgeleitete Information oder eine getroffene Entscheidung erklärbar und dadurch für Menschen verständlich und nachvollziehbar zu machen.

Beispiele:

- *Spielcomputer:* Ein Spielcomputer kann seinem Gegner seine gespielten Züge mittels Sequenzen von (wechselseitigen) Folgezügen begründen und ihm so zu einem besseren Verständnis des Spiels verhelfen.
- *MES:* Das interaktive Dispositionssystem eines MES in der industriellen Fertigung sollte dem Produktionsleiter mittels dynamischer Simulationen aufzeigen, welche Vor- und Nachteile verschiedene Planvarianten haben.
- *Moderner PKW:* Ein (teil-)autonom fahrender PKW zeigt dem menschlichen Fahrer permanent nicht nur die Straßenbegrenzung mit allfälligen Hindernissen an, sondern auch die anderen in seiner direkten Umgebung wahrgenommenen Fahrzeuge, inklusive deren Klassifizierung und Fahrrichtung. Dies ist eine ausgesprochen vertrauensbildende Maßnahme gegenüber dem menschlichen Fahrer.

PIDAGEMS-X:

- P: Perceive
- I: Imagine
- D: Decide
- A: Act
- G: Generalize
- E: Enclose
- M: Motivate
- S: Safeguard
- X: eXplain

Alle Fähigkeiten der Referenzarchitektur

Damit ist die Referenzarchitektur für digitale Agenten vollständig. **Abbildung 4** zeigt, wie all diese Fähigkeiten zusammenspielen. Dabei sind zwei Dinge zu bemerken:

- Die Fähigkeit *Safeguard* ist bewusst ohne Verbindungen zu den anderen Fähigkeiten aufgezeigt, da sie jede der anderen Fähigkeiten schützen muss.
- Die Fähigkeit *eXplain* ist nicht separat gezeigt, da sie als eng verwobener Bestandteil von *Imagine* und *Decide* betrachtet wird.

Technisch gesehen sind die in dieser Referenzarchitektur enthaltenen Fähigkeiten nicht notwendigerweise als separat zu implementierende Prozesse zu verstehen, sondern können auch eng ineinander verwoben sein.

Fazit

Die vorgestellte Referenzarchitektur ist eine Checkliste, mit der Softwaresysteme jeglicher Art, von einfachen Berechnungssystemen bis zu komplexen autonomen Systemen aus dem Bereich der künstlichen Intelligenz konstruiert, analysiert und kritisch hinterfragt werden können. In der absehbaren Zukunft werden wir es vermehrt mit solchen digitalen Agenten zu tun haben, welche sich im Auftrag von uns ganz autonom in einer digitalen Machine-to-Machine(M2M)-Ökonomie bewegen.

Und wenn wir wie Nick Bostrom [Bos14] oder Stuart Russel [Rus19] noch etwas weiterdenken, so ist es nicht ganz ausgeschlossen, dass wir in (ferner) Zukunft einmal so etwas wie eine künstliche „Superintelligenz" schaffen. Spätestens dann wird insbesondere das ausgewogene Zusammenspiel zwischen *Enclose* und *Safeguard* essenziell:

- *Safeguard* schützt ein solches System vor Angriffen aus seiner Umwelt,
- wogegen *Enclose* die Umwelt vor unerwünschten Aktionen eines solchen Systems schützt. ||

Der Autor

Markus Schacher
(markus.schacher@knowgravity.com)
ist Mitbegründer von KnowGravity Inc. und hat sich auf modellbasiertes Engineering spezialisiert. Er hat bereits 1997 die ersten öffentlichen UML-Kurse in der Schweiz durchgeführt und vielen großen Projekten geholfen, modellbasierte Techniken einzuführen und nutzbringend anzuwenden. Als Mitglied der Object Management Group (OMG) ist er in der Entwicklung verschiedener Modellierungssprachen involviert. Außerdem ist Markus Schacher Koautor dreier Bücher zu den Themen Geschäftsregeln, SysML sowie operationellen Risiken und häufiger Präsentator auf internationalen Konferenzen.