Industrielle Agenten und Agenten-basiertes Lernen im technischen Kontext

Stefan Bosse
Universität Bremen, Fachbereich Mathematik & Informatik, Bremen, Deutschland

Datenverarbeitungsprozesse werden immer komplexer hinsichtlich der Datenmenge, Datendimension, und dem Zusammenhang der abgeleiteten Informationen mit den Eingangsdaten. Dieses wird bei sensorischen Messprozessen besonders deutlich, wo Messunsicherheiten, Kalibrierungsfehler, und Unzuverlässigkeit von Sensoren signifikanten Einfluss auf die Informationsgewinnung haben. Gerade im technischen und industriellen Kontext stellt die zunehmende Komplexität und Verteilung der Datenverarbeitung ein zunehmendes Problem dar. Häufig stehen hinter der Informationsableitung mathematische Modelle und Funktionen, die aber nicht immer vollständig sind. Geht es um die Gewinnung von Zustandsaussagen eines Systems oder um Adaption, können Lernverfahren eine Alternative darstellen. Traditionell werden Messdaten zentral gesammelt und ausgewertet. Es soll aufgezeigt werden, wie verteiltes Maschinelles Lernen mit mobilen Agenten und selbst-organisierenden Systemen einen signifikanten Beitrag zur Verbesserung der Datenverarbeitung in technischen und industriellen Systemen leisten kann, dieses sowohl hinsichtlich der Qualität der Aussagen von Schlussfolgerungen, der Effizienz, als auch der Robustheit. Dieser Beitrag soll einen Überblick der verschiedenen Teilbereiche Lernen, Agenten, und Architekturen geben.

Einführung

In den letzten Jahrzehnten gab es einen Wandel von passiven einzelnen Sensoren hin zu Netzwerken aus smarten Sensoren, die mit Informations- und Kommunikationstechnologien ausgestattet sind. Weiterhin kann man einen rasanten Anstieg der Sensordichte in Sensornetzwerken feststellen **[1]**, zu sehen in Abb. **1**. Fortschreitende Miniaturisierung von Sensoren und neue Möglichkeiten der Mikrosystemtechnik ermöglichen die Einbettung von sensorischen Netzen in Materialien und technische Strukturen **[2]**. Die material-integrierten Sensornetzwerke stellen besondere Anforderungen an die Informationsverarbeitung hinsichtlich Ressourcenbedarf, Latenz, Energiebedarf, und Robustheit dar. Technische Fehler können in solchen Systemen i.A. nachträglich nicht korrigiert werden und müssen durch adaptive und selbst-organisierende Systemeigenschaften ausgeglichen werden. Die Größe eines autonomen Smarten Sensors ausgestattet mit Informations- und Kommunikationstechnologien erreicht derzeit die mm^3 Skala, wie z.B. die Smart Dust Mote **[3]**.

Auf der anderen Seite der Skalierbarkeit gibt es einen zunehmenden Bedarf und Einsatz von Cloud Lösungen, z.B., um Fertigung, Entwurf, und Produkte zur Optimierung zu koppeln **[4]**. Eine Cloud ist charakterisiert durch die ortsungebundene Virtualisierung von Speicher und Rechenkapazität, häufig unter Anbietung von Dienstleistungen für die Datenverarbeitung (Service-orientierte Architekturen). Eine Cloud ist grob-granuliert verteilt, primär mit Rechenzentren physisch besetzt, und skalierbar durch Hinzunahme weiterer Server und Rechenzentren. Cloud Umgebungen leisten einen wesentlichen Beitrag zur Bewältigung des Big-Data Problems und der Informationsgewinnung aus einer großen Menge von teils nicht direkt korrelierten Daten.

Zukünftige industrielle Umgebungen, die sich aus Produktion, Entwurf, und Endkunden zusammen setzen, werden Datenverarbeitung auf diversen Skalen abdecken: Von der Mikro- bis zur Makroebene. Diese Umgebungen werden durch stark heterogene Netzwerkumgebungen charakterisiert sein.

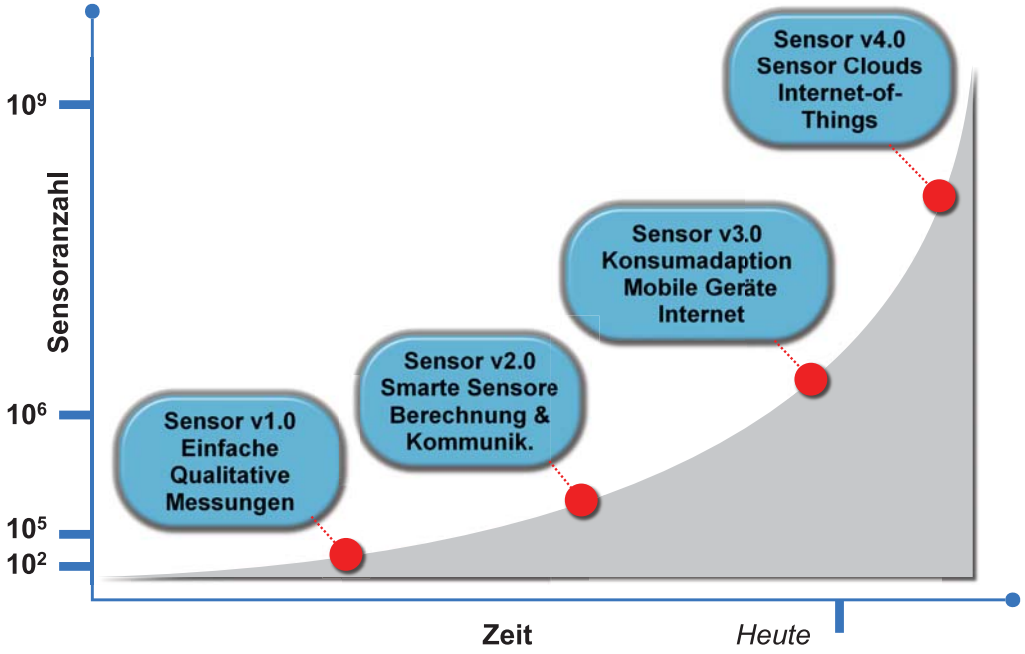

Abb. 1. Generationenfolge von Sensoren: Von passiven Sensoren hin zu Sensor Clouds

In diesem Artikel sollen neue Möglichkeiten durch den Einsatz von mobilen und lernenden Agenten aufgezeigt werden, um auch zukünftig effiziente und skalierbare Datenverarbeitung in Sensor- und industriellen Netzwerken zu ermöglichen. Gerade auch in Hinblick auf selbst-organisierende und selbst-adaptive Strukturen, die den Administrationsaufwand deutlich verringern, und die Zuverlässigkeit erhöhen, sowie Cloud-basierte Lösungen ermöglichen.

Multi-Agenten Systeme und Industrielle Agenten

Agenten sind teil- oder voll autonome Datenverarbeitungseinheiten, die grundlegend aus einem Steuerungs- und Planungsmodul sowie Daten bestehen, zusammengefasst in einem informatorischen Prozess, der durch seinen veränderlichen Kontroll- und Datenzustand bestimmt ist. Diese privaten Daten des Agenten stammen zum einen aus einem Perzeptionsprozess mit der Umgebung (z.B. Sensorische Daten), und zum zweiten aus berechneten Daten. Wesentliche Eigenschaft von Agenten ist ihre Adaptierbarkeit ihres Verhaltens (des Steuerungsmoduls) basierend auf aktuellen und vergangenen Daten, z.B. Sensordaten. Diese Adaptierbarkeit geht unmittelbar mit dem Konzept des Lernens einher.

Das Verhalten von Agenten kann an das von natürlichen Lebewesen angelehnt sein, z.B. wie in der weit verbreiteten Belief-Desire-Intention (BDI) Architektur **[5]**, häufig unter Verwendung von deklarativen Beschreibungen. Ein vereinfachtes Verhaltensmodell ist das graphen-basierte Aktivitäts-Übergangs Modell (ATG, engl. Activity-Transition Graph, siehe Abb. **2**), welches das Verhalten eines Agenten mit einer Vielzahl von Aktivitäten beschreibt, die Aktionen ausführen, d.h. Daten verarbeiten und mit der Umgebung interagieren. Die Aktivitätsübergänge werden durch interne Daten des Agenten bestimmt (d.h. sind i.A. konditional).

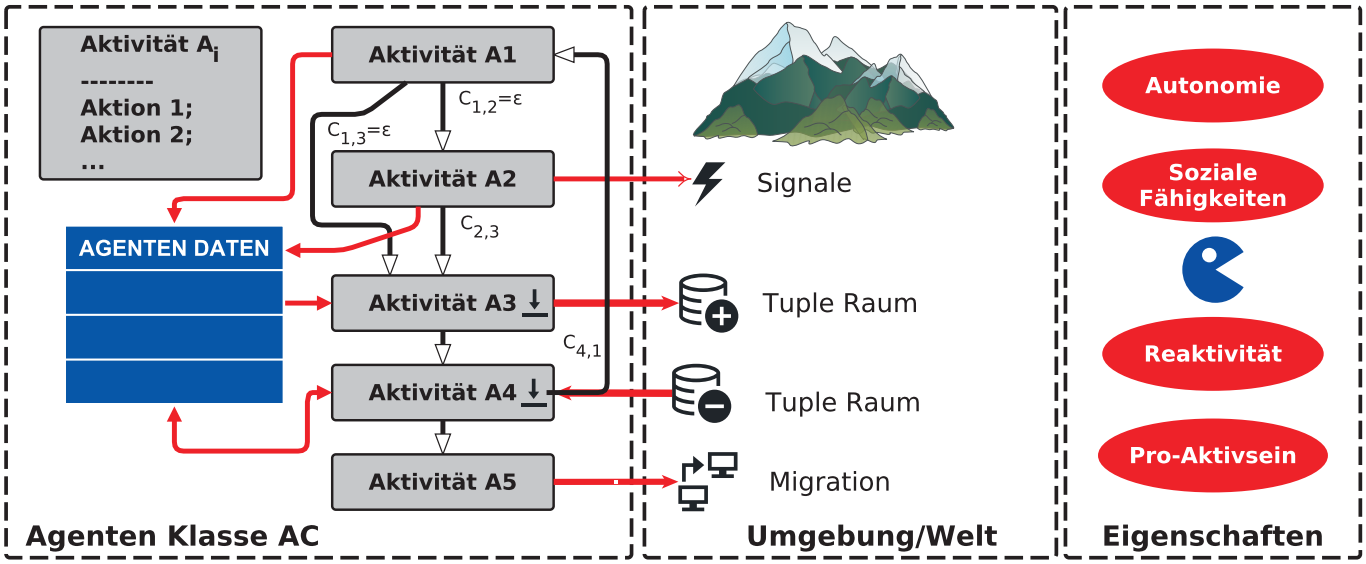

Abb. 2. Aktivitäts-Übergangs Graphen als einfaches Modell zur Beschreibung des Verhaltens von Agenten

Die Aktivitäten (Anweisungsblöcke) erlauben eine Partitionierung des Gesamtverhaltens eines Agenten, auch mit Bezug auf die Ziele des Agenten, und stellen grob granulierte Ausführungsschritte dar. Aktivitäten werden von einer Agentenplattform als "Wirt" ausgeführt. Ein ATG ist zur Laufzeit vom Agenten modifizierbar (Dynamischer ATG), wodurch das Verhalten eines Agenten selbst-adaptierbar wird. Die Modifikation des ATG bedeutet eine Änderung der Übergänge (Kanten) und der Aktivitäten (Austausch, Löschung, oder Hinzufügung) des Graphen.

Agenten werden bereits erfolgreich im industriellen Umfeld eingesetzt, vornehmlich für die Planung **[6]** und Ablaufsteuerung von Fertigungsprozessen **[7]**, aber auch zunehmend für eine weitreichende Vernetzung von z.B. Produktion und Entwurfsprozessen **[4]**.

Neben der Steuerung von Produktionsprozessen sind Agenten auch für die Bereiche Wartung, modulare Fertigungssysteme, Qualitätskontrolle und Energiemanagement einsetzbar. Fähigkeiten der Selbst-Organisation und Adaptivität spielen eine zunehmende Rolle im industriellen Umfeld, insbesondere durch die zunehmende Einbeziehung von Sensoren und Sensornetzwerken. Das neue Paradigma der industriellen Agenten kann einen wesentlichen Beitrag leisten für zukünftige modulare und flexible industrielle Umgebungen die in Cloud Umgebungen eingebettet werden.

Mobile Agenten sind in der Lage ihren Zustand, der sich aus Daten- und Kontrollzustand zusammensetzt, als "Schnappschuss" von einer Ausführungsplattform zu einer anderen mitzunehmen, wodurch eine nahtlose Ausführung des Agenten möglich wird. Mobile Agenten reflektieren daher eine mobile Servicearchitektur.

Multi-Agenten Systeme setzten sich aus einer Vielzahl von einzelnen Agenten zusammen, die miteinander kommunizieren. Kommunikation kann direkt mit Nachrichten erfolgen (z.B. FIPA-ACL), oder indirekt und stärker entkoppelt durch z.B. Tupleräume mit Muster-basierter Suche **[8]** geschehen. In verteilten Systemen wird i.A. eine große Aufgabe auf eine Vielzahl von kleinen Aufgaben und somit Agenten verteilt, dem Divide-and-Conquer Ansatz folgend. Ein Beispiel ist verteiltes Data Mining mit Map&Reduce Ansätzen.

Die Skalierung von industriellen Datenverarbeitungsanwendungen auf komplexe Cloud-basierte und weit ausgedehnte verteilte Netzwerke inklusive Sensornetzen führt zu Systemen

mit tausenden bis Millionen von Agenten. Die Agentenplattform ist daher eine zentrale Schlüsseltechnologie, die später diskutiert wird.

Verteiltes Maschinelles und Agenten Lernen

Maschinelles Lernen (ML) kann in unterteilt werden in überwachtes, rückgekoppeltes, und nicht-überwachtes Lernen, gezeigt in Abb.**3**. Trainiertes überwachtes Lernen dient häufig der Ableitung einer Klassifikationsfunktion *K:* f(***x***) $\rightarrow$ *l* mittels eines Lerners (Modellbilder) *M*: f(***D***) $\rightarrow$ *K* aus einer Vielzahl von gekennzeichneten Datensätzen ***D,*** die sich aus ***x*** $\in$ ***X*** n-dimensionalen Vektoren, z.B. Sensordaten ***S***, und einer Menge von zugeordneten Symbolen (Labels) *l* $\in$ ***L*** zusammensetzen. Die Klassifikationsfunktion wird durch Training in der Lernphase abgeleitet, wo durch einen Trainer (Mensch oder Programm) bekannten Datensätzen die entsprechenden Symbole zugeordnet werden (beide sind Eingabedaten, "labeled data"). Nach der Lernphase folgt die Anwendungsphase, wo die Klassifikationsfunktion eine Vorhersage eines Symbols (das Klassenmerkmal) für einen unbekannten Datensatz ***x*** berechnet. Ein Beispiel wird später gezeigt, wo ML für die Erkennung von unterschiedlichen Lastsituationen einer mechanischen Struktur aus Sensordaten genutzt wird. Rückgekoppeltes Lernen (z.B. Belohnungslernen) hingegen bewertet aktuelle Perzeption hinsichtlich gewählter Aktionen, und versucht in der Planung geeignete Aktionen (oder Pläne) auszuwählen, um vorgegebene Ziele zu erreichen.

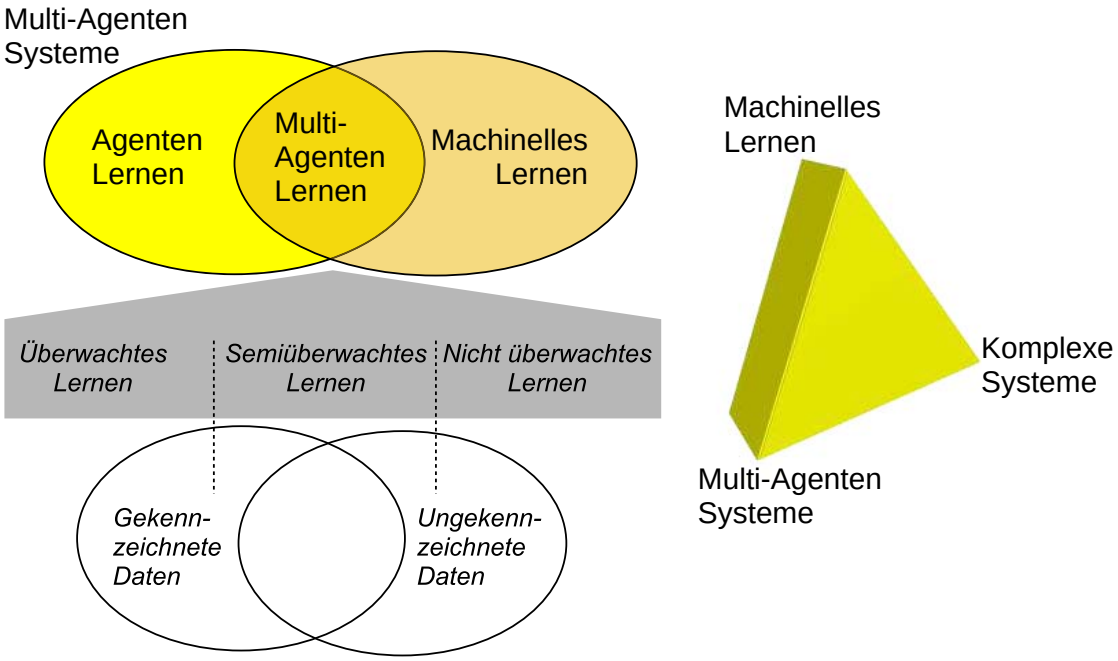

Abb. 3. Die Kombination aus Maschinellen Lernen und Agenten führt zu einem Synergieeffekt bei der Komposition von komplexen Systemen.

ML und Agenten können kombiniert werden, und mobile Agenten sind geeignet für die Exploration in räumlich ausgedehnten Gebieten in einem Netzwerk, d.h. um Sensordaten zu sammeln. Diese können verwendet werden um ein Klassifikationsmodell der Daten bezüglich der gewählten Eigenschaften zu erstellen, wie z.B. verschiedene Lastsituationen eines Bauteils. Dieses Modell kann vom mobilen Agenten transportiert werden, und an anderer Stelle angewendet werden. Diese Eigenschaft ist besonders in mobilen Netzwerkumgebungen von Bedeutung, vor allem bei der Nutzung von mobilen Endgeräten wie Smartphones als Sensorknoten. So kann die

Mobilität eines Lernagenten umgekehrt genutzt werden, um Daten in einem festgelegten räumlichen Bereich von fluktuierenden und "durchströmenden" mobilen Geräten aufzunehmen (Ubiquitous Computing, **[9]**). Weiterhin kann rückgekoppeltes Lernen (Belohnungslernen) mit dem Agentenmodell zur Adaptation und Planung von Aktionen, wie z.B. der Steuerung von Aktoren oder ganzen Maschinen, kombiniert werden.

Neben klassischen Lernalgorithmen die Lern- und Anwendungsphasen besitzen gewinnen inkrementelle (online) Lerner an Bedeutung **[10]**. Sie können während der Laufzeit neue Datensätze zur Verbesserung des gelernten Modells nutzen (z.B. Entscheidungsbaumlerner oder Neuronale Netzwerke) ohne die gesamte Datenbank mit allen bisherigen Datensätzen speichern zu müssen.

Häufig sind Lerner zentralisiert, d.h., alle Eingabedaten werden durch ein Programm gesammelt und ausgewertet, was mit einem zentralen Ausfallpunkt und hoher Datenstromdichte verbunden ist. Es existieren aber Ansätze, das Lernen unter Verwendung von Agenten zu verteilen. Ein möglicher Ansatz basiert auf dem Lernen lokaler Modelle die global zusammengeführt werden. Dies geschieht indem das gesamte Netzwerk räumlich in Regionen (Regions of Interest, ROI) unterteilt wird, und eine Vielzahl Lerner jeweils in diesen Regionen operieren. Beim Klassifizieren wird es eine Vielzahl von Vorhersagen geben, die immer nur auf einer lokalen Sicht beruhen und differieren (falsch sein) können. Ein geeignetes Mittel zur Ableitung der zuverlässigen globalen Klassifikation kann durch Stimmabgabe und Wahlverfahren erfolgen, d.h. z.B. ein Mehrheitsentscheid liefert das wahrscheinlichste Klassifikationsergebnis. Solche verteilten Lernsysteme skalieren deutlich besser als zentrale Lerner, und besitzen keinen zentralen Ausfallpunkt. Der Wegfall einzelner Lerner führt nur zu einer kleineren Anzahl von Stimmen mit herabgesetzter Wahrscheinlichkeit der globalen Korrektheit.

Formell lässt sich die räumliche Verteilung am Beispiel des klassifizierenden Lernprozesses für Sensordaten wie folgt darstellen:

$$
\begin{array}{ll}
M : D \to K(S) & m_{i,j} : d_{i,j} \to k_{i,j}(s) \\
l \in \boldsymbol{L} & k_{i,j} : s_{i,j} \to l_{i,j} \\
K : S \to l & \xrightarrow{\text{Distribution}} \quad K : (l_{1,1}, l_{1,2}, \cdots) \to l \\
D : \{(S^1, l^1), (S^2, l^2), \cdots\} & d_{i,j} : \{(s^1_{i,j}, l^1), (s^2_{i,j}, l^2), \cdots\} \\
S : \begin{pmatrix} x_{1,1} & \cdots & x_{n,1} \\ \vdots & \ddots & \vdots \\ x_{1,m} & \cdots & x_{n,m} \end{pmatrix} & s_{i,j} : \begin{pmatrix} x_{i-u,j-v} & \cdots & x_{i+u,j-v} \\ \vdots & \ddots & \vdots \\ x_{i+u,j-v} & \cdots & x_{i+u,j+v} \end{pmatrix}
\end{array}
\qquad (1)
$$

wenn $\boldsymbol{S}=(s_1,s_2,..)$ eine Matrix (oder Vektor) der einzelnen räumlich verteilten Sensoren mit den Daten $x_{n,m}$ an Position (n,m) darstellt, und $\boldsymbol{S}_{i,j}$ eine Teilmatrix um den räumlichen Mittelpunkt (i,j) davon, und K als globale, $k_{i,j}$ als eine lokal gelernte Klassifikationsfunktion.

Agentenplattformen

In stark heterogenen Einsatzumgebungen ist die Agentenplattform eine Schlüsseltechnologie um mobile Agenten nahtlos und ohne zusätzliche Transformation ausführen zu können. Bisherige Agentenplattformen können noch nicht mit einer großen Anzahl von Agenten umgehen und können nur im Labormaßstab eingesetzt werden (mit weniger als 1000 Agenten) **[7]**. Will man

den Ansatz auf größere Probleme anwenden, so erfordert dies eine deutlich höhere Agentenzahl.

Weiterhin sind in stark heterogenen Einsatzumgebungen verschiedene Implementierungen (Hardware, Software, Simulation, Browser & Server) auf verschiedenen Hostplattformen (Microchip, eingebettete und mobile Systeme, Computer, Server, Browser) erforderlich, wie in Abb. **4** gezeigt ist. Es gibt nur wenige prototypische Agentenplattformen die sehr breitbandig eingesetzt werden können. Ein Beispiel ist die portable JavaScript Agent Machine (JAM) Plattform **[11]**, die Agenten mit mobilen JavaScript-Code effizient ausführt, der auch die Daten des Agenten einbettet. Eine Stack Prozessor basierte Lösung stellt die Pipelined Agent Virtual Machine (PAVM) dar **[12]**, die ebenfalls sehr breitbandig eingesetzt werden kann, mobilen FORTH-Code mit eingebetteten Daten nutzt, und zudem Hardware und Software Implementierungen ermöglicht. Sowohl PAVM als auch JAM basieren auf dem ATG Agenten Modell und sind prinzipiell kompatibel.

Die JAM Plattform bietet als eine der wenigen Maschinelles Lernen als Service an, d.h., ein Lerneragent speichert nur ein gelerntes Modell, aber nicht die Algorithmik (den ML-Code), was zu einer erheblichen Effizienzsteigerung führt.

Der Einsatz von mobilen Agenten und den Plattformen im Internet oder industriellen Netzwerken erfordert zusätzliche Organisations- und Sicherheitsstrukturen. Z.B. in einem materialintegrierten Sensornetzwerk besitzen Knoten allgemein eine geometrische Nachbarschaftskonnektivität, die im Internet nicht gegeben ist, zudem das Internet ein Netz aus Netzen darstellt. Mobilität von Agenten bedarf daher virtueller Vernetzungs- und Kommunikationsstrukturen, die von der Plattform bereit gestellt werden müssen. Eine Möglichkeit besteht in einem verteilten Verzeichnisdienst, der Rechner (Hosts) und Agentenplattformen in Gebiete einteilt **[11][12]**.

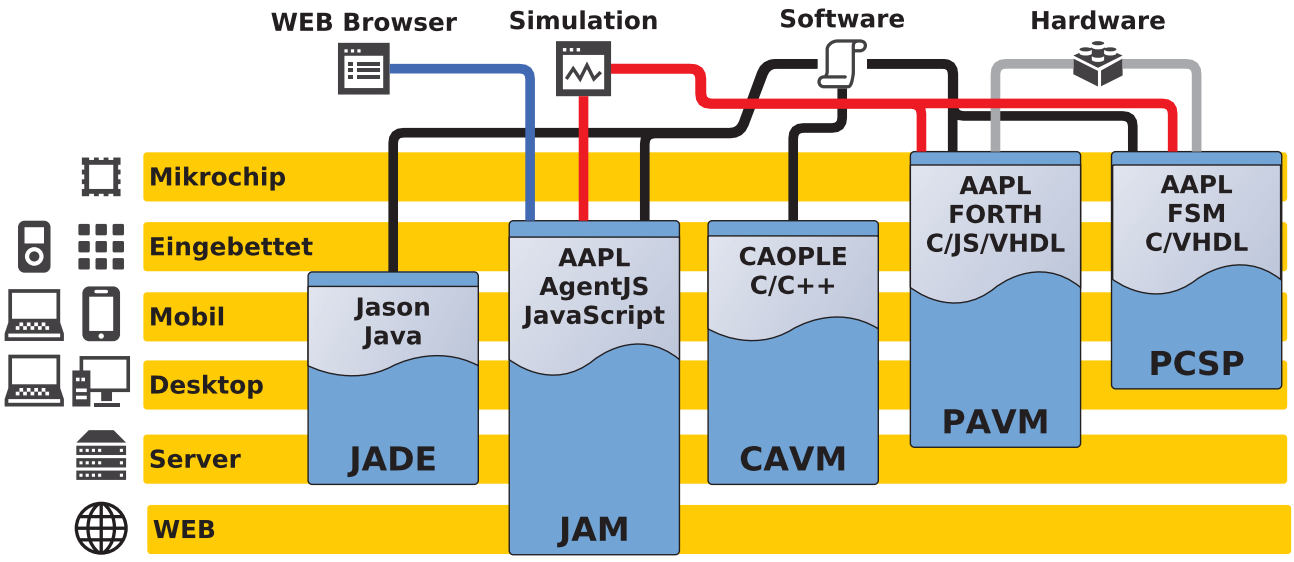

Abb. 4. Verschiedene Hostplattformebenen, Implementierungen und Agenten Plattformen (JADE: **[17]**, JAM: **[11]**, CAVM: **[13]**, PAVM: **[12]** , PCSP: **[14]**)

Mobilität von Agenten (=Prozessen) erfordert zwei wichtige Eigenschaften: (I) Effiziente Erzeugung eines "Schnappschusses" eines Prozesses in Form eines Programms, der den Code und den aktuellen Kontroll- und Datenzustand des Prozesses einbindet (II) Geringe Abhängigkeit des Programms von der Plattformschnittstelle, der Hostplattform (Rechnerarchitektur), und von Netzwerkarchitekturen, d.h. Vermeidung von Binärcode. JavaScript bietet z.B. auf gängigen virtuellen

Maschinen (node.js, Google V8, usw.) eine isomorphe Transformation zwischen dem gerade ausgeführten Code und dem Programmtext. Weiterhin erfordert die Rekonfigurierbarkeit von Agenten die Fähigkeit zum Codemorphing (Änderung des Programmcodes) zur Laufzeit.

Beispiel: Lasterkennung von mechanischen Strukturen

Ein Anwendungsbeispiel für verteiltes Lernen gemäß dem Ansatz in Gl. **1.** ist die Strukturüberwachung von mechanischen Bauteilen. Dabei können durch ein Lernverfahren verschiedene Lastsituationen (l_1,l_2,,..) von Sensordaten gelernt und unterschieden werden, ohne eine genaue Kenntnis des mechanischen Modells des Bauteils zu haben (z.B. ein FE Modell). Die Sensordaten stammen aus einem Sensornetzwerk, welches z.B. auf der Oberfläche oder im Bauteil integriert ist und z.B. mittels Dehnungssensoren einen örtlich aufgelösten Dehnungszustand des Bauteils liefern kann, wie in Abb. **5** auf der linken Seite gezeigt ist. Die Knoten des Sensornetzwerkes stellen neben Sensoren auch eine Agentenplattform zu Verfügung.

In diesem Szenario werden eine Vielzahl von mobilen und immobilen Agenten (siehe Abb. **5**, rechte Seite) mit unterschiedlichen Zielen/Aufgaben und Verhalten eingesetzt:

- **Knotenagent**: Jeder Sensorknoten ist mit einem immobilen Knotenagenten besetzt, der die Sensorvorverarbeitung und Ereigniserkennung durchführt (d.h. ob es einen signifikanten Stimulus gegeben hat)
- **Lerneragent**: Auf jedem Sensorknoten befindet sich ein Lerneragent, der vom Knotenagent aktiviert wird wenn ein sensorischer Stimulus aufgetreten ist. Dieser Lerner kann sich in zwei Phasen befinden: (I) Lernen (II) Klassifizieren. Die Auswahl der Phase erfolgt über **Benachrichtigungagenten**, die im Netzwerk z.B. eine bekannte Lastsituation (das Label *l*) ankündigen, oder Lerner in den Anwendungsmodus schalten. Die Lerneragenten haben zugriff auf Sensordaten aus ihrer unmittelbaren Umgebung, mit denen sie ein lokales Modell (Sensor-Last) lernen.
- **Explorationsagent**: Sowohl für die Vorhersage eines sensorischen Stimulus als auch für das Lernen sind Sensordaten in einem begrenzten Bereich (Region of Interest, ROI) erforderlich. Diese werden mittels eines Divide-and-Conquer Ansatzes mit Explorationsagenten eingesammelt und an den Knoten- oder Lerneragenten ausgeliefert.
- **Stimmagent**: Hat ein Lerneragent eine Klassifikation der Lastsituation aus seiner lokalen Sicht durchgeführt, so sendet er Stimmagenten aus, die von **Wahlagenten** eingesammelt werden, um ein globales Abstimmungsergebnis und somit eine Vorhersage einer aktuellen Lastsituation zu liefern.

Die einzelnen Agenten werden zum Teil dynamisch von anderen Agenten erzeugt, z.B. die Explorationsagenten von den Lerner- oder Knotenagenten. Die Agenteninteraktionen findet über Tupleräume (synchronisierter Datenaustausch) sowie mobile Signale statt.

In **[15]** wurde mit einem beispielhaften Netzwerk bestehend aus 64 Sensorknoten solch ein MAS simuliert (Abb. **5**). Dabei wird die Sensorierung einer mechanischen Struktur mit Dehnungssensoren angenommen. Je nach Lastsituation und Änderung von Dehnungssensoren ist das Netzwerk zeitweise mit einigen Hundert bis Tausend Agenten besetzt, die teilweise zwischen den Sensorknoten migrieren. Eine effiziente Ausführung und Migration ist daher unverzichtbar und konnte durch die JavaScript-Plattform JAM realisiert werden. Ein signifikanter Vorteil liegt in der ereignisbasierten Sensorverarbeitung, wo im Gegensatz zur kontinuierlichen strombasierten Auswertung nur stimulierte Bereiche im Netzwerk aktiviert werden, und sich der erforderliche Rechen- und Kommunikationsaufwand deutlich reduziert (bis zu 90%). Agenten können in diesem Szenario ebenso zwischen verschiedenen Netzwerken und in heterogenen Umgebungen eingesetzt werden, um z.B., Online-Lernverfahren mit numerischen Offline-Verfah-

ren zu koppeln **[16]**, und um Sensor- und Aktornetzwerke in das Internet und Cloud-Umgebungen zu integrieren.

Die Ergebnisse des verteilten Lernens basierend auf dem Wahlverfahren konnten verschiedene Lastsituationen mit einer guten globalen Vorhersagewahrscheinlichkeiten klassifizieren **[15]**, die mittlere Korrektheit der globalen Klassifizierung (richtig positiv) lag meistens bei über 80% (d.h. 20% der Stimmen waren falsch).

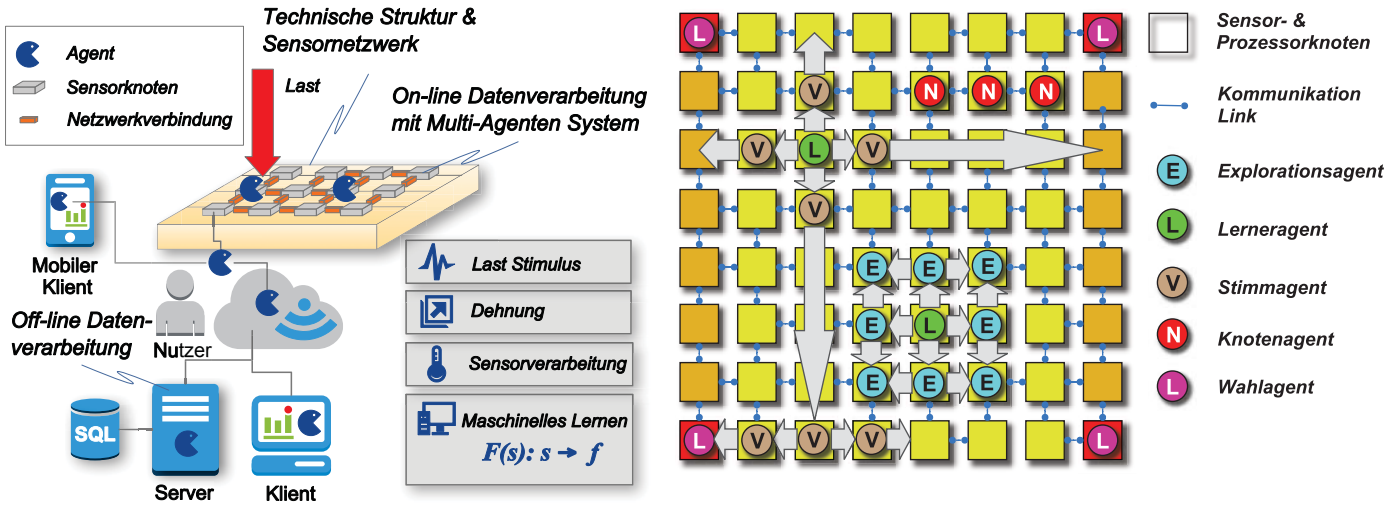

Abb. 5. (Links) Technische Struktur mit Sensornetzwerk und Integration in Intra- und Internet Umgebungen (Rechts) Logische Netzwerkschicht und Besetzung mit verschiedenen Agenten Klassen.

Zusammenfassung

Die zunehmende Komplexität und Heterogenität von industriellen Netzwerken und deren Einbindung in das Internet und Clouds erfordert neue selbst-organisierende und selbst-adaptive Methoden, die durch autonome Basiszellen konstruiert werden können. Multi-Agenten Systeme stellen eine skalierbare Methode dar, die mit lernenden Agenten einen wesentlichen Beitrag zur robusten und effizienten Informationsgewinnung leisten können. Am Beispiel der Strukturüberwachung wurde die Partitionierung einer globalen in viele einzelne Aufgaben und Bearbeitungseinheiten mittels Agenten aufgezeigt, die wesentlich auf einem Divide-and-Conquer Ansatz beruht. Lokales Lernen ist für die Erlangung globaler Aussagen durch Kollektiventscheid nutzbar. Die Agentenplattform ist eine Schlüsseltechnologie für den erfolgreichen und effizienten Einsatz von MAS in stark heterogenen Umgebungen.

Literatur

[1] V. Di Lecce, M. Calabrese, and C. Martines, *From Sensors to Applications: A Proposal to Fill the Gap*, Sensors & Transducers, vol. 18, no. Special Isse, pp. 5–13, 2013.

[2] D. Lehmhus et al., *When nothing is constant but change: Adaptive and sensorial materials and their impact on product design*, Journal of Intelligent Material Systems and Structures, Sep. 2013

[3] B. Warneke, M. Last, and B. Liebowitz, *Smart dust: Communicating with a cubic-millimeter computer*, Computer, 2001.

[4] D. Lehmhus, T. Wuest, S. Wellsandt, S. Bosse, T. Kaihara, K.-D. Thoben, and M. Busse,

Cloud-Based Automated Design and Additive Manufacturing: A Usage Data-Enabled Paradigm Shift, Sensors MDPI, vol. 15, no. 12, pp. 32079–32122, 2015

[5] R. H. Bordini and J. F. Hübner, *BDI agent programming in AgentSpeak using Jason*, Computational Logic in Multi-Agent Systems, Volume 3900 of the series Lecture Notes in Computer Science, Springer, 2006, pp. 143-164.

[6] M. Caridi and A. Sianesi, *Multi-agent systems in production planning and control: An application to the scheduling of mixed-model assembly lines*, Int. J. Production Economics, vol. 68, pp. 29–42, 2000.

[7] M. Pechoucek, V. Marík, 2008. *Industrial deployment of multi-agent technologies: review and selected case studies*. Auton. Agent. Multi-Agent Syst. 17 (3), 397–431

[8] L. Chunlina, L. Zhengdinga, L. Layuanb, and Z. Shuzhia, *A mobile agent platform based on tuple space coordination,* Advances in Engineering Software, vol. 33, no. 4, pp. 215–225, 2002

[9] E. Pournaras, I. Moise, D. Helbing, *Privacy-preserving ubiquitous social mining via modular and compositional virtual sensors*. In 2015 IEEE 29th International Conference on Advanced Information Networking and Applications (pp. 332-338).

[10] F. Jiang, Y. Sui, and C. Cao, *An incremental decision tree algorithm based on rough sets and its application in intrusion detection*, Artif Intell Rev, vol. 40, pp. 517–530, 2013.

[11] S. Bosse, *Mobile Multi-Agent Systems for the Internet-of-Things and Clouds using the JavaScript Agent Machine Platform and Machine Learning as a Service*, in The IEEE 4th International Conference on Future Internet of Things and Cloud , 22-24 August 2016, Vienna, Austria, 2016.

[12] S. Bosse, *Unified Distributed Computing and Co-ordination in Pervasive/Ubiquitous Networks with Mobile Multi-Agent Systems using a Modular and Portable Agent Code Processing Platform*, in The Proc. of the 6th EUSPN 2015, Procedia Computer Science.

[13] B. Zhou and H. Zhu, *A Virtual Machine for Distributed Agent-oriented Programming*, in Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE'2008), San Francisco, CA, USA, July 1-3, 2008, 2008.

[14] S. Bosse, *Distributed Agent-based Computing in Material-Embedded Sensor Network Systems with the Agent-on-Chip Architecture*, IEEE Sensors Journal, vol. 14, no. 7, pp. 2159-2170, 2014.

[15] S. Bosse, *Structural Monitoring with Distributed-Regional and Event-based NN-Decision Tree Learning using Mobile Multi-Agent Systems and common JavaScript platforms*, in Procedia Technology, 2016, DOI 10.1016/j.protcy.2016.08.063

[16] S. Bosse, A. Lechleiter, and D. Lehmhus, *Data evaluation in smart sensor networks using inverse methods and artificial intelligence (AI): Towards real-time capability and enhanced flexibility*, in Proc. of the CIMTEC, - 7th Forum on New Materials, Perugia, Italy, June 5 to 9, 2016

[17] F. Bellifemine and G. Caire, *Developing Multi-Agent Systems with JADE*, John Wiley & Sons, Ltd, 2007.