Echtzeitfähige Softwareagenten zur Realisierung cyber-physischer Produktionssysteme

Kurzfassung

der von der Fakultät Informatik
der Technischen Universität Dresden
zur Erlangung des akademischen Grades Doktoringenieur (Dr.-Ing.)
genehmigten Dissertation

eingereicht von

Dipl.-Inf. Sebastian Theiss
geboren am 6. November 1980 in Leisnig

Betreuender Hochschullehrer:

Prof. Dr.-Ing. habil. Klaus Kabitzsch

Dresden im Dezember 2015

Inhaltsverzeichnis

Tabellen- und Abbildungsverzeichnis

1 Einleitung

«Es gibt Revolutionen, die machen die Welt komplizierter, und es gibt Revolutionen, die machen die Welt einfacher. Und es gibt Revolutionen, die machen die Welt auf komplizierte Art einfacher.»

Christian Kämmerling, in der "Weltwoche" vom 28. August 2003

Die fortschreitende Globalisierung eröffnet produzierenden Unternehmen die Chance zur Erschließung neuer Absatzmärkte, relativiert aber anderseits Standortvorteile und bringt zusätzliche Mitbewerber in Reichweite. Da Konsumenten eine gewisse Produktqualität erwarten und die Spielräume bei der Preisgestaltung nur klein sind, werden Faktoren wie Innovationskraft und -geschwindigkeit sowie Kundenorientierung zunehmend erfolgsentscheidend. Dementsprechend ist in vielen Branchen der Trend zu immer komplexeren, aber dennoch variantenreichen oder gar individualisierten Produkten bei gleichzeitig sinkenden Entwicklungszeiten zu beobachten, etwa in der Rechentechnik, bei Unterhaltungselektronik und bei Kraftfahrzeugen. Für die zur Herstellung solcher Produkte eingesetzten automatisierten Fertigungsanlagen hat das weitreichende Konsequenzen: Während diese Anlagen bisher auf die möglichst effiziente Massenfertigung einer schmalen, über lange Zeit hinweg gleichbleibenden Produktpalette ausgerichtet waren, müssen sie nun in vielerlei Hinsicht hochflexibel ausgelegt werden, um

- die Fertigung vielfältiger Produktvarianten – die sogenannte *Mass Customization* – zu ermöglichen,

darüber hinaus aber auch zu gewährleisten, dass einfach

- auf neue Produktgenerationen umgerüstet,
- auf Änderungen und logistische Probleme in komplexen Zulieferketten reagiert,
- auf Schwankungen der Nachfrage eingegangen und
- intelligent mit Störungen einzelner Anlagenteile umgegangen

werden kann.

Diese Flexibilitätsanforderungen sowie der mit der zunehmenden Technisierung von Konsumgütern einhergehende Anstieg des Fertigungsaufwands erfordern einen grundlegenden Wandel bei der Gestaltung von Automatisierungssystemen, der aktuell in der Fachwelt unter den Schlagwörtern *vierte industrielle Revolution* und *Industrie 4.0* diskutiert wird. Ein wesentlicher Aspekt dieses Wandels ist die Auslegung von Automatisierungsanlagen als *cyber-physische Produktionssysteme (CPPS)*, also engmaschige Netzwerke flexibel kombinierbarer cyber-physischer Komponenten. Die namensgebende Einheit von mechatronischer Baugruppe und eingebetteter Rechenkapazität eröffnet dabei die Möglichkeit, Teile der Engineering-Arbeit zu den Herstellern solcher Komponenten zur verlagern. Diese können typische Einsatzszenarien vordenken und neben Softwarebausteinen zur Ansteuerung beispielsweise auch entsprechende Bedienbilder, die Verrieglungssteuerung, Simulations- und Diagnosefunktionen zur Unterstützung der Inbetriebnahme sowie Asset-Management-Parameter gebrauchsfertig integrieren [HOF+2014]. Mit ausreichend Intelligenz ausgestattet können solche Komponenten sogar die logische Vernetzung untereinander während der Inbetriebnahme, nach Umrüstungen oder in Reaktion auf Betriebsstörungen teilweise selbst übernehmen. Dadurch ergeben sich Fähigkeiten wie Selbstkonfiguration und Selbstregeneration, die in der Fachliteratur unter dem Begriff *Self-X* zusammengefasst werden. Insgesamt lässt sich also sagen, dass CPPS die Komplexität zukünftiger Automatisierungsanlagen einfacher beherrschbar machen und durch Reduktion des (Re-)Engineering-Aufwands die Flexibilität solcher Anlagen deutlich erhöhen. Allerdings stellen sie auch hohe Ansprüche an die zur ihrer Realisierung eingesetzte Basistechnologie. Diese muss:

- die verteilte Struktur von CPPS abbilden können,
- Echtzeitanforderungen, die oft an die Interaktion mit physischen Objekten und Prozessen geknüpft sind, berücksichtigen können. Diese Anforderungen können sowohl einzelne Komponenten als auch Gruppen von ihnen betreffen. Beispiele für Letzteres sind das synchrone Ansteuern mehrerer Antriebskomponenten oder das rechtzeitige Umschalten zwischen redundanten Komponenten im Störfall.
- im Hinblick auf die angestrebten Self-X-Eigenschaften semantisches Wissen über Struktur und Aufgaben des Gesamtsystems sowie Eigenschaften und Fähigkeiten einzelner Komponenten integrieren können,
- der Aufgabenvielfalt einzelner Komponenten Rechnung tragen und diesen den Zugang zu einer Vielzahl von Technologien aus dem IKT-Bereich offenhalten, etwa für den Zugriff auf Datenbanken, die Realisierung multimodaler Benutzeroberflächen oder die Nutzung von Internetdiensten.

2 Stand der Technik

In der Dissertation wurden die genannten Anforderungen von CPPS an den heute in der Automatisierung eingesetzten Technologien gespiegelt und folgende Schlüsse gezogen:

- Die Norm IEC 61131-3 standardisiert Sprachen zur Programmierung automatisierter Abläufe auf einem niedrigen Abstraktionsniveau und bringt daher keine prinzipiellen Restriktionen für die Architektur des Gesamtsystems mit sich, bietet aber andererseits auch keinerlei Unterstützung für spezifische Belange komplexer verteilter Applikationen. Gänzlich ungeeignet ist die auf der Norm aufbauende Produktwelt: Monolithische SPSen eignen sich unter anderem wegen Bauform, Energiebedarf und Preis nicht für die Umsetzung feingranular verteilter CPPS; etablierte Engineering-Werkzeuge sind auf die zentrale Steuerung von Automatisierungsprozessen zugeschnitten.
- Folgerichtig wird mit der Norm IEC 61499 die verteilte Systemgestaltung methodisch und technisch untersetzt. Die Norm abstrahiert von Programmiersprachen hin zu Funktionsbausteinen und beschreibt Automatisierungsanwendungen als Netzwerke solcher Bausteine, die über Daten- und Ereignisleitungen miteinander verknüpft sind. Dabei wird vorwiegend auf grafische Beschreibungsmittel gesetzt, die sich nahtlos in moderne, modellgetriebene Entwurfsabläufe einfügen. Die Norm findet jedoch bis heute kaum Anwendung in der industriellen Automatisierung, was auf eine Reihe von Unzulänglichkeiten zurückgeführt werden kann. Die wesentlichsten sind die teilweise nicht eindeutige Semantik von Programmen [SZCS2011], die auf fixe Punkt-zu-Punkt-Daten- und Ereignisverbindungen beschränkte Kommunikation, die ineffiziente Datenorganisation [WDV2010] sowie die Vernachlässigung des Themas Echtzeitfähigkeit.

Aus diesen Gründen wurde im Rahmen der Dissertation mit Agentensystemen ein alternativer Ansatz zur Realisierung verteilter Automatisierungssysteme untersucht. Agenten sind autonome Softwaremodule, die ihre Umwelt wahrnehmen, in Reaktion auf Ereignisse aus dieser Umwelt oder aufgrund eigener Zielvorgaben handeln und mit anderen Agenten interagieren können. Gemessen an dieser Definition können CPPS per se als Agentensysteme verstanden werden – ein Umstand, der in der Literatur als „natural fit“, also natürliche Passfähigkeit beschrieben wird [B2007]. Ein gewichtiger Vorteil solcher Systeme gegenüber Applikationen auf Basis der IEC 61499 ist die Unterstützung der bei verteilter

Datenorganisation inhärent aufwendigeren Verarbeitung durch entsprechend optimierte Interaktionsmechanismen und Protokolle. Diese erlauben einen flexiblen Informationsaustausch mit zur Entwicklungszeit nicht notwendigerweise festgelegten Kommunikationspartnern. Die dafür notwendigen semantischen Technologien sind ein wesentlicher Bestandteil des Agenten-Paradigmas. Entsprechend ist in der Literatur eine Vielzahl von Beispielen zu finden, bei denen Produktionsschritte und Maschinenfähigkeiten semantisch beschrieben wurden und die Disposition von Fertigungsaufträgen durch Verhandlungen zwischen Produkt- und Maschinenagenten unter Nutzung eben dieser Wissensbasis erfolgt [PP2006a]. Auf die dabei wirkenden Mechanismen wird auf Seite 13 noch einmal genauer Bezug genommen. Wesentlicher Vorteil dieses Vorgehens ist, dass die Entscheidung über die zu verwendende Maschine erst unmittelbar vor der Ausführung eines Produktionsschrittes fällt und damit aktuelle Informationen über die Auslastung aller potentiell geeigneten Maschinen sowie eventuelle Maschinenausfälle einbeziehen kann. Darüber hinaus können Änderungen an den Produkten oder bei der Maschinenausstattung ohne Programmierung, nur durch Anpassung der Wissensbasis, realisiert werden [VROM2009].

Allerdings erlauben existierende Agentenplattformen aufgrund genau dieser komplexen Kommunikations- und Wissensverarbeitungsmechanismen die vollumfängliche Berücksichtigung harter Echtzeitanforderungen nicht und sind daher für die Umsetzung prozessnaher Funktionalität ungeeignet. Oft werden deshalb zweiteilige Architekturen verwendet, die auf oberer Ebene agentenbasiert arbeiten und auf unterer Ebene klassische SPSen einsetzen. Aufgrund des Strukturbruchs zwischen diesen beiden Ebenen ist die Gesamtlösung jedoch weniger praktikabel und es bleiben wesentliche Potentiale der Agententechnologie ungenutzt.

Tabelle 1: Spiegelung bestehender Technologien an den Anforderungen eines CPPS

Anforderung	IEC 61131-3	IEC 61499	Softwareagenten
verteilte Architektur	–	●	●
echtzeitfähige Reaktionen	●	○	–
echtzeitfähige Interaktionen	–	–	–
Integration von semantischem Wissen	–	–	●
Ablage & Verarbeitung komplexer Daten, Zugriff auf IKT-Technologien	–	○	●
durchgängige Architektur ohne Strukturbruch	–	●	–

– nicht erfüllt ○ teilweise erfüllt ● vollständig erfüllt

3 Beitrag der Dissertation

Vor dem eben geschilderten Hintergrund wurde in der Dissertation untersucht, wie Agentensysteme konzipiert werden müssen, um den beteiligten Agenten das Reagieren und Interagieren in Echtzeit zu ermöglichen und mithin allen zuvor genannten Anforderungen cyber-physischer Systeme zu genügen. Die dabei erarbeiteten Ergebnisse lassen sich wie folgt zusammenfassen:

- Zunächst wurde eine echtzeitfähige Laufzeitumgebung für Agenten entworfen, die auf allen Rechnerknoten eines Agentensystems platziert werden muss und den in ihr ablaufenden Agenten die benötigte Infrastruktur bereitstellt.
- Anschließend wurden etablierte Kommunikationsprotokolle für Agenten untersucht und so umgestaltet, dass sie deterministisch und mit möglichst geringen Bandbreitenbedarf ausgeführt werden können. Dabei wurde mit dem Konzept der semantischen Adressierung ein echtzeitfähiges und vielfältig einsetzbares Mittel zur Integration von semantischem Wissen beschrieben.
- Abschließend wurden die geschaffenen Mechanismen in einem Antwortzeitmodell abgebildet, mit dem das rechtzeitige Reagieren eines Agentensystems auf lokal oder verteilt zu behandelnde Ereignisse überprüft und nachgewiesen werden kann.

In den verbleibenden Anschnitten dieses Kapitels werden diese drei Punkte als wesentliche Beiträge der Dissertation weiter untersetzt.

3.1 Echtzeitfähige Laufzeitumgebung für Agenten

Umseitig ist die Architektur der entworfenen Laufzeitumgebung als Klassendiagramm dargestellt. Die Gründe für einen objektorientierten Entwurf wurden in der Dissertation umfangreich erörtert; an dieser Stelle sei besonders die in Sachen Terminologie und Struktur angestrebte Ähnlichkeit zu dem weit verbreiteten, objektorientierten Agenten-Framework JADE[1] hervorgehoben, durch die einerseits die Intuitivität und Praktikabilität des Entwurfs und andererseits die Aussagekraft der erzielten Ergebnisse absichert werden sollte.

[1] Siehe http://jade.tilab.com

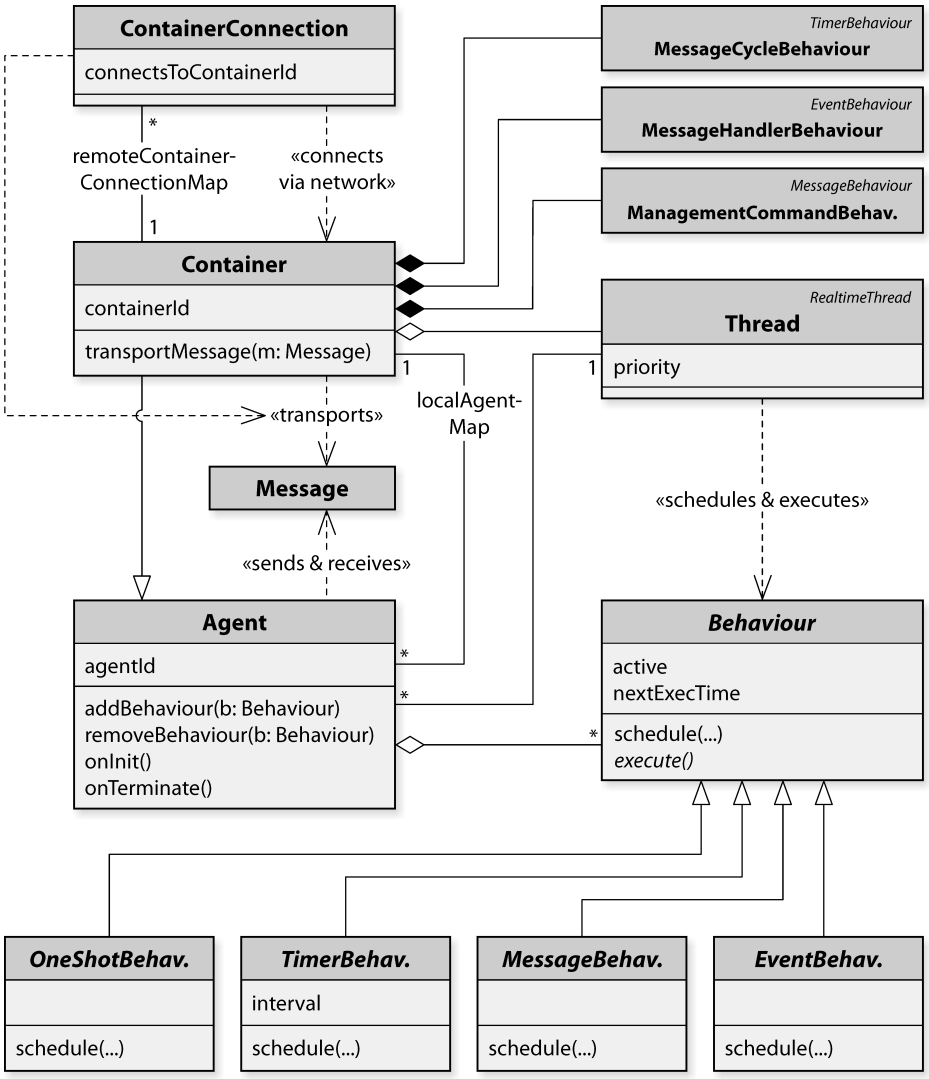

Abbildung 1: Klassendiagramm der in der Dissertation entworfenen Agentenplattform

Das Kernstück der Laufzeitumgebung stellt die Klasse `Container` dar, die in der Regel einmal pro Rechnerknoten instanziiert wird und als Ablaufumgebung für alle Agenten des Knotens dient. In dieser Rolle verwaltet sie die Programmflüsse, also `Thread`-Objekte, zur Ausführung von Agenten und organisiert ferner die Nachrichtenkommunikation zwischen Agenten. Letztere werden durch Objekte der Klasse `Agent` repräsentiert, die selbst vor allem organisatorische Aufgaben haben. Neben ihrer Rolle als Absender oder Adressat von Nachrichten sind sie Träger einer Menge von `Behaviour`-Objekten, die das eigentliche Verhalten des jeweiligen Agenten bestimmen und als in Objekten gekapselte Ereignisbehandlungsroutinen gesehen werden können. Sie verfügen über eine Methode `execute()`, die den Code zur Ereignisbehandlung enthält, sowie eine Methode `schedule()`, die

bestimmt, wie und wann das Behaviour ausgeführt werden soll. Durch geeignete Implementierung dieser Methode oder Ableiten von einer der vorgefertigten Spezialisierungen können Behaviours beispielsweise an eingehende Nachrichten, externe Interrupts oder Timer-Ereignisse gekoppelt werden. Die Vorteile des Behaviour-Konzeptes sind vielseitig. Zum einen folgt es dem Leitsatz „composition over inheritance“, der auf die Steigerung der Flexibilität und Stabilität entsprechender Entwürfe abzielt. Zum anderen besteht zwischen Agenten mit ereignisgesteuert ausgeführten Behaviours und Basisfunktionsbausteinen nach IEC 61499 mit einem ereignisgesteuert ausgeführten Vorrat an Algorithmen eine ausgeprägte Analogie, die die Tauglichkeit des beschriebenen Agentenansatzes für die Realisierung von Automatisierungsfunktionalität unterstreicht. Außerdem nutzt das schon erwähnte Agentenframework JADE Behaviour-Objekte gleichermaßen, um das Verhalten von Agenten umzusetzen (siehe [BCTR2010], S. 21 ff.).

Allerdings weist der Entwurf neben diesen Gemeinsamkeiten mit JADE auch einige charakteristische Eigenheiten auf, die hauptsächlich den spezifischen Gegebenheiten in echtzeitkritischen Anwendungsfeldern geschuldet sind. So ist etwa die Klasse `Container` selbst von `Agent` abgeleitet und kann damit den Behaviour-Mechanismus zur Umsetzung plattforminterner Funktionalität, etwa der getakteten Nachrichtenkommunikation zwischen Containern (`MessageCycleBehaviour` und `MessageHandlerBehaviour`) oder dem Starten und Stoppen von Agenten (`ManagementCommandBehaviour`), einsetzen. Diese effiziente Nutzung ohnehin bereitstehender Funktionalität verringert den Footprint der Plattform und begünstigt so den Einsatz auf ressourcenbeschränkten eingebetteten Rechnerknoten [TVK2009]. Ein weiteres charakteristisches Merkmal ist die Trennung zwischen Agenten und Programmflüssen. Während Agenten bei JADE jeweils über einen eigenen Thread verfügen, in dem sie autonom agieren können, sieht die hier vorgestellte Plattform einen Pool von Threads vor, auf den Agenten entsprechend der ihnen zugeteilten Prioritätsstufe aufgeteilt werden. Damit wird zum einen die Skalierbarkeit der Plattform gegenüber JADE deutlich verbessert und zum anderen dem rein prioritätsbasierten, nicht auf Zeitscheiben beruhenden Scheduling Rechnung getragen, welches von vielen Echtzeitbetriebssystemen sowie der *Real-Time Specification for Java (RTSJ)* [RTSJ2006] eingesetzt wird. In der Dissertation wird ausführlich beschrieben, wie dieses Scheduling-Prinzip von Tasks auf die Ebene der Behaviours übertragen und damit deren deterministische Ausführung sowie die Übertragbarkeit bekannter Antwortzeitmodelle gewährleistet werden kann (siehe Abschnitt 3.3). Da letztendlich auch von entfernten Agenten empfangene Nachrichten durch Behaviours verarbeitet werden, wurde damit nicht nur lokale Echtzeitfähigkeit umgesetzt, sondern darüber hinaus eine wichtige Grundlage für die Realisierung von verteilten Agentensystemen geschaffen, die auch Interaktionen in Echtzeit erlauben.

3.2 Echtzeitfähige Kommunikation zwischen Agenten

Im Hinblick auf ihre Echtzeitfähigkeit sind vor allem jene – für Agentensysteme charakteristischen – Interaktionen kritisch, bei denen die beteiligten Agenten nicht zur Entwicklungszeit festgelegt, sondern zur Laufzeit anhand von semantischem Wissen über Struktur und Aufgaben des Gesamtsystems sowie Eigenschaften und Fähigkeiten einzelner Agenten dynamisch ermittelt werden. Ein Beispiel hierfür ist die Disposition von Fertigungsaufträgen, bei der ein Produktagent mit einer zuvor nicht bekannten Menge von potentiell für den anstehenden Bearbeitungsschritt geeigneten Maschinenagenten in Verbindung tritt. Für solche Szenarien ist in Agentensystemen das in Abbildung 2 dargestellte *Contract Net Protocol (CNP)* [FIPASC00029H] vorgesehen. Dabei schickt der *Initiator* – im Beispiel der Produktagent – einen *Call for Proposals*, also ein Gesuch mit der Beschreibung der zu vergebenden Aufgabe, an alle *Participants* – im Beispiel Maschinenagenten – die zuvor als potentiell geeignete Diensterbringer identifiziert wurden. Diese können die Aufgabe von vornherein ablehnen oder mit einem *Proposal*, also einer Beschreibung ihrer individuellen Rahmenbedingungen für die Übernahme der Aufgabe, antworten. Im Beispiel könnten dabei die jeweilige Maschinenbelegung, eventuell notwendige Rüstzeiten oder anstehende Wartungsarbeiten sowie die durch Nutzung der Maschine anfallenden Kosten einfließen. Anhand dieser Auskünfte trifft der *Initiator* mithilfe einer Metrik zur Gewichtung entscheidungsrelevanter Aspekte die aus seiner Sicht beste Auswahl und benachrichtigt alle beteiligten Akteure entsprechend. Betrachtet man den Gesamtablauf, dann ergeben sich die nachfolgend aufgeführten Nachteile im Hinblick auf die Effizienz eines dergestalt organisierten Automatisierungssystems und mögliche echtzeitkritische Einsatzfälle:

(1) Das CNP setzt die Kenntnis geeigneter Kommunikationspartner voraus. Zuvor muss also gegebenenfalls der *Directory Facilitator (DF)* – ein Verzeichnisdienst – kontaktiert werden, der anhand einer meist ontologiebasierten Beschreibung die Adressen entsprechender Agenten liefert. Dadurch wird sowohl das Verkehrsaufkommen im Netzwerk als auch die Zeitdauer bis zur Vergabe der Aufgabe erhöht.

(2) Der Directory Facilitator ist eine kritische Systemkomponente, deren Ausfall die Funktionalität des gesamten Agentensystems maßgeblich beinträchtigen würde (*Single Point of Failure*). Insbesondere beim feingranularen Einsatz von Agenten stellt ein einzelner DF außerdem einen Flaschenhals dar. Mehrere sogenannte *Federated DF* beheben diese Problem teilweise, können aber Konsistenzprobleme nach sich ziehen und weisen ein schwer vorhersehbares Zeitverhalten auf.

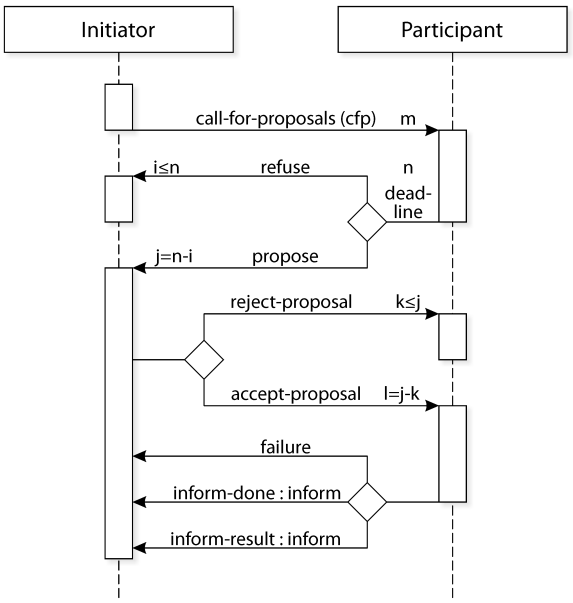

Abbildung 2: *Contract Net Protocol* nach [FIPASC00029H]

(3) Sämtliche Nachrichten werden an symbolische Bezeichner adressiert, die für den Nachrichtentransport zunächst in Netzwerkadressen umgesetzt werden müssen. Dies bedeutet wiederum schwer vorhersehbare Verzögerungen und gegebenenfalls zusätzlichen Netzwerkverkehr.

(4) Jeder Agent, der im Rahmen des CNP einen *Proposal* versendet, ist bis zum Abschluss des Protokolls an diesen gebunden und muss gegebenenfalls entsprechende Ressourcen zur Erfüllung des *Proposals* reservieren, bis entweder *accept-proposal* oder *reject-proposal* empfangen wird. Das kann besonders bei etwa zeitgleich durch verschiedene Agenten angestoßene Instanzen des CNP zu suboptimaler Vergabe der Aufgabe führen.

(5) Gemäß Abbildung 2 werden pro Instanz des CNP m identische *cfp*-Nachrichten verschickt, die von insgesamt n *refuse*- oder *proposal*-Nachrichten beantwortet werden. Hinzu kommen insgesamt j Antworten auf *Proposals* und eine Nachricht nach Abschluss der Bearbeitung der vergebenen Aufgabe. Nur in selten Fällen antworten *Participants* gar nicht oder mit einer *refuse*-Nachricht, da ohnehin nur qualifizierte Diensterbringer kontaktiert werden. Entsprechend liegt das gesamte Nachrichtenaufkommen in der Größenordnung $3 \cdot m$.
Da der *Initiator* auf Antworten von allen *Participants* oder das Verstreichen der Deadline wartet, hat jeder *Participant* unmittelbaren Einfluss auf das Zeitverhalten des Protokolls und macht dieses schwer vorhersehbar.

(6) Keine der dem Autor bekannten Agentenplattformen sieht die Zeitsynchronisation zwischen den einzelnen Knoten vor, welche zwingende Voraussetzung für koordiniertes echtzeitfähiges Handeln wäre. Aus diesem Grund muss die Deadline zur Abgabe von *Proposals* im CNP sehr großzügig bemessen werden, um Fehler im Protokollablauf zu verhindern.

Um diese Nachteile gezielt und im Einklang mit den Fähigkeiten derzeit verfügbarer Kommunikationstechnologien adressieren zu können, wurde zunächst ein Netzwerkmodell erarbeitet, welches sich an modernen, Ethernet-basierten Feldbussen orientiert und deren Eigenschaften abstrakt wie folgt beschreibt:

(A1) Die Netzwerkkommunikation erfolgt in Buszyklen mit der Dauer t_{cycle}.

(A2) Jeder Knoten *i* kann zu Beginn jedes Buszyklus Daten der Größe *fd(i)* absenden. (Die Datenmenge muss nicht zwangsläufig für jeden Knoten gleich sein.)

(A3) Alle versendeten Daten werden auf dem gesamten Netzwerkbus sichtbar. Sie können an einen oder an alle anderen Knoten (*Broadcast*) adressiert werden.

(A4) Am Ende jedes Zyklus empfangen alle Knoten die an sie adressierten Daten.

(A5) Die Uhren aller Knoten sind mit einer Genauigkeit deutlich unterhalb der Buszykluszeit synchronisiert.

Aufbauend auf diesem Netzwerkmodell wurde in der Dissertation das Konzept der *semantischen Adressierung* beschrieben. Dabei werden sämtliche Eigenschaften und Fähigkeiten, die einen Agenten als potentiellen Kommunikationspartner charakterisieren, in einer Taxonomie, ähnlich der nachfolgend abgebildeten, klassifiziert. Wird nun im Rahmen des

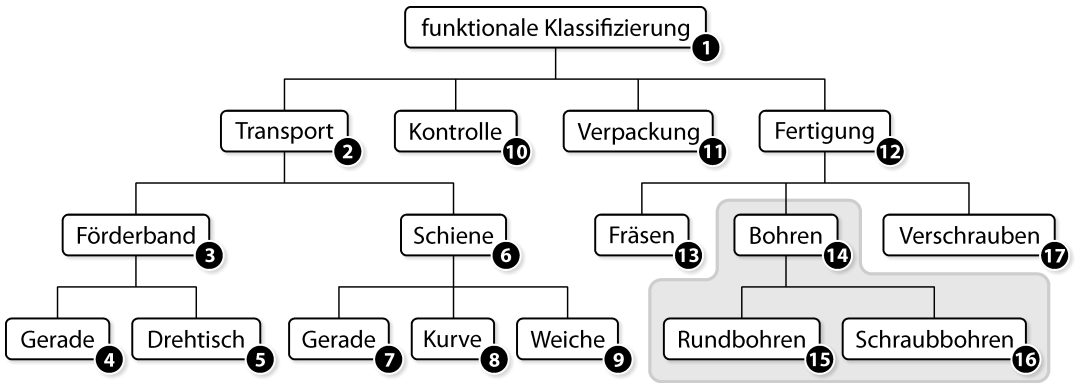

Abbildung 3: Beispiel-Taxonomie für maschinelle Ausstattung einer Fertigungsanlage

CNP eine bestimmte Fähigkeit nachgefragt, beispielsweise das Vermögen zum schienengebundenen Transport, dann sind gemäß der abgebildeten Taxonomie alle Agenten mit einer Eigenschaft aus dem Teilbaum *Schiene* Ansprechpartner; müsste gebohrt werden, dann wären dafür Agenten mit einer Eigenschaft aus dem Teilbaum *Bohren* geeignet. Nimmt man eine *Tiefenindizierung*[2] des Taxonomie-Baums vor, dann kann eine nachgefragte Eigenschaft eindeutig durch den *Taxonomie-Index (TIX)* der Wurzel des entsprechenden Teilbaums identifiziert werden. So umfasst etwa der Teilbaum *Bohren* im Beispiel die Elemente [14, 16] und kann eindeutig durch den TIX 14 referenziert werden. Demnach kann der TIX als eine semantische Gruppenadresse für alle Agenten mit einer Eigenschaft aus diesem Teilbaum verstanden werden. In der Dissertation wird beschrieben, wie sich das Konzept weiter ausgebaut lässt, sodass auch sogenannte multilaterale semantische Adressen formuliert werden können. Diese erlauben die Verknüpfung mehrerer als TIX codierter Eigenschaften, die allesamt bei jedem adressierten Agenten vorliegen müssen. So kann, wenn die beispielhaft verwendete Wissensbasis gemäß Abbildung 4 erweitert wird, etwa die Gruppe jener Maschinenagenten adressiert werden, die im Nordteil von Halle 2 zum Bohren von Aluminiumwerkstücken in der Lage sind.

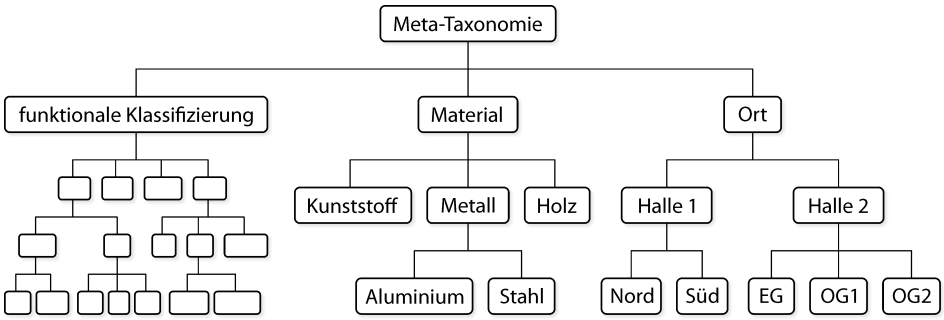

Abbildung 4: Meta-Taxonomie für multilaterale semantische Adressierung

Das zuvor formulierte Netzwerkmodell stellt mit seinen Annahmen (A3) und (A4) sicher, dass semantisch adressierte Nachrichten ohne Mehraufwand auf Seite des Netzwerkbusses allen Containern eines Agentensystems zugestellt werden können. In der Dissertation wird darüber hinaus gezeigt, dass sich die Verarbeitung wie beschrieben aufgebauter semantischer Adressen und letztlich die Zustellung der entsprechenden Nachrichten algorithmisch effizient und damit echtzeitzeitfähig realisieren lässt.

[2] Der Begriff *Tiefenindizierung* wurde in Anlehnung an die verbreitete Bezeichnung *Tiefensuche* gewählt und meint eine Zuweisung von Indizes in der folgenden Reihenfolge: Elternknoten, anschließend Teilbäume von links nach rechts (englisch: depth first, pre-order)

Neben der semantischen Adressierung wird in der Dissertation außerdem beschrieben, wie sich die Entscheidungsmetrik, anhand der der Initiator beim CNP die eingehenden Proposals miteinander vergleicht, im Call for Proposals codieren lässt. Damit können die pro Container erstellten Proposals bereits vor Ort miteinander verglichen werden, und es muss nur der jeweils beste an den Initiator zurückgesendet werden. Dies führt zu einer Reduktion der zu versendeten Netzwerknachrichten und minimiert die Menge der zur Erfüllung abgegebener Proposals gegebenenfalls umsonst vorgehaltenen Ressourcen.

Mithilfe der beiden beschriebenen Konzepte lässt sich für das CNP ein modifizierter Gesamtablauf realisieren, der alle zuvor genannten Kritikpunkte (1) bis (6) ausräumt. In

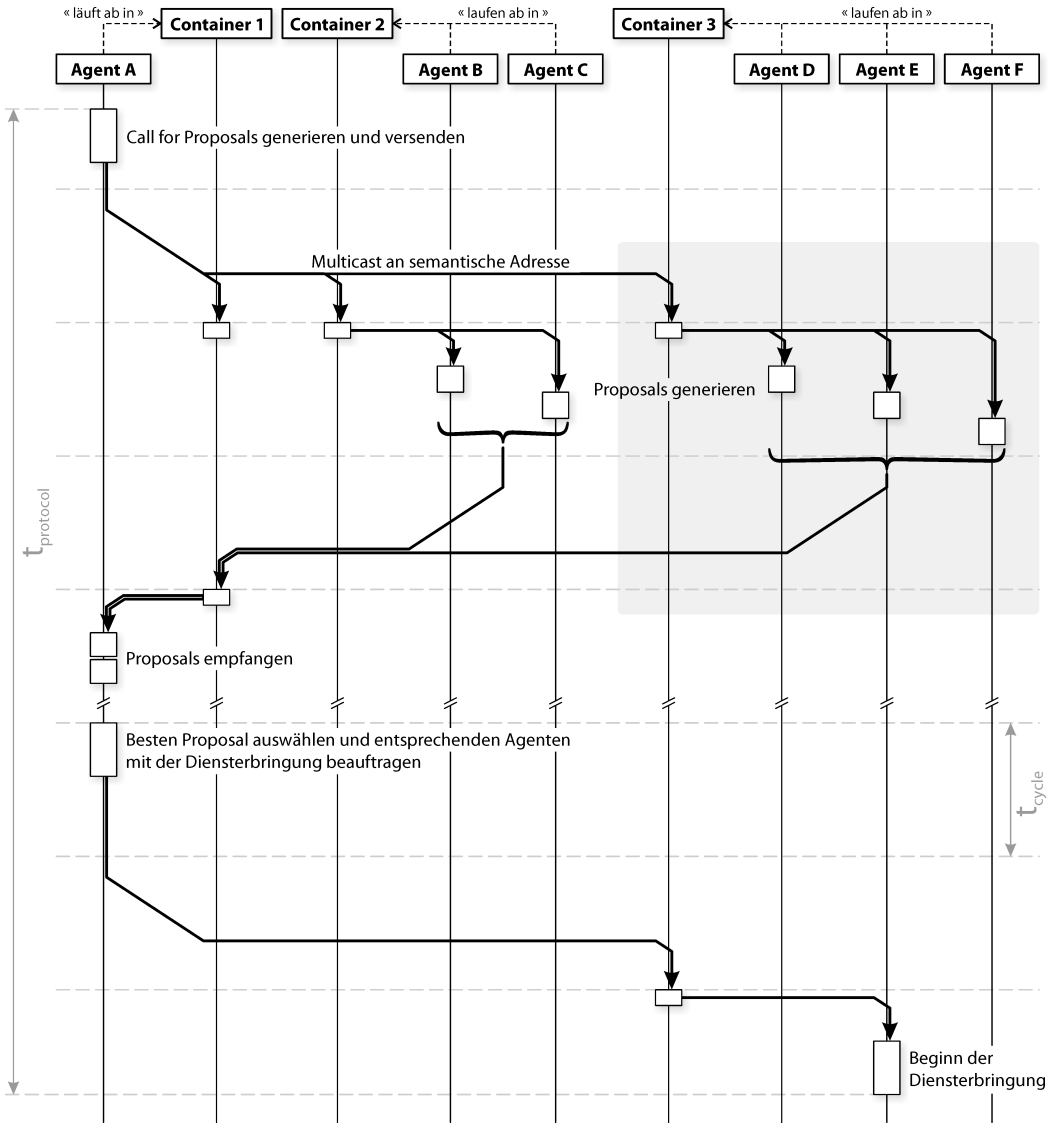

Abbildung 5: Detaillierter Kommunikationsablauf für überarbeitetes Contract Net Protocol

Abbildung 5 ist dieser modifizierte Ablauf beispielhaft dargestellt. Zunächst sendet ein Agent A als Initiator einen Call for Proposals samt kodierter Entscheidungsmetrik an eine per semantischer Adresse spezifizierte Untermenge von Agenten. Gemäß dem zugrunde gelegten Netzwerkmodell erreicht dieser Call alle beteiligten Container spätestens am Ende des nächsten vollständigen Buszyklus und löst dort zunächst die `MessageHandlerBehaviour`-Objekte der Container aus (siehe Abschnitt 3.1), welche die semantische Adresse auflösen und den Call an die lokalen Agenten A bis F weiterleiten. Diese bereiten jeweils einen zum Call passenden Proposal vor, von denen dann der pro Container bezüglich der anfangs übermittelten Entscheidungsmetrik beste zurück an den Initiator gesendet wird. Nach spätestens einem weiteren Netzwerkbuszyklus erreichen die Proposals zunächst das `MessageHandlerBehaviour` des ersten Containers und anschließend ein `MessageBehaviour` des Initiators, welches die Proposals zunächst nur entgegennimmt. Nach dem Verstreichen der anfangs mit dem Call übermittelten Deadline für Proposals wird anschließend der beste Proposal auswählt und dem zugehörigen Agenten, im Beispiel Agent E, die ausgelobte Aufgabe übertragen.

Der so modifizierte Gesamtablauf zeichnet sich neben der Beseitigung der genannten Kritikpunkte vor allem durch sein vorhersehbares Zeitverhalten aus. Offen ist bisher noch, inwiefern andere Formen der Agenteninteraktion von den vorgestellten Konzepten profitieren können. Dazu kann zunächst angemerkt werden, dass durch die direkte semantische Adressierbarkeit von Agenten vermittelnde Instanzen und die damit einhergehenden Protokolle wie das *Brokering Interaction Protocol* und das *Recruiting Protocol* obsolet werden. Die übrigen von der FIPA standardisierten Protokolle [P2007] sehen, mit Ausnahme des CNP, eine bilaterale Kommunikation zwischen einander bekannten Kommunikationspartnern vor, die in der Regel über logische Adressen der Form *(ContainerId, AgentId)* abgewickelt werden kann. Eine Ausnahme stellt die Kommunikation mit mobilen Agenten dar. Diese können jedoch individuell über semantische Adressen angesprochen werden, indem für jeden mobilen Agenten eine ihn kennzeichnende, unikale Eigenschaft im Taxonomie-Baum vorgesehen wird. Darüber hinaus kann mithilfe der semantischen Adressierung auch nach dem *Publish-Subscribe*-Prinzip kommuniziert werden, bei dem jeweils Nachrichten eines bestimmten Typs an eine Gruppe dafür registrierter Abonnenten verteilt werden. Dazu muss die Mitgliedschaft in der Abonnentengruppe jeweils als Eigenschaft im Taxonomie-Baum abgebildet werden. Zusammenfassend lässt sich sagen, dass die entwickelten Konzepte die Agentenkommunikation insgesamt bereichern; ihr größter Nutzen ergibt sich jedoch bei der Kontaktaufnahme mit zuvor nicht feststehenden Kommunikationspartnern. Diese Art der Kommunikation stellt allerdings einen zentralen Charakterzug von Agentensystemen dar.

3.3 Antwortzeitmodell für verteilte Agentensysteme

Abschließend wird nun skizziert, wie sich die erarbeiteten Konzepte in einem Antwortzeitmodell abbilden lassen, mit dem die Reaktionszeit für lokal oder verteilt zu behandelnde Ereignisse ermittelt und so die Echtzeitfähigkeit eines Agentensystems überprüft und nachgewiesen werden kann. Voraussetzung für diese Modellierung ist eine formale Beschreibung des betrachteten Agentensystems. Diese schließt die Beschreibung aller Behaviours ein, ebenso deren Zuordnung zu Agenten und deren Platzierung in Containern. Zusätzlich muss die Gesamtheit aller auftretenden Ereignisse und die jeweils durch sie angestoßene Folge von Behaviours und Netzwerknachrichten formalisiert werden. Auf Details dieser Formalisierung wird an dieser Stelle nicht weiter eingegangen; in der Dissertation sind sie umfangreich dargestellt.

Die Modellierung der Antwortzeit fußt auf der Idee, dass sich verteilte Ereignisbehandlungen in einander abwechselnde Phasen der (verteilten) Abarbeitung von Behaviours und des Versendens von Netzwerknachrichten einteilen lassen. Aus Abbildung 5 auf Seite 16 geht dies anschaulich für das Contract Net Protocol hervor. Lokal behandelte Ereignisse stellen einen trivialen Fall dar, der nur aus einer Phase der Behaviour-Abarbeitung besteht, und für den alle nachfolgend getroffenen Aussagen gleichermaßen gelten. In Bezug auf die Ausführungszeit einer Ereignisbehandlung lassen sich aus dieser Einteilung in alternierende Phasen folgende Aussagen ableiten:

- Für die maximale Ausführungszeit (*Worst Case Execution Time, WCET*) einer Ereignisbehandlung stellt die Summe der maximalen Ausführungszeiten der einzelnen Phasen eine obere Schranke dar.
- Für die Bestimmung der WCET einer Behaviour-Phase müssen die Behaviours der betrachteten Ereignisbehandlung im Kontext der von allen übrigen Ereignisbehandlungen erzeugten maximalen Behaviour-Last betrachtet werden. Analog muss bei der Bestimmung der maximalen Übertragungszeit (*Worst Case Transmission Time, WCTT*) von Netzwerknachrichten der Kontext der von allen übrigen Ereignisbehandlungen erzeugten maximalen Nachrichtenlast berücksichtigt werden.
- Die maximal von einer Ereignisbehandlung erzeugte Behaviour- und Nachrichtenlast ergibt sich bei der schnellstmöglichen Abarbeitung derselben. Durch die serielle Abarbeitung der alternierenden Phasen einer Ereignisbehandlung entstehen Sequenzen von Behaviour- und Nachrichtenlasten, deren minimaler zeitlicher Abstand jeweils ein ganzzahliges Vielfaches der Buszykluszeit t_{cycle} ist.

Zur Berechnung der WCET einer Behaviour-Phase wird in der Dissertation ein lokales Antwortzeitmodell beschrieben, mit dem pro Container die maximale *Summenausführungszeit* der in der Phase enthaltenen Behaviours unter Berücksichtigung des von allen übrigen Ereignisbehandlungen vorgegebenen Kontextes bestimmt werden kann. Da die in Abschnitt 3.1 beschriebene Laufzeitumgebung prioritätsbasiertes Scheduling (fixed-priority preemptive scheduling) von Behaviours umsetzt, wurde das in [JP1986] für ebendieses Scheduling-Verfahren beschriebene analytische Modell als Basis für die Bestimmung der lokalen Antwortzeiten herangezogen:

$$R_i = C_i + \sum_{x \in \mathrm{hp}(i)} \left\lceil \frac{R_i}{T_x} \right\rceil C_x$$

Dieses Modell ermittelt die maximale Reaktionszeit R_i eines Tasks i als Summe aus dessen maximaler Ausführungszeit C_i und der maximalen Verzögerung, die durch die in $\mathrm{hp}(i)$ enthaltenen Tasks mit höherer Priorität und einer Zwischenankunftszeit T_x hervorgerufen werden kann. Bei der Übertragung dieses Modells von Tasks auf Behaviours muss berücksichtigt werden, dass bei Behaviours keine Totalordnung der Prioritäten vorliegt und mithin auch gleich priorisierte Behaviours verzögernd wirken können. Nachfolgend wird die Menge der Behaviours mit potentiell verzögerndem Einfluss in Anlehnung an die vorangegangenen Betrachtungen mit $context$ bezeichnet. Zusätzlich berücksichtigt die nächste Iteration des Modells mit dem Summanden B_i Blockierungen durch Synchronisationsmechanismen und, durch den Übergang von C_i zu $\hat{C}_i$, auch den Scheduling-Overhead; auf beides soll an dieser Stelle aber nicht genauer eingegangen werden.

$$R_i = \hat{C}_i + B_i + \sum_{x \in context} \left\lceil \frac{R_i}{T_x} \right\rceil \hat{C}_x$$

Soll nicht die Reaktionszeit eines einzelnen Behaviours, sondern die Summenreaktionszeit einer Menge $\{b_i\}$ von Behaviours bestimmt werden, dann ist der Berechnungsansatz wie folgt zu modifizieren:

$$R_{\{b_i\}} = \sum_i \hat{C}_{b_i} + \max_i B_{b_i} + \sum_{x \in context} \left\lceil \frac{R_{\{b_i\}}}{T_x} \right\rceil \hat{C}_x$$

Die Korrektheit dieser Formel und die genauen Rahmenbedingungen für ihre Anwendung werden in der Dissertation ausführlich diskutiert. Nach wie vor enthält die Kontextmenge allerdings nur Behaviours, deren Last potentiell zeitgleich mit der Aktivierung der in $\{b_i\}$

enthaltenen Behaviours wirkt. Um die zuvor beschriebenen Lastsequenzen abbilden zu können, ist eine zusätzliche Erweiterung des Modells notwendig:

$$R_{\{b_i\}} = \sum_i \hat{C}_{b_i} + \max_i B_{b_i} + \sum_{x \in context} \left\lceil \frac{\max(0, R_{\{b_i\}} - \Delta_x)}{T_x} \right\rceil \hat{C}_x$$

In dieser endgültigen Variante des lokalen Antwortzeitmodells bezeichnet Δ_x die minimale Zeitverzögerung des Behaviours x aus der Kontextmenge gegenüber dem ersten Behaviour aus $\{b_i\}$. Wie sich die Parameter Δ_x und T_x für die Behaviours einer Ereignisbehandlung bestimmen lassen, ist in der Dissertation umfassend beschrieben.

Für die Bestimmung der maximalen Summenübertragungszeit einer Menge $\{m_i\}$ von Nachrichten wurde analog ein ähnliches Modell aufgestellt. Mit $Q_{\{m_i\}}$ wird zunächst die maximale Anzahl der für die Übertragung benötigten Buszyklen ermittelt. d beziffert die pro Buszyklus und Knoten maximal übertragbare Datenmenge. Durch Multiplikation mit der Buszykluszeit kann aus $Q_{\{m_i\}}$ anschließend die Übertragungszeit ermittelt werden.

$$Q_{\{m_i\}} = \left\lceil \left(\sum_{x \in context} \left\lceil \frac{\max(0, Q_{\{m_i\}} \cdot t_{cycle} - \Delta_x)}{T_x} \right\rceil \cdot size_x + \sum_i size_{m_i} \right) / d \right\rceil$$

Auch hier sei für weitere Details, insbesondere die Bestimmung der Parameter Δ_x und T_x betreffend, auf die vollständigen Ausführungen in der Dissertationsschrift verwiesen.

4 Validierung und Evaluierung

Als Nachweis für die praktische Realisierbarkeit der erarbeiteten Konzepte wurde eine Referenzimplementierung der entworfenen Plattform angefertigt. Dafür wurde die von der Firma aicas zur Verfügung gestellte JamaicaVM – eine RTSJ-konforme Echtzeit-Java-VM – an die Echtzeiterweiterung *Xenomai* [3] für Linux angepasst und für die Nachrichtenkommunikation der echtzeitfähige, Ethernet-basierte Netzwerkstack *RtNet* [4] [KWZYB2005] eingebunden. Die resultierende Struktur dieser Referenzimplementierung ist in Abbildung 6 dargestellt.

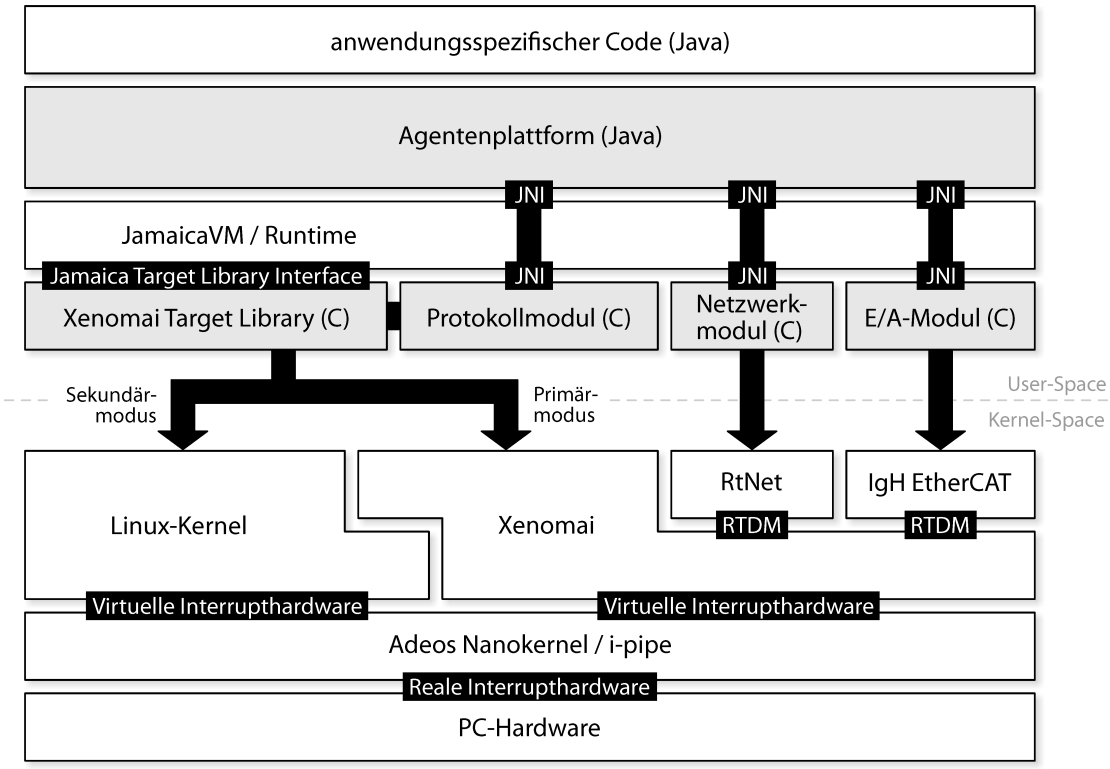

Abbildung 6: Struktur der Referenzimplementierung (Eigenentwicklungen in grau)

[3] http://xenomai.org

[4] Siehe http://rtnet.org

Mit dieser Referenzimplementierung wurde die Voraussetzung für praktische Antwortzeitmessungen geschaffen, die einen Vergleich mit den Resultaten des zuvor beschriebenen Antwortzeitmodells erlauben. Die Durchführung solcher Messungen und die dabei ermittelten Messergebnisse sind in der Dissertation ausführlich beschrieben; nachfolgend sind repräsentativ die Resultate für ein künstlich konstruiertes Worst-Case-Szenario dargestellt, bei dem eine wachsende Anzahl *n* von Instanzen des Contract Net Protocols mit elf beteiligten Agenten, verteilt auf drei Container, zeitgleich ausgelöst wurden.

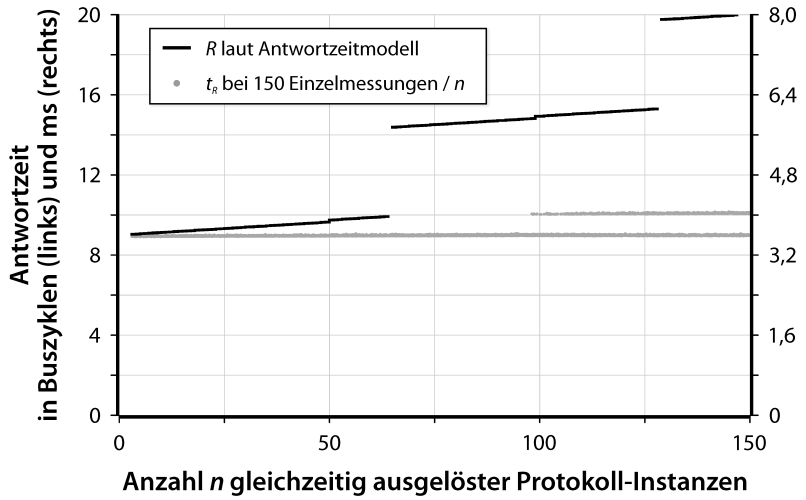

Abbildung 7: Ergebnisse der Antwortzeitmessung bei paralleler Ausführung von *n* Instanzen des Contract Net Protocols

Über die Gesamtheit der durchgeführten Messungen lässt sich wie folgt resümieren: Alle Messungen zeigen deutlich nach oben begrenzte Antwortzeiten und liefern damit zwar keinen endgültigen Beweis, aber ein starkes Indiz für die Echtzeitfähigkeit von auf Basis der entwickelten Plattform realisierten Agentensystemen. Die Vergleiche zwischen gemessenen und von dem Modell berechneten Antwortzeiten zeigen für kleine und mittelgroße Agentensysteme deutliche Übereinstimmungen, die die Korrektheit des Antwortzeitmodells und mithin auch die Echtzeitfähigkeit der Agentenplattform weiter untermauern. Die bei großem Nachrichtenaufkommen auftretenden Abweichungen konnten durchgängig erklärt werden und stellen die Validität des Antwortzeitmodells nicht in Frage.

In der Dissertation wird darüber hinaus auch die Praktikabilität der entworfenen Agentenplattform und des Antwortzeitmodells ausführlich diskutiert. Aus Platzgründen wird an dieser Stelle auf eine Zusammenfassung dieser Diskussion verzichtet und auf die vollständige Arbeit verwiesen.

5 Zusammenfassung und Ausblick

Im Rahmen der Dissertation wurde eine echtzeitfähige Laufzeitumgebung für Softwareagenten entworfen und anschließend die Überarbeitung bestehenden Kommunikationsmechanismen im Hinblick auf ihre Echtzeitfähigkeit vorgenommen. In diesem Kontext wurde mit dem Konzept der semantischen Adressierung eine vielfältig einsetzbare Möglichkeit geschaffen, Nachrichten an ausgewählte Gruppen von Agenten mit bestimmten, semantisch beschriebenen Eigenschaften zur verschicken. Die dabei zur Wissensrepräsentation genutzten Taxonomie-Bäume bieten ein für viele Aufgabenstellungen ausreichendes Maß an Ausdrucksstärke und erlauben zudem die Verarbeitung unter harten Echtzeitbedingungen. Damit wurde ein tragfähiger Ansatz geschaffen, der die in Kapitel 2 formulierten Anforderungen von CPPS vollumfänglich erfüllt.

Darüber hinaus wurden die geschaffenen Mechanismen in einem Antwortzeitmodell abgebildet, mit dem das rechtzeitige Reagieren eines Agentensystems auf lokal oder verteilt zu behandelnde Ereignisse überprüft und nachgewiesen werden kann. Durch den Vergleich von an einer Referenzimplementierung gemessenen und durch das Modell berechneten maximalen Antwortzeiten konnte die Echtzeitfähigkeit der entwickelten Plattform und die Korrektheit des Antwortzeitmodells bestätigt werden. Damit wurde ein Hauptkritikpunkt von Agentensystemen adressiert, was zu einer nachhaltigen Steigerung der Akzeptanz des Agentenparadigmas führen könnte.

Während große Teile der erarbeiteten Lösung als allgemeingültige Grundlagenforschung verstanden werden können, wurde bei der Formulierung von Anforderungen, der Darstellung von Beispielen und der Erläuterung von Entwurfsentscheidungen immer wieder auf automatisierungstechnische Belange Bezug genommen. Außerdem wurde am Ende der Arbeit eine kritische Bewertung der Ergebnisse vor dem Hintergrund eines möglichen Einsatzes in zukünftigen Automatisierungssysteme durchgeführt und so ein stimmiges Gesamtbild präsentiert.

Dennoch legt diese Arbeit nur einen Grundstein für die agentenorientierte Realisierung cyber-physischer Produktionssysteme und soll weitere, stärker anwendungsorientierte Forschungsbeiträge anregen. So muss beispielsweise genauer untersucht werden, wo dem Konzept der semantischen Adressierung durch die begrenzte Ausdrucksfähigkeit von Taxonomien Grenzen gesetzt sind und wie diese durch Kombination mit anderen Technologien

überwunden werden können. Für die in der Dissertation beschriebene Referenzimplementierung wurde aus dort beschriebenen technischen Gründen auf Desktop-PCs als Rechnerknoten zurückgegriffen. Überdies wurden nur Benchmark-Szenarien praktisch umgesetzt. Zukünftig sollte eine auf den Anwendungsfall CPPS zugeschnittene eingebettete Hardwareplattform geschaffen und für die Automatisierung realer Versuchsanlagen genutzt werden. Als Ausgangspunkt könnte die Gnublin-Plattform[5] dienen, die mit halbleitergetriebene 24V-Ein- und Ausgängen sowie einem dedizierten Netzwerkcontroller ausgestattet werden kann und damit geringere technischen Hürden als andere kostengünstige Embedded-Plattformen aufweist. Perspektivisch wäre auch eine Hardwareplattform mit nativer Java-Unterstützung interessant. Darüber hinaus erscheint die Schaffung einer Werkzeug-Infrastruktur zweckmäßig. So ist etwa ein grafischer Editor für Interaktionsprotokolle denkbar, der automatisch das Codegerüst für die zur Umsetzung der Protokolle notwendigen Behaviours und außerdem die formalen Beschreibungen zur Antwortzeitanalyse erzeugt. Überdies wurden im Rahmen der Dissertation mehrfach die Vorteile der Java-Plattform bei der Realisierung möglichst universeller CPPS-Knoten dargestellt, im Zuge der abschließenden kritischen Diskussion aber auch bestehende Nachteile beleuchtet. Zukünftige Arbeiten könnten an diese Betrachtung anknüpfen und überprüfen, inwieweit eine für Automatisierungsanwendungen spezialisierte Variante von Java die beschriebenen Nachteile ausräumen könnte. Alternativ sollte eine weitere Überarbeitung der IEC 61131-3 in Betracht gezogen werden, mit der die Norm noch besser für den Einsatz in zukünftigen Automatisierungssystemen gerüstet werden und als technische Basis für die hier beschriebene oder eine vergleichbare Agentenplattform dienen könnte.

[5] Siehe http://gnublin.embedded-projects.net

Literaturverzeichnis

[B2007] Brennan, R. W.: Toward Real-Time Distributed Intelligent Control: A Survey of Research Themes and Applications. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 37, No. 5, Sep. 2007, S. 744-765.

[BCTR2010] Bellifemine, F.; Caire, G.; Trucco, T. und Rimassa, G.: *JADE Programmer's Guide*. Telecom Italia S.p.A., 2010. Online: http://jade.tilab.com/doc/programmersguide.pdf

[FIPASC00029H] *FIPA SC00029H – FIPA Contract Net Interaction Protocol Specification*, 2002.

[HOF+2014] Holm, T.; Obst, M.; Fay, A.; Urbas, L.; Albers, T.; Kreft, S. und Hempen, U.: Dezentrale Intelligenz für modulare Automation, Lösungsansätze für die Realisierung modularer Anlagen. In: *Automatisierungstechnische Praxis (atp edition)*, Vol. 56, No. 11, Nov. 2014, S. 34-43.

[JP1986] Joseph, M. und Pandya, P.: Finding Response Times in a Real-Time System. In: *The Computer Journal*, Vol. 29, No. 5, 1986, S. 390-395.

[KWZYB2005] Kiszka, J.; Wagner, B.; Zhang Yuchen und Broenink, J.: RTnet – A Flexible Hard Real-Time Networking Framework. In: *Proc. 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2005, S. 449-456.

[P2007] Poslad, S.: Specifying Protocols for Multi-Agent Systems Interaction. In: *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 2, No. 4, 2007, S. 15:1-15:24.

[PP2006a] *State of the Art in Factory Control and Requirement Specification for most advanced most flexible, and most adaptable control of manufacturing systems*. Deliverable 1.1, PABADIS'PROMISE consortium, FP6-IST-016649, 2006.

[RTSJ2006] *The Real-Time Specification for Java*. RTSJ 1.0.2, 2006. Online: http://www.rtsj.org/docs/rtsj_1.0.2_spec.pdf

[SZCS2011] Strasser, T. I.; Zoitl, A.; Christensen, J. H. und Sünder, C.: Design and Execution Issues in IEC 61499 Distributed Automation and Control Systems. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 41, No. 1, Jan. 2011, S. 41-51.

[TVK2009] Theiss, S.; Vasyutynskyy, V. und Kabitzsch, K.: Software Agents in Industry: A Customized Framework in Theory and Praxis. In: *IEEE Transactions on Industrial Informatics*, Vol. 5, No. 2, April 2009, S. 147-156.

[VROM2009] Vrba, P.; Radakovič, M.; Obitko, M. und Mařík, V.: Semantic Extension of Agent-Based Control: The Packing Cell Case. In: *Proc. 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS)*, 2009, S. 47-60.

[WDV2010] Wenbin Dai und Vyatkin, V.: On Migration from PLCs to IEC 61499: Addressing the Data Handling Issues. In: *Proc. 8th IEEE International Conference on Industrial Informatics (INDIN)*, 2010, S. 1142-1147.