Lehrstuhl für
Fördertechnik Materialfluss Logistik
der Technischen Universität München

Kommunikations- und Steuerungsstrategien für das Internet der Dinge

Dipl.-Inf. Univ. Razvan Chisu

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzende:

Prof. Dr.-Ing. Birgit Vogel-Heuser

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. Willibald A. Günthner
2. Univ.-Prof. Dr. med. Michael ten Hompel,
   Technische Universität Dortmund

Die Dissertation wurde am 29.10.2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 05.02.2010 angenommen.

Herausgegeben von:

Univ.-Prof. Dr.-Ing. Willibald A. Günthner
fml – Lehrstuhl für Fördertechnik Materialfluss Logistik
Technische Universität München
Zugleich: Dissertation, TU München, 2010

Layout und Satz: Razvan Chisu

ISBN: 978-3-941702-05-9

Printed in Germany 2010

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Fördertechnik Materialfluss Logistik (fml) der Technischen Universität München.

Mein Dank gilt in erster Linie Herrn Prof. Dr. Willibald A. Günthner, der mir durch sein Vertrauen und seine Unterstützung die Durchführung meiner Forschungstätigkeit ermöglicht hat. Weiterhin danke ich Prof. Dr. Michael ten Hompel für die Übernahme des Koreferats und Prof. Dr.-Ing. Birgit Vogel-Heuser für den Vorsitz der Prüfungskommission.

Dem Bundesministerium für Bildung und Forschung danke ich für die Förderung des Projekts „Internet der Dinge“, im Rahmen dessen die meisten hier vorgestellten Ideen entstanden sind. Auch den Projektpartnern danke ich für die stets gute Zusammenarbeit.

Weiterhin möchte ich mich bei allen Mitarbeiterinnen und Mitarbeitern des Lehrstuhls fml bedanken – in erster Linie bei meinen Bürokollegen Florian Kuzmany, Michael Schipplick und Peter Tenerowicz für die angenehmen Stunden im Südbau und die wertvolle Hilfe bei der Inbetriebnahme der Roboterzelle und der Elektrohängebahnanlage.

Mein persönlicher Dank gilt vor allem meinen Eltern, ohne deren langjährige und geduldige Unterstützung in allen Lebensbereichen all dies nicht möglich gewesen wäre.

Garching, im Oktober 2009 Razvan Chisu

Kurzzusammenfassung

Kommunikations- und Steuerungsstrategien für das Internet der Dinge

Razvan Chisu

Heutige Materialflusssysteme müssen immer höheren Anforderungen gerecht werden: Die steigende Variantenvielfalt bei Konsumgütern sowie die immer kürzer werdenden Innovations- und Produktlebenszyklen führen zu schwer prognostizierbaren Auftragslasten und -strukturen, die auftragsbezogene Produktion gewinnt immer mehr an Bedeutung. Vor diesem Hintergrund müssen innerbetriebliche Materialflusssysteme dynamisch reagieren und ggf. schnell umgebaut werden können und dabei möglichst geringe Kosten verursachen.

Herkömmliche automatisierte Systeme sind aber kaum in der Lage, diesen Anforderungen gerecht zu werden, denn die klassische, zentrale Steuerungsarchitektur ist zum einen hochkomplex und muss zum anderen projektspezifisch ausgelegt werden, womit sie gegenüber Änderungen sehr unflexibel ist.

Ziel der vorliegenden Arbeit ist die Entwicklung technischer Grundlagen für eine neuartige, dezentrale Materialflusssteuerung. Das Internet als verteiltes Datennetzwerk dient dabei als Vorbild für die Schaffung eines „Internet der Dinge“, in dem intelligente Fördertechnik und Transporteinheiten in der Lage sind, kooperativ und hierarchielos den Materialfluss zu steuern.

Die Systemarchitektur besteht aus standardisierten, erweiterbaren und aufeinander aufbauenden Komponenten bzw. Entitäten, sieht aber gleichzeitig Möglichkeiten vor, um das Zusammenspiel dieser Bausteine frei zu konfigurieren oder zu verändern. Die Kommunikation – ein essenzieller Aspekt verteilter Systeme – wird sowohl aus inhaltlichen als auch technischen Blickwinkeln betrachtet und bildet die Grundlage für die Entwicklung von Koordinations- und Kooperationsmustern für die Realisierung typischer logistischer Prozesse und Materialflussstrategien.

Summary

Communication and control strategies for the Internet of Things

Razvan Chisu

Today's material flow systems are facing increasingly difficult requirements: The soaring variety of consumer goods and decreasing innovation and product life cycles are making orders and order structures much less predictable. Thus, order-related manufacturing is becoming more and more prominent. In these circumstances, intra-plant material flow systems must react dynamically and be quickly modified while causing only minimal costs.

Conventional automated systems are barely able to meet these requirements. Their central command architecture is highly complex and must be specifically designed for each new project, making it highly inflexible and difficult to modify.

The present paper aims at developing the technological foundation for a new, decentralized material flow control system. The Internet as a distributed data network presents the paradigm for designing an "Internet of Things" where intelligent conveyors and transport units are cooperatively controlling the material flow without any form of hierarchical command structures.

The system architecture consists of standardized and extendable components or entities which can be iteratively implemented based on each other. At the same time, the system allows for a free definition and reconfiguration of these entities' interplay patterns. Communication, being an essential aspect of distributed systems, is analyzed both in terms of content and technological requirements and forms the basis for designing coordination and cooperation strategies needed for typical processes and material flow strategies in intra-plant logistics.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Formelverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bedeutung
AKL	Automatisches Kleinteilelager
BB	Blackboard
DB	Datenbank
DF	Directory Facilitator
EBS	Early Baggage Store
EHB	Elektrohängebahn
EPC	Elektronischer Produktcode
ERP	Enterprise Resource Planning
FIPA	Foundation for Intelligent Physical Agents
FTF	Fahrerloses Transportfahrzeug
FTS	Fahrerloses Transportsystem
GUI	Graphical User Interface
JADE	Java Agent Development Framework
KLT	Kleinladungsträger
KMU	Kleine und mittlere Unternehmen
KTF	Kleinteileförderer
LAM	Lastaufnahmemittel
LAN	Local Area Network
LVS	Lagerverwaltungssystem
MFR	Materialflussrechner
QVW	Querverschiebewagen
RFID	Radio-Frequenz-Identifikation
SAIL	Softwarearchitektur für die Intralogistik

SOA	Service Oriented Architecture
SPS	Speicherprogrammierbare Steuerung (engl. PLC)
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol / Internet Protocol
TE	Transporteinheit
UHF	Ultra High Frequency
UML	Unified Modelling Language
WLAN	Wireless Local Area Network
WMS	Warehouse Management System
XML	Extensible Markup Language

1 Einleitung

1.1 Ausgangssituation

Die Logistik ist einer der größten und am schnellsten wachsenden Wirtschaftszweige Deutschlands [Kl-06]. Einer der Gründe für diese Entwicklung ist die immer stärkere Vernetzung von Unternehmen, sowohl auf nationaler Ebene als auch weltweit, die durch die anhaltenden Trends zum Outsourcing und zur Globalisierung der Absatz- und Beschaffungsmärkte verursacht wird. Aber auch die gestiegenen Erwartungen der Kunden an Lieferzeit und -qualität wie auch die zunehmende Bedeutung individualisierter Produkte und den damit verbundenen immer kleineren Losgrößen in Produktion und Versand stellen die Logistik vor neue Herausforderungen [Wil-06]. Die Anforderungen an Informationstransparenz, Reaktionsfähigkeit und Wandelbarkeit sind in den letzten Jahren somit stark angestiegen.

In einem solchen, von starken Schwankungen und schlechter Prognostizierbarkeit charakterisierten Umfeld, stoßen heutige Materialflusssysteme an ihre Grenzen. Vor allem hoch automatisierte Systeme gelten im Allgemeinen als eher unflexibel, so dass bei hohen Wandelbarkeitsanforderungen oftmals manuellen Systemen der Vorzug gegeben wird. Diese sind aber weniger effizient als automatisierte Lösungen und durch hohe Personalkosten gekennzeichnet.

Einer der wichtigsten Gründe für die geringe Flexibilität heutiger Materialflussanlagen ist ihre sehr hohe Komplexität. Der hohe Anteil projekt- und kundenspezifischer Lösungen wie auch die zentrale Steuerungsarchitektur solcher Systeme führt zu schwer überschaubaren Wechselwirkungen zwischen den einzelnen Systemkomponenten und behindert durch die große Anzahl heterogener Schnittstellen die Integration neuer Funktionalität und mindert so die Erweiterbarkeit des Systems. Die hohe Komplexität herkömmlicher Systeme wird bereits heute als schwerwiegendes Problem angesehen, wobei davon auszugehen ist, dass sich dieses in Zukunft noch weiter verschärfen wird.

Diese Komplexitätsfalle rührt aber nicht von einzelnen Teilproblemen oder der nicht ausreichenden Performance einzelner technologischer Komponenten her, sondern ist eine inhärente Eigenschaft der heute eingesetzten zentralen und hersteller-

spezifischen Architekturen. Daher ist die Realisierung komplett neuartiger Kommunikations- und Steuerungsmechanismen erforderlich.

Eine Senkung der Komplexität kann nur über eine stringente Modularisierung des Gesamtsystems erreicht werden. Die Gliederung in überschaubare Teilprobleme bringt sowohl einen technologischen als auch einen organisatorischen Vorteil, da Funktionsumfänge gekapselt und nur über relativ einfache, klare Schnittstellen zugreifbar sind. Dies führt zu einer erhöhten Transparenz des Gesamtsystems, da alle Abhängigkeiten zwischen Einheiten offen gelegt und versteckte, schwer nachvollziehbare Wechselwirkungen vermieden werden. Außerdem zeichnet sich ein modulares System durch eine einfachere Austauschbarkeit und Wiederverwendung von Komponenten aus.

Diese Idee der Dezentralisierung und Modularisierung in der Logistik hat sich in den letzten Jahren immer weiter verbreitet. Viele Anlagenbauer setzen bereits auf Baukastensysteme und verteilte Rechenkapazität, beispielsweise in Form frei programmierbarer Frequenzumrichter oder Mikrocontroller. Aber trotz modularer Mechanik und gestiegener Rechenkapazität anlagennaher Komponenten wird der größte Teil der Entscheidungen in einem heutigen Materialflusssystem immer noch von zentralen Systemen, hauptsächlich dem Materialflussrechner, getroffen. Dadurch können nicht alle Vorteile der Dezentralisierung, vor allem bezüglich der Wiederverwendbarkeit von Standardkomponenten und der durch klare Strukturierung der Funktionalität erzielbaren Verringerung der Gesamtsystemkomplexität, ausgenutzt werden.

Eine konsequente Weiterführung und Umsetzung der verteilten Automatisierung bedingt zusätzlich zur modularen Gestaltung von Mechanik und Energieversorgung vor allem die Dezentralisierung der Steuerungslogik. An Stelle der heutzutage üblichen, zentralistisch geprägten Hierarchien werden kleinskalige Einheiten benötigt, die durch intelligente kooperative oder konkurrierende Verhaltensweisen in der Lage sind, den Materialfluss zu steuern. Diese müssen Aufgaben wie Auftragsdisposition, Wegplanung, Bewertung von Wegen bezüglich verschiedener Kriterien, Ressourcenmanagement und Optimierung dezentral erfüllen.

1.2 Zielsetzung und Vorgehensweise

Diese Arbeit befasst sich mit Kommunikations- und Steuerungsverfahren für aus autonomen Einheiten aufgebaute Materialflusssysteme. Dabei soll zum einen ein für diese Aufgabe tragfähiges Kommunikations- und Datenverteilungskonzept entwickelt werden. Zum anderen sollen darauf aufbauende Kooperations-, Koordinations- und Verhandlungsprotokolle implementiert werden, die die Steuerung komplexer und heterogener logistischer Systeme durch dezentrale Entscheidungsträger ermöglichen. Dabei sollen auch die Anbindung eines solchen Materialflusssystems an übergeordnete, zentrale Instanzen, wie z.B. dem Lagerverwaltungssystem und Konzepte zur globalen Überwachung und zur manuellen Beeinflussung durch Mitarbeiter untersucht werden.

Besonders hervorzuheben ist dabei die angestrebte Allgemeingültigkeit der Ergebnisse. Die Betrachtungen sollen sich nicht auf bestimmte Fördertechnologien oder Systeme, z.B. Unstetigförderer oder FTS, beschränken, sondern die Grundlagen für die Steuerung innerbetrieblicher Logistikprozesse im Allgemeinen liefern. Die vorliegende Arbeit versteht sich somit als ein Überblick technischer Grundlagen und Vorgehensweisen für die softwaretechnische Realisierung dezentral gesteuerter, hoch automatisierter Materialflussanlagen, unabhängig von deren mechanischer Auslegung oder der eingesetzten Softwaretools und -bibliotheken. Zwar werden im Rahmen dieser Arbeit einige Modelle und Konzepte an bestimmte Standards für Agentensysteme, vornehmlich den FIPA-Vorgaben, angelehnt. Trotzdem werden dabei vor allem die Kommunikations- und Kooperationsverfahren in Form eines einfachen Nachrichtenaustausches zwischen allgemeinen autonomen Einheiten dargestellt, um so die Übertragbarkeit und allgemeine Realisierbarkeit der erarbeiteten Konzepte sicher zu stellen. Da sich diese Arbeit größtenteils noch im Bereich der Grundlagenforschung bewegt und lediglich einen ersten Schritt in Richtung einer industriellen Implementierung darstellt, werden keine Standards bzgl. der Programmierung oder der Kommunikationsprotokolle vorgeschlagen.

Den ersten Schritt stellt dabei die Aufnahme des aktuellen Stands der Technik dar. Dabei werden zum einen heutige Materialflusssysteme mit den dort gebräuchlichen Steuerungsarchitekturen sowie den zu erfüllenden Funktionen und zum anderen verteilte IT-Systeme, vor allem das Internet und Multiagentensysteme, betrachtet.

Die Analogie zwischen Materialflüssen in der Logistik und dem Informationsfluss im Internet sowie das daraus abgeleitete Modularisierungsprinzip des „Internet der Dinge“ führt zur Definition elementarer Einheiten, die in einem verteilten Logistiksystem als Entscheidungsträger agieren. Methoden der objektorientierten Softwareentwicklung gewährleisten eine effiziente, an die Anforderungen der Logistik angepasste Implementierung sowohl der autonomen Einheiten bzw. Softwareagenten als auch der grundlegenden Datenübertragungsmechanismen. Eine besondere Anforderung an die Softwaretechnischen Einheiten besteht darin, dass sie sowohl einen hohen Standardisierungsgrad aufweisen als auch in der Lage sein sollen, verschiedenste Szenarien umzusetzen.

Der Gestaltung und Analyse der Kommunikation – sowohl in Bezug auf die auszutauschenden Inhalte als auch auf die Kommunikationsmethodik – wird ein eigenes Kapitel gewidmet. Dieses stellt zum einen die Ontologie, also das Vokabular oder den „Sprachschatz“ des Internet der Dinge dar und zeigt zum anderen Möglichkeiten zur Realisierung eines effizienten, transparenten und zuverlässigen Datenaustauschs.

Anschließend werden sowohl für stetig als auch für unstetig fördernde Systeme in Anlehnung an konkrete Logistikszenarien Steuerungskonzepte vorgestellt, die sowohl die lokale Funktionalität eines einzelnen Akteurs als auch komplexe Muster für für die Interaktion zwischen diesen abdecken. Ein weiterer Punkt ist die zentrale Steuer- und Beobachtbarkeit eines dezentralen Systems.

Die entwickelten Konzepte werden in der Versuchsanlage des Lehrstuhls fml der Technischen Universität München und im Rahmen verschiedener Emulationen umgesetzt und demonstrieren die Tauglichkeit der vorgestellten Lösungsansätze.

Die Struktur dieser Arbeit und die thematische Beziehung der Kapitel untereinander werden in Abbildung 1-1 schematisch dargestellt.

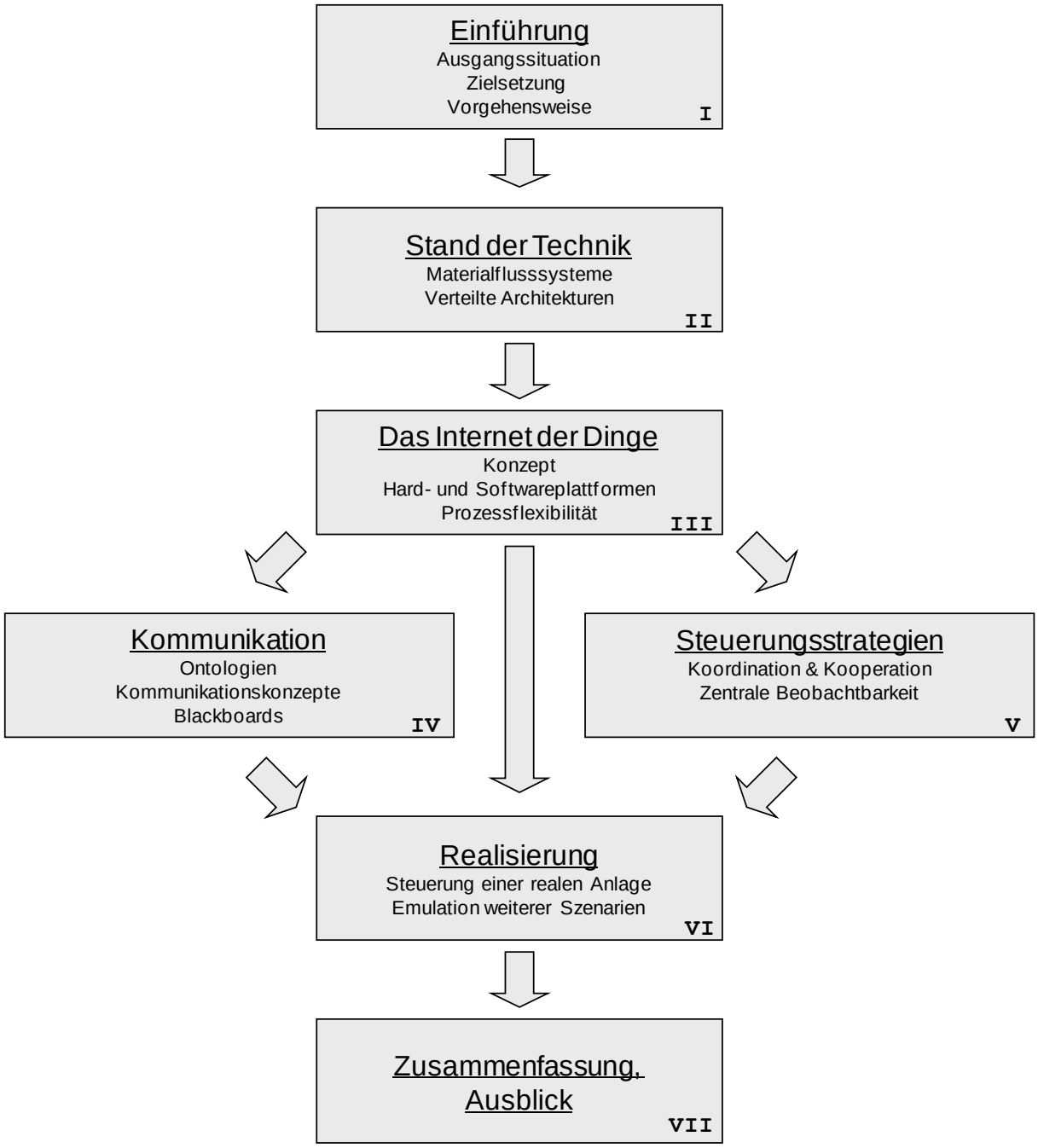


Abbildung 1-1 Vorgehensweise

2 Stand der Technik und Forschung

2.1 Steuerung von Materialflusssystemen

Materialflusssysteme müssen im Allgemeinen komplexe Prozesse umsetzen und steuern. Aufträge müssen möglichst zeitnah erfüllt werden, wobei auf Grund von Interdependenzen zwischen Produktions- oder Kommissionieraufträgen auch komplexe Funktionen wie das Sortieren von Transporteinheiten oder die Einhaltung von Reihenfolgen von großer Bedeutung sind. Um eine hohe Leistung und Auslastung des Gesamtsystems zu gewährleisten, werden Transportaufträge mittels teils komplizierter Algorithmen auf die verschiedenen Transportmittel verteilt und koordiniert.

Herkömmliche Materialflusssteuerungen sind zumeist hierarchisch aufgebaut und gliedern die anfallenden Aufgaben in mehrere Ebenen. So definiert das VDMA-Einheitsblatt 15276 „Datenschnittstellen in Materialflusssystemen“ [VDMA-15276] sechs funktionale Ebenen (siehe Abbildung 2-1):

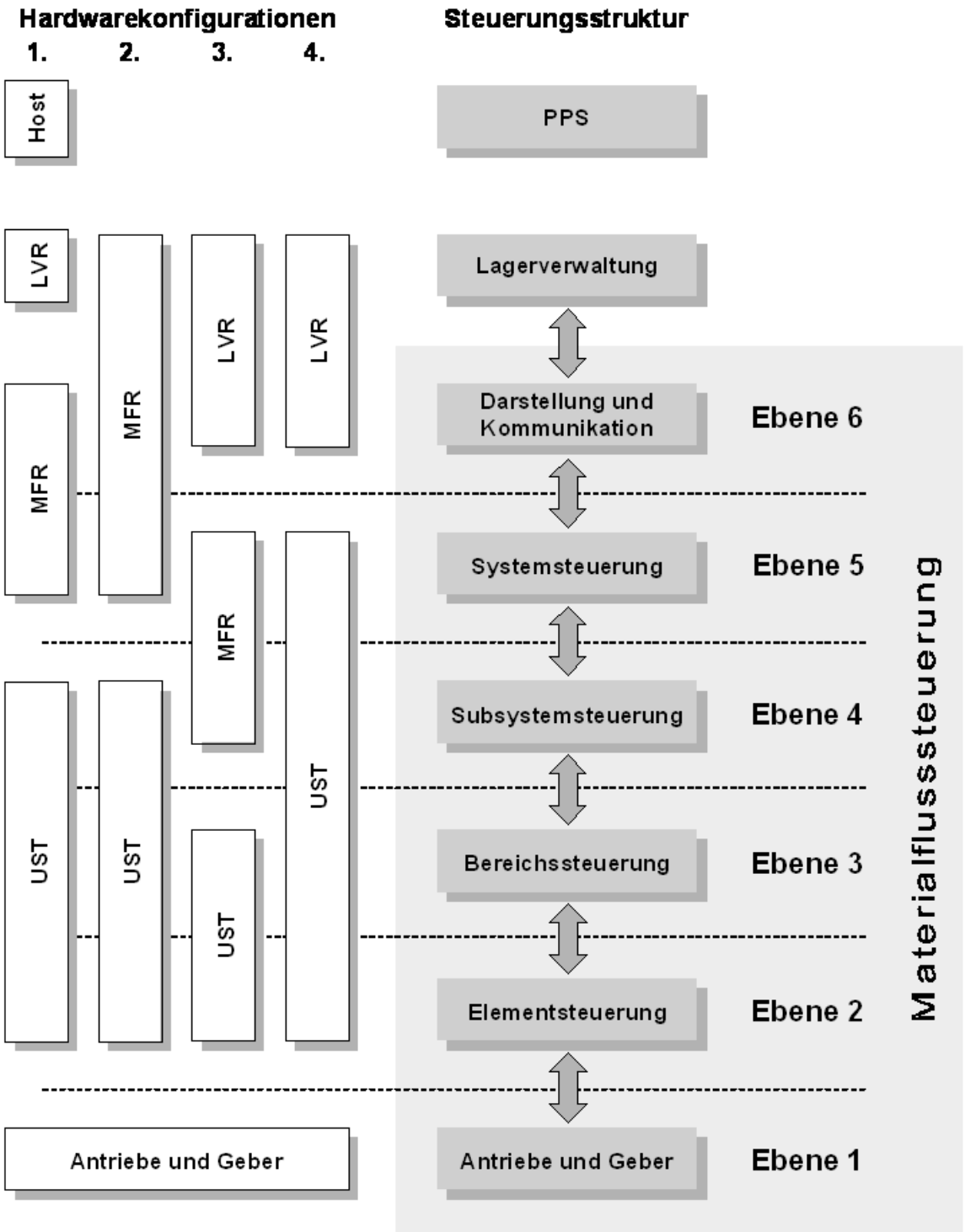
- Ebene 1: Antriebe und Geber
- Ebene 2: Elementsteuerung
- Ebene 3: Bereichssteuerung
- Ebene 4: Subsystemsteuerung
- Ebene 5: Systemsteuerung
- Ebene 6: Darstellung und Kommunikation.

Dadurch wird allerdings keine Festlegung der Hardwareplattform getroffen, denn je nach Aufgabenstellung und Komplexität einer Materialflussanlage können mehrere Ebenen kombiniert und auf derselben Hardware umgesetzt werden. Klassische Systeme bestehen aus einem Materialflussrechner, der die Ebenen drei bzw. vier bis sechs implementiert, sowie mehreren speicherprogrammierbaren Steuerungen (SPS), auf denen die unteren Ebenen realisiert werden. Während die SPSen direkt die Mechanik und Elektrik ansteuern und überwachen und im Allgemeinen eher einfache Logik zur Überprüfung des korrekten Betriebs von Komponenten abbilden, ist der Materialflussrechner für alle Aufgaben mit dispositivem bzw. strategischem Cha-

rakter wie z.B. die Auftragsdisposition, Routenplanung, Koordination von Abläufen oder die Umsetzung von Strategien zuständig [Büc-00].

Durch diese Konzentration von Steuerungslogik auf einem zentralen Materialflussrechner steigt die Komplexität dieser Komponente drastisch an. Zudem werden Materialflussrechner im Regelfall kunden- und projektspezifisch ausgelegt und auf die zur Realisierungszeit bekannten Prozesse zugeschnitten. Die damit sehr starre und unflexible Umsetzung lässt sich nur mit erheblichem Aufwand und einer Vielzahl manueller Änderungen an neue Anforderungen anpassen, was vor allem beim Umbau oder Erweiterung von Anlagen zu hohen Kosten führt [Gün-08c].

Eine solche hierarchische Betrachtung und Modellierung von Materialflusssteuerungen hat den Vorteil, die verschiedenen Funktionen eines Systems zu gliedern und Ebenen mit definierten Schnittstellen zuzuweisen, wodurch die Transparenz des Systems erhöht wird. Dabei sind die Abhängigkeiten zwischen den Ebenen jedoch sehr stark, weil jede Ebene die Aufträge bzw. Anfragen der nächst höheren bearbeiten und dabei die Aufgaben der nächst niedrigeren Ebene überwachen muss. Dies führt zu teils sehr komplexen Abhängigkeiten zwischen den einzelnen Systemteilen, was durch die Existenz zahlreicher herstellerspezifischer Schnittstellen oft zu weiteren Schwierigkeiten bei der Integration der einzelnen Ebenen bzw. Rechenplattformen führt. Zudem wird auch die Komplexität der Entscheidungsvorgänge innerhalb der Ebenen nach oben hin immer höher, was vor allem im Bereich des Materialflussrechners zu kaum mehr beherrschbaren Strukturen führen kann.



PPS: Übergeordnetes Hostsystem (Produktionsplanung und -steuerung)
LVR: Lagerverwaltungsrechner
MFR: Materialflussrechner
UST: Unterlagerte Steuerung

Abbildung 2-1 Hardwarekonfiguration und Steuerungsarchitektur der Materialflusssteuerung nach [VDMA15276]

Für eine Abkehr vom Ebenenmodell hin zu einer funktionsorientierten Betrachtung setzt sich das Forum Intralogistik des VDMA ein [Irr-07]. Eine neue Systemarchitektur für die Intralogistik, kurz SAIL, orientiert sich an der objektorientierten Programmierung und definiert Standardfunktionen und -komponenten mit einheitlichen Schnittstellen. SAIL sieht demnach fünf fördertechnische Funktionen:

- Anlagensteuerung
- Richtungsentscheidung
- Fahrauftragsverwaltung
- Ressourcennutzung
- Transportkoordination

und vier Komponenten vor:

- Förderelemente
- Fördergruppen
- Fördersegmente
- Förderbereiche

Dabei wird jede Komponente als Zusammenfassung des jeweils voran gegangenen Bausteins definiert und erfüllt eine der fördertechnischen Funktionen. So erfüllen Förderelemente die Funktion Anlagensteuerung und können zusammengefasst eine Fördergruppe bilden, die Richtungsentscheidungen trifft. Fördersegmente bestehen wiederum aus mehreren Fördergruppen und übernehmen die Fahrauftragsverwaltung, während Förderbereiche, als Menge von Fördersegmenten, zudem auch die Ressourcennutzung verwalten. Die Schnittstellen und der Nachrichtenaustausch zwischen den einzelnen Komponenten werden von SAIL ebenfalls definiert. Derzeit (Stand August 2009) sind die SAIL-Empfehlungen in einem Richtlinienentwurf zusammengefasst [VDI/VDMA-5100].

Durch diese Kapselung steuerungstechnischer Funktionen wird die Transparenz der Materialflusssteuerung sowie die Austauschbarkeit von Komponenten verbessert [Bal-06]. Trotz der funktionalen Strukturierung stellt aber auch das SAIL-Modell einen hierarchischen Ansatz zur Steuerung von Materialflusssystemen dar.

2.2 Multiagentensysteme

Hierarchische Steuerungen sind aber nicht die einzige Möglichkeit, um komplexe Problemstellungen zu lösen. So werden Mechanismen und Plattformen für das verteilte Rechnen in der Informatik, vor allem in den Bereichen der Hochleistungsrechner und der künstlichen Intelligenz, schon seit Jahrzehnten erforscht und eingesetzt. Dabei kann zwischen verteilter Rechenhardware und verteilter Software unterschieden werden.

Im Bereich der verteilten Rechenhardware sind vornehmlich die so genannten Parallelrechner zu nennen. Diese Computer können bis zu mehrere Tausend Prozessoren haben [Ung-99] und werden hauptsächlich für sehr rechenintensive Aufgaben eingesetzt, beispielsweise die Simulation physikalischer Prozesse. Mehrprozessorsysteme mit zwei bis vier Rechenkernen sind mittlerweile aber auch im PC-Bereich zum Standard geworden, um die immer höheren Anforderungen der Multimedia- und Spieleanwendungen zu bewältigen. Dabei wird eine komplexe Gesamtrechenaufgabe in kleine, unabhängige Probleme aufgeteilt und mit entsprechenden Algorithmen auf möglichst viele, parallel arbeitende Rechenwerke verteilt. Die von jedem Prozessor berechneten Teillösungen werden anschließend wieder zu einem Gesamtergebnis zusammengefasst, das dem Nutzer angezeigt oder als Eingabe für weitere Berechnungen dienen kann. Der Einsatz solcher Computer ist aber nur dann sinnvoll, wenn die Rechenaufgabe an sich parallelisierbar ist. Wird z.B. in jedem Rechenschritt das Ergebnis der voran gegangen Operation benötigt, kann die Berechnung nur sequenziell abgearbeitet und damit nicht auf mehrere Prozessoren verteilt werden.

Ähnlich wie bei Parallelrechnern gibt es auch Softwaresysteme, die aus mehreren autonomen, aber untereinander kommunizierenden Einheiten bestehen. Diese werden beispielsweise im Bereich der Künstlichen Intelligenz [Fer-01] [Rus-04] eingesetzt [Fer-01] [Rus-04]. Diese so genannten Softwareagenten [Woo-02] verfolgen verschiedene Ziele und kooperieren oder konkurrieren miteinander, ohne zentral gesteuert zu werden. Dabei beobachtet ein Agent seine Umgebung und erhält Informationen über diese entweder in Form von Nachrichten von anderen Agenten oder Softwareprogrammen, oder auch in Form von Sensorsignalen. Anhand dieser Daten und weiteren, evtl. schon bei der Initialisierung fest vorgegebenen Wissens, kann der

Agent ein Weltmodell erstellen und aktuell halten. Dieses Weltmodell stellt die Grundlage für die Planung und Durchführung von Aktionen, die sich sowohl auf den internen Zustand des Agenten als auch auf seine softwaretechnische oder physikalische Umgebung auswirken können (siehe Abbildung 2-2).

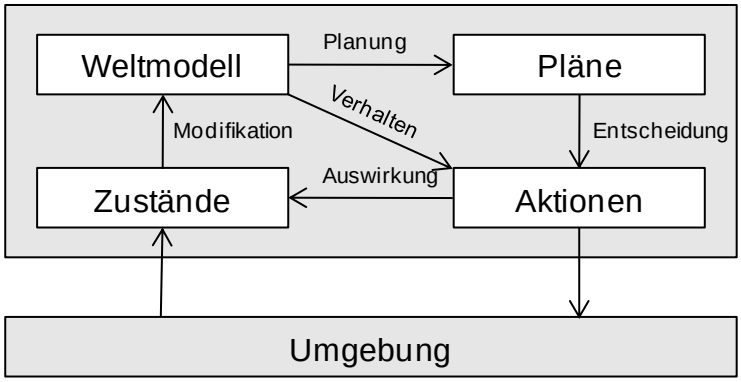


Abbildung 2-2 Basisstruktur eines Agenten [Sca-05]

Für Softwareagenten existiert allerdings keine einheitliche und allgemein anerkannte Definition, wobei eine solche, wegen des sehr breiten Anwendungsspektrums und der Fähigkeiten dieser Programme, wahrscheinlich auch nicht möglich ist [Nwa-96]. Typischerweise werden Agenten jedoch die folgenden Eigenschaften zugesprochen, wobei ein Softwareagent nicht zwingender Maßen alle davon besitzen muss:

- Autonomie – Agenten sind in der Lage, ohne Benutzereingabe Aufgaben wahrzunehmen und Ziele zu verfolgen.
- Persistenz – Agenten werden im Allgemeinen nicht bei Bedarf gestartet bzw. gestoppt, sondern „leben“ über längere Zeiträume und entscheiden selbst, wann sie eine konkrete Aufgabe wahrnehmen und wann sie passiv bleiben.
- Reaktivität – Agenten können ihre Umwelt wahrnehmen und auf Veränderungen in ihrem Umfeld reagieren.
- Proaktivität – Agenten können aus eigener Initiative heraus Aktionen anstoßen.
- Soziales Verhalten – Agenten können über Schnittstellen miteinander kommunizieren und evtl. zusammenarbeiten.
- Lernfähigkeit – Agenten können aus getroffenen Entscheidungen oder auf Grundlage der Wahrnehmung ihrer Umwelt lernen, um ihr Verhalten zu optimieren oder langfristig anzupassen.

- Mobilität – Agenten können zur Laufzeit zwischen verschiedenen Rechnern migrieren. Dies ist z.B. dann sinnvoll, wenn ein Agent auf einen Datenbankserver verschoben wird, um für eine bestimmte Aufgabe minimale Antwortzeiten von der Datenbank zu gewährleisten.

Je nach implementierter Funktionalität ist eine Gliederung von Softwareagenten in bestimmte Klassen möglich, beispielsweise kollaborative Agenten, reaktive Agenten, intelligente Agenten und Schnittstellenagenten (siehe Abbildung 2-3).

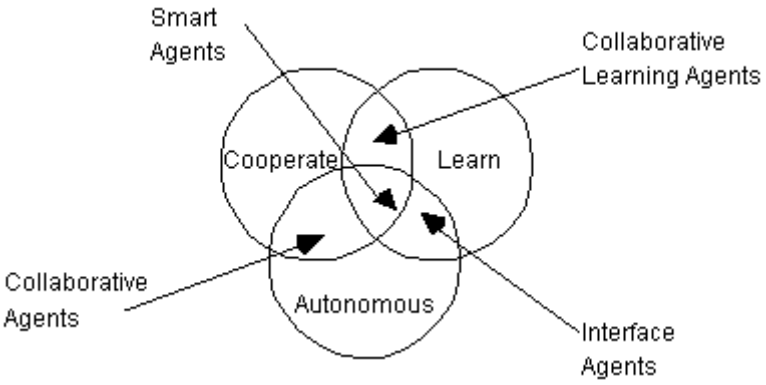


Abbildung 2-3 Beispiel zur Klassifizierung von Agenten [Nwa-96]

Softwareagenten werden jedoch nicht nur für Datenverarbeitungsaufgaben eingesetzt. So können Agenten auch die Steuerung, Überwachung oder Optimierung von Prozessen in der realen Welt übernehmen. Die Einheit von Softwareagent und den dazugehörenden mechatronischen Komponenten, z.B. Sensoren und Antriebe, werden als technischer Agent bezeichnet [Lüt-98]. Technische Multiagentensysteme werden beispielsweise zur Steuerung mobiler Roboter eingesetzt [Guz-97] [Rit-00] [Wan-00] [Yam-00].

Damit Agenten miteinander kommunizieren bzw. im selben Umfeld arbeiten können, müssen sie dieselben Schnittstellen und Kommunikationsprotokolle beherrschen. Eine der wichtigsten Institutionen, die sich mit der Entwicklung von Standards für verschiedene Aspekte der Agentenmodellierung, -kommunikation und -interaktion befasst, ist die Foundation for Intelligent Physical Agents, kurz FIPA. Obwohl die FIPA selbst keine Softwarebibliotheken oder Tools zur Implementierung von Agenten anbietet, sind von Drittanbietern Software-Frameworks zur Erstellung FIPA-konformer Agentensysteme erhältlich. Hier sei z.B. das frei verfügbare Java Agent Development Framework [JADE] [Bel-07] zu nennen.

2.3 Aktuelle Forschung im Bereich dezentraler Automatisierung

Der anhaltende Trend in Richtung Variantenvielfalt und Individualisierung, vor allem bei Konsumgütern [Tse-03] [SFB582-04], stellt Logistik- und Produktionssysteme vor eine immer größere Herausforderung. Denn durch die steigende Anzahl der Produktvarianten und die damit einher gehenden sinkenden Losgrößen [Weh-04], müssen Materialflusssysteme flexibel auf Veränderungen, beispielweise der Auftragslast oder –struktur, reagieren. Darüber hinaus ergibt sich aus den immer kürzeren Produktlebenszyklen auch ein immer kürzerer Planungshorizont für intralogistische Systeme: Das sich häufig verändernde Artikelspektrum macht eine langfristige Prognose der benötigten Leistung und damit der mechanischen und topologischen Auslegung eines Systems äußerst schwierig, Umbauten existierender Systeme werden immer häufiger notwendig [Log-07]. Daher müssen Materialflusssysteme nicht nur im Rahmen der vorgeplanten Anforderungen und Szenarien flexibel reagieren können, sie müssen auch mit möglichst geringem Aufwand umgebaut werden können [Sch-01] [Wir-01].

Untersuchungen der Kostentreiber bei der Realisierung und Inbetriebnahme zentral gesteuerter Systeme haben ergeben, dass die Integration verschiedener Komponenten und Subsystemen, sowie die Überprüfung ihres korrekten Zusammenspiels einen sehr großen Aufwand verursachen [Gün-08c]. Beim Umbau von Anlagen verschärft sich die Situation zusätzlich. Denn die Integration neuer Komponenten erfordert zumeist eine manuelle Anpassung bzw. Umprogrammierung des Gesamtsystems [Lan-97] und damit auch seine vorübergehende Abschaltung. Diese Nachteile rühren zum größten Teil aus der hierarchischen Struktur solcher Systeme – die Steuerungslogik und Materialflussstrategien werden auf einem oder sehr wenigen Materialflussrechnern konzentriert, die Ansteuerung der mechanischen Komponenten übernehmen speicherprogrammierbare Steuerungen, wobei eine SPS häufig für größere Anlagenbereiche zuständig ist. Diese Architektur wird projektspezifisch ausgelegt und ist somit sehr starr und unflexibel.

Ab Ende der achtziger Jahre wurden in der Automatisierungstechnik die Potenziale einer neuen, dezentralen Architektur und ihre Notwendigkeit für zukünftige Organisations- und Steuerungsstrukturen erkannt [Dru-88] [Gol-93]. Seitdem kann auch ein

Trend in Richtung von Dezentralisierung und der Gestaltung von Automatisierungsbaukästen beobachtet werden [Mei-96] [Gün-98] [Mön-99]. Dabei wird versucht, Ansätze aus der Informatik zum verteilten Rechnen bzw. Problemlösen auf Logistik- oder auch Produktionssysteme zu übertragen. Die einzelnen Komponenten einer Anlage sind dann nicht mehr innerhalb einer hierarchischen Struktur fest miteinander gekoppelt, sondern agieren autonom und kooperativ. Je nach Anforderungen und technischen Ausprägungen werden dabei Agenten [Bus-04], Holone [Koe-76] [Bru-98] oder Fraktale [War-93] [War-99] [Kwa-04] zur Implementierung und Modellierung solcher Systeme eingesetzt. Fraktale und Holone definieren im Gegensatz zu Agenten auch eine hierarchische Aggregation funktionaler Einheiten zu komplexeren Systemen, wobei vor allem bei Fraktalen gewisse Attribute und Funktionen auf allen Hierarchieebenen vorhanden sind. Auch serviceorientierte Architekturen, kurz SOA [Sun-05], die auf dem Gedanken der losen Kopplung abstrakter Dienste beruhen, gewinnen im Bereich der Logistik sowohl auf organisatorischer [Hop-07] als auch auf Steuerungsebene [Hom-08] immer mehr an Bedeutung.

Für den Bereich der Transportlogistik wurden beispielsweise agentenbasierte Ansätze zur kooperativen Tourenplanung untersucht. Ein System autonomer Agenten, die für jeweils ein Fahrzeug zuständig sind und die Abarbeitungsreihenfolge der zugeteilten Aufträge optimieren, bzw. im Gesamtsystem vorhandene Transportbedarfe analysieren und auf Fahrzeugagenten verteilen, kann dabei verschiedene, von einem Benutzer vorgegebene, Zielgrößen verfolgen [Wen-07]. Ähnliche Fragestellungen werden auch in [Dah-07] und [Kop-07] behandelt. Der Fokus dieser Arbeiten liegt auf kooperierenden Speditionsunternehmen, wobei durch eine kollaborative Auftragsdisposition eine verbesserte Auslastung der Fahrzeuge und eine Verringerung der Transportkosten erreicht werden kann. Diese Vernetzung und Kooperation zwischen Unternehmen soll es vor allem KMUs erlauben, mit großen Transportunternehmen erfolgreich konkurrieren zu können. Zur Unterstützung der Auftragsdisposition vor allem im Fall auftretender Störungen wie z.B. dem Ausfall von Fahrzeugen, wird in [Kir-07] ein Agentensystem vorgeschlagen, das einen Disponenten bei der Entscheidungsfindung unterstützten soll.

Das Projekt Agent-based Multi-Site Planning and Scheduling Application Framework, kurz AMPA, das an der Carl von Ossietzky Universität Oldenburg bearbeitet wurde, greift das Problem räumlich verteilter Produktionsnetzwerke und virtueller Unterneh-

men auf. Für die Ablaufplanung und Koordination von Produktionsprozessen über mehrere Unternehmen hinweg, auch Multi-Site-Scheduling genannt, wurde ein Multiagentensystem vorgeschlagen [App-99] [Sau-00]. Dabei werden, auf den verschiedenen Hierarchieebenen einer Unternehmensorganisation, den zu koordinierenden Teilprozessen bzw. -strukturen Softwareagenten zugeordnet, die sowohl interne Vorgänge planen und koordinieren als auch Abhängigkeiten von anderen Agenten bzw. Systemen berücksichtigen [App-00]. Das für die Ablaufplanung entwickelte Agentensystem kann damit auch für das Management von Supply-Chains eingesetzt werden [Sau-03].

Dezentral organisierte Systeme werden aber auch für die Steuerung innerbetrieblicher Prozesse eingesetzt. So wurden z.B. im Rahmen des internationalen Konsortiums „Intelligent Manufacturing Systems“ Holone als eine Möglichkeit für die Gestaltung hochflexibler Produktionssysteme untersucht [Knu-94]. Der Begriff „Holon“ wurde von Koestler geprägt und bezeichnet im Allgemeinen ein Ganzes, das wiederum Teil eines größeren Ganzen ist [Koe-72]. Jedes Holon ist in sich abgeschlossen und autonom, kann aber mit anderen Holonen auf derselben Hierarchieebene kommunizieren und interagieren. Auf Produktionssysteme übertragen können beispielweise die Komponenten einer Fertigungsmaschine als Holone angesehen werden. Die Fertigungsmaschine ist ihrerseits ebenfalls ein Holon, das innerhalb eines noch größeren Holons, beispielweise einer Roboterzelle, mit anderen Holon-Maschinen interagiert, um eine bestimmte Aufgabe zu erfüllen und dabei alle zusätzlichen Rahmenbedingungen, wie z.B. Sicherheitsvorschriften, einzuhalten. Es wurde gezeigt, dass dieser Ansatz nicht nur für die lokale Steuerung von Prozessen innerhalb eines Holons geeignet ist, sondern dass auch übergeordnete Koordinations- und Synchronisationsvorgaben eingehalten werden können [Kan-97] [Gou-98].

Multiagentensysteme sind dabei eine Möglichkeit zur Implementierung von holonischen Systemen. Multiagentensysteme verzichten aber im Normalfall auf die hierarchische Kapselung und Kombination autonomer Einheiten zu größeren Systemen, sodass von einer einzigen Steuerungsebene bzw. einer komplett flachen Steuerungshierarchie ausgegangen wird. In [Rit-03] werden, aufbauend auf den FIPA-Vorgaben für Agentensysteme [FIPA-05], autonome Einheiten und Interaktionsprotokolle zur Steuerung von Produktionssystemen definiert. Dabei werden vor

allem Themen wie die Auftragsgenerierung, -disposition und -ausführung und das Management von Agenten und Ressourcen behandelt.

Dezentrale Steuerungen für unterschiedliche Materialflusssysteme werden sowohl in der Forschungslandschaft als auch in der Praxis untersucht bzw. eingesetzt. Unter dem Titel „Realtime Logistics“ [Hom-04a] [Hom-05] wurde am Lehrstuhl für Förder- und Lagerwesen der Universität Dortmund eine agentenbasierte Steuerung für ein Stetigfördersystem entwickelt. Dabei werden nicht nur Mechanismen für die dezentrale Steuerung an sich betrachtet, sondern auch Anforderungen in Bezug auf die Echtzeitfähigkeit von Agentensystemen formuliert [Hom-04b] [Hom-07]. Ebenfalls wurden dabei Verfahren für eine automatische Generierung agentenbasierter Simulationsmodelle für die Intralogistik realisiert, die belegen, dass dezentral gesteuerte Systeme eine mit klassischen Anlagen vergleichbare Durchsatzleistung erreichen [Roi-07].

[May-09] demonstriert den Aufbau und die dezentrale Steuerung eines Stetigfördersystems durch identische, quadratische und in alle vier Richtungen fördernde Module. Der Fokus liegt dabei vor allem auf dem Routing und der Vermeidung von Deadlockzuständen.

Ein Beispiel dafür, dass Dezentralität nicht nur vorhandene Probleme lösen, sondern auch zur Gestaltung völlig neuer Fördertechnologien führen kann, ist das vom Fraunhofer IML in Zusammenarbeit mit der Siemens Dematic AG entwickelte Multi-Shuttle [Jun-04] [Sch-05]. Ein Schwarm autonomer, kostengünstiger Fahrzeuge kann dabei über ein Schienennetz und Aufzüge alle Ebenen eines Hochregallagers bedienen und ausgelagerte Waren anschließend z.B. in einen Kommissionierbereich transportieren. Da mehrere Fahrzeuge gleichzeitig in einer Regalgasse verkehren können, kann das System eine viel höhere Leistung als klassische Regalbediengeräte erzielen. Systeme mit über 100 Multishuttles befinden sich bereits im industriellen Einsatz [Dem-07].

Auch andere dezentral gesteuerte Systeme werden bereits mit Erfolg eingesetzt. Die amerikanische Firma Kiva Systems bietet Kommissioniersysteme, bei denen ein Schwarm kleiner Roboter Regale mit Waren zu einem Kommissionierer befördert. Dieser muss dann nur noch die benötigten Artikel aus dem angelieferten Regal entnehmen und in einen Kommissionierbehälter legen, wodurch die

Kommissionierleistung erhöht werden kann [KS-08] [Gui-08]. Für Sortier- und Verteilaufgaben an Flughäfen bietet die Firma Beumer ein System aus schienengeführten Fahrzeugen. Das autover® System ist nach Herstellerangaben einfach und kosteneffizient zu erweitern, die einzelnen Fahrzeuge werden dezentral von bordeigenen Mikrocomputern gesteuert und können so beispielsweise eine eigene Routenplanung durchführen [Beu-06].

Die aufgeführten Forschungsarbeiten im Bereich der Agentensysteme decken vor allem die Bereiche der Transportlogistik bzw. des Supply-Chain-Managements sowie die Produktionssteuerung ab. Zwar werden dezentrale Systeme auch für die Intralogistik erforscht oder bereits eingesetzt. Dabei handelt es sich aber einerseits um herstellerspezifische, auf eine bestimmte Anwendung und technologische Auslegung zugeschnittene Lösungen. Diese sind nur sehr schwer auf andere Systeme zu übertragen, zumal Details zur Realisierung nicht öffentlich zugänglich sind. Andererseits gehen die in der Forschungslandschaft durchgeführten Arbeiten zumeist von vereinfachten Randbedingungen aus und betrachten nur die grundlegendsten Funktionen, die zur Steuerung eines Materialflusssystems notwendig sind.

Die vorliegende Arbeit ist im Rahmen des vom Bundeministerium für Bildung und Forschung geförderten Forschungsprojektes „Internet der Dinge – Wandelbare Echtzeit-Logistiksysteme auf Basis Intelligenter Agenten für den produktionsnahen Bereich“ [IdD] entstanden. Im Unterschied zu den aufgeführten Veröffentlichungen wird hier das Ziel verfolgt, eine Systemarchitektur und ein Kommunikationskonzept zu entwickeln, die für allgemeine Materialflusssysteme, unabhängig von deren technischer Auslegung, geeignet sind. Darüber hinaus werden relevante und komplexe Materialflussstrategien behandelt, die nur durch die zielgerichtete Kooperation mehrerer autonomer Einheiten realisierbar sind.

3 Das Internet der Dinge

3.1 Funktionsweise des Internets

Die Funktion von Kommunikations- und Computernetzwerken besteht in der Übermittlung von Informationseinheiten von einem Sender an einen oder mehrere Empfänger. Somit erfüllen beispielsweise das Telefonnetz oder das Internet eine ähnliche Funktion wie Materialflusssysteme, die Ladeeinheiten von einer Quelle zu einer Senke fördern. Zwar unterscheiden sich die funktionalen Randbedingungen und Anforderungen wie auch die technologischen Möglichkeiten bei der Übertragung von Datenpaketen teilweise sehr stark von den Gegebenheiten des Stückguttransports. Grob vereinfacht muss aber in beiden Bereichen die zeitgerechte und korrekte Zustellung von Sendungen innerhalb einer sich dynamisch verändernden Topologie gewährleistet werden.

Eines der heutzutage wichtigsten Datennetze ist das Internet, das weltweit mittlerweile über eine Milliarde Nutzer hat [IWS]. Betrachtet man den Aufbau des Internet, fällt auf, dass dieses System eine von heutigen Materialflusssteuerungen grundsätzlich unterschiedliche Struktur aufweist. Zwar ist auch hier eine gewisse Hierarchie zu erkennen (siehe Abbildung 3-1), die sich beispielweise durch die Zusammenschaltung mehrere LANs (Local Area Network) zu einem MAN (Metropolitan Area Network) und der Kombination mehrerer MANs zu einem ein WAN (Wide Area Network) ergibt [Mei-04]. Dabei handelt es sich aber um eine rein strukturelle Gliederung – ähnlich der Vergabe von Postleitzahlen im Postsystem – und nicht um eine Steuerungshierarchie wie im Fall der heutzutage üblichen Materialflussanlagen.

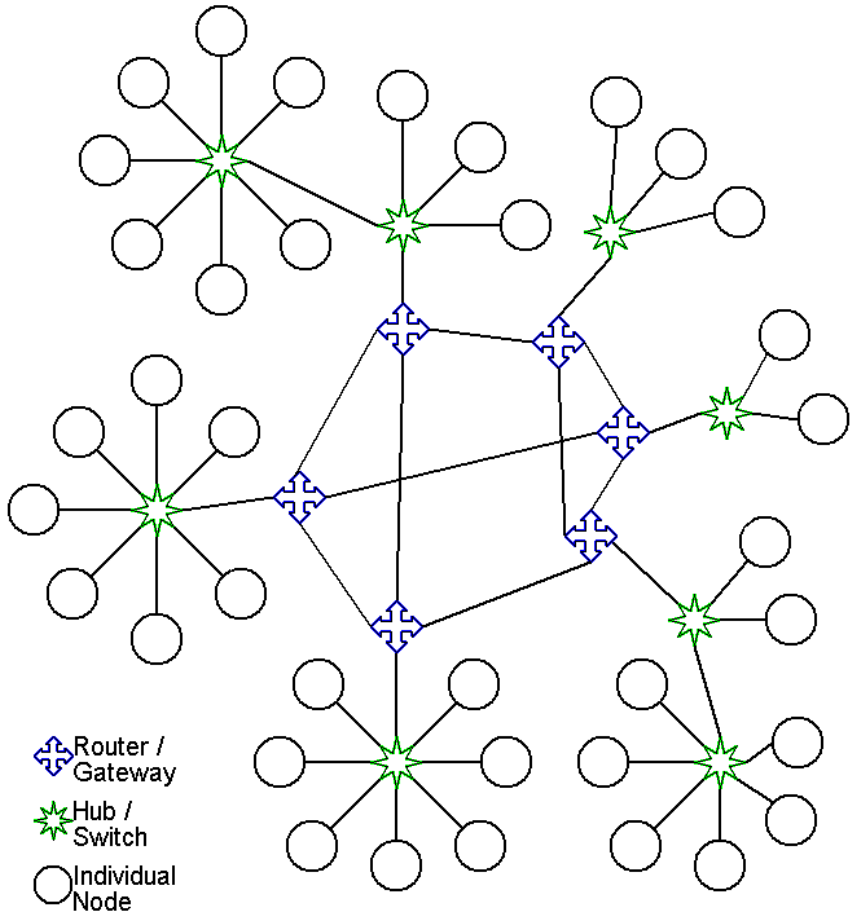


Abbildung 3-1 Vereinfachte Darstellung der Struktur des Internets [Kee-03]

Die eingesetzten Kommunikations- und Steuerungsprotokolle sind zudem so gestaltet, dass diese Verschachtelung von Netzwerken vor dem Benutzer des Internets, also den Absendern und Empfängern von Nachrichten, versteckt wird. So funktioniert die Kommunikation zwischen zwei Computern, die sich im selben Heimnetzwerk befinden, auf genau dieselbe Weise wie die Kommunikation zwischen PCs auf verschiedenen Kontinenten. Dabei ist die Topologie des Internet fast beliebig erweiterbar, (Sub-)Netze oder Verbindungsrouten können jederzeit entfernt, verändert oder hinzugefügt werden, ohne dass an einer zentraler Stelle Routen oder Strategien umprogrammiert oder umkonfiguriert werden müssten.

Als Entscheidungspunkte in Datennetzwerken fungieren so genannte Router. Diese verbinden mehrere Netzwerke und leiten eintreffende Datenpakete entweder an ihr endgültiges Zielnetz – sofern der Router daran direkt angeschlossen ist – oder an einen anderen, dem Ziel des Datenpakets näheren Router. Diese Komponenten arbeiten alle autonom und unabhängig voneinander und sind in der Regel intelligent

genug, um Veränderungen in ihrem Umfeld selbst wahrzunehmen und diese Informationen für die eigenen Berechnungen zu nutzen und ggf. an andere Router weiterzuleiten. Verschiedenste Routing-Protokolle [Sch-04] ermöglichen es, immer einen Pfad vom Absender zum Empfänger eines Datenpaketes berechnen, solange eine physikalische Verbindung existiert.

3.2 Das Internet der Dinge

Vereinfacht betrachtet erfüllt das Internet in der virtuellen Welt der reinen Information also dieselbe Funktion wie ein Materialflusssystem in einer Fabrik oder einem Distributionszentrum. Diese Funktionalität wird aber durch eine von Materialflusssteuerungen grundsätzlich verschiedene Architektur erfüllt, die durch den dezentralen Aufbau große Vorteile in Bezug auf die Flexibilität, Robustheit und Skalierbarkeit des Systems aufweist.

Da diese Eigenschaften in heutigen Logistiksystemen von sehr hoher Bedeutung sind, von den herkömmlichen Technologien aber nur in unzureichendem Maße geboten werden, erscheint eine Übertragung der Steuerungsprinzipien des Internets auf den Materialfluss vielversprechend. Die einfache Analogie zwischen Datenpaketen und Ladeeinheiten bzw. zwischen Routern und Fördertechnik kann dabei als Grundlage für weitere Überlegungen zur Gestaltung dezentraler Logistiksysteme dienen (siehe Abbildung 3-2).

Dies kann aber nicht im Rahmen heutiger Steuerungsarchitekturen geschehen, sondern benötigt ein radikales Umdenken sowohl in der Modellierung als auch der Implementierung von Intralogistiksystemen. Dabei werden zentrale Steuerungscomputer und -verfahren durch ein kooperatives Netzwerk autonomer und intelligenter Einheiten ersetzt, was in der Entstehung eines „Internet der Dinge“ mündet.

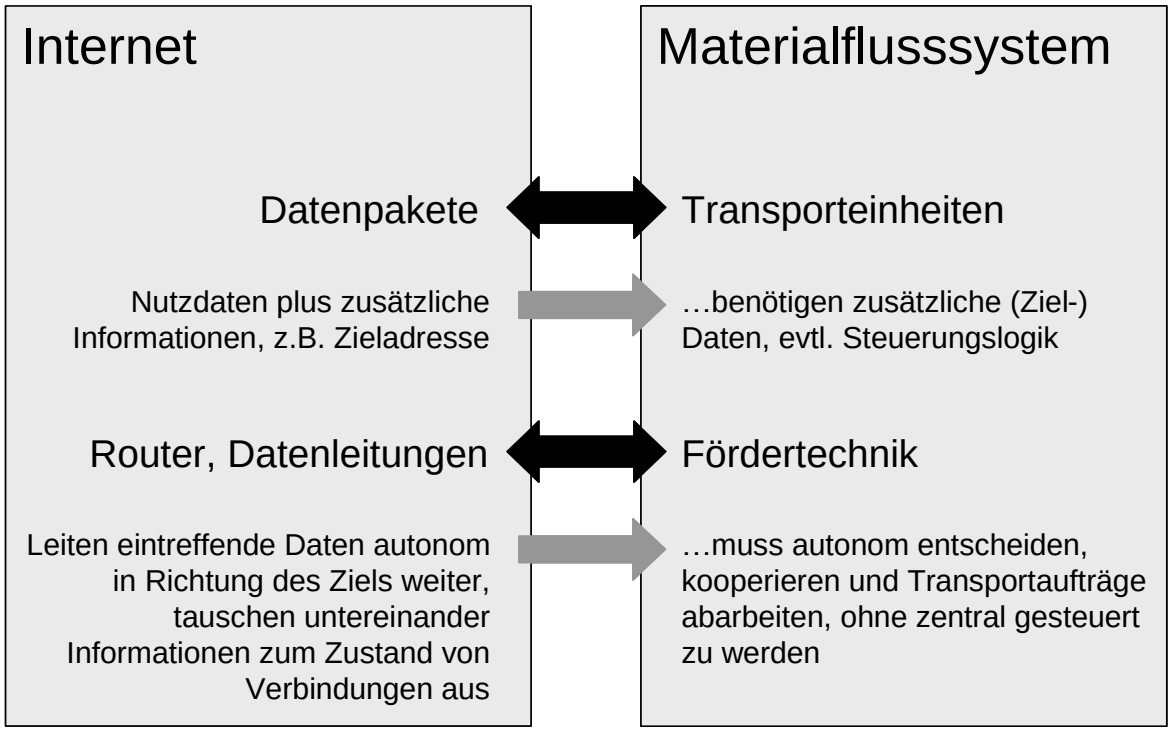


Abbildung 3-2 Analogie zwischen Internet und Materialflusssystemen

So wie die im Internet transportierten Datenpakete nicht nur aus Nutzdaten bestehen, sondern in so genannten Headern noch zusätzliche Informationen, wie z.B. die Zieladresse, eine Sequenznummer oder sonstige protokollspezifische Angaben enthalten, müssen auch physikalische Transporteinheiten wie Behälter oder Paletten mit zusätzlichen Informationen ausgestattet werden. Heutzutage üblich ist dabei der Einsatz von Barcodes, anhand derer Pakete eindeutig identifiziert werden können. Die dazugehörenden Informationen, beispielsweise der Inhalt eines Behälters oder sein Ziel, können dann anhand der ID aus einer zentralen Datenbank abgerufen werden. Neue Technologien, vor allem die Radio-Frequenz-Identifikation, kurz RFID, erlauben es aber, solche Informationen auf preisgünstigen Datenträgern direkt am transportierten Stückgut zu speichern. Durch diese Kopplung von Informations- und Warenfluss können Kommunikationsvorgänge und damit Schnittstellen minimiert und die Transparenz eines Systems erhöht werden. Obwohl wünschenswert, ist dieser Übergang vom Data-On-Network-Prinzip hin zu Data-On-Chip für die korrekte Funktionsweise des Internets der Dinge jedoch nicht zwingend notwendig.

Bei der Analogie zwischen Datenpaketen und Transporteinheiten ist weiterhin der Umstand zu beachten, dass Materialflusssysteme im Allgemeinen nicht nur den einfachen Transport von A nach B gewährleisten müssen. So müssen beispielweise

Kommissionierbehälter mehrere Kommissionierbahnhöfe, evtl. sogar in einer bestimmten Reihenfolge, anfahren und mit den gewünschten Artikeln befüllt werden, bevor sie eine Verpackungsstation und dann den Versand anfahren. Um solche komplexe Abläufe bzw. Abfolgen von sich gegenseitig bedingenden Transportaufträgen umsetzen zu können, genügt es also nicht, einer Transporteinheit nur zusätzliche Informationen durch das Anbringen eines RFID-Chips oder Barcodes mitzugeben. Anders als Datenpakete, die in ihrem Verhalten völlig passiv sind, werden Behälter und Pakete im Internet der Dinge auch ein aktives Verhalten zeigen müssen, um Förderprozesse zu starten und evtl. steuernd in den Betrieb der Anlage einzugreifen. Neben den erwähnten Zielinformationen muss daher jeder Transporteinheit also auch eine aktive Steuerungslogik, z.B. in Form eines Softwareagenten zugeordnet werden, der die korrekte Abarbeitung von beispielsweise Kommissionieraufträgen überwacht und gewährleistet (siehe Abbildung 3-3).

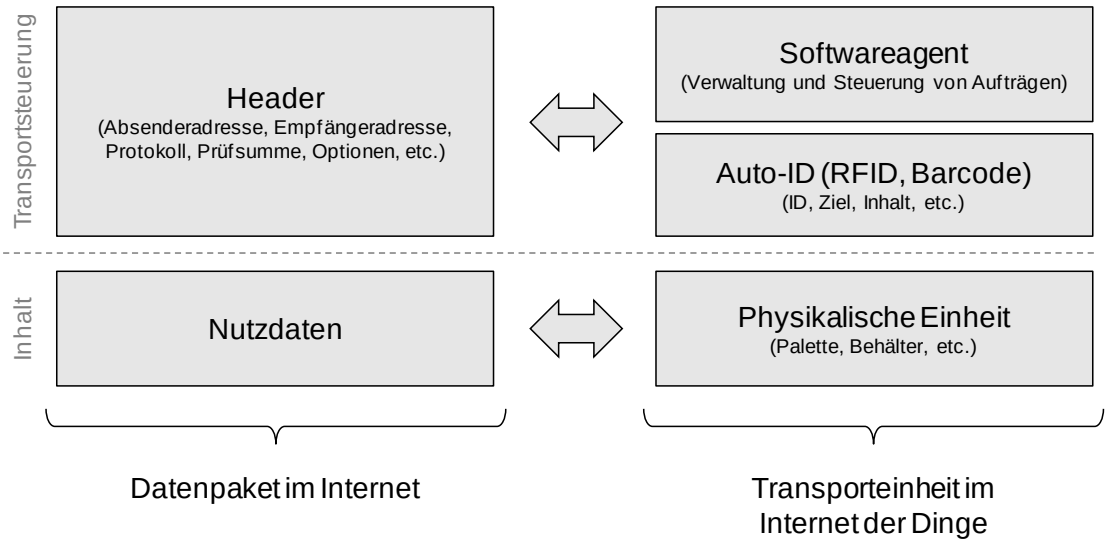


Abbildung 3-3 Analogie zwischen Datenpaketen und Transporteinheiten im Internet der Dinge

Der Transport dieser Güter geschieht in Materialflusssystemen durch verschiedene Arten von Fördertechnik, beispielsweise Unstetigförderer wie Stapler und fahrerlose Transportfahrzeuge (FTF) oder Stetigförderer wie Band- und Rollenförderer, die in herkömmlichen Systemen von zentralen SPS und Materialflussrechnern gesteuert werden (siehe Abschnitt 2.1). Die Intelligenz dieser überlagerten Entscheidungsebenen muss nun auf anlagennahe Komponenten, die „Router“ bzw. „Datenleitungen“ des Internet der Dinge, verteilt werden. Der erste dafür notwendige Schritt ist die Definition fördertechnischer Module, die möglichst generisch sind und sich in ihrer

Standardausführung für möglichst viele unterschiedliche Projekte bzw. Kundenanforderungen einsetzen lassen. Als Leitfaden für die Gestaltung einfach integrierbarer Module bietet sich eine funktionsorientierte Modularisierung [Wil-06] an. Dabei werden die Modulgrenzen zum einen so gezogen, dass Mechanik, Energieversorgung und Steuerungstechnik in einer mechatronischen Einheit mit einheitlichen Systemgrenzen gekapselt sind. Zum anderen ist darauf zu achten, dass ein Modul immer nur eine logistische Grundfunktion, also Fördern, Verteilen, Lagern oder Sammeln bzw. Zusammenführen [Arn-95], erfüllt. Auf diese Weise wird die Komplexität der Module reduziert und ihre Wiederverwendbarkeit erhöht. Ein Modul ist dann für die Steuerung aller eigenen Belange, beispielsweise das Schalten und Überwachen von Aktoren und Sensoren oder die Bearbeitung eines zugeteilten Auftrags, selbst zuständig und erfüllt diese in Eigenregie. Dabei können bzw. müssen Module aber auch miteinander kommunizieren, um bestimmte Prozesse, wie z.B. die Lastübergabe, abzusichern oder um eine dezentrale Auftragsdisposition umzusetzen.

Obwohl Transporteinheiten und Module bereits genügen, um ein funktionierendes Internet der Dinge zu realisieren, sind in Materialflusssystemen auch andere Funktionen notwendig, die über den reinen Transport hinaus gehen. Hier wären beispielsweise Visualisierungsumgebungen, Datenbanken oder Schnittstellen zu externen Systemen zu nennen. Ebenso ist es denkbar, dass in manchen Szenarien eine optimale Systemleistung bzgl. Größen wie Durchsatz oder Durchlaufzeiten nur dann gewährleistet werden kann, wenn Module und Transporteinheiten in ihrem Verhalten beeinflusst bzw. in ihrer Entscheidungsfreiheit eingeschränkt werden. So könnte z.B. ein Verkehrsleitsystem in einem komplexen EHB- oder FT-System dafür sorgen, dass Staus schneller aufgelöst werden oder gar nicht erst entstehen. Solche Agenten, die ausschließlich aus Software bestehen und nicht für die direkte Steuerung oder Überwachung physikalischer Komponenten oder Prozesse verwendet werden, werden im Internet der Dinge als Dienste bezeichnet.

Somit besteht das Internet der Dinge aus gleichberechtigten Entitäten, die sich in

- Module,
- Transporteinheiten (TE) und
- Dienste

gliedern lassen [Gün-08a].

3.3 Hardwareplattformen

Obwohl die Rechenplattform, auf der die Softwareagenten eines Materialflusssystems ausgeführt werden, keinerlei Einfluss auf die eigentliche Funktionalität des Internet der Dinge hat, hat die Wahl einer Hardwareplattform großen Einfluss sowohl auf die Kosten als auch auf die Robustheit eines Systems. Folgende drei Konfigurationen sind denkbar:

- Ein gemeinsamer Rechner für mehrere Agenten
- Ein dedizierter Rechner für jeden Agenten
- Migration eines Agenten über mehrere Rechensysteme hinweg

3.3.1 Hardwareplattformen für Dienste

Als reine Software-Entität, die im Normalfall koordinierende oder überwachende Funktionen für einen größeren Bereich der Anlage, sprich für mehrere Module und/oder Transporteinheiten, übernimmt, werden Dienste im Allgemeinen auf PC- oder Workstation-Systemen ausgeführt. Im Normalfall werden mehrere Dienste auf einem einzigen Rechner ausgeführt werden können, lediglich im Fall sehr komplexer und rechenintensiver Programme, wie z.B. einer Visualisierungsumgebung, erscheint es sinnvoll, einen dedizierten Rechner einzusetzen.

3.3.2 Hardwareplattformen für Transporteinheiten

Transporteinheiten können im Allgemeinen sowohl aus technologischen wie auch aus Kostengründen nicht mit einem eigenen Rechner ausgerüstet werden. Eine Möglichkeit besteht also darin, die Gesamtzahl der TE-Agenten auf einem oder mehreren Server-PCs auszuführen. Werden die Module, die die TEs transportieren, mit eigenen Rechnern ausgestattet, ergibt sich auch die Möglichkeit, den Agenten einer Transporteinheit durch das System zu migrieren, so dass er immer auf dem Rechensystem desjenigen Moduls ausgeführt wird, auf dem sich die TE physikalisch befindet. Der Nachteil bei diesem Vorgehen ist die erhöhte Kommunikationslast, da der gesamte Programmcode synchron zum Materialfluss durch das Netzwerk propagiert wird.

Unabhängig von der ausgewählten Hardwareplattform muss aber der Agent die Position und das Vorankommen des physikalischen Stückguts durch den Materialfluss beobachten können. Zu diesem Zweck muss die Transporteinheit, beispielsweise mittels Barcode oder RFID-Chip, eindeutig identifizierbar sein. Wird sie durch entsprechende Leseantennen oder Scanner im System geortet, muss die Information an den Agenten der Transporteinheit weitergeleitet werden.

3.3.3 Hardwareplattformen für Module

Module sind als mechatronische Einheiten mit einem eigenen Rechner auszustatten. Ist dies aus finanziellen Gründen nicht erwünscht oder möglich, ist es aber ebenfalls denkbar, mehrere Modulagenten auf einer gemeinsamen Plattform auszuführen, um so die Kosten für Rechenhardware zu senken.

Module weisen im Gegensatz zu Transporteinheiten und Diensten jedoch eine Besonderheit auf, die bei der Auswahl eines konkreten Rechensystems unbedingt beachtet werden muss. Die Steuerungslogik von Modulen muss zwei Bereiche mit stark unterschiedlichen Anforderungen abdecken:

- Die Steuerung und Überwachung physikalischer Vorgänge benötigt echtzeitfähige Soft- und Hardware.
- Die Kommunikation und Interaktion mit anderen Entitäten sowie das Treffen komplexer Entscheidungen benötigt keine harte Echtzeitfähigkeit, dafür aber eine viel komplexere Logik und Kommunikation als die einfache E/A-Überwachung.

Daher erscheint es sinnvoll, diese zwei Aufgabenbereiche mit dafür spezialisierten Werkzeugen bzw. Programmiersprachen zu realisieren. Im anlagennahen Bereich werden heute fast durchgängig die verschiedenen IEC-61131-3 Programmiersprachen [IEC-61131-3] eingesetzt. Diese stellen die beste und einfachste Möglichkeit dar, um Ein- und Ausgänge zu überwachen bzw. zu schalten oder funktionale Abläufe festzulegen. Dabei machen es erst die von einer SPS garantierten Taktzeiten möglich, schnell ablaufende Prozesse zuverlässig zu steuern und vor allem auch Sicherheitsbestimmungen einzuhalten. Mechanismen der objektorientierten Programmierung, wie z.B. Kapselung, Vererbung und Polymorphie, fehlen dabei aber weitgehend. Das Paradigma der Objektorientiertheit ist aber eine Voraussetzung für

die Erstellung modularer, übersichtlicher und leicht wieder verwendbarer Software. Die neuere IEC-61499-1 Norm [IEC-61499-1] führt zwar einige dieser Konzepte ein, hat sich aber bisher kaum durchsetzen können.

Im Gegensatz dazu bieten PC-basierte Umgebungen und Betriebssysteme wie Windows oder Linux schon seit Langem den Komfort höherer Programmiersprachen und haben zudem auch den Vorteil vergleichsweise günstiger und sehr leistungsfähiger Hardware. Zahlreiche und sehr umfangreiche Bibliotheken für verschiedensten Anwendungen stehen als Grundlage für die Entwicklung eigener Anwendungen zur Verfügung und verringern so den Programmieraufwand. Der große Nachteil der PC-Systeme in Bezug auf die Automatisierungstechnik besteht aber darin, dass sie zumeist nicht echtzeitfähig sind.

Computer, die sowohl eine PC- als auch eine SPS-Plattform vereinen und somit die Vorteile beider Rechenumgebungen kombinieren, sind bereits seit Jahren zu günstigen Preisen am Markt verfügbar. Diese Industrie- oder Embedded-PCs verwenden üblicherweise Desktop- oder ähnliche Prozessoren und sind sowohl mit einem gängigen Betriebssystem wie Windows oder Linux als auch mit einer Soft-SPS, also einer echtzeitfähigen Laufzeitumgebung für IEC-61131 Programme, ausgestattet. Diese zwei auf einem einzigen Rechner kombinierten Softwareschichtenumgebungen können dabei über verschiedene Schnittstellen miteinander kommunizieren.

Solche Industrie-PCs ermöglichen somit die Umsetzung der für Module notwendigen Zwei-Schicht-Architektur. Dabei werden alle höherwertigen Aufgaben wie Kommunikation und Kooperation, Verwaltung von Aufträgen und Kapazitäten oder Wegplanung und Wegeoptimierung dem Softwareagenten zugeordnet, der wie Dienste und TE-Agenten mit höheren Programmiersprachen erstellt wird. Die Ansteuerung und Überwachung der Ein-/Ausgänge bzw. physikalischer Gegebenheiten werden in eine mit klassischen SPS-Sprachen programmierte und echtzeitfähige Maschinensteuerungsebene ausgelagert, die dem Agenten unterlagert ist und keine Schnittstellen zu anderen Modulen besitzt (siehe Abbildung 3-4).

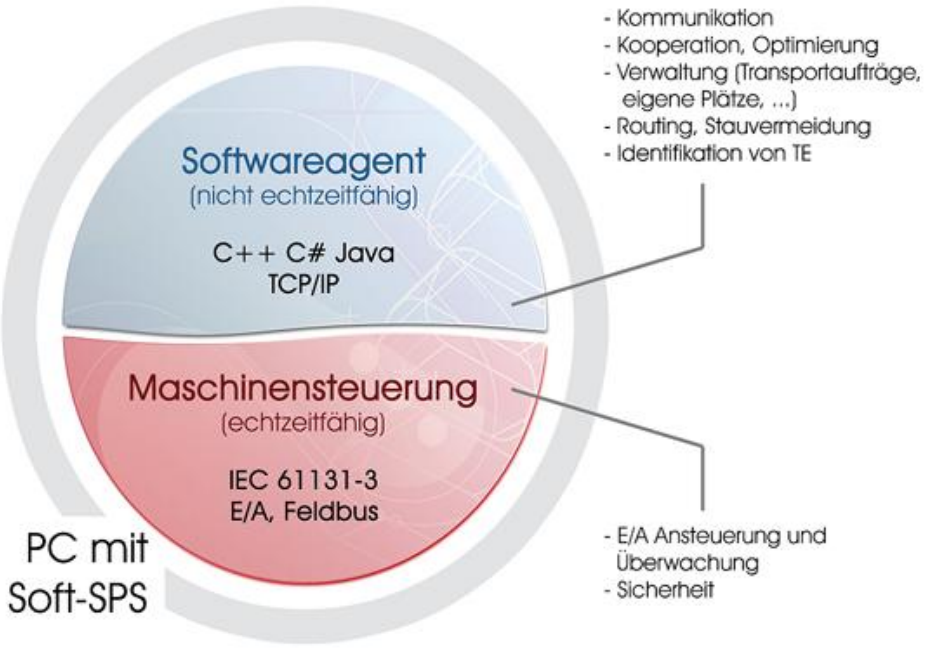


Abbildung 3-4 Zwei-Schicht-Architektur für Module

3.4 Softwarearchitektur

Bei der Entwicklung einer Softwarearchitektur für das Internet der Dinge ist zum einen die interne Struktur der Agenten, also deren verschiedene Funktionen mit ihren jeweiligen Anforderungen sowie deren sequenzielle oder parallele Abarbeitung zu betrachten. Zum anderen ist auch der Aufbau des gesamten Softwarebaukastens von großer Relevanz. Dabei ist auf den effektiven Einsatz objektorientierter Programmiertechniken, vor Allem der Vererbung von Klassen, zu achten, um auf diese Weise die einfache Wiederverwendung von Programmcode zu ermöglichen. Damit wird sowohl der Aufwand für die Entwicklung neuer Software gesenkt als auch deren Zuverlässigkeit erhöht.

3.4.1 Agenteninterne Softwarearchitektur

3.4.1.1 Interne Prozesse einer allgemeinen Entität

Wie im Abschnitt 2.2 beschrieben besteht die Aufgabe eines Agenten darin, anhand eines durch Sensoren und/oder Kommunikation generierten Weltbildes, Entscheidungen zu fällen oder langfristig zu planen und durch das Ausführen verschiedener

Aktionen auf Umweltveränderungen zu reagieren, um so die eigenen Ziele zu verfolgen.

Da alle Entitäten im Internet der Dinge miteinander kommunizieren und Daten austauschen, bildet dieser rein informationstechnische Aspekt der Umweltbeobachtung eine Kernfunktion aller Agenten. Diese wird daher in der Basisklasse *Entitaet* implementiert und steht dann über die Vererbung allen anderen, von *Entitaet* abgeleiteten Agenten des Systems zur Verfügung (siehe Abschnitt 3.4.2.2 bzw. Abbildung 3-8).

Beim Start einer Entität werden also, neben der Initialisierung der internen Variablen, auch nebenläufige Vorgänge, so genannte Threads, für das Empfangen und Abarbeiten von Nachrichten gestartet. Diese haben den Vorteil, dass die verschiedenen Aufgaben eines Agenten, so z.B. die Beobachtung der Umwelt bzw. das Empfangen von Nachrichten, die Verarbeitung dieser Daten und das Planen oder das Durchführen von Aktionen auf Grundlage dieser Informationen auch unabhängig voneinander bzw. parallel ausgeführt werden können. Dies erlaubt eine bessere Strukturierung bzw. Gliederung von agenteninternen Vorgängen, die zwar prinzipiell aufeinander aufbauen, nicht aber zeitlich strikt sequenziell abgearbeitet werden müssen.

Zwar lassen sich das Empfangen und Abarbeiten einer Nachricht theoretisch direkt miteinander verbinden und im selben Thread ausführen, die Entkopplung dieser beiden Vorgänge erlaubt aber ein besseres Scheduling der Datenverarbeitung. So können Nachrichten vor dem Verarbeiten beispielsweise nach Priorität sortiert werden. Während also in einem Thread Nachrichten nur empfangen und in einer evtl. sortierten Warteschlage gespeichert werden, werden diese Nachrichten gemäß ihrer Sortierung von einem anderen Thread zur Verarbeitung durch den Agenten freigegeben (siehe UML-Aktivitätsdiagramm in Abbildung 3-5).

Werden im System Dienste eingesetzt, deren Verwendung für alle Entitäten verbindlich oder zumindest erwünscht ist, so können zusätzliche Threads für das Auffinden und Aufrufen solcher Dienste ebenfalls in der Entität definiert werden. Ein Beispiel hierfür ist die Verwendung eines Blackboards bzw. einer Datenaustauschplattform, wie sie in Abschnitt 0 dieser Arbeit beschrieben wird.

Die konkrete Interpretation der Daten bzw. die Generierung des agenteninternen Weltmodells sowie die Berechnung und Durchführungen möglicher Aktionen sind allerdings von der jeweiligen Aufgabe eines Agenten abhängig und werden sich

somit zwischen Transporteinheiten, Diensten und Modulen stark unterscheiden. Die Implementierung dieser Funktionen wird also erst später vorgenommen und kann nicht in der Basis-Entität hinterlegt werden.

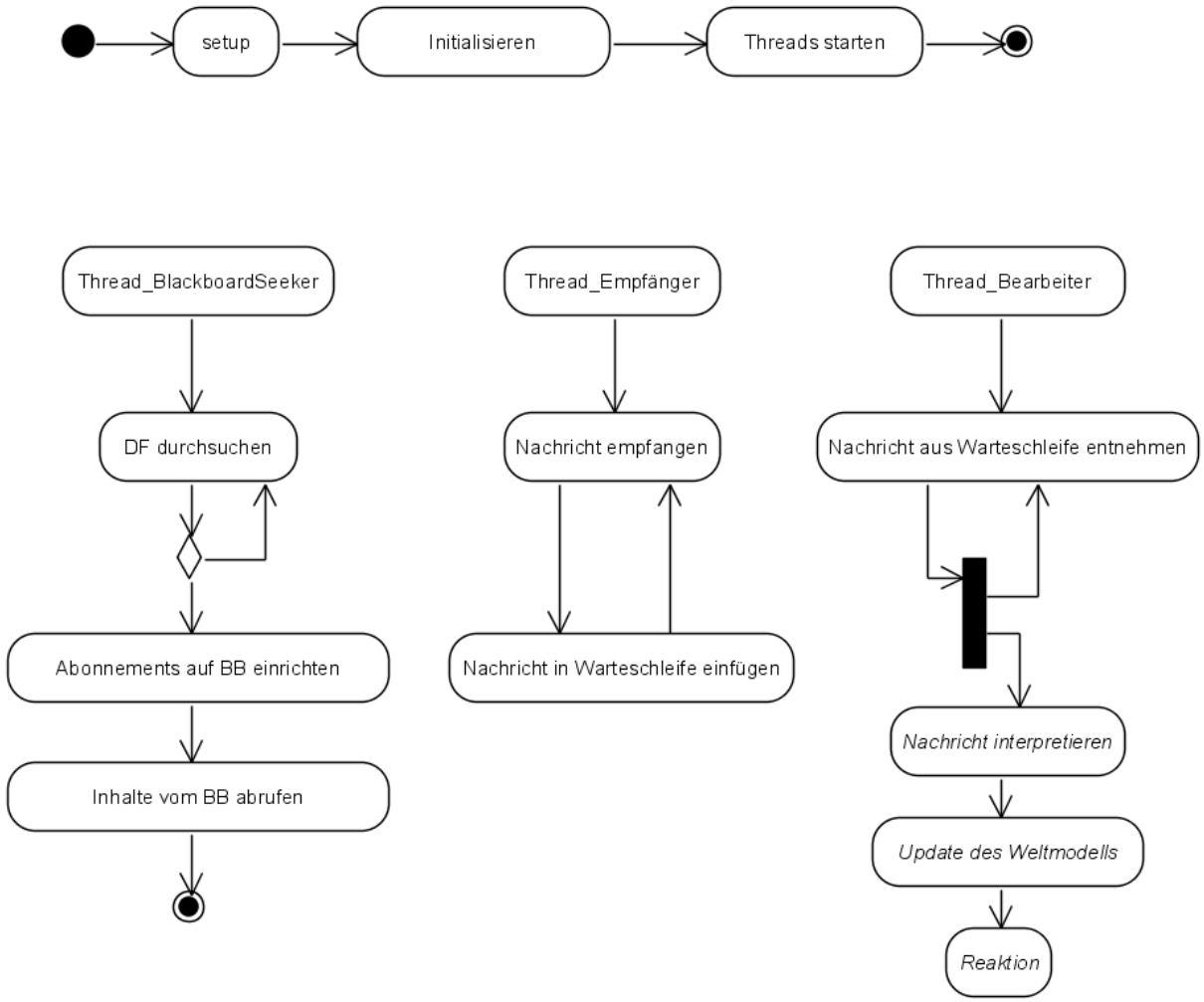


Abbildung 3-5 Interne Abläufe einer allgemeinen Entität

3.4.1.2 Interne Prozesse eines Moduls

Die Steuerungslogik der Module besteht im Unterschied zur Logik von Diensten und Transporteinheiten aus zwei Schichten (siehe auch Abschnitt 3.3.3). Während der Agent komplexere Aufgaben übernimmt, wie z.B. Kommunikation oder Kooperation im Agentensystem und sich in Funktion und Aufbau an elementaren logistischen Funktionen orientiert, muss die Maschinensteuerungsebene an die jeweils eingesetzte Mechanik und Elektrik des Moduls angepasst werden. Diese wird im Normalfall hersteller- und aufgabenspezifisch ausfallen. Um die Wiederverwendbarkeit der Agentenlogik nicht durch Integrationsschwierigkeiten an der Schnittstelle zwischen

den zwei Steuerungsebenen zu mindern, ist es daher notwendig, die internen Kommunikationsprotokolle der eingesetzten Steuerungshardware sowie die Beschreibung der auszutauschenden Variablen von der eigentlichen Agentenlogik zu trennen und in einer gesonderten, frei konfigurierbaren Schicht zu kapseln.

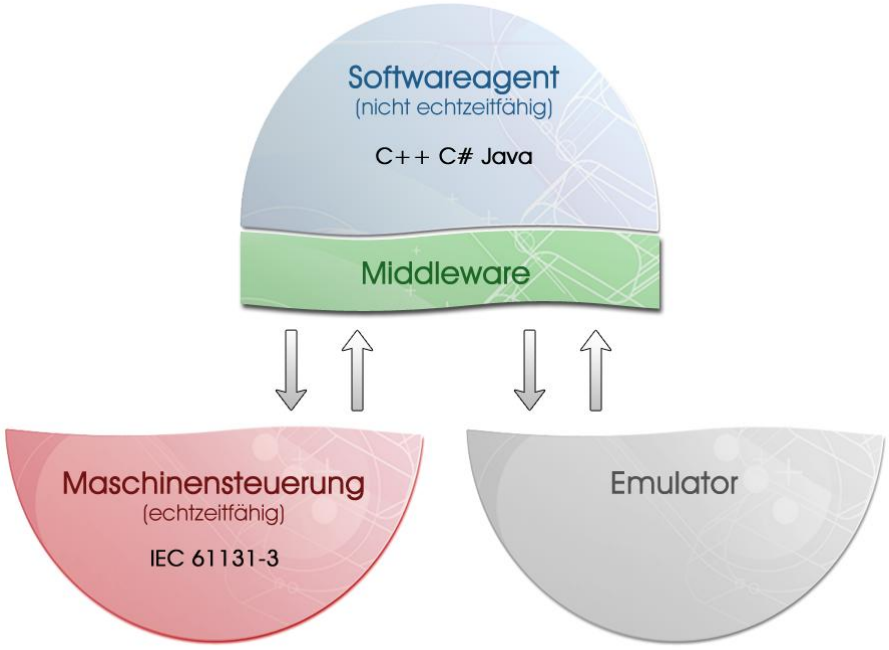


Abbildung 3-6 Modulinterne Middleware

Diese Ebene erfüllt die Funktion einer Middleware, also einer Vermittlungsschicht, die die verschiedene Kommunikationsprotokolle bündelt und die internen Abläufe des Agenten bzw. der Maschinensteuerung vor dem jeweils anderen Steuerungsbaustein versteckt. Diese ist, um der Forderung nach Dezentralisierung, Modularisierung und Kapselung gerecht zu werden, keine eigenständig lauffähige Software, sondern eine Klasse bzw. eine Sammlung von Klassen, die in jedem Agenten instanziiert werden und eine Schnittstelle zum Lesen und Schreiben von Variablen mit verschiedenen Datentypen anbieten.

Wird die Maschinensteuerungsschicht angepasst oder z.B. zu Testzwecken durch einen Emulator ersetzt, kann die Middlewareschicht über eine Konfigurationsdatei oder auch eine Nachricht an den Agenten auch während des Betriebs umkonfiguriert werden. Die Kommunikationsprotokolle für die einzelnen Variablen lassen sich beliebig kombinieren, beispielsweise um sowohl mit der Maschinensteuerungsebene als auch mit anderen beigefügten Komponenten, wie z.B. einem RFID-Reader, zu kommunizieren. Da der Emulator wiederum als Agent realisiert werden kann, muss bei

der Bearbeitung von über Agentenkommunikation erhaltenen Nachrichten geprüft werden, ob diese vom Emulator des Moduls oder von einer anderen Entität im System stammen.

Für einen Querverschiebewagen bedeutet dies beispielsweise, dass der Softwareagent zum Aufnehmen oder Abgeben einer Palette die Millimeterposition anfahren muss, an der sich die entsprechende Zuführ- oder Abtransportstrecke befindet. Der Agent übergibt diese Sollposition als Integer-Wert an die Maschinensteuerung. Diese kann nun die Antriebe anschalten und dem Agenten über das Setzen einer bool'schen Variable mitteilen, dass sich der Verschiebewagen gerade bewegt. Während der Fahrt wird auch die aktuelle Millimeterposition von der Maschinensteuerung dem Agenten als Integer-Wert bereitgestellt. Das Erreichen der Sollposition bzw. das Beenden des Fahrauftrags wird von der Maschinensteuerung über einen weiteren Bool-Wert an den Agenten gemeldet. Somit weiß dieser nun, dass er den Lastwechsel durchführen kann und wird sich zu diesem Zweck beispielsweise mit der entsprechenden Rollenbahn koordinieren müssen.

Somit haben Module, als mechatronische Einheiten, mehr Möglichkeiten zur Informationsgewinnung als die Basis-Entität. Denn neben der Kommunikation mit anderen Agenten erhalten Module einen Großteil ihres Wissens von der eigenen Maschinensteuerung – sei diese nun wiederum ein reines Softwareprogramm oder eine an Sensoren und Aktoren angebundene Logik.

Auch hier muss wiederum zwischen Sensor-/Aktordaten zur Ansteuerung der Modulmechanik und dem Lesen bzw. Schreiben von RFID-Chips unterschieden werden, da diese beiden Prozesse bezüglich Datenformaten und -mengen und vor allem der benötigten Zykluszeiten unterschiedliche Anforderungen aufweisen. Diese beiden Informationsquellen können zyklisch und in verschiedenen Zeitintervallen abgerufen werden, wobei Veränderungen der Eingangsvariablen der Maschinensteuerungsebene oder detektierte RFID-Datensätze sofort zur Verarbeitung freigegeben und vom Agenten interpretiert werden.

Das Basis-Modul erweitert also die Basis-Entität um zwei weitere Threads zur Kommunikation mit der Maschinensteuerungsebene und zum Lesen von RFID-Tags (siehe UML-Aktivitätsdiagramm in Abbildung 3-7: von der Basis-Entität übernommene Funktionen werden weiß dargestellt, neue Elemente sind grau eingezeichnet).

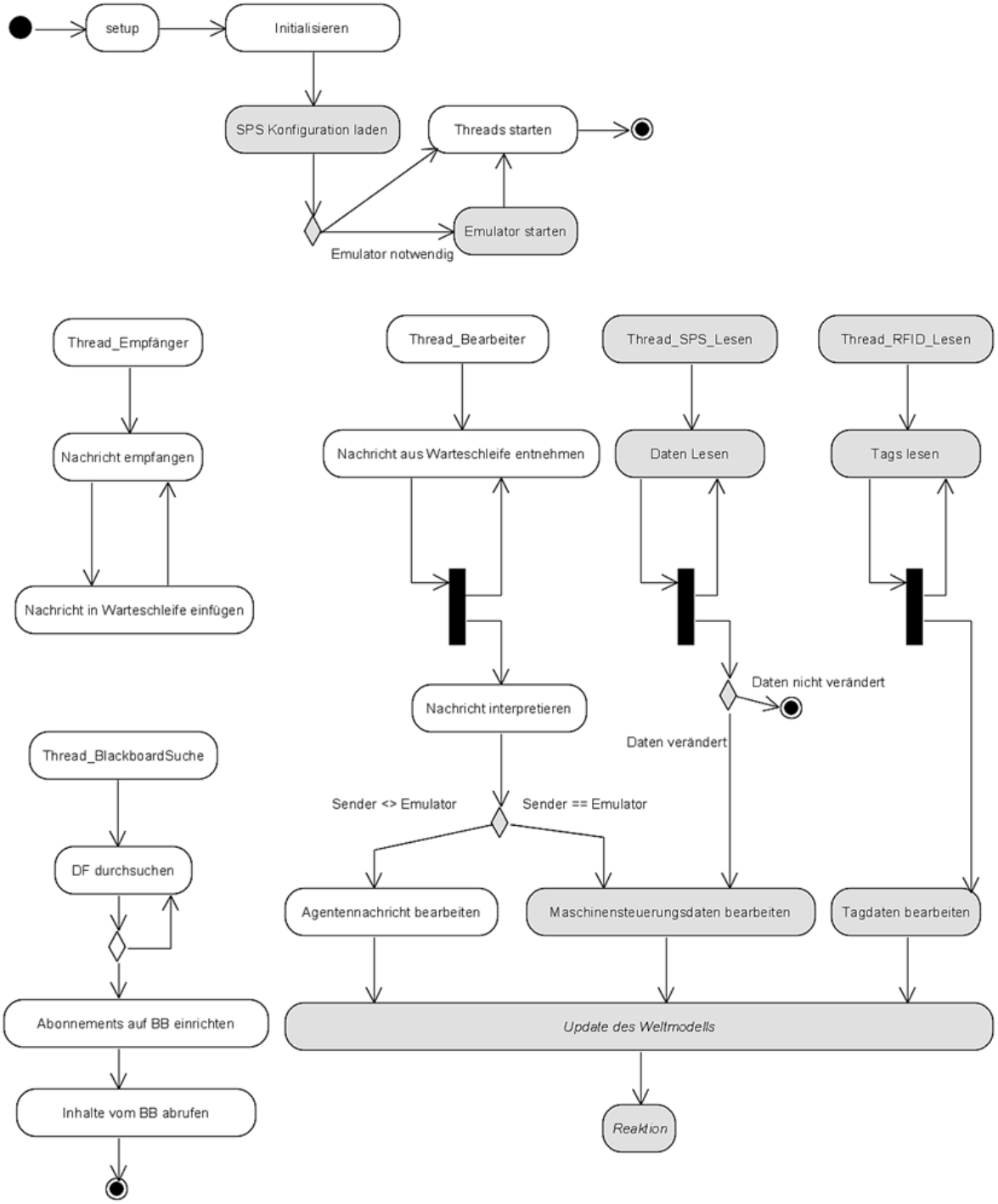


Abbildung 3-7 Interne Abläufe eines allgemeinen Moduls

Die Interpretation der Daten und die Implementierung von Verhaltens- oder Reaktionsmustern muss auch in diesem Fall wieder modulspezifisch ausgeführt werden. Allerdings enthält diese Klasse bereits umfangreiche Methoden zur Kommunikation sowohl mit Entitäten als auch mit der Maschinensteuerung, die ohne weiteren Aufwand für alle Modultypen wiederverwendet werden können.

3.4.2 Architektur des Softwarebaukastens

3.4.2.1 Agentenframeworks

Der Startpunkt für die Entwicklung eines Agentensystems ist ein so genanntes Agentenframework. Dieses enthält Methoden für die Instanziierung und das Management von Agenten, für die Suche von Systemteilnehmern anhand eindeutiger Namen oder abstrakter Funktionalitätsbeschreibungen und für das Senden und Empfangen von Nachrichten. Zusätzlich können auch verschiedene Kommunikationsprotokolle oder Methoden zum automatischen Interpretieren von Nachrichten enthalten sein. Diese Funktionalität wird in Form von Basisklassen bereitgestellt, von denen alle weiteren Agenten abgeleitet werden müssen. Ein solches Framework ist das frei verfügbare Java Agent Development Framework, kurz JADE [JADE]. Der durchgängige Einsatz dieser oder einer ähnlichen Programmierbibliothek für die Erstellung aller Agenten – sowohl für Dienste als auch für Transporteinheiten und Module – erlaubt es, die Vorteile der Objektorientiertheit bzgl. Vererbung und Wiederverwendung von Code voll auszunutzen.

3.4.2.2 Klassenhierarchie für das Internet der Dinge

Die Softwarearchitektur bzw. Klassenhierarchie des Agentensystems sollte die Modularisierungssystematik und die Standardentitäten und -funktionsklassen des Internet der Dinge (siehe Abschnitt 3.2) direkt aufgreifen. Dadurch ergeben sich folgende Klassen, die in Abbildung 3-8 als UML-Klassendiagramm dargestellt werden:

- *Entitaet* wird vom im Agentenframework implementierten Basisagenten abgeleitet und enthält zusätzliche Funktionen, die in einem Materialflusssystem für Module, Dienste und Transporteinheiten gleichermaßen wichtig sind. Dazu zählt z.B. das Anmelden bei einem Blackboard-Dienst (siehe Kapitel 4) oder die Verwaltung einer evtl. priorisierten Warteschlange mit empfangenen aber noch nicht verarbeiteten Nachrichten (siehe Abschnitt 3.4.1). Diese Klasse ist abstrakt, kann also nicht direkt instanziiert werden, dient aber als Grundklase für Module, Dienste und Transporteinheiten.
- *TE* entspricht dem Agenten einer Transporteinheit und wird von *Entitaet* abgeleitet. Die TE enthält zusätzlich eine *WorkflowEngine*, also einen Baustein

zum Interpretieren, Abarbeiten und Verwalten frei definierbarer logistischer Arbeitsabläufe bzw. Workflows (siehe Abschnitt 3.5.3).

- *Dienst* ist eine von *Entitaet* abgeleitete Klasse und dient als Basisklasse für reine Softwareagenten, z.B. eine Visualisierung mit grafischer Benutzeroberfläche (siehe Abschnitt 5.3.2) oder ein Blackboard mit Anbindung an ein externes Datenbanksystem (siehe Kapitel 4).
- *Modul* wird ebenfalls von *Entitaet* abgeleitet und bildet die Grundlage für die Implementierung verschiedener Modultypen wie Unstetig- oder Stetigförderer. Die Klasse *Modul* enthält bereits eine *SPSMiddleware* zur Kommunikation mit der unterlagerten Maschinensteuerung bzw. einem Emulator (siehe Abschnitt 3.4.1.2) und einen standardisierten Mechanismus zur Lastwechselkoordination mit anderen Modulen.
- *Unstetigfoerderer, Verzweigung* und *Stetigfoerderer* sind spezialisierte Module und enthalten für den jeweiligen Typ charakteristische Funktionen oder Bausteine. So können Unstetigförderer oder Verzweigungen eine Wegplanung durchführen. Diese Klassen dienen wiederum als Grundlage für die Ableitung konkreter Module wie z.B. EHB-Katzen, Querverschiebewägen (QVW), Weichen oder Rollenförderern, die ihrerseits noch speziellere Funktionalität enthalten – so z.B. die Ansteuerung eines Lastaufnahmemittels für EHB-Katzen oder das Anfahren festgelegter Millimeterpositionen für den QVW. Obwohl die Agentenklassen auf dieser Detaillierungsebene bereits instanziiert, also direkt genutzt werden können, können auch weitere, noch speziellere Agenten von diesen abgeleitet werden.
- *Emulator* kann direkt vom Basisagenten des Frameworks abgeleitet werden und dient als Grundlage für Emulatoren für konkrete Module, wie z.B. EHB-Katzen, Weichen oder Querverschiebewägen. Die *Emulator*-Klasse enthält dabei Methoden zur Kommunikation mit der *SPSMiddleware* eines Moduls, während die abgeleiteten Klassen (*Emulator_Katze*, *Emulator_Weiche*, etc.) die Funktion der jeweiligen Maschinensteuerungsschicht implementieren – so z.B. das Durchführen von Schaltvorgängen inkl. dem Einhalten von Schaltzeiten oder die Veränderung der aktuellen Position des Gerätes unter Berücksichtigung der Gerätegeschwindigkeit.

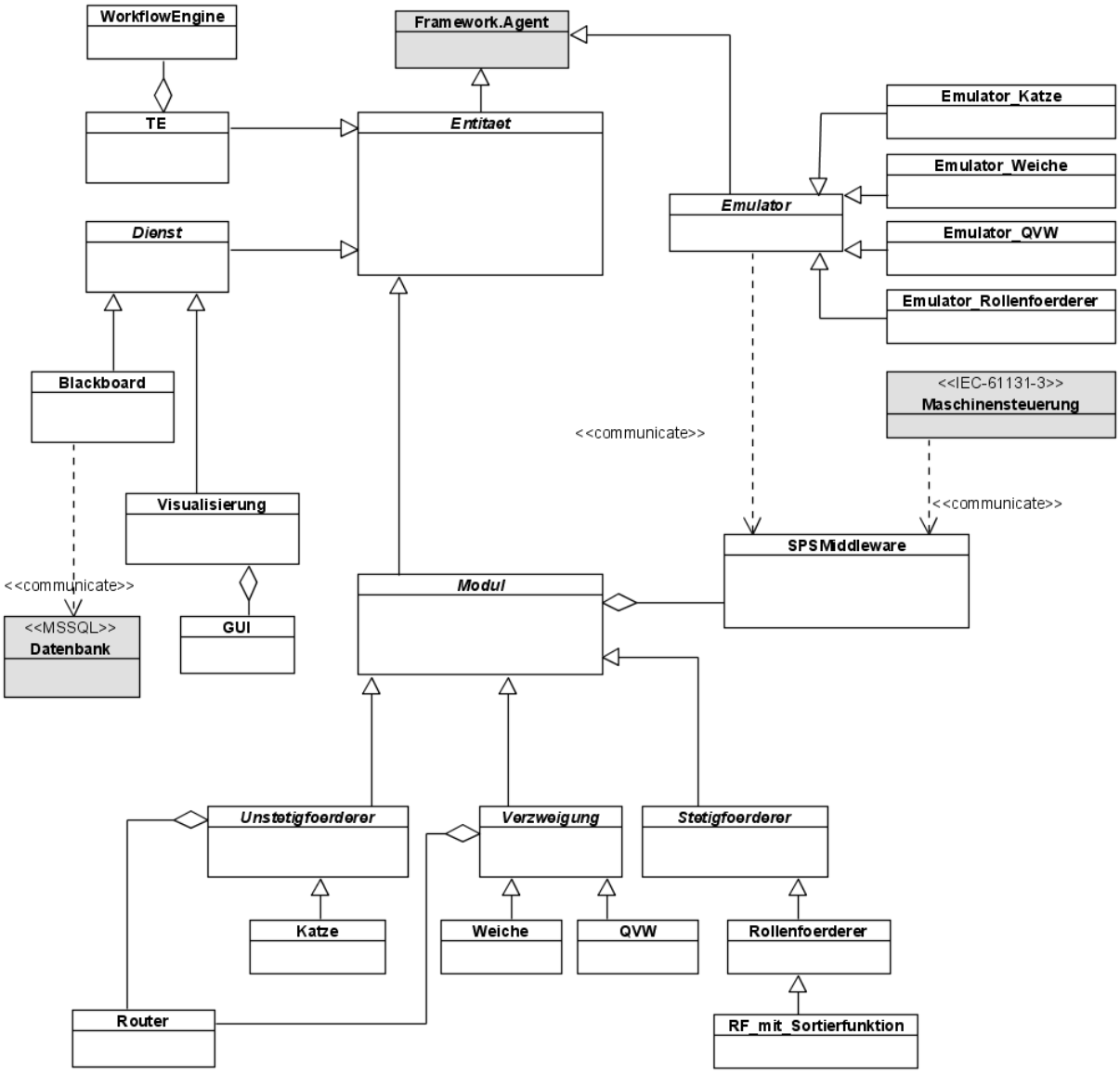


Abbildung 3-8 Klassenhierarchie für das Internet der Dinge

3.5 Das WWW des Internet der Dinge: Wandelbarkeit, Wiederverwendbarkeit, Workflows

Obwohl die Internet-Technologie größtenteils schon seit den 70er Jahren verfügbar war, hat erst der WWW-Boom der 90er Jahre zu einer weitflächigen Verbreitung dieses Datennetzwerkes geführt. Ein Grund dafür könnte in der Tatsache liegen, dass erst das World Wide Web die einfache Verwendbarkeit des Internet und vor allem seinen Nutzen für eine riesige Anzahl an Unternehmen und privaten Benutzern sichtbar und greifbar gemacht hat.

Auf ähnliche Weise hängt auch der Erfolg des Internet der Dinge nicht nur von einer zukunftsorientierten und ausgereiften technologischen Basis ab, sondern vor allem von seinem Nutzen für die Industrie bzw. Logistikbranche. Wie eingangs erwähnt ist in diesem Bereich eine hohe Flexibilität und Wandelbarkeit von Systemen ein essenzieller Erfolgsfaktor, wobei niedrige Betriebs- und Investitionskosten eine wichtige Randbedingung sind. Das Internet der Dinge verspricht durch den Einsatz standardisierter, intelligenter und „Plug and Play“-fähiger Fördertechnikmodule und anderer Entitäten eine erhebliche Senkung des Engineering- und Inbetriebnahmeaufwands [Gün-08c]. Dabei ist eine zentrale Frage bisher aber noch offen: Können die in der Logistik typischen, äußerst heterogenen Kundenanforderungen, Prozesse und Systeme denn überhaupt mit Standardkomponenten abgedeckt werden oder ist ein erheblicher Aufwand für projektspezifische Anpassungen und Erweiterungen unvermeidbar? Wenn dies der Fall ist, könnten Konzepte wie das Internet der Dinge nur in sehr begrenztem Umfang Vorteile bieten.

Der vorliegende Abschnitt befasst sich daher mit der Analyse der Wandelbarkeit eines Systems vor allem unter funktionalen Aspekten und zeigt Möglichkeiten auf, die einen hohen Wiederverwendungsgrad von Komponenten in der Logistik ermöglichen.

3.5.1 Prozessflexibilität

Frühere Arbeiten auf dem Gebiet dezentral gesteuerter Materialflusssysteme definieren drei Aspekte, die die Wandelbarkeit eines Systems beeinflussen [Han-01] [Gün-02a]:

- Layoutflexibilität,
- Fördergutflexibilität und
- Durchsatzflexibilität.

Diese Faktoren berücksichtigen vor allem physikalische Gegebenheiten, z.B. die Position von Übergabestellen im System oder die Geometrie, Beschaffenheit und Anzahl der zu transportierenden Einheiten. Die zusätzlich geforderten Eigenschaften der Integrations- und Erweiterungsfähigkeit [Gün-04] berücksichtigen darüber hinaus auch die mechanischen, energetischen und informationstechnischen Schnittstellen der Systemkomponenten bzw. Module. Dieses Konzept erlaubt damit die Bewertung eines Systems bzw. den Vergleich mehrerer technischer Systemvarianten, die für eine spezifische Aufgabe entwickelt und ausgelegt werden (siehe Abbildung 3-9).

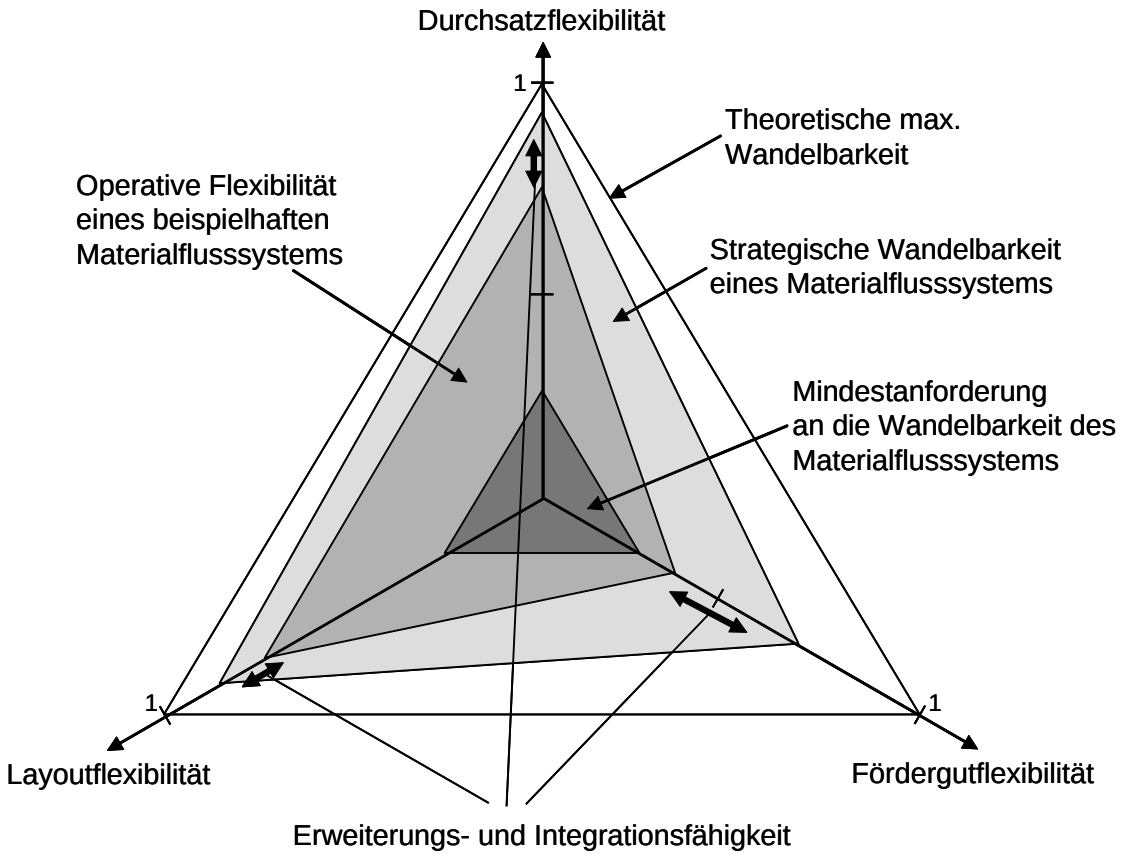


Abbildung 3-9 Beurteilung der Wandelbarkeit eines Materialflusssystems [Wil-06]

Dabei bleibt aber die Flexibilität bezüglich der logistischen Prozesse im System weitgehend unbeachtet. Zwar sind die Layout-, Fördergut- und Durchsatzflexibilität sowie die Integrations- und Erweiterungsfähigkeit notwendig, um Prozesse in einer Anlage

mit geringem Aufwand verändern zu können. Aber Abläufe wie z.B. Kommissionier- oder Sortierstrategien stellen eine andere Perspektive auf das Problem der Flexibilität bzw. Wandelbarkeit eines Systems dar und beziehen sich vor allem auf die logistische Gesamtaufgabe und nicht auf die konkrete, mechanische Auslegung oder Erweiterung einer Anlage. Dabei sind gerade die zunehmende Dynamik und immer schwierigere Prognose zukünftiger funktionaler Anforderungen an ein Materialflusssystem Aspekte, mit denen die Logistik immer häufiger konfrontiert wird. Diese Problematik erfordert die Abkehr von der in Produktion und Logistik bisher vorherrschenden Orientierung an festen, vorgeplanten und anlagenspezifischen Prozessen und den Wandel hin zu einer losen Kopplung funktionaler Einheiten [Hom-08].

Die *Prozessflexibilität* beschreibt somit die Möglichkeit, die in einer Anlage ablaufenden logistischen Prozesse mit minimalem Aufwand zu verändern, ohne dabei einen zentralen Steuerungscomputer oder zahlreiche autonome Einheiten umprogrammieren oder umkonfigurieren zu müssen. Das System muss, im Rahmen der physikalischen und mechanischen Gegebenheiten, sozusagen in der Lage sein, auf Knopfdruck eine neue oder veränderte logistische Aufgabe umzusetzen. Diese Prozessflexibilität lässt sich als eine neue Dimension des bisherigen Wandelbarkeitsmodells auffassen (siehe Abbildung 3-10), da sie sich auf die Veränderbarkeit von Funktionalitäten und Interaktionsmustern bezieht, nicht aber auf die Fähigkeit der Einzelkomponenten, unter unterschiedlichen Randbedingungen korrekt zu funktionieren.

Demnach lässt sich die Prozessflexibilität auch nicht durch die Erweiterungs- und Integrationsfähigkeit eines Systems verbessern oder vermindern, sondern ist eher eine Voraussetzung für diese: Ein System bzw. Systemkomponenten, die sich auf rein technischer Ebene dank standardisierter informationstechnischer oder auch mechatronischer Schnittstellen leicht integrieren lassen, müssen auch in Bezug auf die Geschäftslogik des Systems so konfiguriert werden können, dass sie zielgerichtet mit anderen Entitäten interagieren.

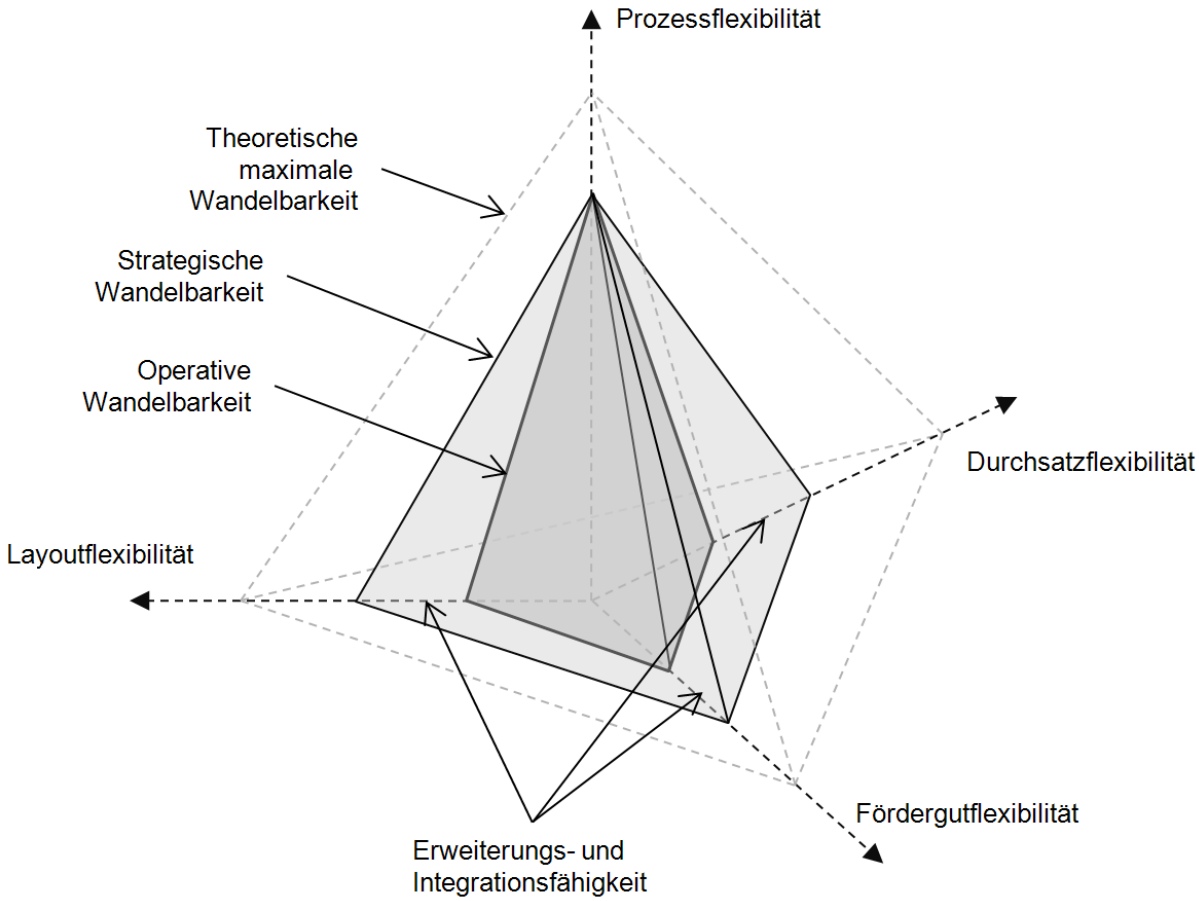


Abbildung 3-10 Bedeutung der Prozessflexibilität für die Wandelbarkeit eines Systems

Bei der Entwicklung der grundlegenden architektonischen Konzepte für das Internet der Dinge ist die Berücksichtigung und Sicherstellung einer möglichst hohen Prozessflexibilität sogar von besonderer Bedeutung. Denn die Implementierung eines Baukastens aus intelligenten und hochgradig standardisierten Entitäten ist nur dann technisch und vor allem auch wirtschaftlich sinnvoll, wenn sich diese Standardkomponenten ohne Veränderung für die Realisierung möglichst vieler unterschiedlicher Materialflusssysteme wiederverwenden lassen. Die Prozessflexibilität als ein Maß für die „funktionale Wandelbarkeit“ einer Systemarchitektur ist somit die Voraussetzung für eine hohe Wiederverwendbarkeit dieser Komponenten.

3.5.2 Materialflusssteuerung und logistische Prozesse

Da gerade die innerbetriebliche Logistik von einer sehr hohen Heterogenität der funktionalen Anforderungen und Prozesse gekennzeichnet ist, sind technologische Entwicklungen notwendig, die den scheinbaren Widerspruch zwischen dem Einsatz standardisierter mechatronischer Einheiten und der Umsetzbarkeit kunden- und

projektspezifischer Anforderungen auflösen. Das vorgestellte Modularisierungskonzept des Internet der Dinge verzichtet aus diesem Grund ganz bewusst auf die Definition übergeordneter Abläufe und beschränkt sich auf die Betrachtung elementarer logistischer Funktionen wie Fördern, Verzweigen oder Zusammenführen. Darauf aufbauend muss aber die übergeordnete, meist projektspezifische Geschäftslogik einer materialflusstechnischen Anlage realisiert werden. Denn die elementaren Funktionen, die von den Modulen bereit gestellt werden, liefern erst in ihrer Kombination bzw. in der sinnvollen Verkettung in Form eines (anlagenspezifischen) Arbeitsablaufs bzw. Workflows einen konkreten Nutzen.

Es stellt sich also die Frage, welche Entität oder Entitäten im Internet der Dinge für die Verwaltung des oder der Workflows in einer Anlage, also für die Umsetzung der kundenspezifischen logistischen Prozesse, verantwortlich sind. Bei genauer Betrachtung stellt sich heraus, dass der scheinbare Widerspruch zwischen dem Einsatz standardisierter Komponenten und der Umsetzbarkeit kundenspezifischer Wünsche im Internet der Dinge durch eine strikte Trennung der

- Materiaflusssteuerung einerseits (die von den Modulen umgesetzt wird) und der
- logistischen Prozesse andererseits (die von der Transporteinheit in Form eines individuellen Workflows verwaltet und abgearbeitet werden)

aufgelöst werden kann.

Anders ausgedrückt weiß eine Transporteinheit, *was* zu tun ist – nicht aber, wie dies geschieht – während ein Modul weiß, *wie* es die von den TEs angeforderten Funktionen möglichst effizient erfüllt und dabei ein optimales Anlagenverhalten ermöglicht, ohne zu wissen, was die Anlage in ihrer Gesamtheit leistet.

3.5.3 Workflowmodellierung

Die Geschäftslogik einer Anlage, also die kunden- oder projektspezifischen Abläufe und Prozesse können als sich auf Transporteinheiten beziehende Arbeitsabläufe oder „Workflows“ aus evtl. mehreren Einzelschritten aufgefasst werden. Die Prozessflexibilität des Internet der Dinge ist dann umgekehrt proportional zum dem Aufwand, der für die Definition, Veränderung und Implementierung eines Workflows notwendig ist. Eine hohe Prozessflexibilität kann also nur dann erreicht werden, wenn Arbeits-

abläufe auf einfache Weise von einem Planer modelliert und von den Entitäten im System verarbeitet und umgesetzt werden können.

Um dies zu erreichen, muss für die Beschreibung von Workflows eine einheitliche, formale Modellierungssystematik entwickelt werden. Diese muss sowohl für einen Systemplaner leicht einzusetzen als auch von den Softwareagenten im Internet der Dinge automatisiert interpretierbar sein. Dabei muss diese Systematik bzw. Workflow-Beschreibungssprache in der Lage sein, folgende Strukturen und Zusammenhänge abzubilden:

- Folgen von Workflowschritten, die sequenziell abgearbeitet werden
- priorisierte Alternativen für einen Workflowschritt
- Verzweigungen im Workflow, wobei die Wahl einer Verzweigung in Abhängigkeit zweier verschiedener Faktoren getroffen wird:
    - Status der TE (z.B. bereits abgearbeitete Workflowschritte und dabei generierte und gesammelte Daten), oder
    - Systemzustand bzw. Verfügbarkeit von Modulen, die die im jeweiligen Workflowschritt notwendige Funktion anbieten

Als Grundlage für die Untersuchung verschiedener Alternativen zur Workflowmodellierung soll ein Szenario aus dem Bereich der Flughafenlogistik dienen, welches die wichtigsten Fälle bei der Bearbeitung von Workflows enthält: Nachdem ein Gepäckstück am Check-In Schalter aufgegeben wurde und das System betritt, muss es in einem ersten Schritt aus Sicherheitsgründen durchleuchtet werden (Schritt „Durchleuchten 1“). Im Fall eines identifizierten Sicherheitsrisikos wird der Koffer in einem zweiten Scanner mit höherer Auflösung und besserer Erkennungsgenauigkeit nochmals untersucht (Schritt „Durchleuchten 2“). Ein zum wiederholten Mal als unsicher eingestufter Koffer wird aus dem System ausgeschleust (Schritt „Ausschleusen“). In der ersten oder zweiten Stufe als sicher erkannte Koffer werden, sobald sich ihr Flug am Gate befindet und für die Gepäckverladung geöffnet wurde, zum Gate gefördert und verladen (Schritt „Verladen“). Ist der Flug beim Passieren des Sicherheitsscanners noch nicht offen, ist eine Zwischenlagerung in einem Frühgepäckspeicher bzw. Early Bagage Store, kurz EBS, notwendig. Dabei wird zwischen EBS-Bahnen, die für einen bestimmten Flug reservierbar (Schritt „EBS für Flug“) und EBS-Bahnen, die für beliebige Flüge innerhalb einer bestimmten Zeitscheibe vorgesehen sind (Schritt

„EBS für Zeitscheibe“), unterschieden. Beim Öffnen des Fluges muss der Koffer die EBS-Bahn verlassen und sich zum richtigen Gate fördern lassen.

Dabei werden in der nachfolgenden Betrachtung die Transportvorgänge nicht explizit dargestellt. Im Allgemeinen ist aber davon auszugehen, dass eine Transporteinheit für die Bearbeitung eines Workflowschrittes erst zu einem neuen Ort bzw. Modul (z.B. Röntgenanlage, Verladetor) transportiert werden muss. Ein Transportvorgang kann zwar ebenfalls als eigener Workflowschritt angesehen werden, muss aber für die Modellierung der Geschäftslogik nicht explizit berücksichtigt werden und ergibt sich dynamisch während des Betriebs.

3.5.3.1 Workflow-Modellierung mittels UML

Ein klassisches Modellierungswerkzeug ist die Unified Modelling Language oder UML. Für die Beschreibung von Abläufen sind im UML-Standard so genannte Aktivitätsdiagramme definiert [Oes-06]. Diese bestehen aus verschiedenen Arten von Knoten (Aktionen, Objekt- und Kontrollknoten), die mit gerichteten Pfeilen verbunden werden und auf diese Weise Abläufe oder Zustandsveränderungen eines Objekts in grafischer Form darstellen.

UML-Aktivitätsdiagramme ermöglichen aber zum einen keine formale Priorisierung verschiedener Pfade bzw. Zustandsübergänge, zum anderen kann die Nichtverfügbarkeit von Zuständen zu einem gewissen Zeitpunkt nicht direkt modelliert werden. Beide Effekte können zwar durch das Einfügen zusätzlicher Kontrollknoten erreicht werden, in denen die Prioritäten bzw. Verfügbarkeiten als Bedingungen abgefragt werden, jedoch macht dieses Vorgehen das Modell schnell unübersichtlich und erschwert weiterhin stark eine direkte Interpretation dieser Beschreibung durch einen automatisierter Prozess bzw. Softwareagenten, da weitere formale Beschreibungen der einzelnen Bedingungen und der dort referenzierten Terme notwendig wären (siehe Abbildung 3-11).

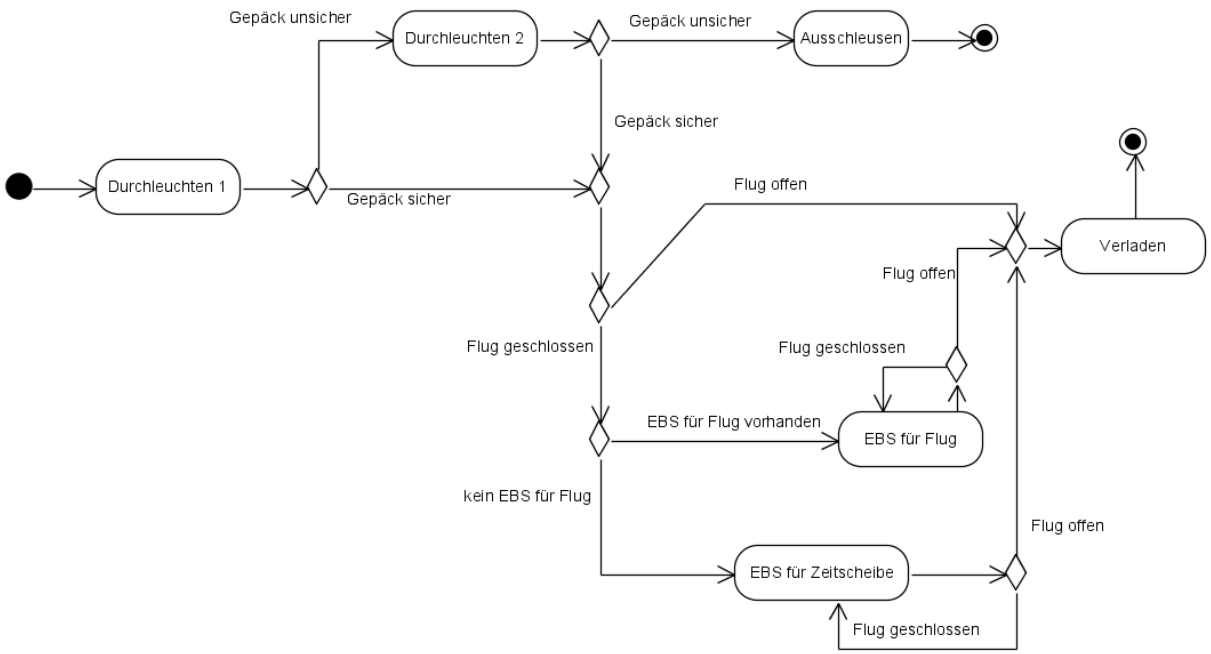


Abbildung 3-11 Beispielhafter Workflow als UML-Aktivitätsdiagramm

3.5.3.2 Workflow-Modellierung mittels Petri-Netzen

Im Gegensatz zu UML sind Petri-Netze ein formales, mathematisches Modell zur Modellierung von Systemtransitionen [Abe-90] [Bau-96]. Petri-Netze können in Form eines gerichteten, bipartiten Graphen oder auch als Matrizen dargestellt werden. Damit sind sie als graphisches Modellierungswerkzeug geeignet, sind aber gleichzeitig einer mathematischen Analyse zugänglich, die beispielsweise für das Auffinden von Deadlockzuständen oder unerreichbaren Stellen nützlich sind.

Der bipartite Graph eines Petri-Netzes besteht aus folgenden Elementen:

- **Stellen** entsprechen den Zuständen eines Systems und werden als Kreise dargestellt. Die in einem Zustand des Gesamtsystems aktiven Stellen werden mit einer oder mehreren Marken gekennzeichnet. Dabei kann jeder Stelle eine Kapazität zugeordnet werden, die angibt, wie viele Marken diese maximal aufnehmen kann. In Bezug auf die Modellierung von Workflows können Stellen als Workflowschritte aufgefasst werden, wobei die (einzige) aktive Stelle dem Workflowschritt entspricht, der gerade bearbeitet wird.
- **Transitionen** repräsentieren einen Zustandsübergang im System und werden als Vierecke dargestellt. Eine Transition kann dann stattfinden bzw. schalten, wenn sich an allen Eingangsstellen genügend Marken befinden, und an allen

Ausgangsstellen ausreichend freie Kapazitäten vorhanden sind, um die in der Transition entstehenden Marken aufzunehmen.

- **Kanten** verbinden eine Stelle mit einer Transition, jedoch nie zwei Transitionen oder zwei Stellen. Sie werden als gerichtete Pfeile dargestellt und haben ein Gewicht, das ihren Kosten entspricht. Dabei entsprechen die Kosten der Anzahl an Marken, die eine Transition beim Schalten aus der voran gegangenen Stelle verbraucht, bzw. der Anzahl an Marken, die durch die Transition in den nachfolgenden Stellen erzeugt werden.
- **Marken** werden als Punkte innerhalb von Stellen dargestellt und kennzeichnen die gerade aktiven Zustände im System. Beim Schalten einer Transition werden Marken aus den Eingangsstellen vernichtet und in den Ausgangsstellen erzeugt, wandern aber niemals von einer Stelle zur nächsten. Alle Marken sind untereinander identisch und nicht unterscheidbar. In Bezug auf den Workflow stellt die Marke den Schritt dar, mit dessen Abarbeitung die Transporteinheit gerade beschäftigt ist.

Bei Petri-Netzen ist vor allem zu beachten, dass einer Stelle zwar mehrere Transitionen folgen können, diese Alternativen dabei aber nicht priorisiert oder an andere Bedingungen geknüpft werden können. Sind an einer Stelle genügend Marken vorhanden, um mehrere Transitionen zu schalten, ist nicht definiert, welche der Transitionen tatsächlich aktiviert wird. Das Verhalten ist somit nichtdeterministisch. Obwohl Petri-Netze eine formale Methode darstellen, sind sie in ihrer einfachsten Form für die Modellierung eines Workflows daher ungeeignet, da weder bedingte Verzweigungen, noch priorisierte Transitionen definiert werden können (siehe Abbildung 3-12).

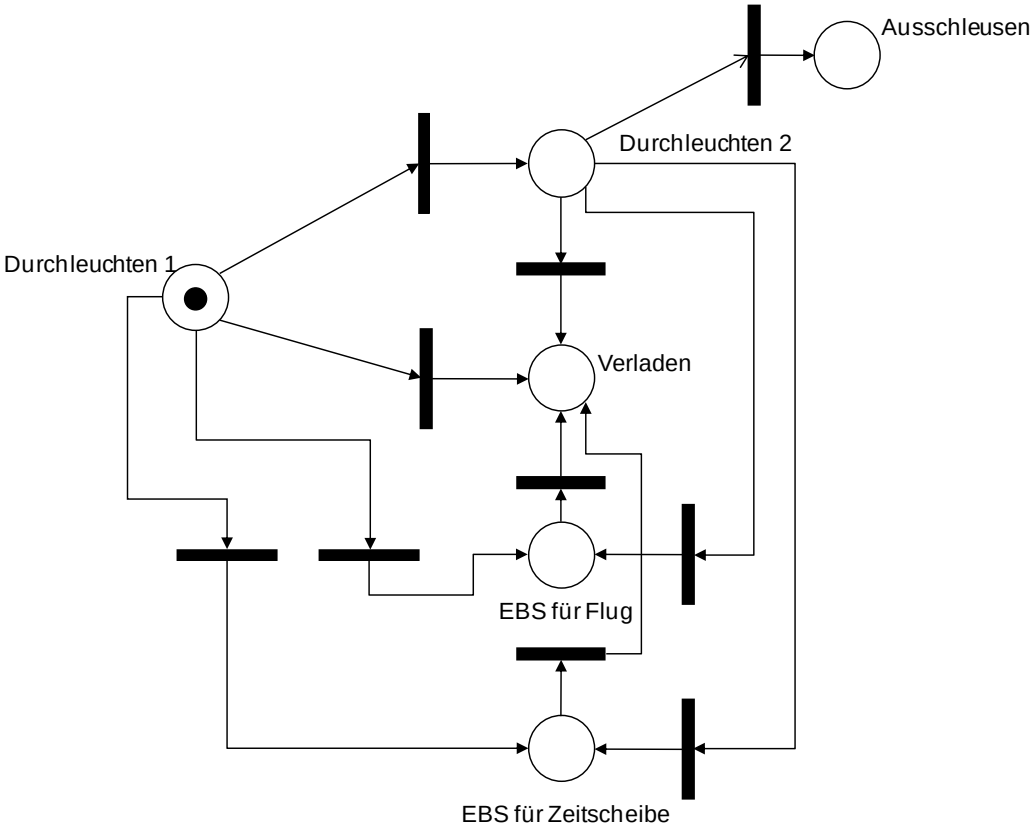


Abbildung 3-12 Beispielhafter Workflow als klassisches Petri-Netz

3.5.3.3 Workflow-Modellierung mittels erweiterter Petri-Netze

Für Petri-Netze sind verschiedene Erweiterungen vorhanden, die es erlauben, Systeme mit speziellen Eigenschaften und Anforderungen zu behandeln. Die wichtigsten Konzepte sind dabei:

- **Zeiterweiterte Petri-Netze:** Dabei verbrauchen die Transitionen während eines Schaltvorgangs Zeit. Je nach Verteilung der Schaltzeiten wird beispielsweise zwischen stochastischen, deterministisch stochastischen oder allgemein stochastischen Petri-Netzen unterschieden.
- **Prioritäten und hemmende Kanten**: Transitionen werden Prioritäten als ganze Zahlen größer gleich 1 zugewiesen, sodass im Falle mehrerer gleichzeitig aktiver Transitionen diejenige mit der höheren Priorität schaltet. Hemmende Kanten hingegen verhindern das Schalten einer Transition, wenn sich an ihrer Ausgangsstelle eine Marke befindet.
- **Farbige Petri-Netze**: Den Marken können verschiedene Eigenschaften oder „Farben“ zugewiesen werden, so dass sie voneinander unterscheidbar sind.

Weiterhin werden Funktionen definiert, die bei der Erzeugung von Marken ihre Farbgebung bestimmen oder an Farben geknüpfte Bedingungen für die Aktivierbarkeit von Transitionen spezifizieren [Jen-92] [Jen-91]. Auf diese Weise sind farbige Petri-Netze eher als Kombination von Petri-Netzen mit Programmiersprachen anzusehen und werden auch „High-level Petri-Netze“ genannt.

Obwohl für die Modellierung von Workflows im Internet der Dinge eine zeitliche Komponente nicht unbedingt notwendig ist, stellen die Priorisierung von Transitionen und die Färbung von Marken die Werkzeuge dar, die eine einfache und formale Workflowbeschreibung auf Grundlage von Petri-Netzen ermöglichen. Der formale, für eine automatisierte Interpretation notwendige, Charakter von Petri-Netzen bleibt auch bei Einsatz solcher Erweiterungen erhalten [Hee-94] [Jen-92] [Mur-89]. So wurden auch verschiedene Konzepte für die Modellierung und das Management von Workflows im Allgemeinen bereits entwickelt [Aal-94] [Aal-96]. Die Anwendung eines erweiterten Petri-Netzes für das gewählte Flughafenszenario wird in Abbildung 3-13 dargestellt.

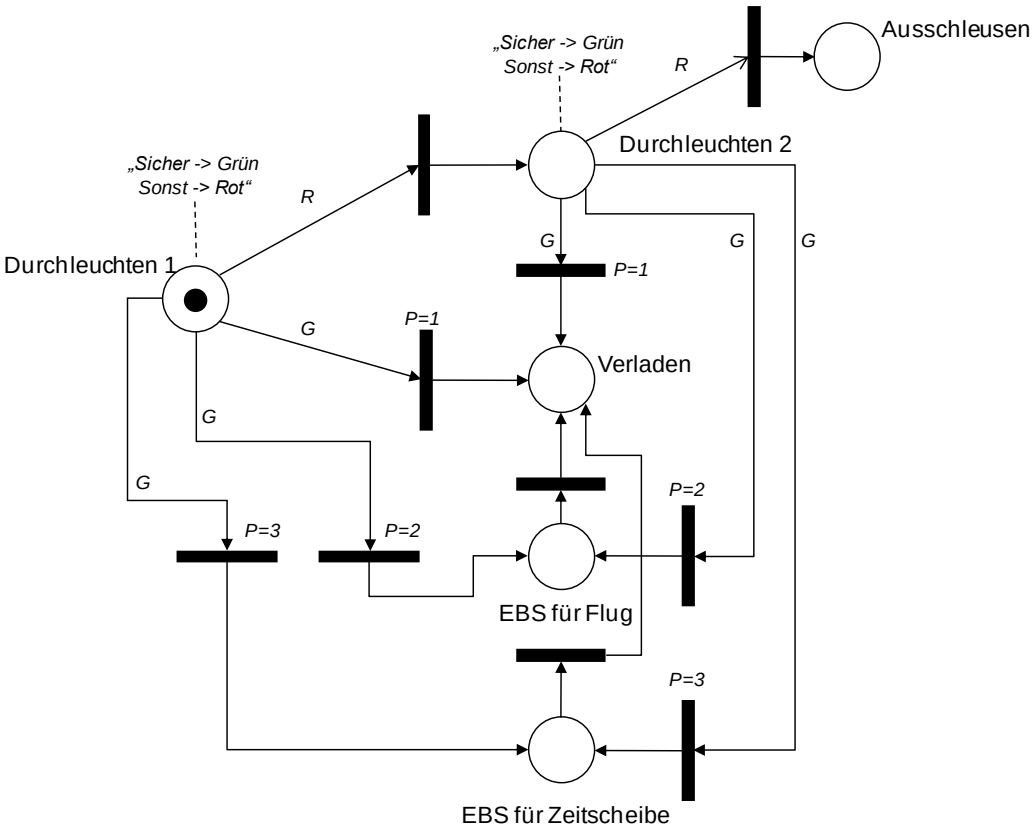


Abbildung 3-13 Beispielhafter Workflow als priorisiertes, farbiges Petri-Netz

Diese Modellierungsform erfüllt alle für Workflows definierten Anforderungen:

- Jedem Workflowschritt wird eine Stelle im Netz zugeordnet, eine Sequenz von Workflowschritten wird gemäß der Petri-Netz Semantik als Folge von Stellen und Transitionen dargestellt.
- Alternativen im Workflow werden durch mehrere Transitionen mit einer gemeinsamen Eingangsstelle dargestellt, wobei eine Priorisierung der Alternativen vorgenommen werden kann. Sind mehrere Transitionen mit gleicher Priorität schaltbereit, ist die Auswahl nicht deterministisch.
- Entscheidungen an Verzweigungen im Workflow werden anhand der Farbe einer Marke getroffen. Die Markenfarbe repräsentiert dabei den Zustand der TE bzw. Informationen über diese, die entweder von vorne herein bekannt sind oder erst an einer bestimmten Stelle im Workflow generiert werden.
- Transitionen können in Petri-Netzen nur dann schalten, wenn die Endstelle eine Kapazität größer Null besitzt. In Bezug auf das Internet der Dinge haben die Stellen nur dann eine Kapazität größer Null, wenn ein Modul vorhanden ist, welches die benötigte Funktion (z.B. „In Flug XY123 verladen“) anbietet. Ist keines vorhanden, muss die Transition mit der schlechteren Priorität gewählt werden.

Da hierbei nur der Workflow einer einzelnen TE betrachtet wird, müssen

- Initialisierungszustände mit keiner oder mehr als einer Marke,
- Transitionen mit mehreren Ausgangsstellen und
- Kanten mit einem Gewicht ungleich 1

als unerlaubt gelten.

3.5.3.4 Verwenden und Abarbeiten von Workflow-Modellen in Softwareagenten

Bisher wurden hauptsächlich die funktionalen Möglichkeiten verschiedener Methoden zur Workflowbeschreibung untersucht und in graphischer Form dargestellt. Diese graphische Methode ist vor allem für die Entwicklung, Modellierung und Fehleranalyse von Workflows einsetzbar.

Soll aber das so entwickelte Modell für die tatsächliche Steuerung des Internet der Dinge Anwendung finden, ist eine nicht-graphische und von Softwareprogrammen automatisiert interpretierbare Darstellungsform notwendig. Zu der in der ISO/IEC 15909-1 Norm [ISO/IEC-15909-1] beschriebenen graphischen Modellierungssprache für High-Level Petri-Netze wird derzeit auch ein auf XML beruhendes Transferformat entwickelt [ISO/IEC-15909-2].

Diese auch PNML (Petri Net Markup Language) genannte Beschreibungssprache (siehe Abbildung 3-14) unterstützt verschiedene Arten von Petri-Netzen. Da PNML auf XML aufbaut, können verschiedene Arten von Netzen mit ihren jeweiligen Erweiterungen und Regeln in Form einer PNTD oder Petri Net Type Definition (analog zur Document Type Definition, oder DTD, für allgemeine XML-Dokumente) beschrieben werden [Web-02] [Bil-03].

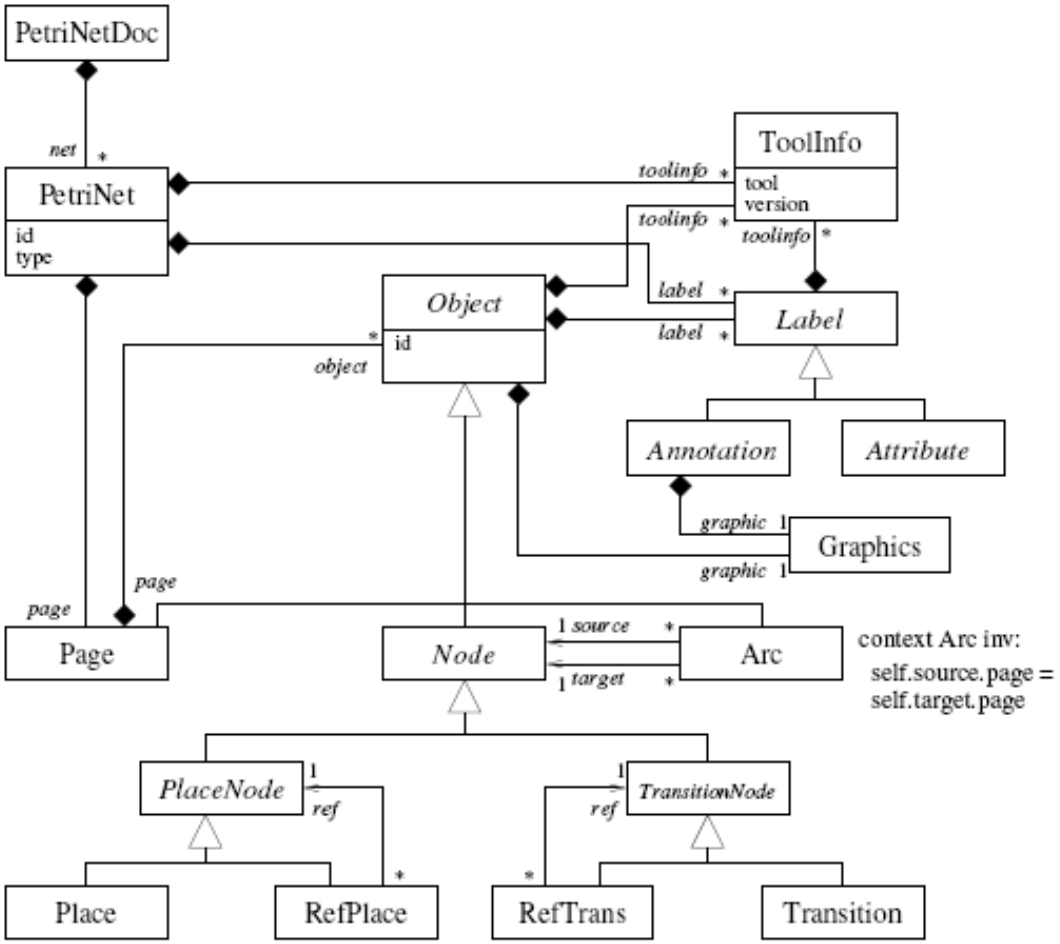


Abbildung 3-14 Das PNML-Core-Modell

Die Humboldt Universität Berlin, die an der Entwicklung von PNML beteiligt ist, bietet zudem einen so genannten Petri Net Kernel zum Download an. Dieser ist als Python- und Java-Version im Internet verfügbar und erlaubt die Erstellung, Veränderung und Simulation beliebiger Petri-Netze [PNK].

PNML bietet somit, zusammen mit einem Tool zum Simulieren von Petri-Netzen, wie z.B. dem Petri Net Kernel, die Möglichkeit, modellierte Workflows als XML-Datei zur Laufzeit in einen Softwareagenten zu laden und schrittweise zu durchlaufen.

4 Kommunikation: Inhalte und Übertragung

4.1 Eine Ontologie für das Internet der Dinge

Soll ein Softwareprogramm Entscheidungen treffen, benötigt es als Entscheidungsgrundlage Informationen über seine Umwelt. Da in den meisten Anwendungen verteilter Architekturen von einer sich dynamisch ändernden (Daten-)Umwelt auszugehen ist, ist die Möglichkeit zur Beschaffung aktueller und zuverlässiger Informationen eine grundlegende Voraussetzung für die korrekte Funktionsweise des Systems. Jede Entität im Internet der Dinge muss also in der Lage sein, durch Kommunikation mit anderen Entitäten, die von ihr benötigten Informationen zu beschaffen.

Bei der Entwicklung von Kommunikationsschnittstellen und der Auswahl einer Kommunikationstechnologie müssen in erster Linie die im Internet der Dinge zu übertragenden Inhalte definiert und formalisiert werden. Die formale Beschreibung einer Konzeptbildung wird in der Informatik als Ontologie bezeichnet [Gr-93]. Anders ausgedrückt beschreibt eine Ontologie eine bestimmte Problemdomäne, legt die erforderlichen Begrifflichkeiten fest und setzt die verschiedenen Elemente in Beziehung zueinander. Für die Beschreibung allgemeiner Ontologien sind mehrere Sprachen im Einsatz, wie z.B. OWL [W3C-04] oder Topic Maps [ISO/IEC-99].

Für den Bereich der Multiagentensysteme existiert auch eine FIPA-Spezifikation für Ontologien, die nicht nur die für dezentral gesteuerte Systeme notwendigen Formalismen enthält, sondern bei Einsatz des JADE-Frameworks mittels Java-Klassen direkt als Grundlage für die Softwareerstellung [Ca-04] verwendet werden kann (siehe Abbildung 4-1). Die Grundelemente einer Agentenontologie sind *Konzepte*, *Prädikate* und *Aktionen*:

- *Konzepte* beschreiben die Bestandteile der Anwendungsdomäne – im Internet der Dinge beispielsweise ein Modul, eine Transporteinheit oder einen Transportauftrag.
- *Prädikate* sagen etwas über den aktuellen Zustand des Systems aus und können entweder wahr oder falsch sein. Prädikate werden zwischen Agenten ausgetauscht, um Informationen über bestimmte Zusammenhänge weiterzu-

geben bzw. zu erfragen. Ein Prädikat gibt beispielsweise an, ob ein Modul eine gewisse Funktionalität anbietet oder ob ein Lastwechsel zum aktuellen Zeitpunkt durchgeführt werden darf.

- *Aktionen* werden von Agenten ausgeführt, um ein bestimmtes Ziel zu erreichen. Im Kontext einer Ontologie werden sie oftmals im Rahmen einer Anfrage eingesetzt, über die ein Agent einen anderen darum bittet, eine bestimmte Aktion durchzuführen. Ein typisches Beispiel für eine Aktion ist der Transport eines Behälters.

Dabei können Aktionen und Prädikate direkt als Inhalt eines Kommunikationsvorgangs bzw. eines Sprechaktes zwischen Agenten auftreten, während Konzepte ihrerseits nur in Verbindung mit Aktionen oder Prädikaten eingesetzt werden.

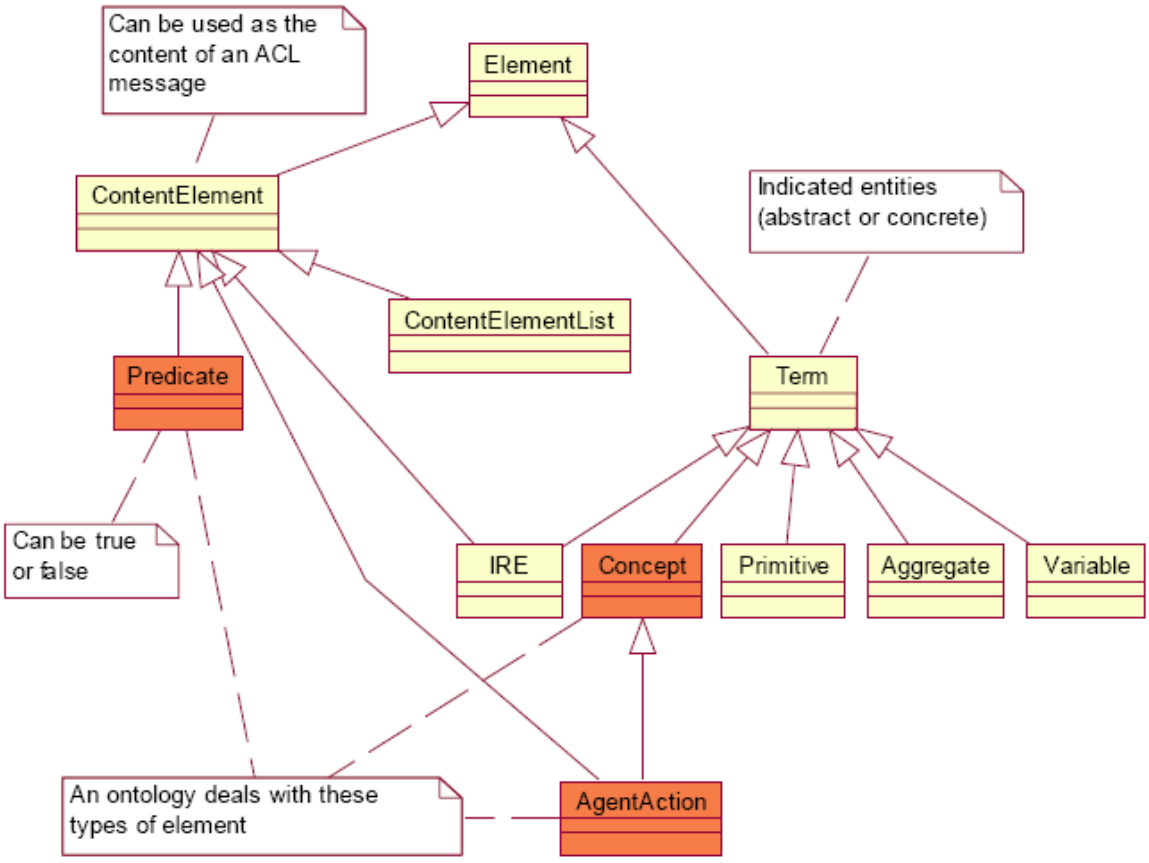


Abbildung 4-1 Modell zur Wissensbeschreibung nach JADE [Cai-04]

4.1.1 Basisontologie

Als Ausgangspunkt für die Entwicklung der Basisontologie für das Internet der Dinge gilt zum einen das vorgestellte Modularisierungskonzept und zum anderen eine all-

gemeine, für möglichst alle Materialflussanlagen gültige, Beschreibung der vom System zu erbringenden Funktion. Diese wird als Workflow aufgefasst, bei dem eine Transporteinheit eine Menge von Schritten durchlaufen muss, bis ihr Status als abgearbeitet gilt (siehe auch Abschnitt 3.5.3). Bei der Ausführung eines Einzelschrittes im Workflow wirken andere Entitäten, im Normalfall Module, auf eine Transporteinheit ein. Dies kann zu Veränderungen der TE selbst (z.B. bei der Kommissionierung oder Verpackung) oder ihrer Position und Orientierung (z.B. beim Transport oder während der Handhabung) führen oder dient der Überwachung und Gewinnung von Daten über die Transporteinheit – beispielweise beim Identifizieren einer TE durch ein Auto-ID-Lesegerät (RFID, Barcode, etc.) oder dem Durchleuchten eines Koffers in einem Flughafengepäckfördersystem.

Um diesen Workflow abzuarbeiten, muss ein Behälter weiterhin in der Lage sein, einen Pfad durch das Materialflusssystem zu bestimmen und Module mit dem Transport zum nächsten Teilziel zu beauftragen.

4.1.1.1 Domänenontologie

Die aus Prädikaten, Aktionen und Konzepten bestehende Domänenontologie kann in Form eines UML-Klassendiagramms dargestellt werden (siehe Abbildung 4-2). Dieses gibt die statischen Beziehungen zwischen den Konzepten, Prädikaten und Aktionen wieder, ohne aber Kommunikationsprotokolle bzw. Absender und Empfänger für die verschiedenen Sprechakte zu definieren.

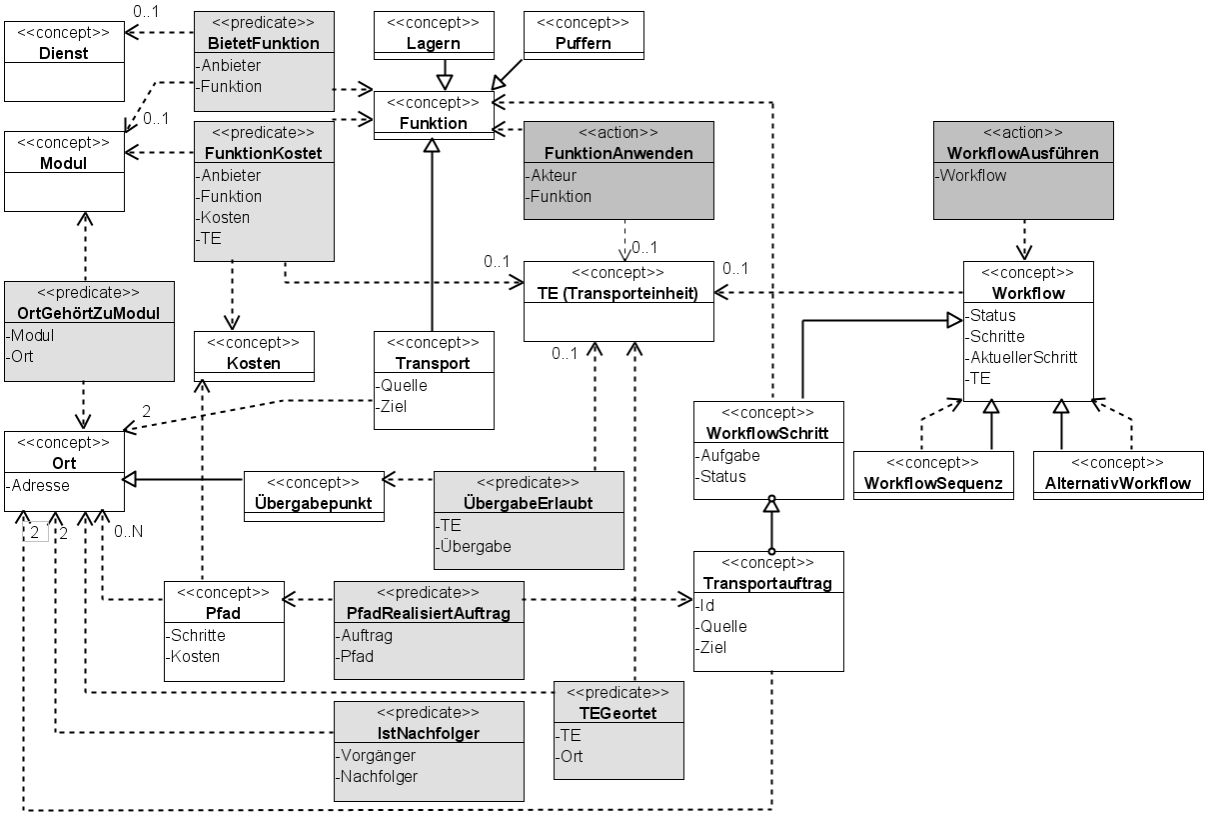


Abbildung 4-2 Basisontologie für das Internet der Dinge [Lib-10]

Diese Ontologie [Lib-10] beschreibt vier grundlegende Aspekte einer Materialflusssteuerung nach dem Internet der Dinge Modell und lässt sich somit in folgende Teile gliedern:

- **Die Kernontologie** beschreibt durch die Konzepte *Dienst*, *TE* und *Modul* die Grundbausteine des Internet der Dinge, so wie sie in Abschnitt 3.2 definiert wurden.
- **Die Funktionenontologie** modelliert die von den Modulen und Diensten angebotenen Funktionen bzw. Dienstleistungen sowie alle Mechanismen, die zum Suchen, Bewerten und Anfordern dieser Funktionen notwendig sind. Das Konzept *Funktion* beschreibt eine logistische oder auch informationstechnische Dienstleistung in System. Konkrete Funktionen, beispielsweise das Lagern oder Puffern von Transporteinheiten oder auch die Visualisierung bestimmter Daten, können von diesem Konzept abgeleitet werden. Die Prädikate *BietetFunktion* und *FunktionKostet* schaffen die Verknüpfung zwischen einer Funktion und dem sie anbietenden Dienst bzw. Modul und den für die tatsächliche Funktionserbringung anfallenden *Kosten* und bilden somit die Grundlage

für die Suche nach Funktionsanbietern sowie für die Auswahl des günstigsten Anbieters. Die Ausführung einer Funktion kann beispielsweise von einer *TE* durch die Aktion *FunktionAnwenden* angestoßen bzw. angefordert werden.

- **Die Workflowontologie** bildet komplexe Arbeitsabläufe ab. Ein *Workflow* enthält einen oder mehrere *WorkflowSchritte*, in denen die *TE* die Ausführung jeweils einer *Funktion* veranlassen muss. Diese Einzelschritte können in einer festen oder beliebigen Reihenfolge, also als *WorkflowSequenz*, abgearbeitet werden oder sich auch im Sinne einer Verzweigung oder eines *Alternativ-Workflows* gegenseitig ausschließen – z.B. kann ein Koffer am Flughafen nach einer Sicherheitsprüfung (Workflowschritt 1) entweder in das Flugzeug verladen (Workflowschritt 2a) oder als unsicher aus dem System ausgeschleust werden (Workflowschritt 2b). Neben diesen vier Konzepten enthält diese Teilontologie die Aktion *WorkflowAusführen*, mit dem eine bestimmte *TE* zum Abarbeiten eines *Workflows* beauftragt wird.
- **Die Transportontologie** detailliert den *Transport* als eine spezielle und essenzielle *Funktion* eines Materialflusssystems. Dies benötigt zum einen die Definition eines *Transportauftrages* als speziellen *Workflowschritt*, in dem eine *TE* auf einem bestimmten *Pfad* von einer Quelle zu einer Senke, also zwischen zwei *Orten*, gefördert wird. Zum anderen werden Prädikate festgelegt, die für die Routenfindung notwendig sind: so gibt *IstNachfolger* an, ob zwei *Orte*, die über das Prädikat *OrtGehörtZuModul* einem *Modul* zugeordnet werden, in einer Vorgänger-Nachfolger-Beziehung stehen bzw. miteinander verbunden sind. Anders ausgedrückt kodiert *IstNachfolger* die Topologie einer Anlage. Die Routenfindung wird über das Prädikat *PfadRealisiertAuftrag* ermöglicht, das angibt, ob ein bestimmter *Pfad* durch das System geeignet ist, um einen *Transportauftrag* für eine bestimmte *TE* zu erfüllen. Das Prädikat *ÜbergabeErlaubt* ermöglicht es, während des Fördervorgangs bei jedem Lastwechsel am *Übergabepunkt* zweier *Module* eine einfache Lastwechselkoordination durchzuführen. Hat ein Modul eine Transporteinheit übernommen bzw. diese detektiert, kann es den TE-Agenten über das Prädikat *TEGeortet* über seinen neuen Aufenthaltsort informieren, womit die TE in die Lage versetzt wird, den Transportvorgang bzw. das Erreichen ihres Zieles zu überwachen.

4.1.1.2 Kommunikationsontologie

Zusätzlich zur Domänenontologie muss auch eine so genannte Kommunikationsontologie definiert werden, deren Fokus auf den tatsächlichen Sprechakten, also auf dem Informationsaustausch zwischen den Entitäten, liegt (siehe UML-Klassendiagramm in Abbildung 4-3). Diese enthält die im System vorhandenen Agenten bzw. Leistungserbringer und zeigt auf, welche Prädikate und Aktionen über welche Kommunikationsprotokolle ausgetauscht werden. Dabei wird in diesem Fall der Einfachheit halber auf folgende, von der FIPA spezifizierte Protokolle [FIPA-SC00037J] verwiesen:

- INFORM: Der Sender informiert den Empfänger darüber, dass ein Prädikat wahr ist.
- QUERY: Der Sender befragt den Empfänger über einen bestimmten Zusammenhang.
- REQUEST: Der Sender fordert den Empfänger auf, eine bestimmte Aktion auszuführen.
- CFP (Call for Proposal): Eine Aufforderung an andere Agenten, Vorschläge zum Ausführen einer Aktion abzuliefern.

Die Kommunikationsontologie für das Internet der Dinge sieht demnach folgenden Ablauf für die Steuerung eines Materialflusssystems vor (siehe Abbildung 4-3):

1. In der Initialisierungsphase oder auch während des Betriebs informieren die *Module* einen *Routingdienst* über die Anlagentopologie, die durch die Prädikate *IstNachfolger* und *OrtGehörtZuModul* dargestellt wird. Der Routingdienst wurde hier als eigener Agent modelliert, kann aber auch als interner Funktionsbaustein eines Moduls und/oder einer Transporteinheit ausgeführt werden. Die Ontologie gibt diesbezüglich keine Einschränkungen vor und ist verschiedenen Implementierungen gegenüber offen.
2. Wird eine *TE* von einem *Modul* mittels einer Auto-ID-Technologie identifziert, z.B. am Wareneingang einer Anlage oder auch während des Transports, wird die *TE* über ihren aktuellen Aufenthaltsort informiert – dies geschieht über das *INFORM* Protokoll mit dem Prädikat *TEGeortet* als Inhalt.

3. Ein *Auftragsmanager*, also ein Dienst, der die Transportbedarfe und sonstige Abläufe einer Anlage steuert – beispielsweise das LVS oder eine Schnittstelle zu diesem – beauftragt eine *TE* mit der Ausführung eines Workflows.
4. Die *TE* holt (für jeden Workflowschritt) über das *CFP*-Protokoll und das Prädikat *FunktionKostet* Angebote von allen *Modulen* im System ein, um so den günstigsten Dienstleister für die im aktuellen Workflowschritt vorgesehene Funktion zu finden.
5. Die *TE* kann zur Auswahl eines Zieles auch eine Wegplanung durch das System durchführen, was in der Kommunikationsontologie durch das Versenden eines *QUERY* bezüglich *PfadRealisiertAuftrag* an einen *Routingdienst* möglich ist.
6. Hat sich die *TE* für ein Ziel entschieden, kann sie über das *REQUEST*-Protokoll und die *FahreZu*-Aktion den Transport zu dem Ziel veranlassen.
7. *Module* führen während des Transports weitere Wegplanungen durch, sind also, genau wie die *TE*, in der Lage, einen *Routingdienst* nach einem Pfad zu befragen.
8. Während des Transports wird u.U. ein Lastwechsel zwischen *Modulen* notwendig – dieser wird über das *QUERY* Protokoll in Bezug auf das Prädikat *ÜbergabeErlaubt* koordiniert.
9. Weiterhin kann es vorkommen, dass ein *Modul* während des Transports andere *Module* mit dem Ausführen von Aktionen beauftragen muss, beispielsweise wenn eine Elektrohängebahnkatze eine Weiche schaltet. Dies wird für den allgemeinen Fall mit dem *REQUEST*-Protokoll und der Aktion *FunktionAnwenden* modelliert.
10. Hat die *TE* ihr Ziel erreicht (worüber sie durch *TEGeortet* in Kenntnis gesetzt wird), beauftragt sie das aktuelle *Modul* mit der Ausführung der in ihrem gegenwärtigen Workflowschritt vorgesehenen Funktion. Nach Erhalt einer Meldung über die erfolgreiche Ausführung der Funktion kann die Transporteinheit, sofern in ihrem Workflow weitere Schritte vorhanden sind, wieder zu Schritt 4 springen.

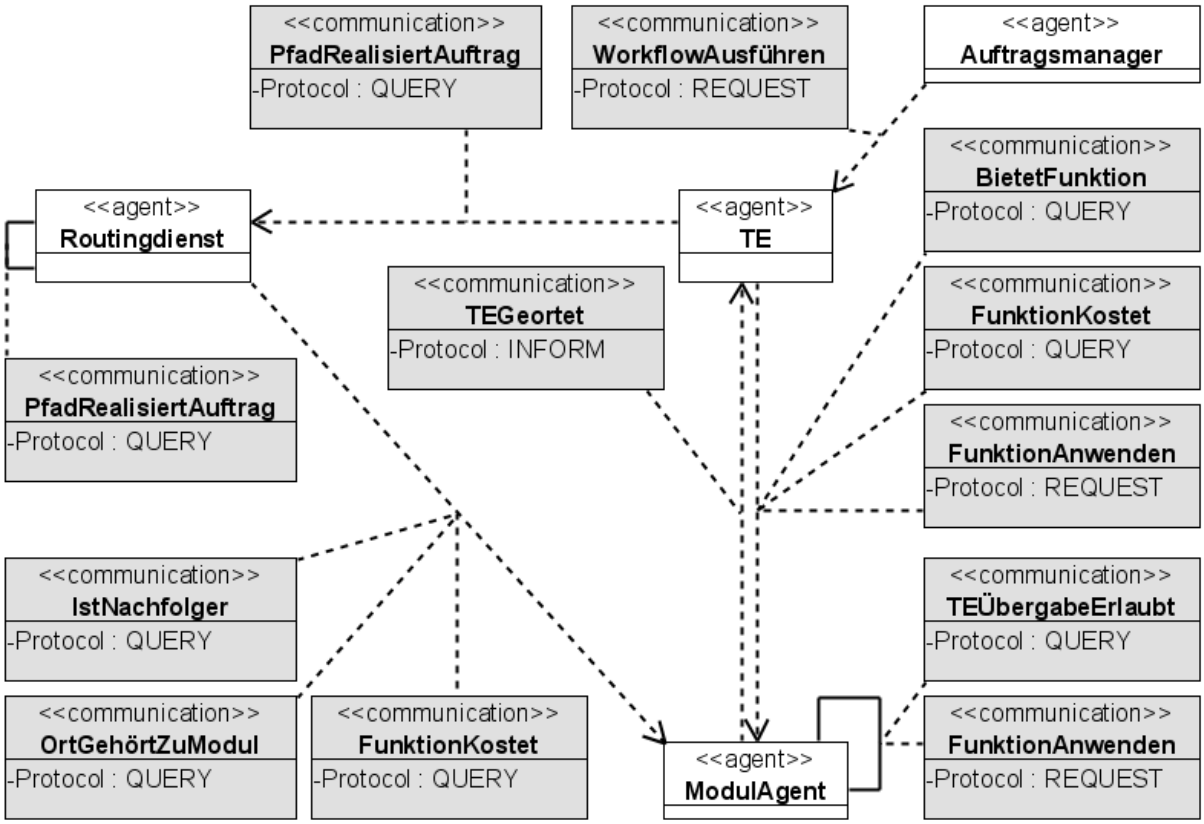


Abbildung 4-3 Kommunikationsontologie für das Internet der Dinge [Lib-10]

4.1.1.3 Nutzen der Ontologie

Die vorgestellte Ontologie hat den Vorteil, generisch aufgebaut zu sein und verschiedenste Materialflussprozesse oder auch Implementierungen eines verteilten Steuerungssystems abbilden zu können. Zum einen kann die Basisontologie durch komplett neue Bausteine (z.B. für die Beschreibung komplexer Lastwechselprozesse; siehe nächster Abschnitt) oder durch Ableitung spezialisierter Elemente erweitert werden. Zum anderen räumt die Kommunikationsontologie eine große Freiheit bei der Gestaltung von Protokollen bzw. der Zuordnung von Kompetenzen zu den Modulen ein. So ist die Entscheidung, wer für das Routing innerhalb eines dezentral gesteuerten Materialflusssystems zuständig ist, eine der wichtigsten Fragen bei der Implementierung des Internet der Dinge, wobei die Basisontologie dabei allen Möglichkeiten (Routing durch die TE, Routing durch die Module, Routing durch einen dedizierten Routing-Dienst bzw. Mischformen dieser drei Möglichkeiten) gegenüber offen ist.

4.1.2 Ergänzungen zur Basisontologie

Die Basisontologie bietet eine hochflexible und universell anwendbare Grundlage für die Modellierung von Kommunikationsprozessen und die Definition von Funktionalitäten für die Systemteilnehmer des Internet der Dinge. Um den höchst heterogenen Anforderungen der innerbetrieblichen Logistik gerecht zu werden, muss die Basisontologie an spezielle Bedürfnisse angepasst werden können. Dies kann auf zwei Arten geschehen:

1. Durch die Ableitung spezialisierter Elemente von bereits Vorhandenen – beispielweise „Durchleuchten“ oder „Konturenkontrolle“ als Unterklassen des Konzeptes „Funktion“ oder „Gabelstapler“ und „FTF“ als Unterklassen des Konzeptes „Modul“.
2. Durch die Definition komplett neuer Konzepte, Aktionen und Prädikate.

Ein Beispiel für eine solche spezielle Ontologie ist die Beschreibung komplexer Lastwechselprozesse, wie sie beispielsweise in einer automatisierten Roboterzelle für die (De-)Palettierung vorkommen (siehe UML-Klassendiagramm in Abbildung 4-4; Elemente aus der Basisontologie werden dabei weiß, neue Elemente grau dargestellt). Die Basisontologie enthält zwar das Prädikat „ÜbergabeErlaubt“, das angibt, ob eine Transporteinheit von einem Übergabepunkt auf einen anderen transferiert werden darf. Spielen aber zusätzliche Parameter wie Palettiermuster, Greifpunkte oder eine geometrische Beschreibung der Transporteinheit eine Rolle bei der Lastübergabe, muss eine spezielle Lastwechselontologie (siehe Abbildung 4-4) definiert werden, die diesen Bedürfnissen gerecht wird.

Diese enthält neue Konzepte für den *Lastwechsel* an sich sowie für die Beschreibung unterschiedlicher *Greifprinzipien* – z.B. zur Unterscheidung kraft- und formschlüssiger Lastaufnahmemittel – und für *3DKoordinaten,* die zusammen mit dem Prädikat *TEBefindetSichBei* die genaue Position einer *TE* im Raum angeben. Die Prädikate *ModulUnterstützt* und *KannGegriffenWerden* geben an, ob ein Modul ein bestimmtes Greifprinzip bietet bzw. ob eine TE mittels eines Greifprinzips gegriffen werden kann. Module, die mehrere oder sehr komplexe Lastaufnahmemittel besitzen, können über die Aktion *GreiferUmkonfigurieren* dazu aufgefordert werden, sich auf einen bestimmten Lastwechselfall einzustellen. Sind alle Voraussetzungen erfüllt, kann der Lastwechsel über die Aktion *LWAusführen* angestoßen werden.

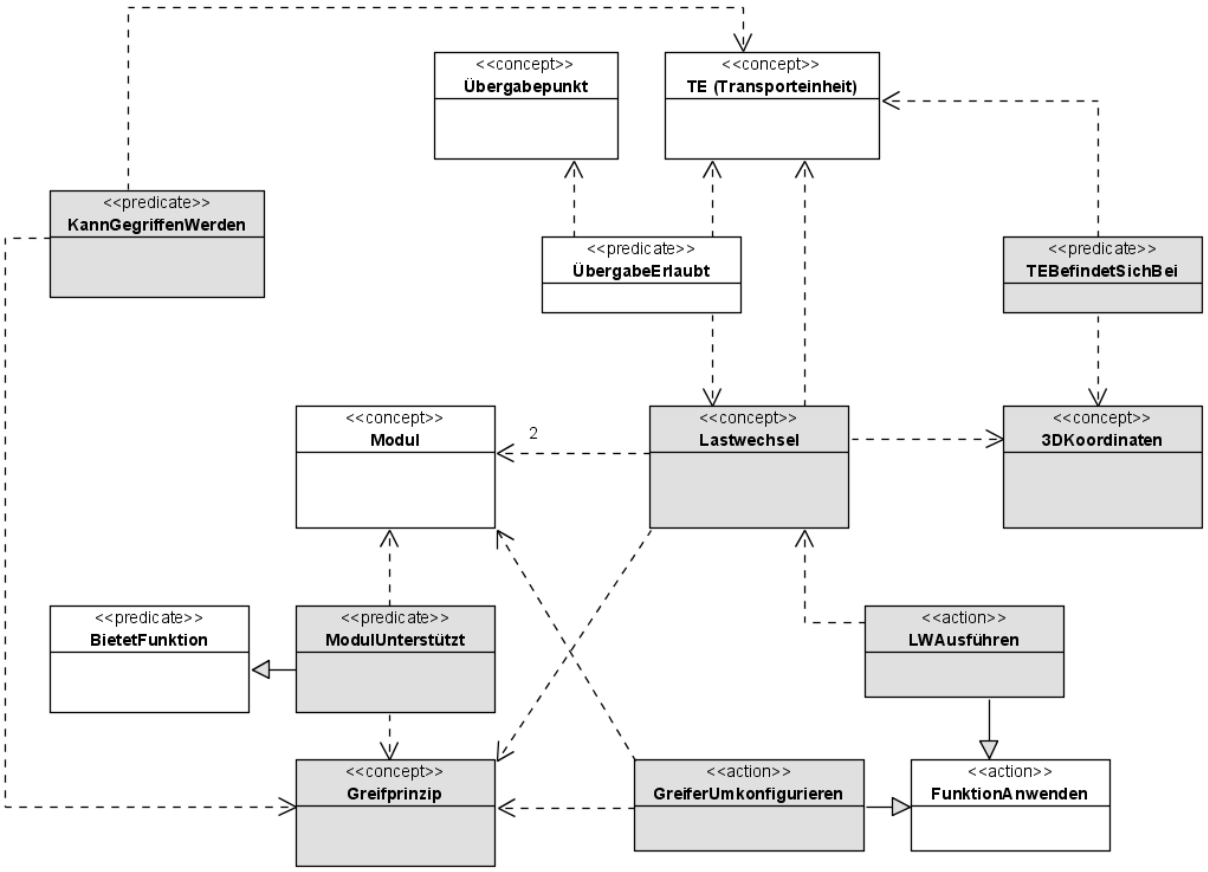


Abbildung 4-4 Lastwechselontologie

Eine weitere, in materialflusstechnischen Anlagen oft sinnvolle Ergänzung der Ontologie ergibt sich durch die Notwendigkeit, Statusinformationen der Entitäten bzw. Informationen über den aktuellen Zustand des Systems (z.B. Streckenbelegungen, Wegreservierungen oder Störungen) zu übertragen. Die dafür notwendigen Prädikate oder auch Aktionen werden jedoch stark von den tatsächlichen Dateninhalten der Ontologie-Klassen abhängen und wurden daher in der Basisontologie nicht berücksichtigt. Trotzdem sind bei der Gestaltung der Softwarearchitektur und vor allem bei der Wahl der Kommunikationsinfrastruktur auch diese Nachrichten zu berücksichtigen.

4.2 Kommunikationsformen

Der Prozess der dezentralen Informationsbeschaffung bzw. ganz allgemein der Kommunikation zwischen zahlreichen Einheiten ist als sehr komplex einzustufen, denn prinzipiell ist nicht bekannt, wer eine bestimmte Information besitzt. Weiterhin ist es möglich, dass im Netzwerk verschiedene Versionen derselben Information

vorhanden sind. Diese Probleme, die in ähnlicher Form jedem Internetnutzer bekannt sind, werden in einem hochdynamischen Netz wie dem Internet der Dinge zusätzlich durch die grundlegende Funktionsweise der eingesetzten Kommunikationstechnologien, nämlich TCP/IP und Ethernet, verschärft. Denn Ethernet arbeitet, zumindest in der am weitesten verbreiteten Standardversion, nicht-deterministisch, d.h. die Zeit zwischen dem Versenden und Empfangen einer Nachricht ist grundsätzlich unbekannt [Küv-07] und kann von Nachricht zu Nachricht ggf. stark variieren. Ebenso ist die Route, auf der ein Datenpaket zum Ziel gelangt, im Voraus nicht bekannt und kann für jedes Datenpaket eine andere sein. Dies führt dazu, dass in der richtigen Reihenfolge versendete Nachrichten unter Umständen in einer anderen, falschen Reihenfolge beim Empfänger ankommen. So kann eine Entität z.B. die Aufforderung, eine Aktion zu beenden noch vor der Aufforderung, eine Aktion zu beginnen, erhalten, obwohl diese beiden Nachrichten richtig versendet wurden (siehe Abbildung 4-5). Dazu ist noch zu sagen, dass diese Probleme durch die Funktionsweise der Kommunikationstechnologie an sich entstehen und nicht durch die Definition ungeeigneter Kommunikationsschnittstellen oder Protokolle.

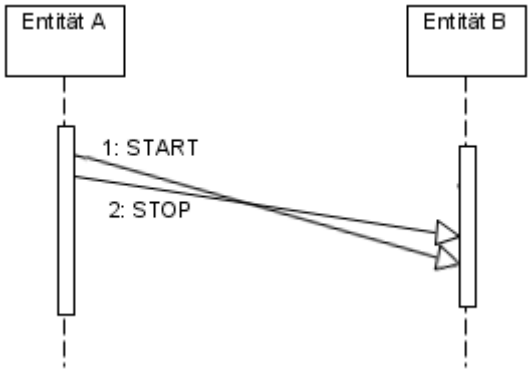


Abbildung 4-5 Mögliche Reihenfolgevertauschung von Nachrichten bei der Kommunikation über Ethernet

Diese und ähnliche Probleme der Kommunikation sind allgemein bekannt, sodass Verfahren für das Auflösen oder Verhindern solcher Situationen entwickelt wurden und verfügbar sind. So können z.B. Vektor-Uhren [Mat-88] über die Aktualität von Daten Auskunft geben oder in jeder Nachricht mitgeführte Zeitstempel dem Empfänger erlauben, diese wieder in die richtige Reihenfolge zu bringen. Trotzdem ist festzuhalten, dass die Kommunikation im Internet der Dinge – durch die möglichen Probleme und die dafür notwendigen Korrekturverfahren – komplexer gestaltet und schwieriger nachvollziehbar sein kann als in herkömmlichen, zentralen Systemen.

So gesehen verringert das Internet der Dinge erst einmal *nicht* die Komplexität einer Materialflusssteuerung, sondern verlagert sie. Die sehr komplexe und umfangreiche Logik bzw. interne Struktur eines Materialflussrechners wird zwar in überschaubare Teilprobleme gegliedert und auf relativ einfache, leicht handhabbare Agenten verteilt. Dafür aber erzeugt die verteilte Beschaffung und Verteilung von Informationen durch diese zahlreichen Agenten die erwähnte Kommunikationskomplexität. Man kann also sagen, dass das Internet der Dinge eine „Komplexität der Logik“ durch eine „Komplexität der Kommunikation“ ersetzt, dadurch aber noch keinen pauschalen Vorteil bietet. Dies wirft zwei Fragen auf:

1. Kann eine generelle Aussage darüber getroffen werden, welche Art der Komplexität „wünschenswerter“, d.h. leichter zu beherrschen ist?
2. Kann die Kommunikationskomplexität im Internet der Dinge verringert werden und wenn ja, wie?

Die erste Frage lässt sich relativ leicht beantworten. Der wichtigste Unterschied zwischen der Logik eines Softwareprogramms und der Kommunikation zwischen Programmen besteht darin, dass die Logik eine ganz bestimmte Funktionalität umsetzt, d.h. anwendungsspezifisch ist, während Kommunikationsmechanismen viel allgemeingültiger und unabhängig von der Anwendung (oder den dabei notwendigen konkreten Schnittstellen) sind. Eine allgemeingültige Aussage darüber, ob für beliebige Szenarien die „Komplexität der Logik“, also eine zentrale Steuerungsarchitektur, der „Komplexität der Kommunikation“ vorzuziehen ist, lässt sich wegen der großen Vielfalt an Systemen und Anforderungen nur schwer treffen. Das Internet der Dinge zielt aber weniger auf die Optimierung bzw. Vereinfachung einer einzelnen Anlage ab, sondern will eine in großem Maße wieder verwendbare Grundlage für die aufwandsarme Erstellung beliebiger Anlagen bieten. Die Informationsbeschaffung bzw. Kommunikation zwischen Agenten ist ein grundlegender Baustein des Internet der Dinge und kann somit in die Basisklasse der „allgemeinsten Entität“ verlagert werden (siehe Abschnitt 3.4.1). Wird an dieser Stelle eine zuverlässige, robuste und performante Kommunikation implementiert, können alle weiteren Module, Dienste und Transporteinheitsagenten von Haus aus auf diese Mechanismen zurückgreifen.

Somit lässt sich die erste Frage auf folgende Weise beantworten: Der Aufwand zur Beherrschung der Logik- bzw. Kommunikationskomplexität an sich kann nur schwer

verglichen werden. Kommunikationsmechanismen müssen aber nur ein einziges Mal implementiert werden, um für beliebige Anwendungen wiederverwendet werden zu können, während die Logik in jedem einzelnen System eine Neue ist.

Die Frage nach den Möglichkeiten zur Vereinfachung der Kommunikation erfordert eine genauere Analyse der verschiedenen Möglichkeiten zum Datenaustausch, sowie ihrer Vor- und Nachteile. Die Kommunikation zwischen mehreren Einheiten kann nach zwei grundlegenden Mustern gestaltet werden (siehe Abbildung 4-6):

1. das Versenden von Nachrichten von einem Sender an einen (oder mehrere) Empfänger,
2. die Nutzung eines gemeinsamen Speichers.

Bei der ersten Form des Informationsaustausches, der vornehmlich als *Peer-to-Peer*-Kommunikation bekannt ist, tauschen die Systemteilnehmer Informationen über einzelne Nachrichten aus, die von einem Sender an den oder die Empfänger verschickt werden, was voraussetzt, dass der Sender die Empfänger kennt. Daten werden also nur von den einzelnen Agenten gespeichert, wobei Informationen oftmals redundant, d.h. von mehreren Agenten, vorgehalten werden.

Als Gegenpol dazu kann der Informationsaustausch über einen gemeinsamen Speicher angesehen werden. Dabei werden keine Nachrichten direkt zwischen den Netzwerkteilnehmern ausgetauscht, was bedeutet, dass sich diese auch nicht kennen müssen. Eine Datenaustauschplattform bzw. ein *Blackboard* enthält dabei das „globale Wissen“ des Systems, das als Grundlage für die Berechnungen aller Agenten dient. Blackboards erlauben es den Akteuren eines Systems, Daten zu lesen, zu schreiben oder auch zu löschen. Sie implementieren außerdem eine Zugriffssteuerung, die z.B. gewährleistet, dass ein Datum nicht gleichzeitig von verschiedenen Agenten verändert wird, was zu einem inkonsistenten Zustand des Datenpools führen würde.

Darüber hinaus sind auch Mischformen realisierbar, die sowohl einen gemeinsamen (zentralen oder wiederum verteilten) Speicher als auch direkte Kommunikation verwenden.

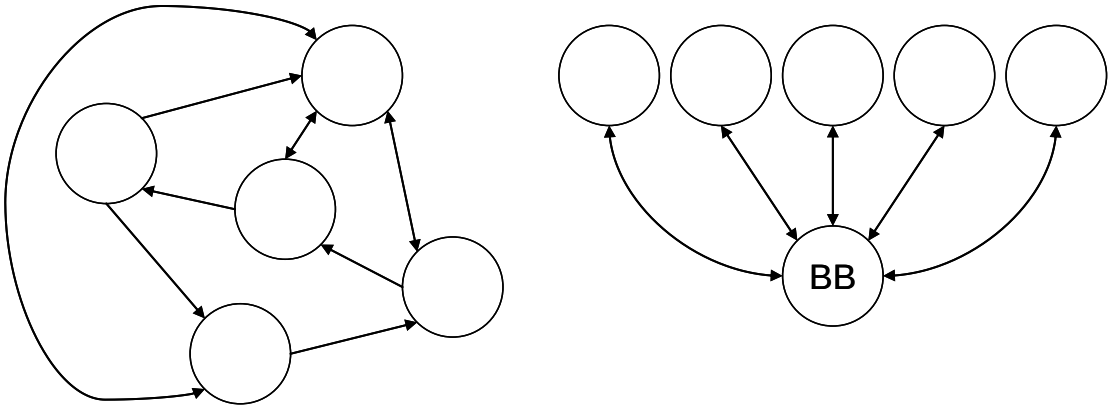


Abbildung 4-6 Peer-to-Peer- und Blackboard-Kommunikation

Sowohl in Peer-to-Peer- als auch in Blackboard-basierten Systemen kann die Informationsbeschaffung auf zwei verschiedene Arten ablaufen. Die erste Möglichkeit kann als klassisches „Fragen und Antworten“ bezeichnet werden: Möchte eine Entität eine Information erhalten, fragt sie eine andere Entität an und bekommt die gewünschte Information oder auch eine Fehlermeldung als Antwort. Das Versenden einer Anfrage und einer Antwort sind dann jedes Mal von Neuem notwendig, wenn eine Entität etwas in Erfahrung bringen möchte. Dieses „Pull-Prinzip“ wird mittels eines UML-Sequenzdiagramms in Abbildung 4-7 dargestellt.

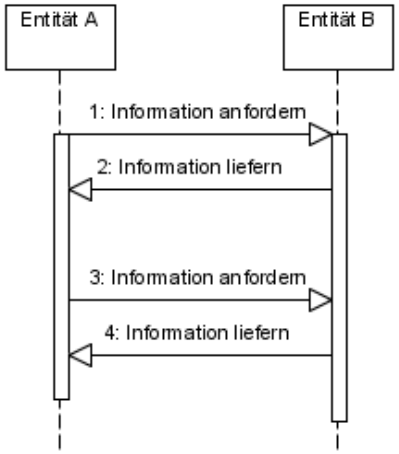


Abbildung 4-7 Informationsbeschaffung über direktes Anfragen ("Pull-Prinzip")

Die zweite Möglichkeit funktioniert über Abonnements. Dabei kann eine Entität ein Abonnement für bestimmte Informationen bei einer zweiten Entität anlegen und wird ab da automatisch von dieser benachrichtigt, sobald sich die betreffenden Daten ändern. Eine explizite Anfrage ist dabei nicht mehr nötig. Dieses „Push-Prinzip“ funktioniert wie vom Sequenzdiagramm in Abbildung 4-8 beschrieben.

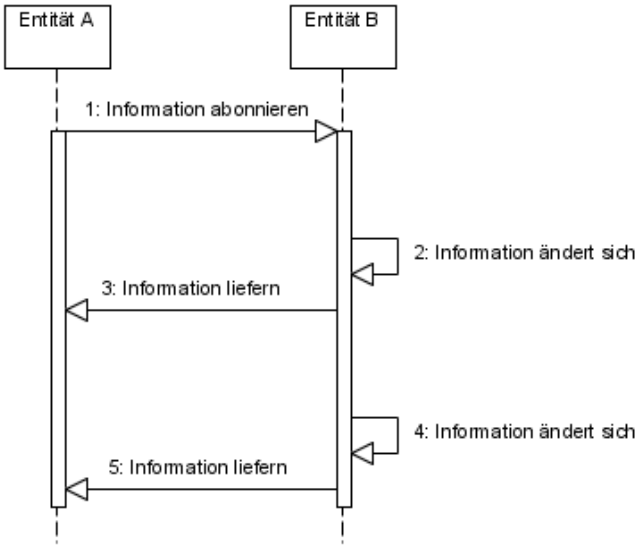


Abbildung 4-8 Informationsbeschaffung über Abonnements ("Push-Prinzip")

Das Push-Prinzip hat den Vorteil, dass es – sobald das Abonnement angelegt wurde – nur halb so viele Nachrichten benötigt, wie das Pull-Prinzip. Dabei ist aber zu beachten, dass das Pull-Prinzip nur dann geeignet ist, wenn:

- informationstechnische Abhängigkeiten zwischen Entitäten relativ statisch sind bzw. sich über längere Zeiträume nicht ändern. Ansonsten könnte das häufige Anlegen, Ändern und Löschen der Abonnements (sowie der Rechenaufwand für deren Verwaltung innerhalb der Entitäten) das Wegfallen der Anfrage-Nachrichten aufwiegen.
- die abonnierten Informationen sich nicht bedeutend häufiger ändern, als sie vom Abonnenten benötigt werden. Ändert sich z.B. bei einem Querverschiebewagen dessen Millimeterposition kontinuierlich, während dieses Datum nur im Sekundentakt von einer Visualisierung benötigt wird, würde das Abonnement-basierte Versenden dieser Information Unmengen nutzloser Nachrichten erzeugen. Um diesem Effekt entgegenzuwirken ist es ratsam, abonnierte Informationen nicht bei jeder einzelnen Änderung an die Abonnenten zu verschicken, sondern in einem vorgegebenen Takt, z.B. jede Sekunde. Zusätzlich ist zu beachten, dass eine allgemeingültige Aussage darüber, ob in einem System die Häufigkeit der lokalen Datenänderung kleiner oder größer ist, als die Häufigkeit, in der diese Daten von anderen Entitäten benötigt werden, nur sehr schwierig gemacht werden kann, zumal sich dieses Verhältnis auch während des Betriebs ständig ändern kann. Um das obige Beispiel des Verschie-

bewagens und der Visualisierung nochmals aufzugreifen, verändern sich während der Fahrt die (Positions-)Informationen viel häufiger, als sie benötigt werden – was für eine Informationsbeschaffung nach dem „Pull-Prinzip“ sprechen würde. Steht der Verschiebewagen aber still, benötigt die Visualisierung trotzdem im Sekundentakt aktuelle Positionsdaten und würde bei Einsatz des „Pull-Prinzips“ zyklisch Anfragen versenden, nur um immer wieder dieselbe Antwort zu erhalten.

Zur Auswahl bzw. Gestaltung eines geeigneten Kommunikationssystems für das Internet der Dinge ist daher eine Gegenüberstellung dieser zwei Datenaustauschprinzipien vorzunehmen. Zum einen müssen funktionale Aspekte, wie z.B. die Gewährleistung der Datenkonsistenz und der dafür zu betreibende Aufwand qualitativ bewertet werden. Zum anderen ist auch eine möglichst geringe Kommunikationslast, also eine geringe Anzahl auszutauschender Nachrichten, wünschenswert.

4.2.1 Analyse funktionaler Aspekte

Folgende Faktoren spielen in einer dezentralen Materialflusssteuerung eine wichtige Rolle und müssen für eine Bewertung der Blackboard- und Peer-to-Peer-basierten Kommunikation betrachtet werden.

- Auffinden von Gesprächspartnern: Benötigt eine Entität bestimmte Informationen, muss erst diejenige(n) Entität(en) im System gefunden werden, die diese Information besitzen. Gleiches gilt für den Fall, dass eine Einheit von einer anderen Entität die Ausführung einer Funktion anfordern muss (z.B. eine Palette, die von einem Stapler den Transport zu ihrem Zielpunkt anfordert).
- Datenkonsistenz: Informationen sind in verteilten Architekturen prinzipiell mehrfach vorhanden. Dabei muss sichergestellt werden, dass alle Kopien ein und derselben Information identisch sind, um ein konsistentes und sinnvolles Systemverhalten zu gewährleisten.
- Datenredundanz und damit Robustheit gegenüber Ausfällen: Fällt der einzige Anbieter einer bestimmten Information aus, sind alle anderen Entitäten, die auf die betreffenden und nun nicht mehr beschaffbaren Daten angewiesen sind, ebenfalls betroffen. Das redundante Vorhalten von Informationen erhöht somit die Robustheit des Systems.

- Zentrale Verfügbarkeit umfangreicher Daten: Obwohl die Steuerungsarchitektur im Internet der Dinge komplett verteilt ist, gibt es auch Funktionsbausteine, die alle oder einen Großteil aller Informationen im System benötigen. Dazu zählen beispielsweise Visualisierungsumgebungen, global optimierende Dienste oder Backupserver.
- Aufzeichnung von Historien: Die Vorgänge im System müssen auch im Internet der Dinge nachvollzogen werden können, um beispielsweise die Diagnose von Fehlerfällen zu ermöglichen. Neben der Erstellung lokaler Logfiles, die die agenteninternen Abläufe und Entscheidungen protokollieren, ist dabei vor allem auch die Analyse der Kommunikation („Wer hat wann was an wen geschickt?“) von Bedeutung.

In der folgenden Tabelle werden die Blackboard- und Peer-to-Peer-Kommunikation in Bezug auf diese Aspekte näher betrachtet.

	Blackboard	Peer-to-Peer
Auffinden von Gesprächspartnern	Die Agenten kommunizieren nicht direkt, sondern nur mit dem Blackboard. Jede Einheit ist für das Veröffentlichen und Beschaffen relevanter Informationen selbst verantwortlich. Eine Suche nach Gesprächspartnern ist also nicht erforderlich.	Der gewünschte Gesprächspartner muss explizit gefunden und angesprochen werden. Dies kann über einen Verzeichnisdienst oder durch das Fluten des gesamten Netzwerks mit einer entsprechenden Anfrage erreicht werden.
Datenkonsistenz	Die Zugriffssteuerung gewährleistet, dass sich das Blackboard in einem konsistenten Zustand befindet.	Durch die hochgradige Verteilung und der hohen Anzahl von Kopien der Datensätze kann es leicht zu Inkonsistenzen kommen, die entdeckt und ggf. wieder aufgelöst werden müssen.

Daten-redundanz	Das Blackboard enthält die einzige vertrauenswürdige Kopie aller Informationen des Systems. Damit ist das Blackboard ein besonders kritischer „Single Point of Failure“.	Im Rahmen des Speicherplatzes der einzelnen Einheiten können Informationen beliebig oft repliziert werden. Ein Peer-to-Peer-Netz ist gegenüber Ausfällen daher als sehr robust anzusehen.
Zentrale Verfügbarkeit umfangreicher Daten	Ja. Das Blackboard ist ein zentraler Datenspeicher für das Gesamtsystem.	Daten können durch Fluten des gesamten Netzwerkes gesammelt werden.
Aufzeichnung von Historien	Das Blackboard kann alle vorgenommenen Lese-, Schreib-, Löschoperationen protokollieren oder einen externen Agenten mit der Protokollierung beauftragen.	Jeder Agent kann ein lokales Protokoll führen. Zur Nachverfolgung größerer Zusammenhänge müssen die lokalen Historien zusammengeführt und synchronisiert werden.

Tabelle 4-1 Gegenüberstellung funktionaler Aspekte der Blackboard- und Peer-to-Peer-Kommunikation

4.2.2 Analyse der Kommunikationslast

Neben den funktionalen Aspekten der Blackboard und Peer-to-Peer-Architekturen muss auch das Nachrichtenaufkommen in den beiden Systemen analysiert werden, wobei für jede der beiden Möglichkeiten sowohl die Informationsbeschaffung nach dem Push- als auch nach dem Pull-Prinzip betrachtet werden muss. Dabei seien der Einfachheit halber die oben angeführten Probleme der Datenredundanz und Datenkonsistenz als gelöst angesehen.

Dies soll analytisch geschehen, da eine simulative Untersuchung dieser Fragestellung anhand eines Szenarios zwar konkrete Zahlen liefern könnte, diese aber keine Allgemeingültigkeit bzw. Übertragbarkeit auf andere Fälle aufweisen würden. Da die Kommunikationsarchitektur einer der grundlegenden Aspekte des Internet der Dinge

ist, muss sie aber für verschiedenste Szenarien ohne Änderung wieder verwendet werden und dabei immer mit möglichst hoher Effizienz funktionieren.

Für die analytische Betrachtung der Kommunikationslast seien folgende Werte definiert:

- *n* – die Gesamtanzahl der Entitäten im System
- *k* – die durchschnittliche Anzahl an Entitäten, von denen eine Entität Informationen benötigt, um Entscheidungen zu treffen oder Aktionen durchzuführen.

In den folgenden Diagrammen sei zudem die Entität, die für eigene Zwecke Informationen beschaffen muss mit „V“ (für Informations-Verbraucher) gekennzeichnet, während die Entitäten, die diese Informationen liefern, mit „L“ (für Informations-Lieferant) gekennzeichnet werden. Ein Blackboard wird mit „BB“ gekennzeichnet.

4.2.2.1 Peer-to-Peer, Pull-Prinzip

Die erste und einfachste Möglichkeit für eine Entität, notwendige Informationen zu beschaffen, besteht darin, alle Entitäten, die die benötigten Informationen besitzen, anzufragen und ihre Antworten auszuwerten (siehe Abbildung 4-9). Dies benötigt pro Verbraucher also *2k* Nachrichten (für jeden der *k* Informationslieferanten eine Frage- und eine Antwortnachricht) und für das Gesamtsystem *2kn* Nachrichten – denn prinzipiell muss jede der *n* Entitäten Informationen nach diesem Muster beschaffen.

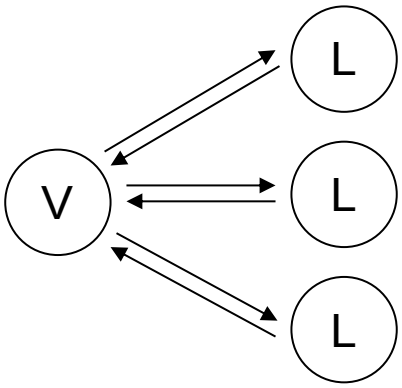


Abbildung 4-9 Informationsbeschaffung in einer Peer-to-Peer-Architektur nach dem Pull-Prinzip

Dabei ist jedoch zu beachten, dass der Verbraucher im Allgemeinen nicht a priori weiß, wer die benötigten Informationen besitzt. Das Auffinden der Gesprächspartner bzw. Informationslieferanten lässt sich wiederum auf zwei Arten lösen: entweder es ist im System ein zentrales Verzeichnis „D“ (für Directory) vorhanden, in dem der

Informationsverbraucher zuerst nachsehen kann, wen er anzufragen hat, was 2 zusätzliche Nachrichten benötigt, insgesamt also *k+2* bzw. *n(k+2)*. Oder die Anfrage wird an alle Entitäten im System versendet, wobei nur diejenigen antworten, die die Information tatsächlich besitzen, was *n+k* Nachrichten pro Verbraucher bzw. n^2+nk Nachrichten für das Gesamtsystem entspricht (siehe Abbildung 4-10; angefragte Gesprächspartner, die die gesuchten Informationen nicht besitzen, sind grau eingezeichnet).

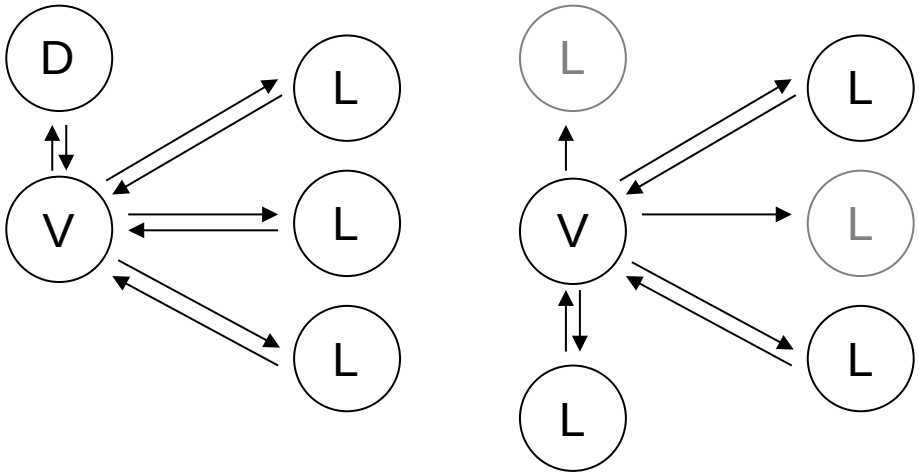


Abbildung 4-10 Auffinden der Gesprächsteilnehmer in einer Peer-to-Peer Architektur

4.2.2.2 Peer-to-Peer, Push-Prinzip

Wie bereits erwähnt kann der Einsatz einer Abonnement-basierten Kommunikation unter gewissen Umständen die Anzahl der Nachrichten halbieren. Nach dem Anlegen der Abonnements, was wiederum eine (im Idealfall einmalige) Suche der Informationslieferanten durch ein Verzeichnis oder durch Fluten des Systems erfordert, sind in einem Peer-to-Peer-System nur noch *k* Nachrichten pro Verbraucher, also *nk* Nachrichten im System notwendig (siehe Abbildung 4-11).

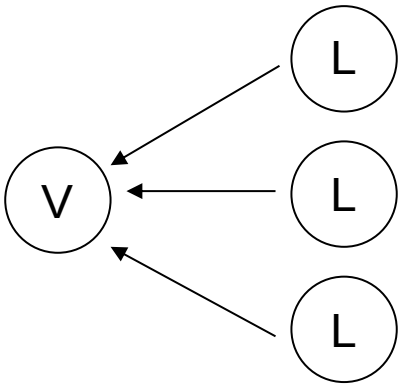


Abbildung 4-11 Informationsbeschaffung in einer Peer-to-Peer-Architektur nach dem Push-Prinzip

4.2.2.3 Blackboard, Pull-Prinzip

Das Blackboard kann in seiner Funktion als gemeinsamer Datenspeicher des Systems als eine Entität aufgefasst werden, die alle Informationen aller anderen Entitäten abonniert hat – also nach dem „Push-Prinzip“ alle Informationen im System sammelt. Für das Füllen bzw. Aktualisieren des Blackboards fallen also, unabhängig von der Anzahl der Verbraucher, immer *n* Nachrichten an (siehe Abbildung 4-12). Für das Abfragen der Informationen fallen pro Verbraucher 2 Nachrichten an, für das Gesamtsystem also $n+2n=3n$ Nachrichten.

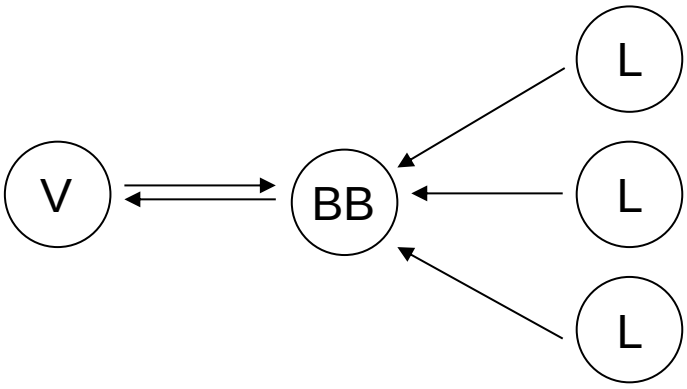


Abbildung 4-12 Informationsbeschaffung in einer Blackboard-Architektur nach dem Pull-Prinzip

4.2.2.4 Blackboard, Push-Prinzip

Wird das Blackboard selbst wiederum mit einem Benachrichtigungsmechanismus ausgestattet, so dass es Veränderungen an die Abonnenten, bzw. Informationsverbraucher, automatisch weiterleitet, sinkt die Gesamtzahl der Nachrichten auf *2n,*

nämlich *n* für das Befüllen des Blackboards und je 1 Benachrichtigung an die *n* Abonnenten (siehe Abbildung 4-13).

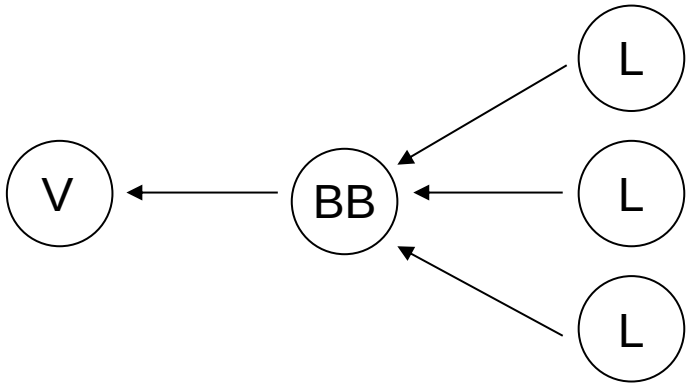


Abbildung 4-13 Informationsbeschaffung in einer Blackboard-Architektur nach dem Push-Prinzip

4.2.2.5 Gegenüberstellung

Die gerade vorgestellten Ergebnisse lassen sich folgender maßen zusammenfassen:

	Peer-to-Peer	Blackboard
Pull-Prinzip	*2nk*	*3n*
Push-Prinzip	*nk*	*2n*

Tabelle 4-2 Gegenüberstellung der Kommunikationslast in Blackboard- und Peer-to-Peer-Systemen

4.2.2.6 Sonderfall globale Datensammlung

Die bisherigen Betrachtungen und Ergebnisse zur Kommunikationslast hängen, vor allem für Peer-to-Peer-Netze in großem Maße vom Faktor *k* – der durchschnittlichen Anzahl der Informationslieferanten pro Informationsverbraucher – ab. Im Allgemeinen kann davon ausgegangen werden, dass zur Steuerung des Materialflusses ein relativ kleines *k* genügt. Im einfachsten denkbaren Fall muss jedes Modul nur mit dem Nachfolgermodul kommunizieren, um zu prüfen, ob dieses eine Transporteinheit zu ihrem Ziel befördern kann und freie Kapazitäten besitzt.

Darüber hinaus können in einem dezentralen Materialflusssystem aber auch Entitäten vorkommen, die von *allen* Entitäten im System Daten benötigen. Dies sind z.B. Visualisierungsumgebungen, Backup-Server, globale Systemoptimierer oder Entitäten mit einer ähnlichen, globale Informationen benötigenden Funktionalität. Da davon

auszugehen ist, dass es in einem System nur eine relativ geringe Anzahl solcher Entitäten geben wird, empfiehlt es sich, diese gesondert zu betrachten. Mit *g* Entitäten diesen Typs in einem System mit insgesamt *n+g* Entitäten gestaltet sich die Gesamtkommunikationslast daher folgendermaßen:

	Peer-to-Peer	Blackboard
Pull-Prinzip	*2nk + 2gn*	*3n + 2g*
Push-Prinzip	*nk + gn*	*2n + g*

Tabelle 4-3 Gegenüberstellung der Kommunikationslast in Blackboard- und Peer-to-Peer-Systemen bei Einsatz globaler Datensammlungsdienste

Besonders auffällig ist dabei, dass in Peer-to-Peer-Systemen das Vorhandensein solcher Entitäten zu einem sehr starken Anstieg der Kommunikationslast führt, während es in Blackboard-Systemen die Kommunikationslast nur linear – wie jede „gewöhnliche“ Entität – ansteigen lässt.

4.2.2.7 Schlussfolgerung

Die bisherigen Betrachtungen zeigen, dass ein Blackboard-System bereits für *k = 2* dieselbe bzw. eine geringere Kommunikationslast aufweist als Peer-to-Peer-Systeme. Für größere *k* benötigt ein Blackboard-System klar weniger Nachrichten als ein Peer-to-Peer-System. Befinden sich im System Entitäten, die sehr viele globale Informationen benötigen, wie z.B. Visualisierungsumgebungen, haben Blackboard-Systeme einen noch deutlicheren Vorteil. Zusammengefasst lässt sich also sagen:

- je stärker ein System vernetzt ist, desto eher bietet sich der Einsatz eines Blackboards an,
- je weniger vernetzt ein System ist, desto eher bietet sich eine Peer-to-Peer-Kommunikation an.

4.3 Kopplung an externe Informationssysteme

Neben der Kommunikation zwischen den Agenten des Internets der Dinge muss ein Materialflusssystem aber auch auf externe Datenquellen zugreifen können. Diese sind vor allem übergeordnete Systeme, wie das Lagerverwaltungssystem, die Produktionssteuerung oder das Warenwirtschaftssystem, die alle in großem Umfang auf

Datenbanken basieren. Die dort gespeicherten Informationen müssen also auch den Akteuren eines Multiagentensystems zugreifbar sein. Ein Hindernis dabei ist aber die große Heterogenität von Datenbanksystemen und der verschiedenen SQL-Varianten, die für die Formulierung von Abfragen eingesetzt werden (z.B. MySQL, Microsoft SQL, Oracle).

Soll das Internet der Dinge nicht nur innerhalb der eigenen Systemgrenzen flexibel sein und eine einfache Integration neuer Teilnehmer ermöglichen, sondern auch selbst mit geringem Aufwand an verschiedene Logistik- und Informationssysteme gekoppelt werden können, ist auch diese Schnittstellenproblematik zu lösen. Eine Umprogrammierung bzw. Anpassung aller Agenten an die jeweilige Systemlandschaft wäre im Allgemeinen sehr aufwändig. Eine viel einfachere Methode ist der Einsatz eines „Übersetzers“, der Anfragen von anderen Agenten aus einer für das Internet der Dinge typischen Ontologie in die jeweilige SQL-Syntax überträgt und auch die Antworten vom Datenbanksystem als mit der Ontologie des Internet der Dinge konforme Nachrichten versendet.

Diese Übersetzerfunktionalität lässt sich sehr gut mit einem Blackboard kombinieren. Durch die Kopplung eines Blackboards an ein Datenbanksystem können zudem die Möglichkeiten zur Datenverwaltung solcher Systeme für das Internet der Dinge übernommen werden.

4.4 Hybrider Ansatz

Blackboard- und Peer-to-Peer-basierte Ansätze haben sehr verschiedene Vor- und Nachteile, die in unterschiedlichen Bereichen der Materialflusssteuerung relevant sind. So spricht beispielsweise die Verfügbarkeit globaler Daten, die für die Visualisierung eines Materialflusssystems oder dessen Optimierung unerlässlich sind, für den Einsatz eines Blackboards, während die höhere Robustheit komplett verteilter Netze für ein Peer-to-Peer-System sprechen.

Um die Vorteile beider Konzepte optimal zu nutzen, wird ein hybrider Ansatz empfohlen, in dem sowohl eine gemeinsame Datenplattform als auch direkte Kommunikation zum Einsatz kommen. Zudem sollte die Blackboard-Architektur an sich verteilt werden, um die Kommunikationslast dieser Agenten zu senken und vor allem die Robustheit des Systems zu erhöhen. Dabei wird, um den Synchronisationsaufwand

zwischen den Blackboards gering zu halten, keine hochgradige Verteilung, sondern der parallele Betrieb weniger, synchronisierter Blackboards angestrebt.

Die Entscheidung über den Einsatz des einen oder anderen Systems zur Kommunikation wird anhand einer Analyse der Prädikate und Aktionen aus der Basisontologie (siehe Abschnitt 4.1) und der für jedes Element geltenden Anforderungen getroffen.

	Interessenten	Zentrale Verfügbarkeit notwendig	Externes DB-System	Geeignete Kommunikation
Basisontologie				
DienstBietet	Absender der Anfrage	Nein	Nein	Peer-to-Peer
FahreZu	1 Modul, 1 TE	Nein	Nein	Peer-to-Peer
FunktionAnwenden	2 beteiligte Module bzw. 1 Modul und 1 TE	Nein	Nein	Peer-to-Peer
FunktionKostet	Absender der Anfrage	Nein	Nein	Peer-to-Peer
IstNachfolger	Gesamtsystem	Ja, für Visualisierung	Denkbar	Blackboard
PfadRealisiert-Auftrag	1 TE bzw. Modul	Nein	Nein	Peer-to-Peer
OrtGehörtZuModul	Gesamtsystem	Ja, für Visualisierung	Denkbar	Blackboard
TEGeortet	TE bzw. TE-Manager-Agent	Ja, für Visualisierung	Denkbar	Peer-to-Peer an TE, Blackboard

ÜbergabeErlaubt	2 beteiligte Module	Nein	Nein	Peer-to-Peer
WorfklowAusführen	1 TE	Nein	Nein	Peer-to-Peer
Typische Ergänzungen der Basisontologie				
Status- & Fehlermeldungen	Gesamt-system, Anla-genbetreiber	Ja, für Visua-lisierung und evtl. zentrale Optimierung	Denkbar	Blackboard
Dynamischer Sys-temzustand (Weg-reservierungen, Streckenbelegung, etc.)	Gesamt-system	Ja, für Visua-lisierung und Optimierung	Nein	Blackboard

Tabelle 4-4 Auswahl des geeigneten Kommunikationssystems für verschiedene Elemente der Ontologie

Eine Analogie zum Kommunikations- und Kooperationsverhalten von Menschen vermag es dabei, die bisher rein technischen Betrachtungen von Kommunikations-systemen auch intuitiv zugänglicher zu machen. Trifft sich eine Gruppe von Mitarbeitern, um zusammen ein Problem zu lösen oder neue Ideen zu generieren, so ist es zweckmäßig, bestimmte Informationen auf einer Tafel oder auf Präsentationsfolien bereit zu stellen und dort auch Zwischenergebnisse festzuhalten oder zu protokollieren. Während der Erarbeitung der Ergebnisse werden sich aber auch Diskussionen zwischen einzelnen Mitarbeitern oder in kleinen Arbeitsgruppen herausbilden. Versucht man, sich eine solche „Brainstorming-Runde“ vorzustellen, die entweder ausschließlich auf direkten Gesprächen (ohne Tafel) beruht oder aber in der direkte Gespräche verboten sind und jeglicher Informationsaustausch nur zentral erfolgen darf, wird schnell klar, dass dabei ein kaum vertretbarer Koordinationsaufwand entsteht, der die Produktivität des Teams stark verringern wird. Der richtige Ansatz besteht also in der Verwendung aller zur Verfügung stehenden Kommunikationsmittel in Abhängigkeit der jeweiligen Situation bzw. Anforderungen.

4.5 Blackboard-Architektur

Die über eine Peer-to-Peer-Kommunikation ausgetauschte Information ist inhaltlich direkt an die Fähigkeiten der beteiligten Agenten gebunden. Ein Blackboard hingegen dient als Speicher für eine Vielzahl verschiedener Datenformate, die es selbst aber inhaltlich nicht verstehen oder verarbeiten muss. Um eine hohe Flexibilität und Erweiterbarkeit der Kommunikationsinfrastruktur zu gewährleisten, muss diese Datenplattform möglichst generisch gestaltet sein. Vor allem sollen dabei keine oder nur möglichst geringe Einschränkungen bzgl. der unterstützten Daten und Datenformate bestehen.

Die Funktionalität eines Blackboards ist an sich sehr einfach: es erlaubt anderen Agenten, Daten zu schreiben, zu lesen und zu löschen. Um diesen Datenaustausch effizient und nachvollziehbar zu gestalten, muss es ein Blackboard anderen Entitäten erlauben, detaillierte Suchanfragen zu formulieren und deren Aktivitäten in Form einer Historie aufzeichnen oder an einen externen Logging-Dienst melden. Darüber hinaus ist auch die Übersetzung von Nachrichten zwischen dem Internet der Dinge und externen Datenbanken eine typische Aufgabe für an weitere Systeme angeschlossen Blackboards.

Diese grundlegenden Methoden müssen aber durch zusätzliche Funktionen ergänzt werden, um die Effektivität des Gesamtsystems zu gewährleisten. Zu diesen Erweiterungen gehören vor allem:

1. eine Zugriffssteuerung, die z.B. gleichzeitige Schreibzugriffe verhindert und die Konsistenz der vom Blackboard gespeicherten Daten gewährleistet,
2. ein Benachrichtigungsmechanismus, der interessierten Agenten automatisch mitteilt, dass bestimmte Dateninhalte verändert wurden und
3. der Betrieb paralleler Blackboards zum Zweck der Performance- und v.a. Robustheitssteigerung (siehe Abbildung 4-14).

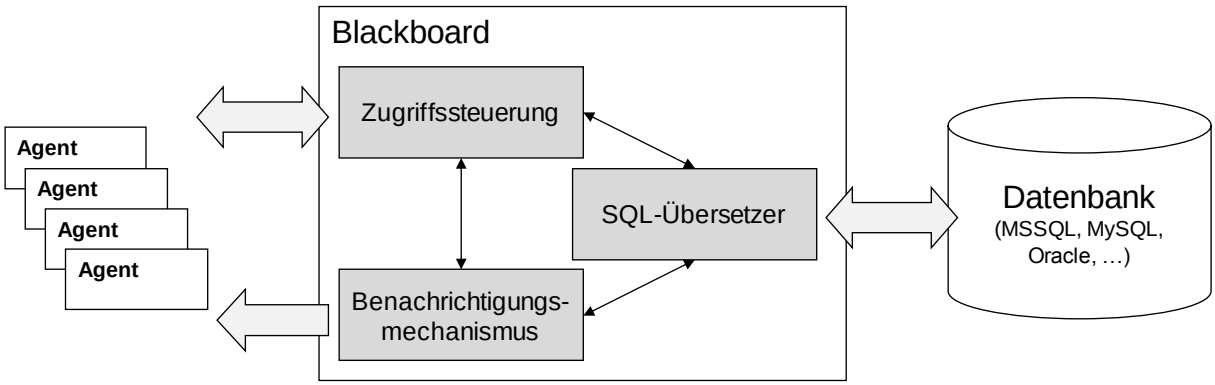


Abbildung 4-14 Komponenten eines Blackboards

4.5.1 Zugriffssteuerung

Jede Entität in einem verteilten System trifft auf Basis ihres Wissens über die Umgebung Entscheidungen, die wiederum den Zustand des Gesamtsystems und damit die Entscheidungsgrundlage für andere Teilnehmer verändert. Daher ist die Verfügbarkeit eines konsistenten, für alle Entitäten verbindlichen datentechnischen Abbilds des Materialflusses besonders wichtig. Wegen der hohen Nebenläufigkeit des Systems kann es aber auch beim Einsatz eines Blackboards zu inkonsistenten Zuständen kommen, beispielsweise wenn Anforderungen zum Lesen oder Verändern von Daten nur teilweise verarbeitet werden können, oder wenn mehrere Schreibzugriffe gleichzeitig stattfinden.

Zur Zugriffssteuerung kann auch auf das in fast allen Datenbanksystemen implementierte Transaktionskonzept zurückgegriffen werden. Dieser Mechanismus stellt sicher, dass ein einzelner oder eine Menge von Datenbankzugriffen entweder vollständig oder gar nicht abgearbeitet werden. Werden beispielsweise ein Schreib- und ein Lesezugriff als eine Transaktion ausgeführt, wird, im Falle eines Fehlers bei der Verarbeitung des Lesebefehls, beispielsweise wegen fehlerhafter Syntax oder einer Zugriffsverletzung, auch die voran gegangene Schreibaktion rückgängig gemacht. Erst wenn alle Schritte einer Transaktion erfolgreich abgearbeitet wurden, wird der neue Stand für die Datenbank festgeschrieben. Das Blackboard im Internet der Dinge verwendet diesen Mechanismus, um Fehler wie nicht verständliche Nachrichten oder falsche Statements von Agenten abzufangen und bei Bedarf eine Fehlermeldung an den Agenten zurück zu senden.

Um Inkonsistenzen durch gleichzeitige Schreibzugriffe zu vermeiden, müssen exklusive Schreibrechte an Agenten vergeben werden. Dies kann auf zwei Arten geschehen:

1. Ein „Token“ wird zyklisch von einem Agenten zum nächsten gegeben, wobei der den Token besitzende Agent als einziger das Recht hat, schreibend auf das Blackboard zuzugreifen (siehe Abbildung 4-15: Der mit dem Blackboard kommunizierende Agent sowie die von ihm gesperrten Daten sind grau eingezeichnet). Diese Methode hat jedoch mehrere Nachteile:
    - Die Tokendurchlaufzeit erhöht sich mit der Anzahl der Agenten, so dass gerade bei sehr großen Systemen nur eine niedrige Performance erreicht werden kann.
    - Der Token bezieht sich auf das gesamte Blackboard, so dass mehrere Agenten auch dann nicht einen gleichzeitigen Schreibzugriff haben, wenn sie verschiedene, komplett unabhängige Daten verändern wollen.
    - Gibt ein Agent seinen Token nicht mehr ab, beispielsweise wegen eines Programmabsturzes, ist das Gesamtsystem nur noch sehr eingeschränkt lauffähig (da niemand den Token bekommt, kann kein Agent Daten auf das Blackboard schreiben). Um dies zu vermeiden, muss die Vergabe des Tokens zentral, beispielsweise vom Blackboard selbst oder von einem weiteren Dienst, verwaltet und überwacht werden. Erst auf diese Weise kann einem Agenten der Token entzogen werden, wenn dieser nicht innerhalb einer bestimmten Zeitspanne zurück gegeben wird.

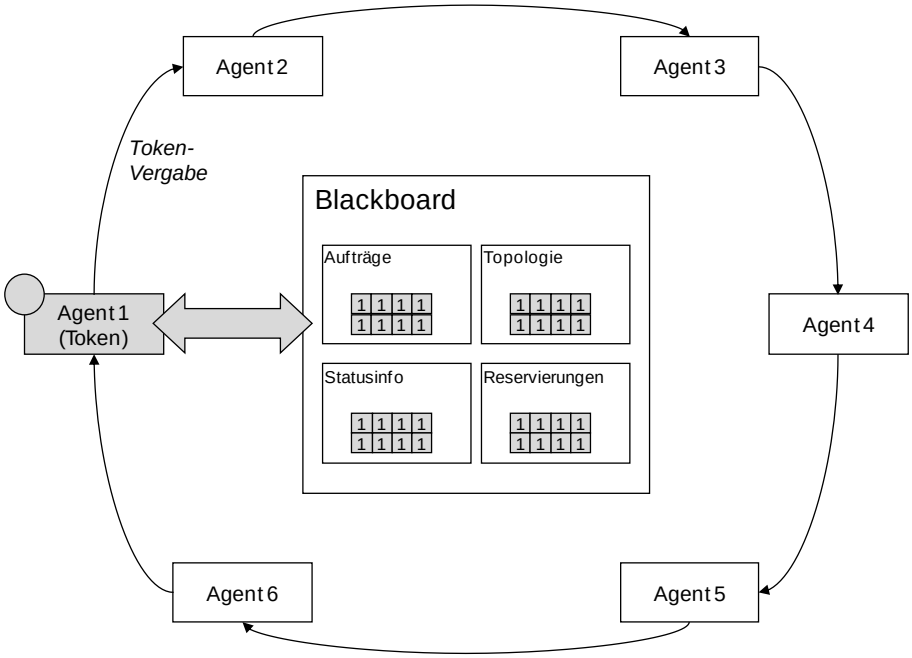


Abbildung 4-15 Zugriffssteuerung über Token

2. Agenten fordern bei Bedarf exklusive Schreibrechte oder „Sperrungen“ an, die sich auf bestimmte Dateninhalte des Blackboards beziehen. Die Schreibrechte werden ihnen dann, sofern keine andere Sperrung vorliegt, vom Blackboard zugeteilt und die betroffenen Dateninhalte für andere Agenten gesperrt (siehe Sequenzdiagramm in Abbildung 4-17). Diese Methode hat den Nachteil einer höheren Kommunikationslast, da bis zum Erhalt der Schreibrechte evtl. mehrere Anfragen und Absagen zwischen Agent und Blackboard versendet werden müssen. Da es keine festgelegte Reihenfolge oder Priorität bei der Vergabe von Schreibrechten gibt, besteht jedoch die theoretisch Möglichkeit, dass durch auftretende Konflikte manche Agenten niemals, oder nur nach sehr langer Zeit, eine benötigte Sperrung anlegen können. Um dies zu vermeiden, kann das Blackboard eine Liste der abgelehnten Anfragen führen und auf Grundlage dieser Daten Agenten bevorzugt behandeln. Neben diesen Schwächen hat diese Methode gegenüber dem Token-Ansatz jedoch folgende Vorteile:

    - Es muss kein Token verwaltet und überwacht werden, Durchlaufzeiten entfallen.
    - Sperrungen gelten im Allgemeinen nicht für das gesamte Blackboard, sondern nur für ganz bestimmte Bereiche. Auf diese

Weise können mehrere Agenten gleichzeitig verschiedene Daten schreiben, sodass beispielsweise die Reservierung eines Transportauftrages durch einen Agenten und das Verändern von Kosten für einen Streckenabschnitt durch einen anderen simultan durchgeführt werden können (siehe Abbildung 4-16: Die mit dem Blackboard kommunizierenden Agenten und die gesperrten Daten sind grau eingezeichnet. Die gesperrten Daten sind mit der Nummer des Agenten, der die Schreibrechte besitzt, versehen). Dies kann die Effizienz des Systems drastisch erhöhen.

- Wenn gleichzeitige Zugriffe auf Grund der Systemlogik ausgeschlossen werden können, können Schreibvorgänge komplett ohne Sperrung durchgeführt werden. Dies ist beispielsweise für die Aktualisierung von moduleigenen Statusinformationen der Fall, die beispielsweise bei der häufigen Übertragung aktueller Positionsdaten einen nicht zu vernachlässigenden Anteil des systemweiten Datenverkehrs ausmachen können.

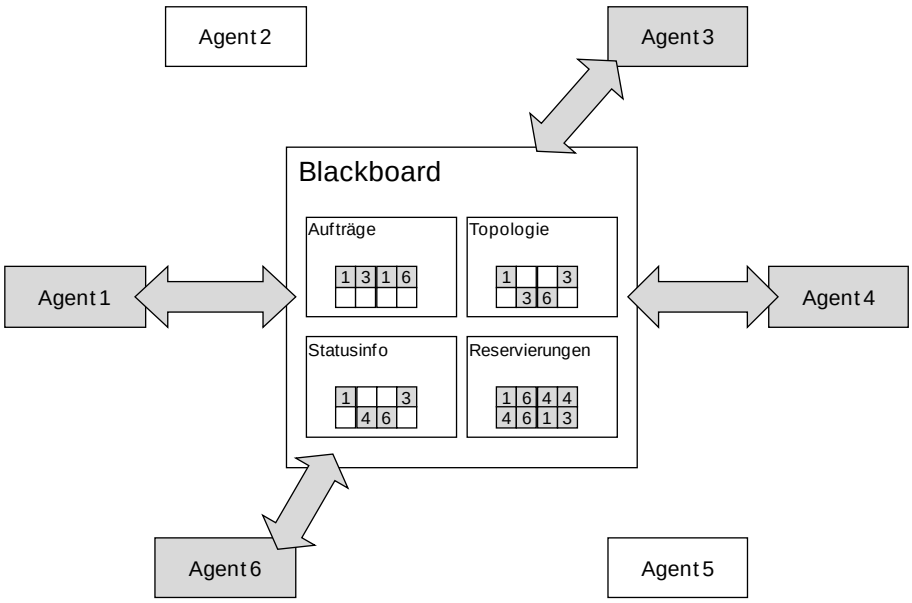


Abbildung 4-16 Zugriffssteuerung über Sperrungen.

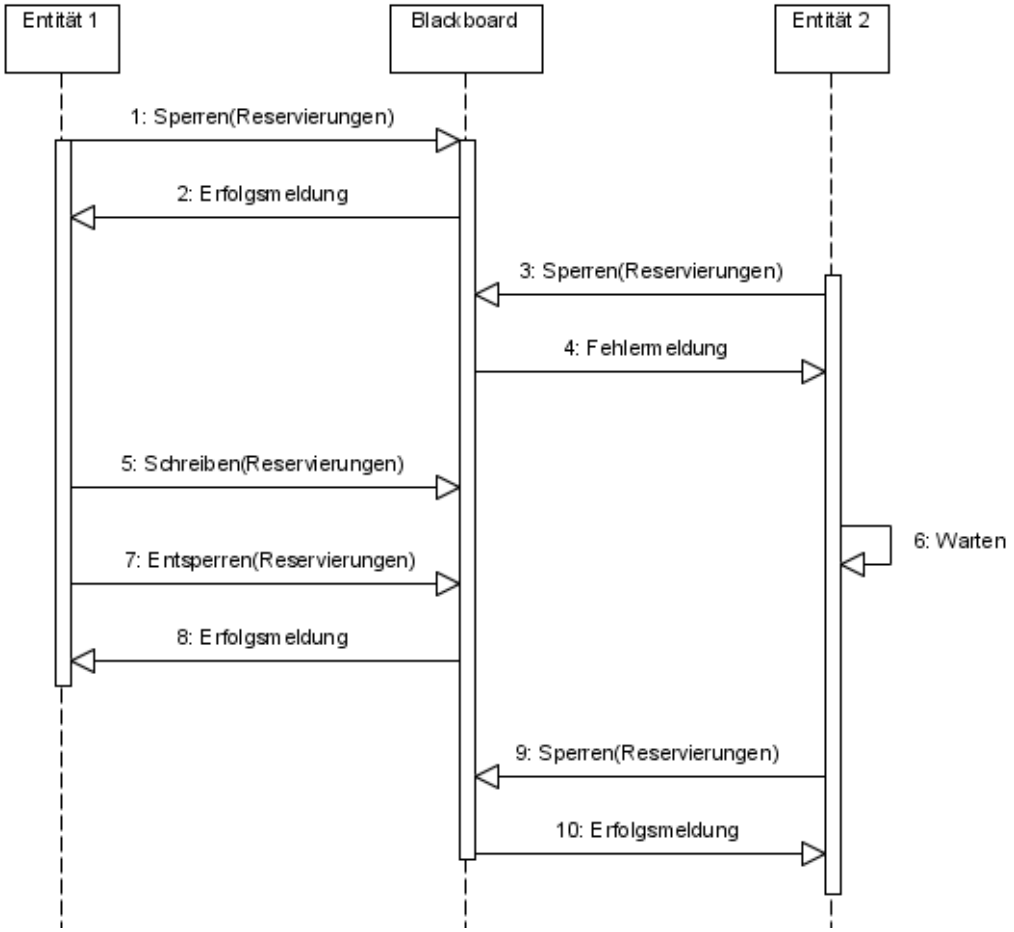


Abbildung 4-17 Blackboard-Zugriffsteuerung

4.5.2 Benachrichtigungsmechanismus

Wie im Abschnitt 4.2.2 gezeigt, ist eine Kommunikation nach dem „Push-Prinzip“, also die Verwendung von Abonnements an Stelle wiederholter Anfragen, in der Lage, die Kommunikationslast im System zu reduzieren. Dies gilt vor allem dann, wenn die Beziehungen zwischen Anbietern und Verbrauchern von Informationen statisch sind und das Anlegen bzw. Verändern von Abonnements relativ selten notwendig ist.

Daher ist die Verwendung eines Benachrichtigungsmechanismus gerade beim Einsatz eines Blackboards sinnvoll, da hier Sender und Empfänger bzw. Informationslieferant und -verbraucher entkoppelt sind – das Blackboard ist somit für alle Systemteilnehmer der einzige Kommunikationspartner. Ein Agent, der über Datenänderungen benachrichtigt werden möchten, meldet sich unter Angabe der gewünschten Informationstypen (z.B. Angaben zu Streckenreservierungen oder -belegungen) beim Blackboard an. Verändert nun eine andere Entität im System einen dieser Einträge,

kann das Blackboard die neuen Inhalte bzw. eine Benachrichtigung über das Löschen des Datensatzes an alle Interessenten weiterleiten (siehe Abbildung 4-18).

Dabei ist es sinnvoll, die Benachrichtigungen mit einem Zeitstempel zu versehen, der den Zeitpunkt des Schreibvorgangs auf dem Blackboard angibt. Da in einem TCP/IP-Netz die Reihenfolge von Nachrichten nicht unbedingt erhalten bleibt, d.h. dass eine später versendete Nachricht eine frühere Nachricht während der Übertragung „überholen“ kann, erlaubt ein Zeitstempel den benachrichtigten Agenten, die korrekte Reihenfolge der Nachrichten zu rekonstruieren.

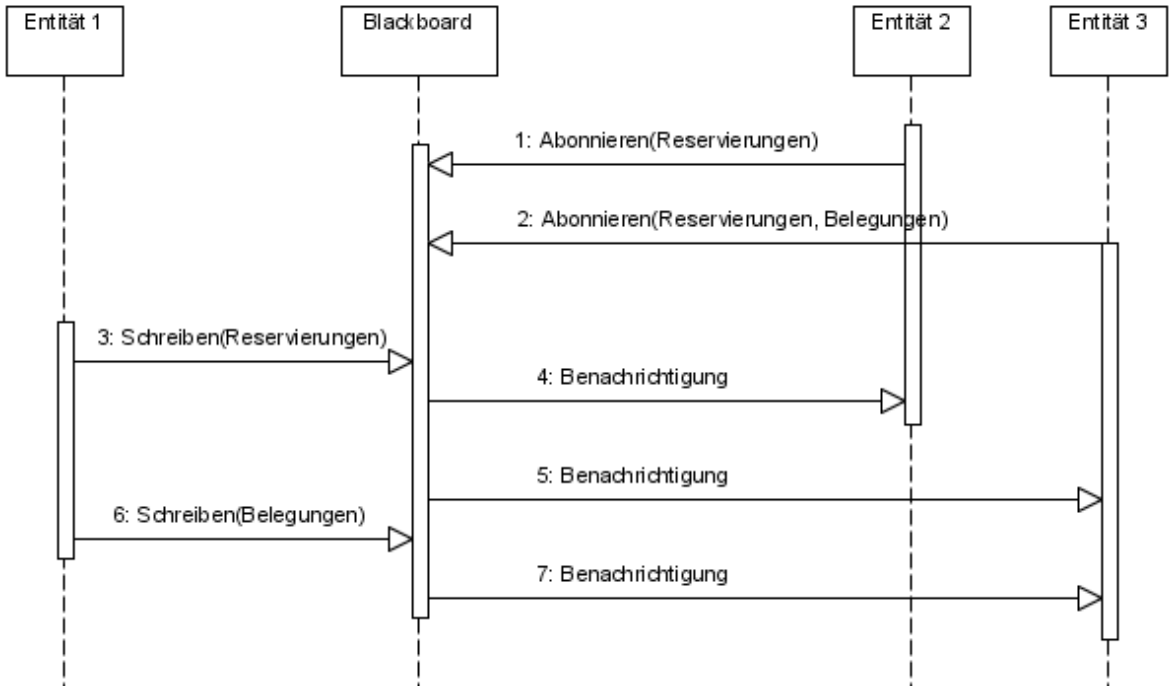


Abbildung 4-18 Blackboard-Benachrichtigungen

4.5.3 Parallelbetrieb von Blackboards

Wie jedes zentrale System hat ein Blackboard den Nachteil, dass es sowohl einen Bottleneck als auch einen Single-Point-of-Failure darstellt. Die Leistungsfähigkeit des Gesamtsystems ist somit durch die Leistungsfähigkeit der Datenaustauschplattform begrenzt. Die klassische Methode, um mit diesen Herausforderungen umzugehen, besteht in der redundanten Auslegung kritischer Systemkomponenten, sodass der Ausfall einer davon im schlimmsten Fall zu einer Verringerung der Gesamtperformance, jedoch nicht zu einem kompletten Systemzusammenbruch führt.

Dieses Vorgehen wird auch in heutigen Materialflussanlagen und Logistiksystemen eingesetzt, beispielsweise durch das doppelte Vorhandensein des Materialflussrechners. Ein wesentlicher Unterschied zwischen dem Einsatz einer redundant ausgelegten zentralen Steuerung und einem Internet der Dinge, in dem die Steuerungslogik komplett dezentralisiert ist und nur ein Teil der Kommunikation auf einer zentralen Architektur beruht, besteht in der Tatsache, dass die zentrale Komponente des Internet der Dinge, also das Blackboard, anwendungsunabhängig ist und eine sehr geringe interne Komplexität aufweist. Wurde ein Blackboard mit den grundlegenden Mechanismen implementiert, kann es für jede beliebige Anwendung – auch außerhalb der Logistik – ohne Änderung oder Umkonfiguration eingesetzt werden.

Die vorgestellten Konzepte zur Zugriffssteuerung und zum automatischen Versenden von Benachrichtigungen genügen bereits, um nebenläufige Blackboards zu realisieren. Da im Internet der Dinge auch ein Blackboard eine ganz gewöhnliche Entität darstellt, die in der Lage ist, mit allen anderen Entitäten zu kommunizieren, können zwei oder mehrere Blackboards über die gegenseitige Abonnierung aller Daten synchronisiert werden. Weiterhin muss ein Blackboard, beim Eingehen einer Anforderung zum Sperren bestimmter Daten, dieselbe Sperrung bei allen anderen Blackboards anfordern, bevor es diese aktivieren kann (siehe Sequenzdiagramm in Abbildung 4-19). Auf diese Weise lässt sich ein Blackboard-basiertes System mit minimalem Aufwand redundant auslegen. Fällt nun ein solcher Informationsknotenpunkt aus, gehen nur solche Informationen verloren, die noch nicht an die anderen Blackboards weitergeleitet wurden, wobei in diesem Fall die Entität, die die ursprüngliche Nachricht versendet hat, keine Erfolgsmeldung erhält und somit ihre Nachricht an eines der übrigen Blackboards nochmals versenden kann.

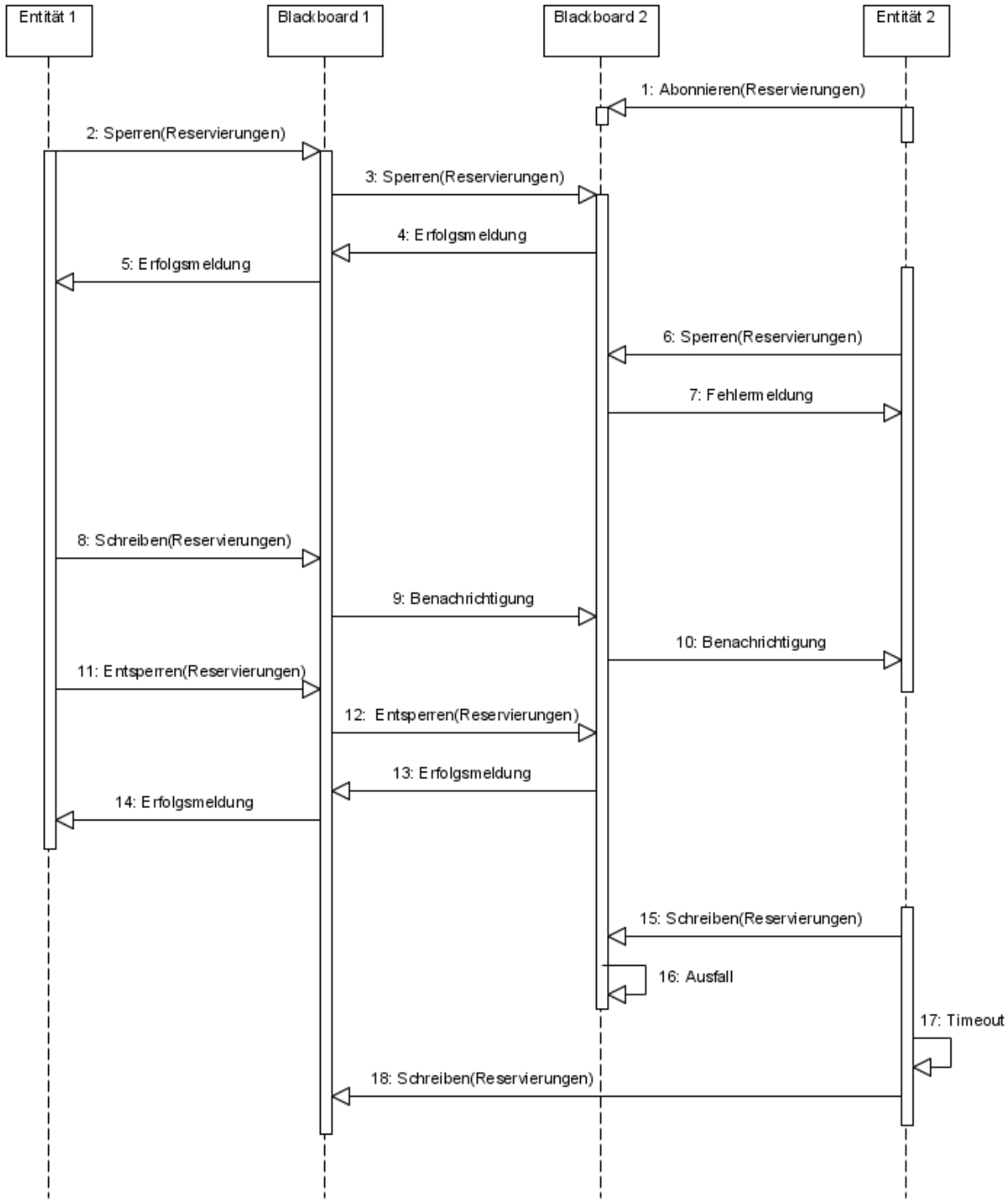


Abbildung 4-19 Parallelbetrieb von Blackboards

5 Steuerungsstrategien für das Internet der Dinge

Die Ontologie und das vorgestellte Kommunikationskonzept bilden die Grundlagen für die Realisierung eines Netzwerks aus Software- und Mechatronik-Einheiten. Zur Umsetzung einer intralogistischen Anlage werden jedoch konkrete Strategien benötigt, die z.B. die Auftragsdisposition und den Transport regeln und einen sicheren, deadlockfreien und durchsatzoptimalen Betrieb des Systems gewährleisten. Dabei sind folgende Aspekte zu betrachten.

1. Die **Zielbestimmung und Auftragsdisposition** erfordern die Suche nach und die Auswahl von Modulen für einen Auftrag, der gemäß der Ontologie einem Schritt im Workflow einer Transporteinheit entspricht.
2. Die **Wegplanung und der Transport** regeln die Berechnung, Bewertung und Auswahl von Pfaden, über die eine Transporteinheit gefördert werden muss, um ihren Workflow abzuarbeiten, sowie die physikalische Durchführung der Transporte. Dabei sind verschiedene Randbedingungen, wie z.B. Besonderheiten in der Topologie einer Anlage und bestimmte Materialflussstrategien zu berücksichtigen.
3. Dabei muss auch im Internet der Dinge eine **zentrale Steuer- und Beobachtbarkeit** des Systems möglich sein. Obwohl der aktive Eingriff in die Steuerungsvorgänge einer Anlage nur in Ausnahmefällen, z.B. beim Auftreten von Fehlern oder bei der Durchführung von Wartungs- oder Umbauarbeiten, vorbehalten bleibt, ist vor allem die zentrale Visualisierbarkeit des gesamten Anlagenzustands eine Anforderung, die auch eine dezentrale Materialflusssteuerung erfüllen muss.

5.1 Zielbestimmung und Auftragsdisposition

5.1.1 Service Discovery

Transporteinheiten im Internet der Dinge sind mit eigener Intelligenz ausgestattet und selbst dafür verantwortlich, ihren Workflow ordnungsgemäß abzuarbeiten. Eine Methode zur Modellierung und datentechnischen Abbildung und Verarbeitung von

Workflows mittels High-level Petri-Netzen wurde in Abschnitt 3.5.3 bereits vorgestellt. An dieser Stelle sollen nun die Mechanismen betrachtet werden, die zur Abarbeitung eines Workflowschrittes notwendig sind.

Ein Workflowschritt bezieht sich immer auf eine Funktion, die von einem oder mehreren Modulen angeboten und von der Transporteinheit in Anspruch genommen werden muss. Diese Orientierung an Funktionen und nicht an bestimmten Teilnehmern eines Systems entspricht dem Konzept der ursprünglich von Sun geprägten Service Oriented Architecture, kurz SOA [Sun-05]. Ein wichtiger Aspekt dieser dienst- bzw. funktionsorientierten Vorgehensweise ist die so genannte „Service Discovery“ oder das Auffinden von Entitäten, die die gesuchte Funktion anbieten. Oftmals werden für diesen Zweck spezielle Verzeichnisse eingesetzt, die ähnlich den „Gelben Seiten“ Einträge über Entitäten und ihre jeweilige Funktionalität enthalten [Gar-04]. Genauso kann aber auch ein Blackboard, das normalerweise für die Vermittlung allgemeiner Dateninhalte genutzt wird, für das Publizieren und Suchen von Funktionen bzw. Funktionsanbietern verwendet werden.

So müssen im Internet der Dinge alle Entitäten die von ihnen angebotenen Funktionen in ein Verzeichnis (bzw. Blackboard) eintragen. Eine Transporteinheit durchsucht das Verzeichnis nach einem geeigneten Dienstleister und kann das gefundene Modul dann direkt kontaktieren, um die Funktion in Anspruch zu nehmen (siehe Abbildung 5-1). Auch die FIPA-Spezifikation sieht für diesen Zweck unter dem Namen „Directory Facilitator“ einen Verzeichnisdienst vor, in dem Agenten die von ihnen angebotenen Funktionen veröffentlichen und diese Einträge bei Bedarf verändern können [FIPA-SC00023]. Dabei können nicht nur der Name der angebotenen Funktion, sondern auch zusätzliche Parameter – im Internet der Dinge z.B. die Kosten für die Funktionserbringung – in das Verzeichnis eingetragen werden.

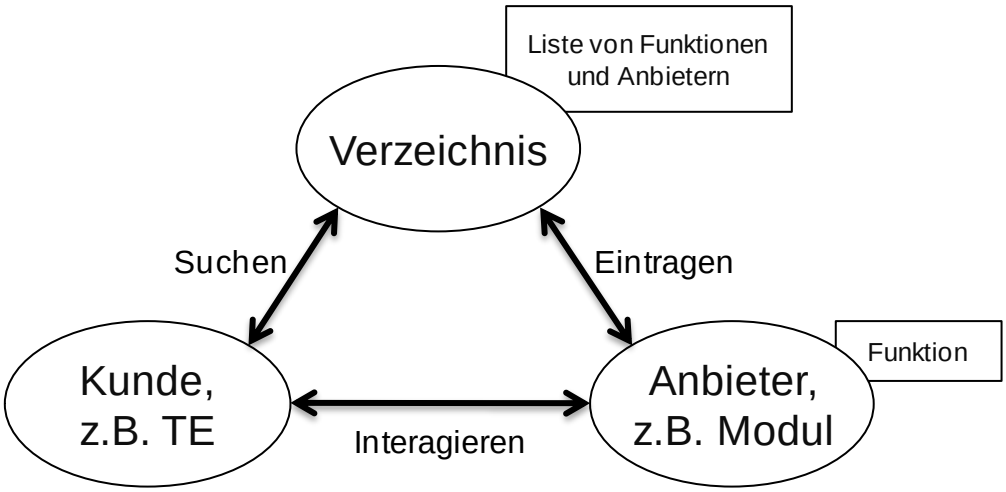


Abbildung 5-1 Service Discovery mit einem Verzeichnis

Eine weitere Möglichkeit zum Auffinden von Funktionsanbietern besteht in der Umkehrung des bereits beschriebenen Vorgehens. Dabei publizieren nicht die Anbieter ihre Funktionen, sondern die Kunden ihren Bedarf. Die Funktionsanbieter sind wiederum dafür verantwortlich, das Verzeichnis nach potenziellen Kunden zu durchsuchen, diese zu kontaktieren und ihnen das Erbringen der gewünschten Funktion zu einem gewissen Preis anzubieten.

Neben den Kosten für die eigentliche Funktionserbringung, die bereits im Rahmen der Service Discovery bestimmt werden können, sind aber auch die Kosten für den Transport zum betreffenden Modul zu betrachten. Denn der Grund für die niedrigen Kosten zum Durchleuchten eines Gepäckstücks kann z.B. an einer sehr schwachen Auslastung der Röntgenanlage liegen, die wiederum daher rührt, dass die Zuführstrecke zu diesem Modul ausgefallen ist. Eine Transporteinheit muss demnach eine Wegplanung von ihrem aktuellen Standort zu jedem der in Frage kommenden Zielmodule durchführen und die berechneten Pfade bewerten. Dabei ist zu beachten, dass diese Abschätzung der Transportkosten nicht sehr genau ist, da die Eigenheiten der Materialflusssteuerung, wie z.B. bestimmte Strategien, der TE unbekannt sind. Eine genaue Berechnung der Transportkosten ist nur durch die Fördermodule an sich möglich. Dies ist aber zum relativ frühen Zeitpunkt der Zielbestimmung aus zwei Gründen nicht zu empfehlen:

- Die Kommunikationslast im System und die Rechenlast der Module würden durch die größere Anzahl erforderlicher Wegplanungen u.U. stark steigen.

- Die berechneten Kosten berücksichtigen nur den aktuellen Systemzustand. Je länger der Transport zum Ziel ist, desto größer ist aber die Wahrscheinlichkeit, dass Veränderungen der Systemlast oder anderer Faktoren eintreten und sich die tatsächlichen Transportkosten daher stark ändern.

Die Wegplanung durch die Transporteinheit dient also nicht dem Zweck, eine bestimmte Route für den Transport durch das System zu wählen, sondern lediglich einer groben Abschätzung des am besten erreichbaren Zieles. Die Wegplanung an sich wird in Abschnitt 5.2 eingehender betrachtet.

5.1.2 Auktionen

Die Mechanismen der Service Discovery erlauben beispielsweise einer Transporteinheit, Module zu finden, die eine bestimmte Funktion bieten. Dabei können aber mehrere Module gefunden werden, zwischen denen sich die Transporteinheit entscheiden muss. Diese Entscheidungsfindung bzw. Auswahl des optimalen Dienstleisters für eine gesuchte Funktion kann in dezentralen Systemen über Auktionen stattfinden.

5.1.2.1 Auktionsmechanismen

Zur Durchführung von Auktionen, bei denen aus einer Menge von Anbietern ein einziger den Zuschlag erhält, sind hauptsächlich vier verschiedene Mechanismen bekannt (siehe Abbildung 5-2): die holländische, die englische, die Höchstpreis- und die Vickrey-Auktion [San-96] [Ski-99].

Die englische Auktion basiert auf offenen Geboten, wobei mehrere Anbieter über mehrere Runden das bisher beste Gebot überbieten, bis keine höheren Gebote mehr abgegeben werden. Bei der holländischen Auktion wird ein Gebot sukzessive herabgesetzt, bis ein Erster das Angebot annimmt und die Auktion somit beendet ist. Im Gegensatz dazu basieren die Höchstpreis- und Vickrey-Auktionen auf verdeckten Geboten. Jeder Anbieter kann dabei einen einzigen Preis vorschlagen, wobei das beste Angebot den Zuschlag erhält. Der Unterschied zwischen den zwei Mechanismen besteht darin, dass bei der Höchstpreisauktion der Gewinner den höchsten und von ihm selbst gebotenen Preis zahlt, während bei der Vickrey-Auktion der Gewinner nicht den eigenen, sondern den zweit-höchsten gebotenen Preis bezahlen muss.

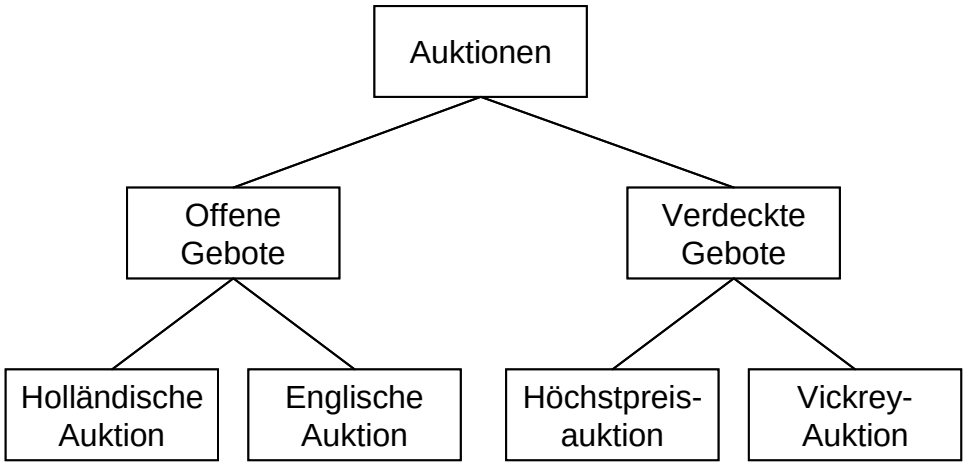


Abbildung 5-2 Auktionsmechanismen

Diese Mechanismen haben verschiedene Vor- und Nachteile in Bezug auf die für den Auktionator erzielbaren Gewinne, bzw. die für die Bieter erzielbaren Einsparungen. Außer bei der Vickrey-Auktion spielt auch die Spekulation über das Verhalten und die Zahlungsbereitschaft anderer Bieter eine wichtige Rolle beim Abgeben von Geboten.

Bei der Auswahl eines Zielmoduls im Internet der Dinge steht jedoch nicht das Erreichen individueller Vorteile im Vordergrund, sondern das Finden einer im Sinne des Gesamtsystems optimalen Lösung. Offene, über mehrere Runden abgegebene Gebote bieten daher keinen Vorteil. Ebenso können Spekulationen über die Gebote anderer Teilnehmer ausgeschlossen werden, sodass die Höchstpreisauktion die einfachste und am besten geeignete Methode für die Auswahl eines Moduls darstellt.

5.1.2.2 Auktionsprotokolle

Eine Transporteinheit muss für jeden Workflowschritt eine Liste möglicher Zielmodule bestimmen und jedem davon eine Bewertung, als Summe der Kosten für die Funktionserbringung und für den Transport zum betreffenden Ziel, zuweisen. Auf dieser Grundlage kann eine Transporteinheit ein konkretes Zielmodul auswählen und den Transport dorthin veranlassen (siehe UML-Sequenzdiagramm in Abbildung 5-3).

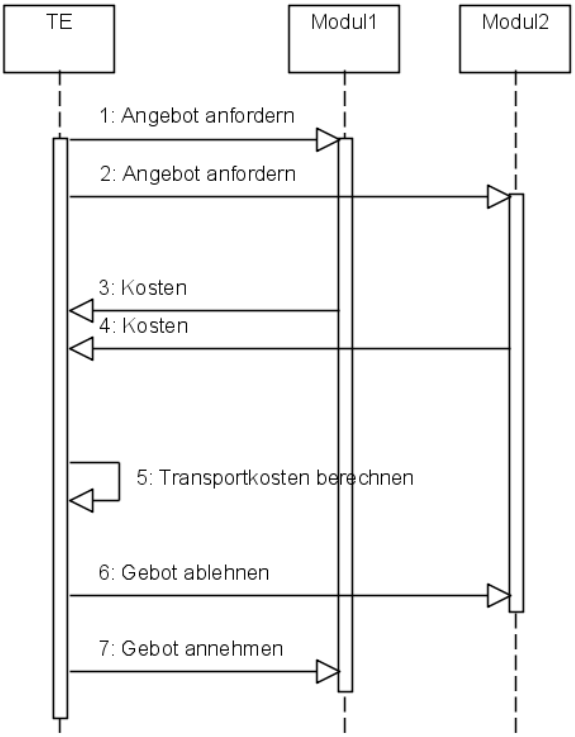


Abbildung 5-3 Von einer TE moderierte Auktion

Dabei ist jedoch zu beachten, dass eine TE im Normalfall nur dann nach Modulen sucht, wenn sie einen neuen Workflowschritt bearbeitet, denn nach der Wahl des Zielmoduls bleibt die Transporteinheit weitgehend passiv und wartet auf eine Meldung über die erfolgreiche Ausführung der benötigten Funktion. Der eigentliche Transport zum Ziel wird nur durch die Module gesteuert. Dafür müssen u.U. aber weitere Auktionen stattfinden, die darüber entscheiden, welches konkrete Modul einen Transportauftrag durchführt. Dies ist z.B. dann der Fall, wenn eine Palette auf ihrem Weg zu ihrem Lagerplatz von einem Querverschiebewagen umgesetzt werden muss, dabei aber mehrere Wägen zur Verfügung stehen. Da solche Details des Transportvorgangs für die transportierte Einheit unbekannt sind, müssen in einem solchen Fall die entsprechenden Module untereinander und ohne Wissen der TE eine Auktion durchführen (siehe Abbildung 5-4).

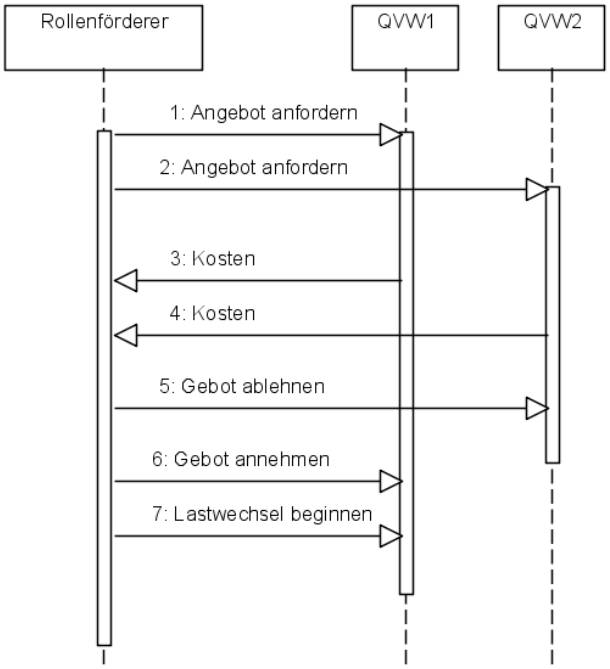


Abbildung 5-4 Von einem Modul moderierte Auktion

Diese einfache Auktionsform hat jedoch den Nachteil, dass die Anfragen bzgl. einer Funktionserbringung unabhängig voneinander und in nicht vorhersehbarer Reihenfolge und Menge bei einem Bieter ankommen und somit unabhängig voneinander verarbeitet werden müssen. Der Bieter kann somit nicht zwischen verschiedenen Auktionen wählen bzw. sein Angebot in Abhängigkeit mehrerer Auktionen berechnen, da er beim Eintreffen einer Anfrage nicht wissen kann, ob nicht kurze Zeit später eine andere, für ihn günstiger zu erbringende Funktion angefordert wird.

Um diesen Fall zu erläutern soll das obige Beispiel einer Auktion zwischen einem Rollenförderer und zwei Querverschiebewägen um einen weiteren Rollenförderer, der ebenfalls eine Palette an einen der QVWs abgeben will, ergänzt und anhand des Layouts in Abbildung 5-5 durchgespielt werden.

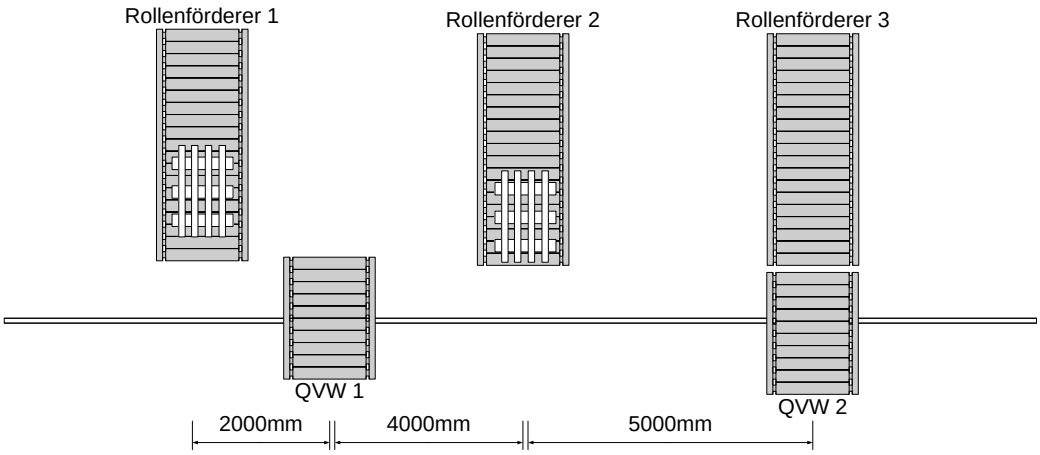


Abbildung 5-5 Beispielhaftes Layout mit zwei Querverschiebewägen

In diesem Szenario möchte Rollenförderer 2 eine Palette abgeben und fragt QVW 1 und QVW 2 an. QVW 1 wird auf Grund der geringeren Entfernung zum Rollenförderer diese Auktion gewinnen. Sehr kurze Zeit später möchte auch Rollenförderer 1 eine Palette abgeben – QVW 1 ist nun aber schon mit der Abarbeitung des ersten Auftrags beschäftigt und kann somit einen eigentlich günstigeren Auftrag nicht mehr annehmen (siehe Abbildung 5-6). Zusätzlich kann im vorliegenden Fall auch QVW 2 kein Angebot für Rollenförderer 1 abgeben, da ihn QVW 1 behindert.

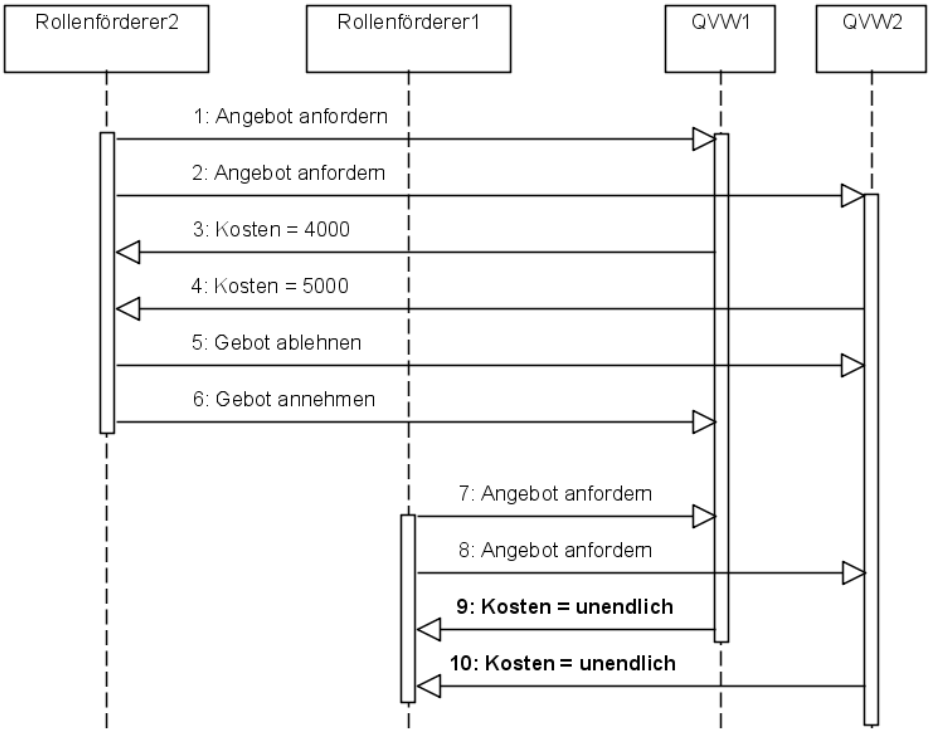


Abbildung 5-6 Probleme bei der Auftragsdisposition beim Einsatz einfacher Auktionen

Dieses Problem kann vermieden werden, indem Angebotsanfragen nicht sofort bearbeitet, sondern über einen gewissen Zeitraum gesammelt und nach Ablauf des Zeitfensters als Gruppe betrachtet werden. Zwar besteht hier wieder prinzipiell die Möglichkeit, dass auch sehr schnell aufeinanderfolgende Anfragen in verschiedene Zeitfenster fallen – trotzdem ist dieses Protokoll in der Lage, die Probleme der einfachen Auktion stark abzumildern. In dem vorliegenden Szenario würden QVW1 und QVW2 beide Anfragen – sowohl die von Rollenförderer 1 als auch die von Rollenförder 2 – erhalten, bevor sie ein Gebot abgeben. Die erste Möglichkeit besteht nun darin, dass sich jeder Bieter nur an der Auktion beteiligt, für die er das individuell günstigste Angebot liefern kann (siehe Abbildung 5-7). Dieses Vorgehen führt im vorliegenden Szenario bereits zur optimalen Verteilung von Aufträgen.

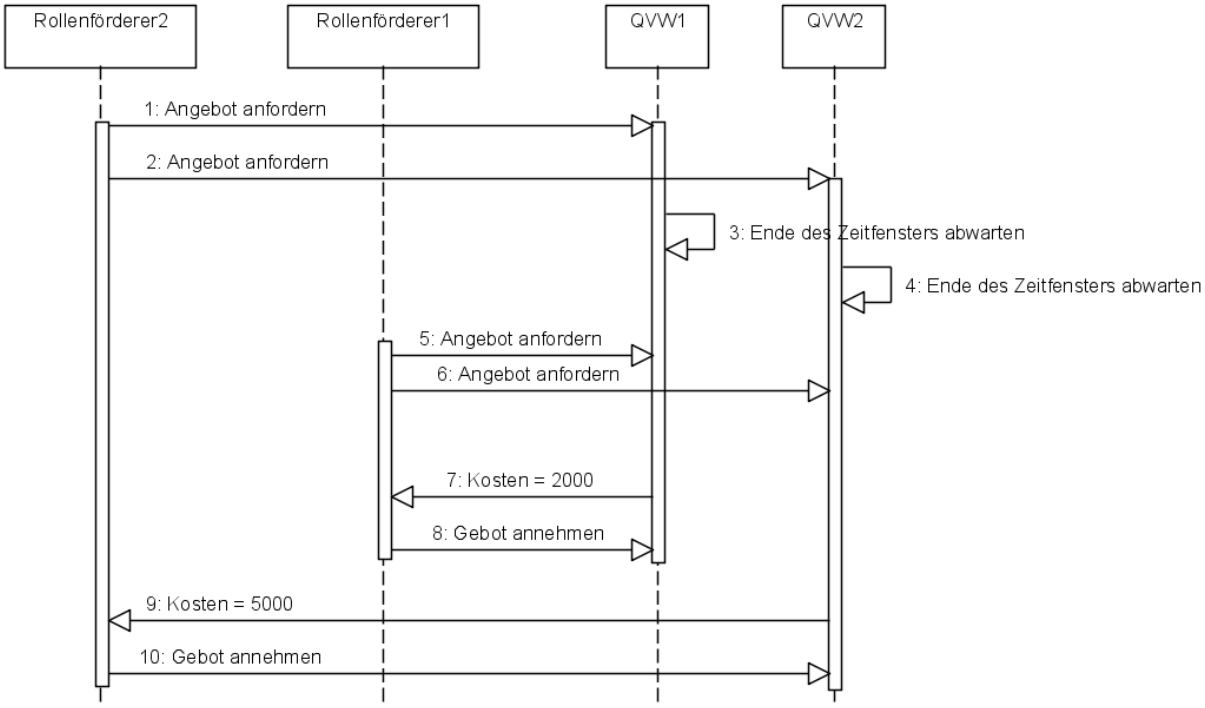


Abbildung 5-7 Optimierung der Auftragsdisposition durch gruppierte Auktionen

Diese Methode kann wiederum dazu führen, dass eine Auktion gar kein Gebot erhält, weil sie für keinen der Bieter die günstigsten Kosten verursacht, während andere Auktionen mehrere Gebote erhalten. Dies kann vermieden werden, indem das Auktionsmodell dahingehend erweitert wird, dass sich Bieter an allen für sie offen stehenden Auktionen beteiligen. Dadurch wird nun eine zweite Auktionsrunde erforderlich, da ein Bieter unter Umständen gleich mehrere Auktionen gewinnen, aber nur eine Funktion erbringen kann. In dieser zweiten Runde wählt der Bieter aus den von ihm gewonnen Auktionen die günstigste aus und bestätigt sein entsprechendes Gebot

beim Auktionator; die Gebote für alle anderen gewonnenen Auktionen werden widerrufen. Dies veranlasst einen Auktionator, dessen Auktionsgewinner sein Gebot widerruft, das zweitbeste Gebot auszuwählen und dem Bieter eine Bestätigung zu schicken (siehe Abbildung 5-8).

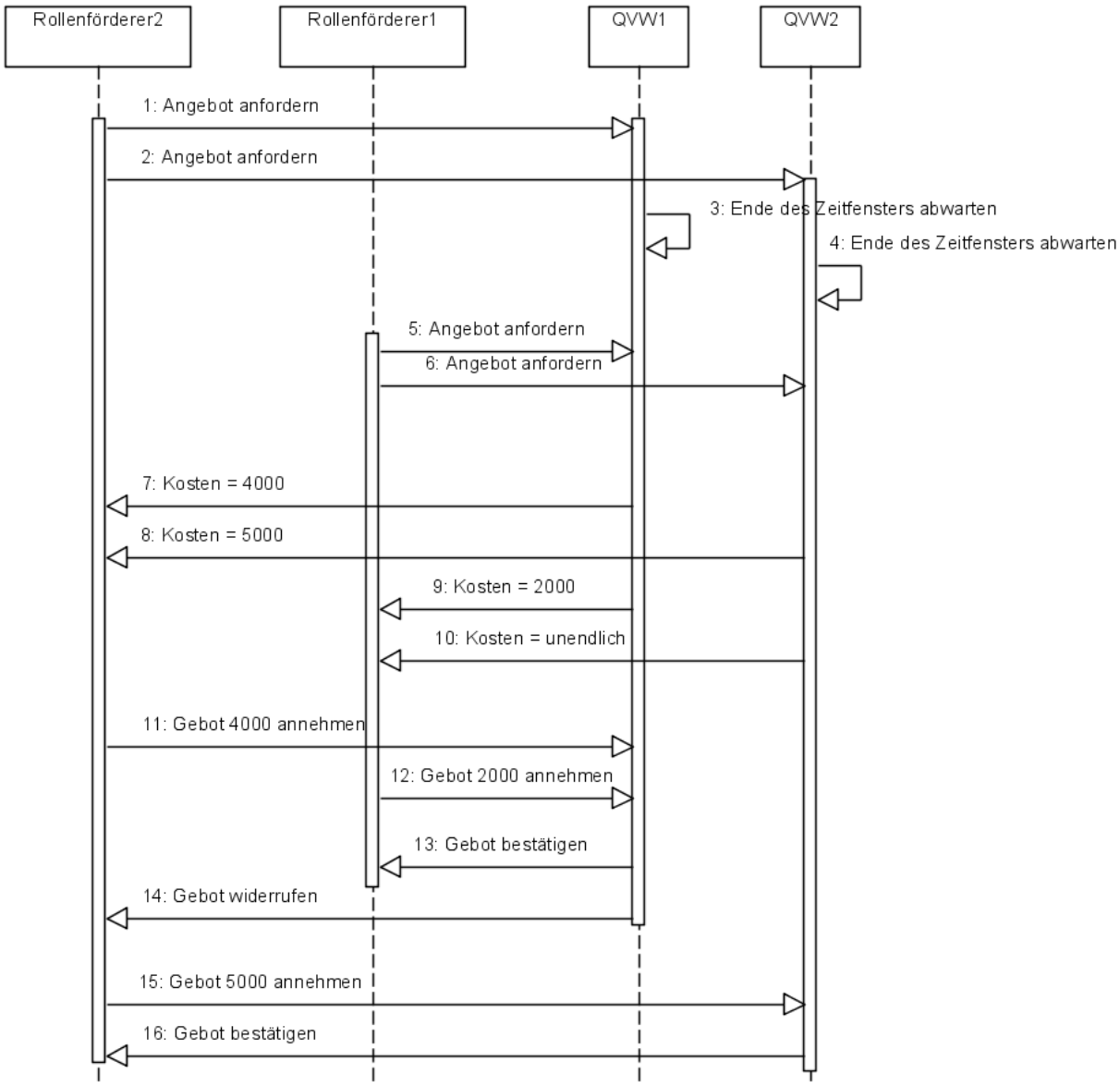


Abbildung 5-8 Optimierte Auftragsdisposition durch erweiterte Auktionen

5.1.3 Allgemeingültige Steuerungslogik für TE

Wie in den Abschnitten 3.5.2 und 4.1.1 beschrieben, sind die Transporteinheiten im Internet der Dinge dafür verantwortlich, die logistischen Prozesse einer Anlage umzusetzen und können somit als Träger der Geschäftslogik des Systems angesehen werden. Durch Verwendung der in Abschnitt 5.1.1 und 5.1.2 vorgestellten Service-

Discovery- und Auktionsmechanismen lässt sich eine einfache und allgemeingültige Steuerungslogik für Transporteinheiten definieren. Das UML-Sequenzdiagramm in Abbildung 5-9 beschreibt diesen Ablauf:

- Die TE bzw. der ihr zugeordnete Softwareagent bestimmt anhand ihres Workflows, welche Funktion als nächstes abzuarbeiten ist (für ein Gepäckstück am Flughafen z.B. „Durchleuchten“).
- Die TE nutzt einen Verzeichnisdienst (oder eine äquivalente Service-Discovery-Methode), um eine Liste der im System verfügbaren Anbieter dieser Funktion zu bekommen.
- Jede der dabei gefundenen Entitäten wird um ein Kostenangebot für die Erbringung der gesuchten Funktion gebeten.
- Für jedes erhaltene Angebot berechnet die TE zusätzlich die Kosten für den Transport zum entsprechenden Modul.
- Aus der Summe von Transportkosten und Kosten für die Funktionserbringung kann die TE das günstigste Modul auswählen.
- Alle an der Auktion beteiligten Entitäten werden von der TE darüber benachrichtigt, ob sie die Auktion gewonnen oder verloren haben.
- Die TE veranlasst den Transport zum neuen Ziel.
- Dort angekommen, fordert sie vom Modul die Erbringung der Funktion an und kann nach Erhalt einer Bestätigung und der bei der Funktionserbringung generierten Daten (z.B. die Einstufung als „sicher“ oder „unsicher“) mit der Bearbeitung des nächsten Workflowschritts fortfahren.

Dieses Vorgehen lässt sich wegen seiner hohen Allgemeingültigkeit ohne eine Veränderung der TE-Logik für beliebige Szenarien einsetzen, beispielsweise auch für die Steuerung von Kommissionier- oder Fertigungsprozessen.

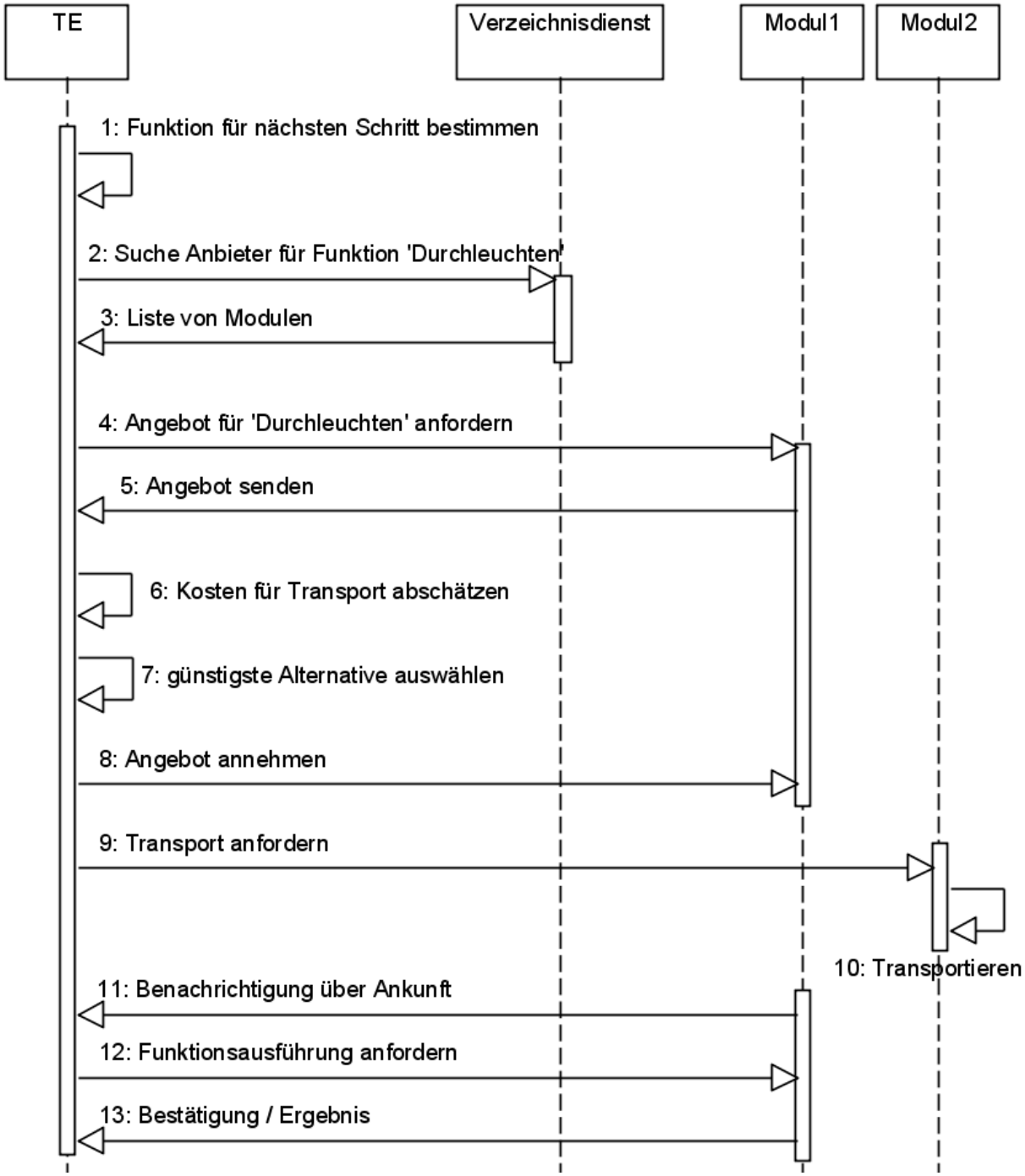


Abbildung 5-9 Allgemeingültiger Steuerungsablauf einer TE

5.2 Wegplanung und Transport

5.2.1 Definition der Topologie

Eine Materialflussanlage kann als Menge von Modulen aufgefasst werden, die miteinander verbunden sind. Die Topologie einer Anlage kann dann als gerichteter und gewichteter Graph dargestellt werden, wobei die Module als Knoten und die Verbindungen zwischen ihren Übergabepunkten als Kanten aufgefasst werden. Als Kantengewicht kann im einfachsten Fall die physikalische Länge der Verbindung angenommen werden, wobei eine den aktuellen Anlagenzustand berücksichtigende Kostenfunktion ebenfalls denkbar ist und durch ihre Dynamik bessere Ergebnisse bei der Wegbewertung verspricht. Der die Topologie einer Anlage beschreibende Graph ergibt sich dann als:

$G = (V_m, E)$, wobei

V_m = Menge aller Module im System,

$E \subseteq V_m \times V_m$ = Verbindungen zwischen Modulen

Dieser Graph enthält allerdings noch keine Informationen über die Übergabepunkte der Module. Die Anzahl und auch die modulinterne Verbindung von Übergabepunkten sind aber für viele Wegplanungs- und Transportaufgaben durchaus relevant, da erst auf dieser Ebene notwendige Schaltvorgänge, z.B. im Fall von Weichen, identifiziert werden können. Weiterhin ist es möglich, dass nach dem Betreten eines Moduls über einen bestimmten Übergabepunkt das Modul nicht über jeden beliebigen Ausgang verlassen werden kann. Ein auf Modulebene definierter Graph würde also Pfade enthalten, die mechanisch nicht umsetzbar sind. Daher ist auch die interne Struktur eines Moduls, also die Anzahl der Übergabepunkte und ihre Verbindungen, abzubilden. Diese Struktur kann ebenfalls als Graph mit Übergabepunkten als Knoten und modulinternen als auch modulexternen Verbindungen als Kanten modelliert werden. Durch die Detaillierung der Topologie auf die Ebene der Übergabepunkte ergibt sich folgende Graphdefinition:

$G = (V_ü, E_{ext}, E_{int})$, wobei

$V_ü$ = Menge aller Übergabepunkte im System,

$E_{ext} \subseteq V_ü \times V_ü$ = Verbindungen zwischen Übergabepunkten verschiedener Module

$E_{int} \subseteq V_ü \times V_ü$ = Verbindungen zwischen Übergabepunkten desselben Moduls

Beide Typen von Kanten werden in der Ontologie über das Prädikat *IstNachfolger* abgebildet. Bei Kanten aus E_{int} gehören aber beide Übergabepunkte zum selben Modul, während Kanten aus E_{ext} Übergabepunkte verschiedener Module verbinden.

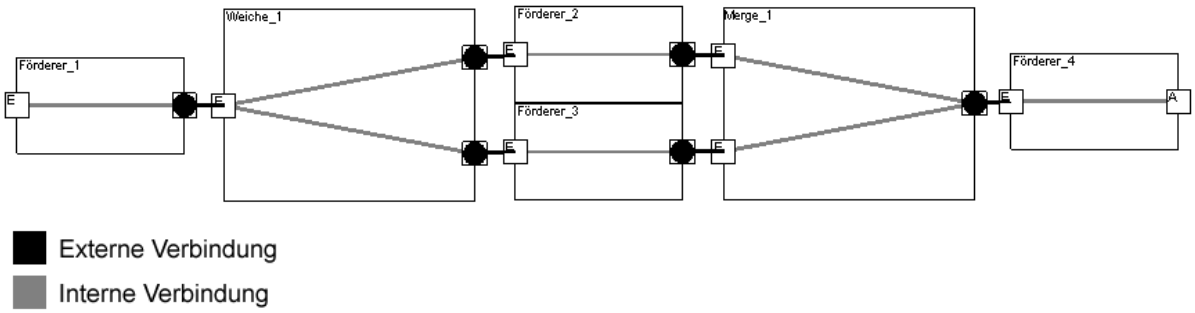


Abbildung 5-10 Beispielhafte Topologie aus vier Förderern, einer Weiche und einer Zusammenführung

Da im Internet der Dinge aber auch verschiedene Transportsysteme wie z.B. Stetig- und Unstetigfördertechnik zusammen arbeiten sollen, muss die datentechnischen Abbildung der Topologie um einen weiteren Kantentypus erweitert werden. Vor allem beim Übergang von Stetig- auf Unstetigförderer, z.B. bei der Übergabe einer Transporteinheit von einer Rollenbahn an einen Querverschiebewagen oder einen Stapler, lassen sich die Verknüpfungsbeziehungen zwischen den Übergabepunkten verschiedener Module nicht mehr als eindeutige 1:1 Beziehung darstellen. Der Grund dafür liegt in der Tatsache, dass die Transporteinheit prinzipiell an mehrere Module übergeben werden kann, beispielsweise beim Parallelbetrieb mehrerer Verschiebewägen auf einer Schiene. Zwar ist es möglich, auch diese Vorgänger-Nachfolger-Beziehungen auf herkömmliche Weise zu definieren und wie alle anderen Verbindungen zu handhaben. Verändert sich aber die Anzahl der Module, mit denen ein Übergabepunkt verbunden ist, beispielsweise beim Einfügen eines zusätzlichen Querverschiebewagens auf der selben Schiene oder dem Entfernen eines Staplers aus dem System, müssen diese Änderungen auch in der Topologie eingetragen werden. Gerade in hoch dynamischen Systemen, die häufig angepasst werden müssen, kann dies den Konfigurationsaufwand unnötig erhöhen.

Dieser zusätzliche Aufwand kann durch das Anwenden einer dienstorientierten Vorgehensweise vermieden werden. Dabei wird als Nachfolger eines Übergabepunktes eines Moduls nicht ein anderer Übergabepunkt verwendet, sondern eine nicht genau definierte Menge von Modulen, deren Gemeinsamkeit darin besteht, dass sie Teil

eines bestimmten Subsystems sind, z.B. „Querverschiebewagen Wareneingang“. Diese Zugehörigkeit zu einem Subsystem kann auch als Funktion aufgefasst werden; ein Modul, das diese Funktion anbietet, kann dann über einen Service-Discovery-Mechanismus gefunden werden. Dies bedeutet, dass bei jeder Routenplanung bzw. bei jedem Transport das Nachfolgemodul neu und dynamisch bestimmt wird. Durch dieses Vorgehen kann die Anzahl der unstetig fördernden Module in einem Subsystem beliebig verändert werden, ohne Änderungen am datentechnischen Abbild der Topologie zu erfordern.

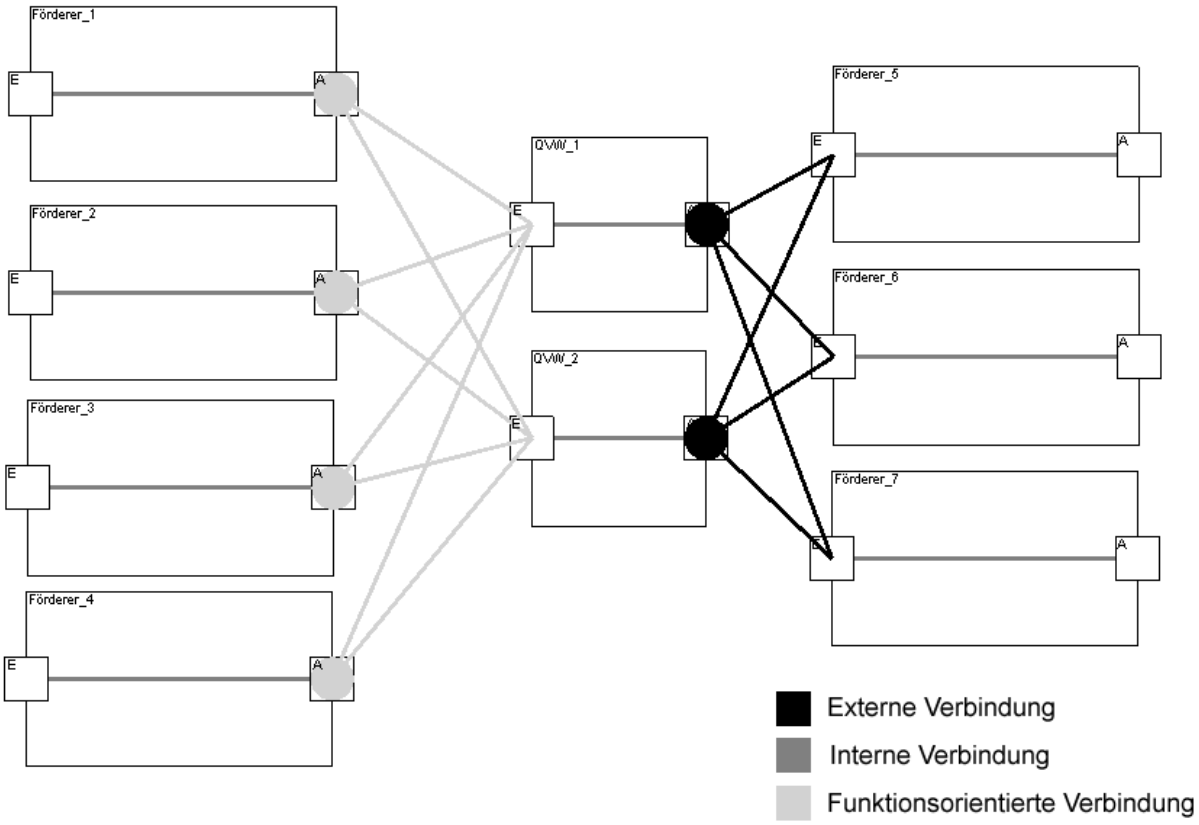


Abbildung 5-11 Funktionsorientierte Verbindungen beim Übergang zwischen Stetig- und Unstetigförderern

Neben den Übergabepunkten und den internen und externen Verbindungen zwischen diesen, gibt es im Topologiegraphen eines Internet der Dinge Systems also zwei weitere Elemente:

$G = (V_ü, V_s, E_{ext}, E_{int,} E_s)$, wobei

$V_ü$ = Menge aller Übergabepunkte im System,

V_F = Menge definierter Subsysteme bzw. Funktionen im System,

$E_{ext} \subseteq V_ü \times V_ü$ = Verbindungen zwischen Übergabepunkten verschiedener Module

$E_{int} \subseteq V_{ü} \times V_{ü}$ = Verbindungen zwischen Übergabepunkten desselben Moduls

$E_F \subseteq V_F \times V_F$ = Verbindungen zwischen einem festen Übergabepunkt und einem Subsystem, die erst zur Laufzeit, also bei einer konkreten Wegplanungs- oder Transportaufgabe, konkretisiert werden.

Ein Graph lässt sich beispielsweise mittels einer Adjazenzmatrix oder Adjazenzliste als maschinenlesbare Datenstruktur abbilden [Wip-06] und kann beispielsweise über ein Blackboard allen Entitäten eines Systems zur Verfügung gestellt werden. Eine erweiterbare, auf XML basierende Kodierung des Graphen und der Attribute der einzelnen Kanten und Knoten wurde in [Wil-06] entwickelt und vorgestellt. Dabei wird jedem Modul bzw. „globalen Wegpunkt“ eine XML-Datenstruktur zugewiesen, die sowohl statische Attribute (z.B. Name, Länge oder Kapazität) als auch dynamische Eigenschaften (z.B. Belegung, Reservierungen oder Kosten) enthält. Zusätzlich werden dort auch Informationen zu den Übergabepunkten bzw. „lokalen Wegpunkten“ und ihrer Verbindung mit modulinternen und modulexternen Übergabepunkten festgehalten (siehe Abbildung 5-12).

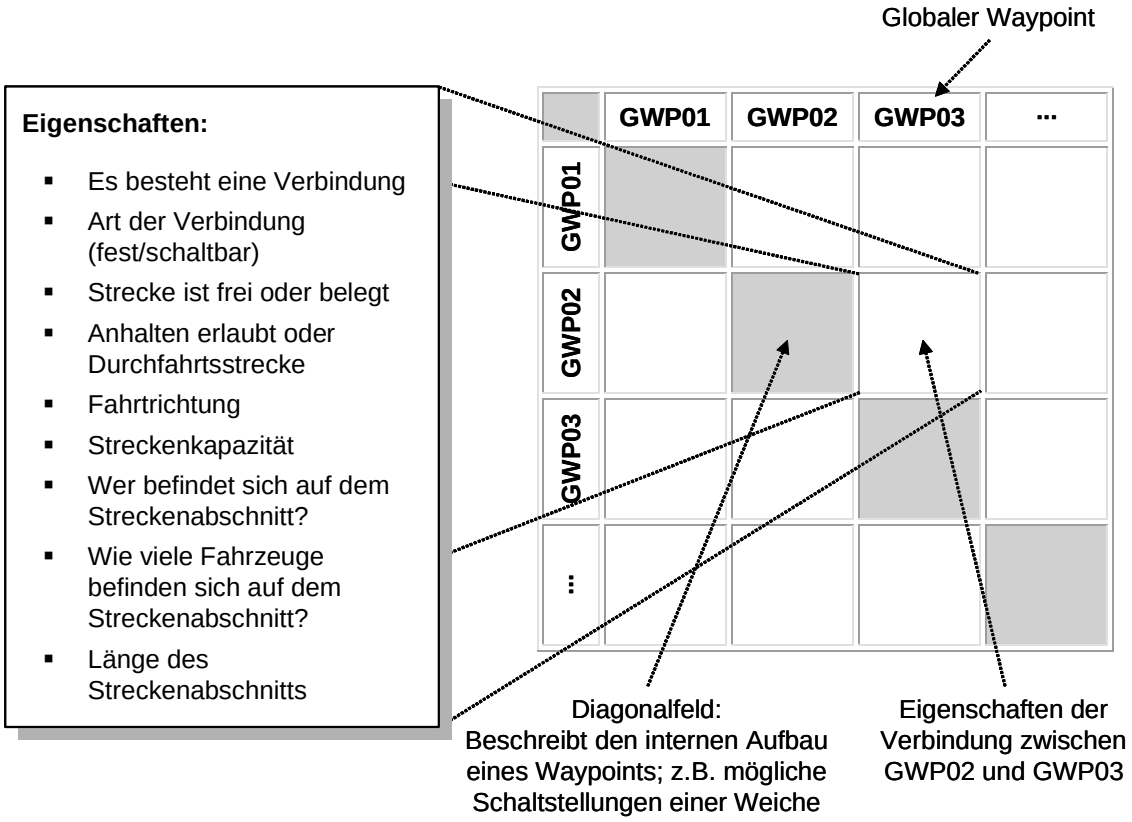


Abbildung 5-12 Struktur und Eigenschaftsfelder einer Topologiematrix [Wil-06]

Diese Datenstruktur bildet sowohl die statische Topologie als auch den dynamischen Anlagenzustand ab. Da es sich hierbei um systemweit relevante Informationen handelt, deren Konsistenz und Aktualität zu jedem Zeitpunkt gewährleistet werden muss, werden diese über ein Blackboard allen Entitäten im System zur Verfügung gestellt.

5.2.2 Verteilung der Routinglogik

Zur Durchführung der Wegplanung in einem Netzwerk aus intelligenten Transporteinheiten und Fördertechnikmodulen lassen sich zwei unterschiedliche Prinzipien anwenden. Zum einen ist es denkbar, die Wegplanung dem Aufgabenbereich der Transporteinheit zuzuschreiben. Ähnlich einem Autofahrer wäre dabei die Transporteinheit für die Suche und Bewertung von Wegen zuständig und würde jedes Modul mit dem Transport von dem Eingangs- zu einem Ausgangspunkt des Moduls beauftragen. Zum anderen können Wegentscheidungen, wie auch im Internet, von der Transportinfrastruktur bzw. der Fördertechnik getroffen werden. Die Transporteinheit selbst verhält sich dabei eher passiv und gibt nur Auskunft über ihr aktuelles Ziel – entweder durch Kommunikation mit den Modulen oder durch das Mitführen dieser Daten auf einem RFID-Chip, der an entscheidenden Stellen im Materialfluss ausgelesen wird.

Aus rein informationstechnischer Sicht lassen sich beide Möglichkeiten umsetzen. Da eine Transporteinheit anders als ein Datenpaket im Internet eine eigene Intelligenz bzw. einen eigenen Softwareagenten besitzt, könnte sie die Topologie des Systems auslesen, analysieren und eine detaillierte Wegplanung durchführen. In der Tat benötigen Transporteinheiten die Fähigkeit, Routen zu berechnen und zu bewerten, da sie sonst nicht in der Lage sind, das für ihre aktuelle Aufgabe bzw. den aktuellen Workflowschritt günstigste Zielmodul zu bestimmen (siehe auch Abschnitt 5.1.1). Wird aber den Transporteinheiten die Wegplanung im Sinne der Materialflusssteuerung übertragen, entspricht dies eher einer zentralen Steuerungsarchitektur. Denn die Transporteinheiten müssten dabei nicht nur die Topologie, sondern auch alle Besonderheiten und Betriebsstrategien des Systems kennen und beachten. Der zentrale Materialflussrechner würde somit durch eine Vielzahl, an einzelne Transporteinheiten gekoppelte, Materialflussrechneragenten mit ähnlichem Umfang und ähnlicher Komplexität ersetzt werden. Vorteile gegenüber aktuellen Systemen,

z.B. in Bezug auf Wiederverwendbarkeit oder eine reduzierte Komplexität, sind bei diesem Vorgehen daher nicht oder nur in sehr geringem Maße zu erwarten.

Die eigentliche Durchführung des physikalischen Transports, sowie die Auswahl von Routen unter Berücksichtigung zusätzlicher Faktoren und Strategien (siehe Abschnitt 5.2.3), ist daher auf die Fördertechnikmodule zu verlagern. Die grundlegenden Mechanismen zur Routenplanung können dabei in eine Grundklasse der Softwarearchitektur (z.B. in die Klasse Modul, siehe Abschnitt 3.4.2.2) verlagert und bei Bedarf in den abgeleiteten Agenten verändert oder verfeinert werden. Auf diese Weise ist auch eine anforderungsgemäße, stufenweise Erweiterung der Wegplanungsmechanismen möglich. Während für eine Verzweigung in einem Stetigförderernetz die Erstellung einfacher Routingtabellen aus einer Topologiebeschreibung genügt, müssen beispielsweise im selben Bereich verkehrende Unstetigförderer zusätzliche Faktoren wie Reservierungen beachten.

Die Routenplanung der Transporteinheiten hingegen kann in einer sehr einfachen Form implementiert werden, die lediglich die Anlagentopologie sowie die aktuelle Auslastung oder eventuelle Störungen berücksichtigt.

5.2.3 Materialflussstrategien

Eine einfache Wegplanung im Sinne der Berechnung eines Pfades von einem Start- zu einem Zielpunkt auf Grundlage eines Graphen, der die Anlagentopologie abbildet, stellt keine besondere Herausforderung dar. Effiziente Algorithmen für diesen Zweck, wie z.B. der Dijkstra- [Dij-59] oder der Bellman-Ford-Algorithmus [For-56] [Bel-58], sind in der Informatik schon seit Jahrzehnten bekannt und werden auch im Bereich der Logistik für die Berechnung von Wegen eingesetzt [Arn-95]. Weiterhin ist es auch möglich, Protokolle und Algorithmen aus dem Bereich der Computernetzwerke und des Mobilfunks, wie z.B. das Routing Information Protocol, kurz RIP [RFC-1058] [RFC-1723], oder Dynamic Source Routing, kurz DSR [Joh-94] [RFC-4728] mit einigen Anpassungen auf Materialflusssysteme zu übertragen und für die Planung von Transportrouten zu nutzen [Hom-08] [Nag-08].

Mindestens genauso wichtig, aber im Hinblick auf eine dezentrale Steuerung eines Systems weitgehend unbehandelt, sind Randbedingungen und besondere funktionale Anforderungen, die sich auf die Routenplanung auswirken. In einem zentral ge-

steuerten System können Aufgaben wie die Sequenzierung und Sortierung von Transporteinheiten oder die Koordination von Warenströmen in Form fest definierter Strategien hinterlegt und von einem Materialflussrechner koordiniert und umgesetzt werden. Im Internet der Dinge müssen diese Funktionalitäten jedoch von autonomen, standardisierten Komponenten erbracht werden. Es stellt sich daher folgende Frage: Wie können höherwertige Funktionen eines Materialflusssystems nur durch dezentrale Entscheidungen und einfache Mechanismen, wie z.B. Service-Discovery oder Auktionen, realisiert werden?

Die Randbedingungen und Anforderungen, die sich auf die Wegplanung in einem Materialflusssystem auswirken und daher spezifische Materialflussstrategien erfordern, lassen sich folgendermaßen gliedern:

- **Konkurrenz** von Modulen: Oftmals befinden sich in einem System mehrere Module, die eine gewünschte Funktion erbringen können. Beispiele dafür sind der Einsatz mehrerer Verschiebewägen auf einer Schiene oder mehrerer EHB-Katzen in einem Elektrohängebahnsystem. Die erreichte Systemleistung hängt hier stark von der Auswahl des optimalen Kandidaten für einen konkreten Auftrag ab.
- **Aggregation** von Modulen: Höherwertige Funktionen wie z.B. das Sortieren von Transporteinheiten nach bestimmten Kriterien können häufig nur durch die Kooperation mehrerer Module umgesetzt werden. So besteht z.B. ein Sorter für Paletten oder ein Frühgepäckspeicher in einem Flughafen aus mehreren Pufferstrecken. Diese müssen sich untereinander abstimmen, um die gewünschte „aggregierte“ Funktion erfüllen zu können.
- **Physikalische Aspekte** der Anlage, die zu Abhängigkeiten zwischen Routen bzw. Warenströmen führen: Vor allem Module mit mehreren Lastaufnahmemitteln, die gleichzeitig Transporteinheiten mit mehreren Nachbarmodulen austauschen müssen, führen zu solchen Abhängigkeiten zwischen Materialflüssen, die bei einer Betrachtung nur der Anlagentopologie unabhängig und parallel erscheinen. Als Beispiel gilt ein Doppelverschiebewagen, der gleichzeitig Paletten von zwei Rollenförderern aufnimmt bzw. an diese abgibt.
- **Zeitliche Aspekte** wie sie für die Herstellung und Einhaltung von Sequenzen berücksichtigt werden müssen. Sollen z.B. mehrere Paletten in denselben

LKW verladen oder mehrere Artikel in denselben Behälter kommissioniert werden, so ist sicherzustellen, dass diese ihr Ziel zur gleichen Zeit und evtl. in einer fest definierten Reihenfolge erreichen.

5.2.3.1 Konkurrenz

In vielen Fällen können Ressourcen oder Aufträge nicht über eine eindeutige 1:1 Beziehung an Module vergeben werden. Dies kommt beispielsweise in Unstetigfördersystemen vor – dort stehen für einen Transportauftrag gleich mehrere Transportmittel wie Stapler oder EHB-Katzen zur Verfügung.

Ist der Transportauftrag auf dem Blackboard gespeichert oder wird er von einem Modul an potenzielle Auftragsbearbeiter verteilt, kann die Auftragsdisposition über eine Auktion abgewickelt werden (siehe Abschnitt 5.1.2). Entscheidend für die Leistung und auch Auslastung des Gesamtsystems ist dabei die gewählte Kostenfunktion bzw. die Methode, anhand derer die mitbietenden Module ihre Angebote berechnen. Im einfachsten Fall kann dabei eine Abschätzung der Entfernung – und damit der Fahrzeit – vom Modul zur zu transportierenden Einheit als Angebot genutzt werden. Darüber hinaus können aber auch Faktoren wie Auslastung des Moduls, Streckenbelegungen oder auch vom Modul gemachte Erfahrungswerte – beispielsweise in Form aufgezeichneter Statistiken über Fahrzeiten oder Staus – in die Berechnung einfließen.

Grundsätzlich lässt sich dabei die Annahme treffen, dass mit der Anzahl der Faktoren, die in der Kostenfunktion berücksichtigt werden, die tatsächliche Fahrzeit – die für die Systemleistung entscheidend ist – besser abgeschätzt werden kann. Dabei muss aber berücksichtigt werden, dass gerade in einem dezentralen System die Beschaffung aller dazu notwendigen Informationen mit sehr großem Kommunikations- und Zeitaufwand verbunden sein kann. Zudem steigen mit der Komplexität der Kostenfunktion und der Menge der dabei auszuwertenden Informationen auch die algorithmische Komplexität und damit die Rechenzeit.

Die Gestaltung einer Kostenfunktion ist aus diesen Gründen ein komplexes Problem, für das es wahrscheinlich keine allgemeingültige Lösung geben kann. Je nach Eigenschaften und Anforderungen eines Systems lassen sich – im Rahmen der verfügbaren Kommunikations- und Rechenleistungen – verschiedene Lösungen entwi-

ckeln, die dann simulativ im Hinblick auf die resultierende Systemleistung untersucht werden müssen.

5.2.3.2 Aggregation

In herkömmlichen, zentral gesteuerten Materialflusssystemen lassen sich Systemgrenzen fast beliebig ziehen, Steuerungsalgorithmen und -hardware können je nach Bedarf für beliebig große Anlagenbereiche und komplexe Funktionen neu programmiert oder ausgelegt werden. Dies ist vor allem dann von Vorteil, wenn mehrere einfache mechanische Komponenten so zusammenspielen müssen, dass eine logistisch höherwertige Aufgabe erfüllt wird.

Eine in der Intralogistik häufig auftretende Aufgabe ist die Sortierung von Transporteinheiten nach verschiedenen Kriterien, wie z.B. dem Zielort oder den Produkteigenschaften. Häufig werden die unsortiert eintreffenden Transporteinheiten über eines oder mehrere Verzweigungselemente auf Pufferbahnen verteilt, sodass alle Transporteinheiten einer Kategorie hintereinander auf einer Bahn liegen. Das sortierte Stückgut kann dann bei Bedarf aus dem Sorter ausgeschleust und zum nächsten Zwischenziel weitertransportiert werden.

Ein Beispiel dafür ist das Zwischenspeichern von Frühgepäck am Flughafen. Zu früh eingecheckte Koffer werden dabei auf Pufferbahnen in einem Frühgepäckspeicher, auch Early Baggage Store oder EBS genannt, verteilt. Bei Flügen mit sehr vielen Passagieren kann eine solche Bahn für ein einziges Flugzeug reserviert werden, das Sortierkriterium ist somit der Zielort der Koffer. Genauso ist es möglich, eine Sorterbahn mehreren Flügen, die in einem gewissen Zeitfenster abfliegen und damit gleichzeitig beladen werden, zuzuordnen. Die Gepäckstücke werden dann also nicht nach dem Ziel, sondern nach der geplanten Verladezeit sortiert. Ähnliche Beispiele lassen sich auch in der Automobilindustrie bei der Herstellung von Karrosseriesequenzen oder in Kommissionierlagern bei der Vorsortierung von Paletten finden. In herkömmlichen Materialflusssystemen werden alle notwendigen Aufgaben, wie z.B. die Verwaltung der einzelnen Bahnen, die Zuordnung von Transporteinheiten zu einer Bahn sowie das Ausschleusen der Einheiten von einem zentralen System wie der Bereichssteuerung oder dem Materialflussrechner durchgeführt. Die Aggregation einfacher Funktionsmodule wie Förderer und Verzweigungen zu einer komplexeren Funktion geschieht also über einen zentralen Steuerungsalgorithmus.

Da das Internet der Dinge aber nur aus autonomen Modulen besteht und auf übergeordnete Logik verzichtet, sind Konzepte notwendig, die es solchen dezentralen Entitäten ermöglichen, aggregierte logistische Funktionen umzusetzen. Die folgenden Betrachtungen und Beispiele orientieren sich am konkreten Fall eines Early Baggage Stores, die vorgestellten Konzepte lassen sich aber auf andere Anwendungsfälle, z.B. in der Produktion oder Lagerlogistik, übertragen.

Das EBS-Szenario lässt sich grob in folgende Schritte gliedern:

1. Eine Transporteinheit bzw. ein Gepäckstück wurde am Check-In-Schalter aufgegeben, wurde in einer Röntgenanlage für sicher befunden und muss nun in ein bestimmtes Flugzeug verladen werden.
2. Ist der Flug bereits geöffnet, kann der Koffer direkt zum Flugzeug befördert werden.
3. Ist der Flug noch nicht offen, muss das Gepäckstück in einem EBS zwischengelagert werden. Ein EBS besteht aus mehreren Bahnen mit Pufferfunktion, die jeweils einem bestimmten Flug oder einem bestimmten Abflugzeitfenster zugeordnet sind.
4. Beim Öffnen des Fluges, bzw. eine gewisse Zeit davor, muss der Koffer aus dem EBS ausgeschleust und zum Ziel transportiert werden.

Im Internet der Dinge ist der Koffer durch seinen Softwareagenten intelligent und kann somit seine eigenen Belange selbst verwalten. Dazu gehört in erster Linie das Abarbeiten des anlagenspezifischen Workflows, wie gerade beschrieben. Die Instanziierung und Initialisierung des TE-Agenten sowie die datentechnische Darstellung des Workflows wurden bereits in Abschnitt 0 beschrieben. Hier soll daher hauptsächlich auf die Aspekte der Materialflusssteuerung eingegangen werden.

Neben den TE-Agenten und dem Blackboard, das die Anlagentopologie und einen Plan mit den Abflugzeiten der einzelnen Flieger bereitstellt sowie einem Verzeichnisdienst, befinden sich im System vier verschiedene Modultypen:

- Förderer und Weichen für den Transport der Koffer,
- Gates, die zu der im Flugplan angegebenen Zeit (oder bei Benachrichtigung von einem externen System oder Bediener) einen Flug als offen bekannt geben und dies in den Verzeichnisdienst eintragen,

- EBS-Bahnen.

Soll ein Gepäckstück verladen werden, so muss dieses über einen Service-Discovery Mechanismus das Gate suchen, an dem das entsprechende Flugzeug zum Verladen bereit steht. Dies kann, wie in Abschnitt 5.1 dargestellt, auf unterschiedliche Weise geschehen. Falls kein Gate gefunden wird, muss der TE-Agent die nächste vom Workflow vorgegebene Möglichkeit ausprobieren, in dem Fall eine EBS-Bahn.

Die einzelnen Bahnen sind voneinander völlig unabhängig und müssen nicht wissen, wie viele EBS-Bahnen im System noch vorhanden sind oder für welchen Flug bzw. welches Zeitfenster diese reserviert sind. Eine EBS-Bahn bietet als Dienst die Zwischenpufferung von Gepäckstücken, die in einen bestimmten Flug oder zu einer bestimmten Zeit verladen werden. Diese Information kann wiederum entweder in einem für die TE-Agenten einsehbaren Verzeichnis veröffentlicht oder auch nur intern verwaltet werden, wenn die EBS-Bahn selbst auf Dienstanfragen auf dem Blackboard reagieren soll.

Da für ein Gepäckstück im Normalfall mehrere EBS-Bahnen als Ziel in Frage kommen, müssen die Kosten für das Zwischenlagern so gestaltet sein, dass sich eine optimale Ausnutzung der vorhandenen Bahnen ergibt. Die Funktion *k* (siehe Formel 5-1) erlaubt die lokale Berechnung der Kosten zur Aufnahme einer Transporteinheit *TE* durch eine Bahn.

$$k(TE) = \begin{cases} 0, \text{ wenn } b \triangleq z(TE) \text{ und Bahn nicht voll} \\ 1, \text{ wenn } b \triangleq t(TE) \text{ und Bahn nicht voll} \\ 2, \text{ wenn } b = \emptyset \text{ und Bahn reservierbar für } z(TE) \text{ oder } t(TE) \\ \infty, \text{ sonst} \end{cases}$$

Formel 5-1 Kostenfunktion für einen einfachen Sorter

Dabei sind:

- *b* die aktuelle Belegung der die Funktion berechnenden Bahn (für eine freie Bahn sei $b = \emptyset$)
- *z(TE)* das Ziel einer Transporteinheit *TE* bzw. der Flug, in den ein Koffer verladen werden soll
- *t(TE)* das Abflugzeitfenster einer Transporteinheit *TE*.

Hat eine Transporteinheit Angebote von mindestens einer Bahn erhalten, kann sie die günstigste Alternative auswählen. Gibt es mehrere günstigste Alternativen, kann die Auswahl nach dem Zufallsprinzip erfolgen. Eine bisher freie Bahn ist ab jetzt dem Flug oder der Abflugzeit des Koffers zugeordnet und wird allen nachfolgenden Transporteinheiten andere, dem neuen Zustand entsprechende Kosten anbieten.

Hat die TE ihr Ziel erreicht und befindet sich auf der EBS-Bahn, kann sie wiederum versuchen, ein Gate zum Verladen zu finden. Wurde der Flug geöffnet und das neue Ziel von der Transporteinheit gefunden, erteilt die TE der Bahn, auf der sie sich befindet, einen Transportbefehl. Befindet sich am Ende der Bahn ein RFID-Schreibgerät, kann das neue Ziel, z.B. die Nummer des Verlade-Gates, auf den RFID-Tag des Gepäckstücks geschrieben werden. Anschließend wird das Gepäckstück gemäß der neuen Zielangabe von der Anlage zum Gate befördert.

Dieser Ablauf wird in Abbildung 5-13 in Form eines UML-Sequenzdiagramms erläutert. Angaben zur Validierung dieser Strategie anhand einer Emulation finden sich im Abschnitt 6.2.1.

Die EBS-Bahn verwaltet dabei eine Liste aller Transporteinheiten, die sich auf ihr befinden, und weiß somit, welche Koffer ausgeschleust werden müssen und welche noch nicht. Wenn sich Koffer mit mehreren Zielen bzw. Zielflügen auf einer Bahn befinden, kann es beim Ausschleusen dazu kommen, dass auch Gepäckstücke, deren Flug noch geschlossen ist, die Bahn verlassen müssen. Steht auf den RFID-Tags dieser Transporteinheiten noch die EBS-Bahn als Ziel, werden sie automatisch wieder in den EBS transportiert, ohne dass weitere Aktionen notwendig sind.

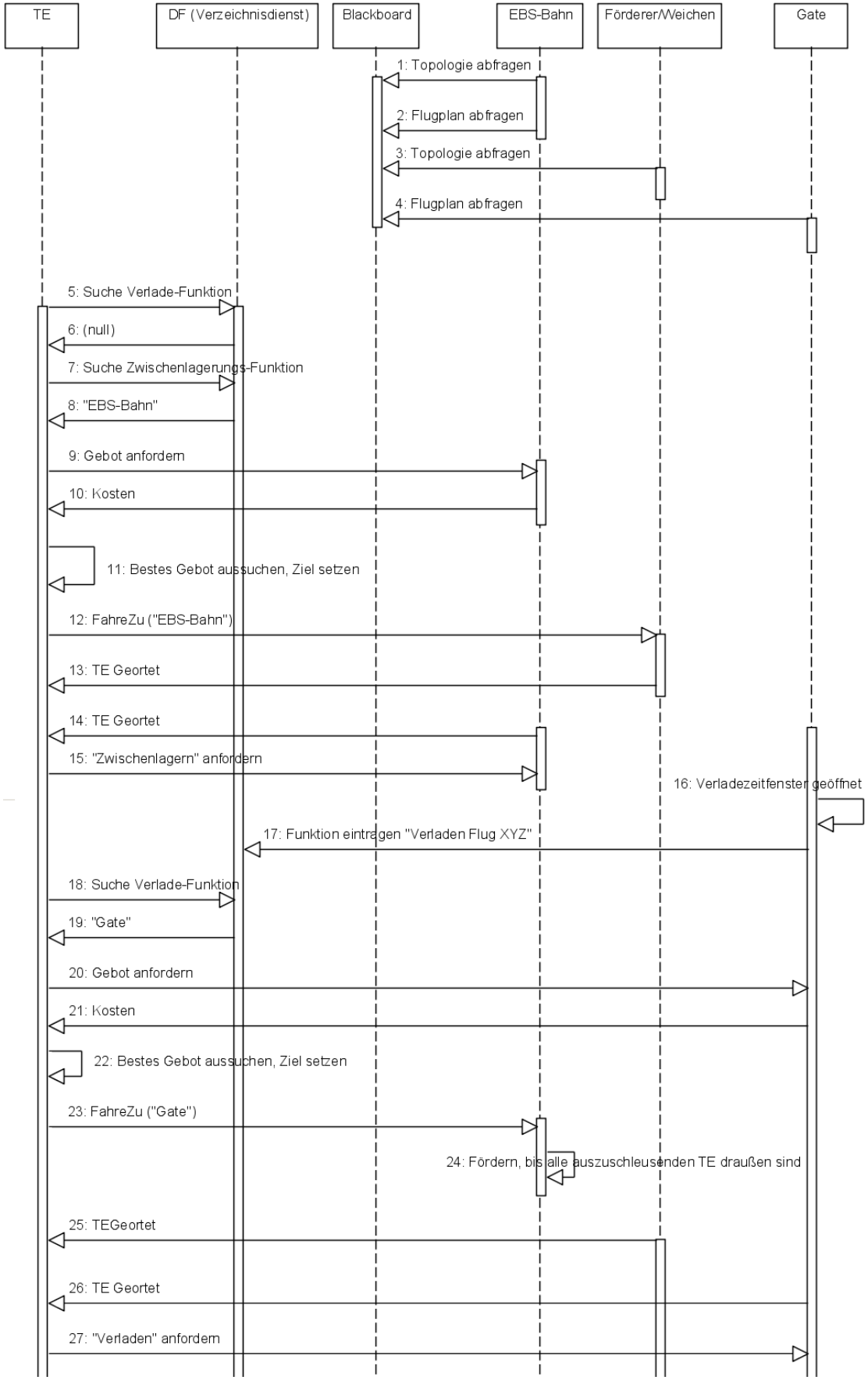


Abbildung 5-13 Steuerungsstrategie für einfachen Sorter bzw. EBS

5.2.3.3 Physikalische Aspekte

Um Module mit mehreren Lastaufnahmemitteln, wie z.B. ein Doppelverschiebewagen, optimal auszulasten, muss sichergestellt werden, dass die zu befördernden Transporteinheiten bereits auf den Zuführstrecken so vorsortiert werden, dass beide Lastwechsel gleichzeitig stattfinden können. Dies bedeutet, dass Transporteinheiten mit verschiedenen Zielen und unabhängigen Routen vom Materialflusssystem bei ihrem Transport automatisch so organisiert werden müssen, dass ein doppelter Lastwechsel möglich ist.

Als Beispiel soll folgendes Szenario aufgegriffen werden: Kartons, die von einem Roboter depalettiert werden, sollen in vier automatische Kleinteilelager eingelagert werden. Die Kartons werden nach dem Depalettieren in einem Sorter mit 6 Bahnen vorsortiert und danach an einen Doppelquerverschiebewagen abgegeben. Dieser führt die Kartons den AKLs zu. Um eine möglichst gute Auslastung des Verschiebewagens – und damit einen guten Durchsatz des Systems – zu gewährleisten, müssen schon bei der Verteilung der Kartons auf die Sorterbahnen die erwünschten Paare gebildet werden. So sollen im vorliegenden Beispiel Kartons mit Ziel AKL1 und AKL2 bzw. mit Ziel AKL4 und AKL3 auf benachbarte Bahnen verteilt werden.

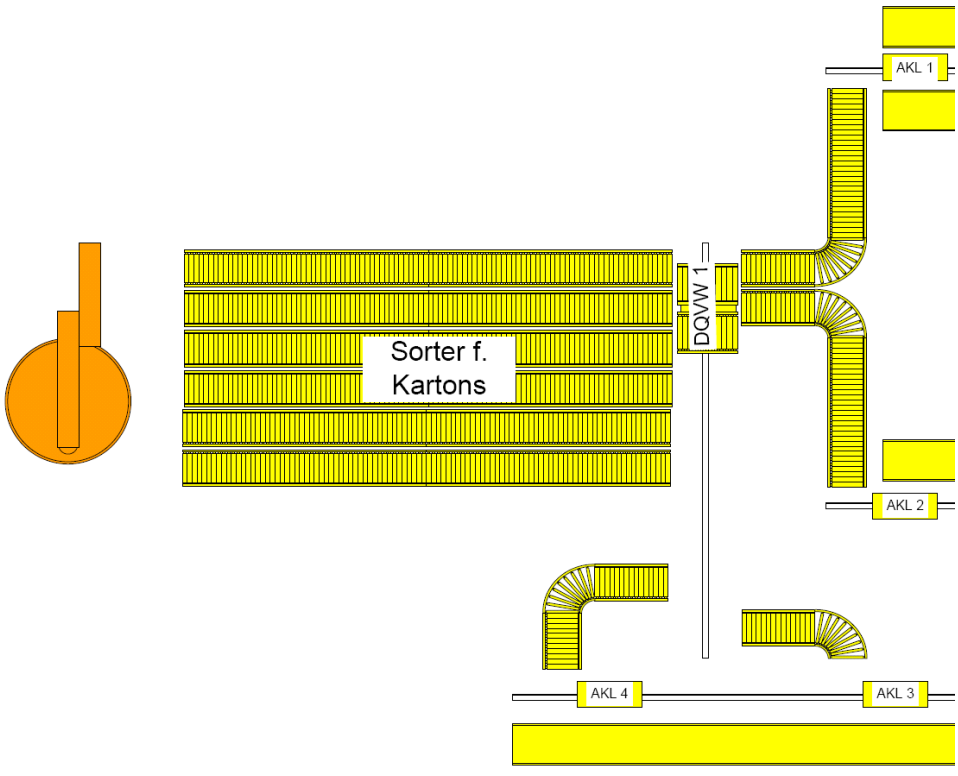


Abbildung 5-14 Sorter mit Doppelquerverschiebewagen

Wie im Szenario zum Frühgepäckspeicher sollen auch hier die einzelnen Bahnen des Sorters, die als autonome Module ausgeführt werden, für eine optimale Belegung und auch die Realisierung der gewünschten Paare sorgen.

Um diese Funktionalität umsetzen zu können, müssen die Agenten der Sorterbahnen folgenden Informationen zur Verfügung haben:

- Sorterstruktur oder Nachbarschaftsbeziehungen zwischen den einzelnen Bahnen
- Zu realisierende Paare
- Aktuelle Belegung aller Bahnen

Die ersten zwei Punkte können als statische Daten angesehen werden, die sich nur bei einem Umbau oder einer Erweiterung der Anlage ändern und können somit entweder auf einem Blackboard oder auch in einfachen Konfigurationsdateien hinterlegt sein.

Die aktuelle Belegung des Sorters ändert sich jedoch dynamisch und muss daher zwischen den einzelnen Bahn ständig ausgetauscht und aktualisiert werden.

Ein Karton, der das System betritt, besitzt in diesem Fall einen Workflow aus zwei sequenziell abzuarbeitenden Schritten:

1. Vorsortieren_AKLx
2. Einlagern_AKLx.

Sucht eine Transporteinheit nach einem Modul für die erste Funktion, prüft jede Sorterbahn, ob und zu welchen Kosten sie den Karton aufnehmen kann. Die Kostenfunktion ist auch in diesem Fall der ausschlaggebende Faktor für das korrekte Funktionieren des Systems und muss die Paarbildung sicherstellen. Hat ein Koffer die günstigste Bahn gefunden und wurde er von dem Roboter auf diese gelegt, kann er sich um den zweiten Schritt, nämlich das Einlagern, kümmern. Dabei wird nur ein Modul als mögliches Ziel gefunden, nämlich das Ziel-AKL, und die Transporteinheit kann sich von der Sorterbahn, dem Doppelverschiebewagen und der anschließenden Fördertechnik zum Ziel transportieren lassen. Dass dabei ein weiterer Karton, gemäß der Paarbildung, gleichzeitig auf den Verschiebewagen übergeht, ist keiner der Transporteinheiten bekannt.

Die Funktion *k* (siehe Formel 5-2) die für jede Bahn die Kosten zur Aufnahme einer Transporteinheit *TE* berechnet, kann folgendermaßen gestaltet werden:

$$k(TE) = \begin{cases} 0, \text{wenn } b \triangleq z(TE) \text{ und } b_n \triangleq c(z(TE)) \\ 1, \text{wenn } b = \emptyset \text{ und } b_n \triangleq c(z(TE)) \\ 2, \text{wenn } b \triangleq z(TE) \text{ und } b_n = \emptyset \\ 3, \text{wenn } b = \emptyset \text{ und } b_n \neq \emptyset \\ 4, \text{wenn } b \triangleq z(TE) \text{ und } b_n \neq c(z(TE)) \text{ und } b_n \neq \emptyset, ODER \\ \quad \text{wenn } b \neq z(TE) \text{ und } b \neq \emptyset \text{ und } b_n \triangleq c(z(TE)), ODER \\ \quad \text{wenn } b = \emptyset \text{ und } \mathrm{c}(b_n) \neq \emptyset \text{ und } c(b_n) \neq z(TE) \\ 5, \text{wenn } b \neq \emptyset \text{ und } b \neq z(TE) \text{ und } b_n \neq \emptyset \text{ und } b_n \neq c(z(TE)) \\ \infty, \text{wenn Bahn voll} \end{cases}$$

Formel 5-2 Kostenfunktion für einen Sorter mit Paarbildung

Dabei sind:

- *b* die aktuelle Belegung der die Funktion berechnenden Bahn (für eine freie Bahn sei *b* =∅)
- b_n die Belegung einer Nachbarbahn
- *z(TE)* das Ziel der Transporteinheit *TE*
- *c(z(TE))* das Ziel, welches eine Transporteinheit haben muss, um mit *TE* ein Paar zu bilden

Dabei ist zu beachten, dass B_n und *p(TE)* jeweils zwei Mal auszuwerten und zu vergleichen sind – ein Mal für die linke und ein Mal für die rechte Nachbarbahn bzw. Transporteinheit.

Die Kostenfunktion erlaubt in dieser Form die Zuordnung von Transporteinheiten zu Bahnen auch dann, wenn keine Paare gebildet werden können. Dieser Fall, der mit höheren Kosten (4 bzw. 5) verbunden ist, gewährleistet, dass die maximale Kapazität des Sorters komplett ausgenutzt wird. Erst wenn alle Bahnen voll sind und die Kosten somit unendlich werden, können Transporteinheiten kein Zielmodul für ihren Workflowschritt mehr finden. Ist dies nicht erwünscht, z.B. weil die korrekte Paarbildung unter allen Umständen umgesetzt werden soll, kann die Kostenfunktion auch im fünften und sechsten Fall auf unendliche Kosten ausgelegt werden.

Das Aktivitätsdiagramm in Abbildung 5-15 bietet einen Überblick über das Vorgehen bei der Auswahl und Reservierung einer Sorterbahn entsprechend des vorgestellten

Prinzips. Zur Validierung dieses Vorgehens wurde eine Emulation durchgeführt (siehe Abschnitt 6.2.2)

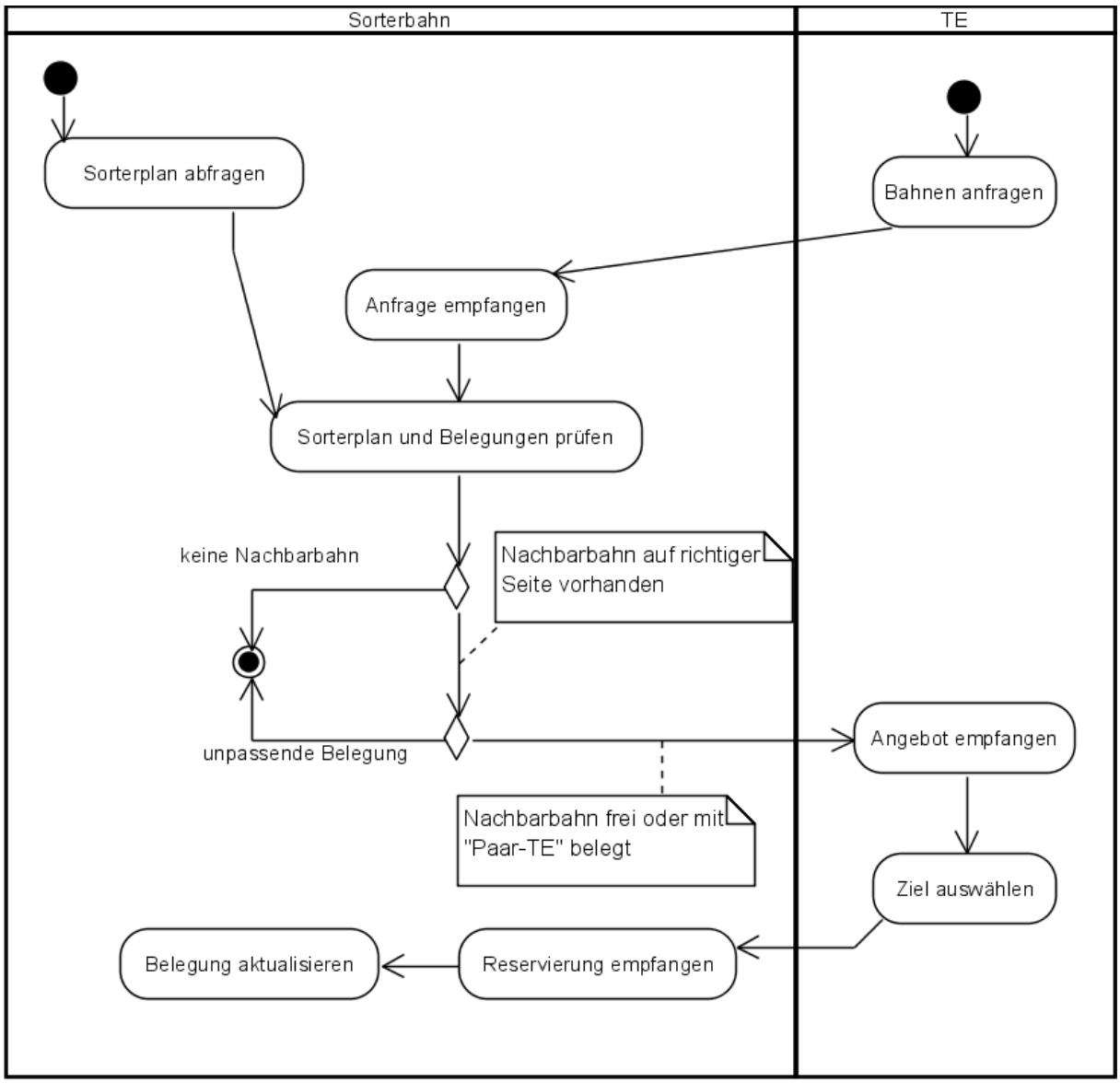


Abbildung 5-15 Steuerungsstrategie für Sorter mit Paarbildung

5.2.3.4 Zeitliche Aspekte

Auch wenn ein optimaler Transportweg für eine TE gefunden wurde, muss unter Umständen mit der Abarbeitung des Auftrags gewartet werden. Dies ist beispielsweise der Fall, wenn eine zu hohe Belastung eines Anlagenteils vermieden werden muss, um das Auftreten von Staus oder sogar Deadlocks zu verhindern. Dieses Problem lässt sich durch die Berücksichtigung von Kapazitätsgrenzen bei der Berechnung der Transportkosten und der Einführung einer oberen Schranke für akzeptable Angebote lösen. Sind die auf dem von der TE im Rahmen der Zielbestimmung

gefundenen Weg (siehe auch Abschnitt 5.1.1) befindlichen Module sehr stark ausgelastet, wird der Transport dem entsprechend teuer – überschreiten die Kosten nun einen vom Anlagenplaner vorgegebenen Schwellenwert, wird die Transporteinheit an ihrer Position verbleiben und zu einem späteren Zeitpunkt eine neue Wegplanung und Kostenbewertung durchführen.

Eine andere in der Logistik sehr wichtige Anforderung bezieht sich auf die Gruppierung mehrerer Transporteinheiten, die für die Abarbeitung eines Kommissionierauftrags erforderlich oder für den gemeinsamen Versand bestimmt sind. Die TEs müssen direkt hintereinander an ihrem Ziel ankommen – wobei unter Umständen auch eine feste Reihenfolge eingehalten werden muss. Funktional betrachtet handelt es sich auch hier um eine Form der Sortierung, ähnlich der in Abschnitt 5.2.3.2 bereits Beschriebenen. Der Unterschied besteht lediglich in den Kriterien, nach denen einer Transporteinheit ein Ziel – z.B. ein LKW-Verladetor oder ein Kommissionierbahnhof – zugewiesen wird. Der in Abbildung 5-13 beschriebene Ablauf kann damit in derselben Form für die Bildung von Sequenzen verwendet werden. Lediglich die modulinterne Logik muss dahingehend verändert bzw. erweitert werden, dass sie nur solchen Transporteinheiten ein Gebot unterbreiten, die zum aktuell bearbeiteten Kommissionierauftrag und evtl. einer bestimmten Auftragsposition gehören.

5.2.4 Transport und Lastwechsel

Der physikalische Transport kann im Normalfall ohne Zutun der Transporteinheit von den Modulen erledigt werden. Die Voraussetzung dafür ist jedoch, dass die zum Transport notwendigen Informationen, allem voran das Ziel der TE, den Modulen bekannt oder zugänglich sind. Ähnlich wie im Internet, wo ein Datenpaket neben den Nutzdaten auch Informationen über Absender und Empfänger der Nachricht enthält, muss im Internet der Dinge jede Transporteinheit relevante Informationen mit sich führen. Dafür kommen verschiedene Auto-Ident Technologien wie 1D- oder 2D-Barcodes, vor allem aber auch RFID in Frage. Der wichtigste Vorteil von RFID gegenüber anderen Alternativen liegt in der Wiederbeschreibbarkeit der Transponder. So können jedes Mal, wenn die TE ein neues Ziel oder Zwischenziel anfahren muss, die neuen Zieldaten auf den Tag geschrieben und – je nach Speicherkapazität – die bisherige Information verworfen oder zum Zwecke der Rückverfolgbarkeit von Abläu-

fen auf dem Transponder verbleiben. Da aber die Bandbreite von Anforderungen und Einsatzgebieten materialflusstechnischer Anlagen sehr groß ist und RFID auch gewisse Nachteile gegenüber den Barcodes aufweist – vor allem in Bezug auf die höheren Kosten und die hohe Empfindlichkeit gegenüber metallischen Umgebungen und/oder Flüssigkeiten – ist es nicht zielführend, das Internet der Dinge vom Einsatz einer bestimmten Technologie abhängig zu machen.

Als Minimalanforderung für die an der Transporteinheit mitzuführenden Informationen kann somit eine eindeutige Identifikationsnummer angesehen werden. Da sich diese im Verlauf des Transportes bzw. der Abarbeitung eines Workflows nicht ändert und die Ladeeinheit somit nicht neu beschriftet werden muss, können sowohl Barcodes als auch RFID-Tags oder andere Identifikationsmethoden verwendet werden.

Ein mit einem entsprechenden Lesegerät ausgestattetes Modul kann die so gespeicherten Informationen auslesen und, wenn notwendig, anhand der Identnummer über einen Verzeichnisdienst den TE-Agenten ausfindig machen und von diesem zusätzliche Daten wie z.B. das Transportziel erfragen. Dieses Vorgehen hat allerdings den Nachteil der nicht vorhersehbaren Kommunikationszeiten. Daher ist von diesem „Data-On-Network“ Prinzip beispielsweise auf sehr schneller Fördertechnik, wo während des Transports nach Lesen der TE-Nummer nur eine sehr kurze Zeitspanne für das Treffen einer Wegentscheidung zur Verfügung steht, abzusehen.

Das lokale, dezentrale Treffen von Entscheidungen und die damit einhergehende Notwendigkeit zur dezentralen Datenbeschaffung können zu einem deutlichen Kostenanstieg der Fördertechnik führen, wenn jedes Modul mit eigenem Barcode-Scanner oder RFID-Reader ausgestattet wird. Dies kann vermieden werden, indem Modulen die Möglichkeit gegeben wird, gelesene RFID-Daten an ihre Nachfolger weiterzugeben und ihnen somit die Ankunft einer bestimmten Transporteinheit im Voraus anzukündigen. In diesem Fall kann aber nicht mehr von einer festen Kopplung von Informations- und Warenfluss gesprochen werden, es handelt sich eher um eine Parallelisierung der beiden: Waren und Daten bewegen sich in derselben Richtung durch das System, die Informationen eilen aber der physikalischen Ladeeinheit voraus, bis sie am nächsten Modul, welches mit einem Auto-ID-Leser ausgestattet ist, wieder miteinander synchronisiert werden. Obwohl diese Methode zu einer Kostenersparnis führen kann, ist darauf zu achten, dass es in bestimmten Situationen zu Fehlfunktionen bzw. Fehlentscheidungen der Module kommen kann. Wird beispiels-

weise eine Transporteinheit durch einen Mitarbeiter manuell von einem Förderer entfernt, stimmen Informations- und Warenfluss nicht mehr miteinander überein: Das nächste Förderermodul erwartet aufgrund der bereits erhaltenen RFID-Daten nun eine Transporteinheit, die sich nicht mehr im Materialfluss befindet (siehe Abbildung 5-16). Da es keine Möglichkeit hat, erhaltene TEs zu identifizieren, wird es möglicherweise die nächste Ladeeinheit dem mittlerweile ungültigen Datensatz zuordnen und sie evtl. in eine falsche Richtung befördern. Ähnliches gilt für den Fall, dass eine Transporteinheit manuell an einer ungünstigen Stelle in den Materialfluss eingeschleust wird. Gerade an kritischen Entscheidungsstellen oder in Bereichen, in denen ein manueller Eingriff in den Warenfluss möglich ist, ist daher eine feste Kopplung von Waren und Daten durch das Anbringen von Identifikationsgeräten an den Modulen dringend erforderlich.

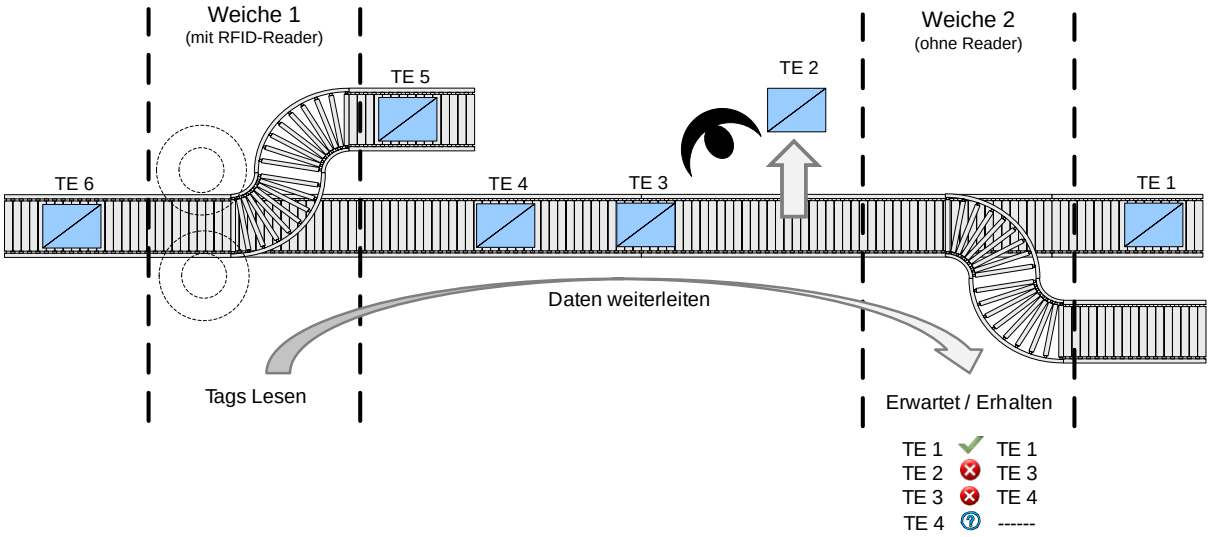


Abbildung 5-16 Entkopplung von Waren- und Datenfluss als mögliche Fehlerquelle

Die Ontologie des Internet der Dinge (siehe Abschnitt 4.1) sieht vor, dass TE-Agenten während des Transports – zumindest jedoch beim Erreichen ihres Transportziels – von den sie identifizierenden Modulen über ihren aktuellen Standort unterrichtet werden. Dies erlaubt es den TE-Agenten prinzipiell, ihren Transport zu überwachen. Werden sie mit der Fähigkeit ausgestattet, Transportzeiten abzuschätzen oder erhalten sie von den transportierenden Modulen eine Prognose über die zu erwartende Transportdauer, könnten sie bei deutlicher Überschreitung dieser Zeitspanne eine Warnung absetzen. So kann ein Anlagenbediener Transporteinheiten mit einer sehr hohen Priorität, die zufällig in einen Stau geraten sind oder deren Transportmodul ausgefallen ist, sofort erkennen und bei Bedarf veranlassen, dass

sie manuell vom defekten Modul entfernt und auf anderem Wege zum Ziel transportiert werden.

Während des Transports einer TE über mehrere Module hinweg kann es für die Fördertechnik notwendig werden, sich beim Lastwechsel abzustimmen. Diesem Umstand wird bereits durch die Basisontologie (siehe Abschnitt 4.1.1) und ihre lastwechselspezifische Erweiterung (siehe Abschnitt 4.1.2) Rechnung getragen. Bisher wurde jedoch nicht definiert, wie ein Modul erkennt, ob es sich mit seinem Nachfolger überhaupt abstimmen muss oder eine Ladeeinheit ohne Nachfrage weitergeben darf.

Zu diesem Zweck sind zwei unterschiedliche Methoden denkbar:

- Die Topologie der Anlage (siehe Abschnitt 5.2.1) wird so ergänzt, dass sie für jede Verbindung zwischen zwei Modulen beispielsweise anhand einer boolschen Variable angibt, ob die Lastübergabe koordiniert werden muss, oder nicht. Wenn ja, müssen die beteiligten Module dies über Kommunikation und dem in der Ontologie definierten Prädikat *ÜbergabeErlaubt* umsetzen.
- Jedes Modul trägt sich im Directory Facilitator bzw. Verzeichnisdienst des Agentensystems als Anbieter einer Funktion zur Lastwechselkoordination für die eigenen Übergabepunkte ein. Möchte ein Modul nun an einem bestimmten Übergabepunkt eine TE abgeben oder annehmen, prüft es zuerst, ob es andere Module gibt, die an diesem Ort eine Lastwechselkoordination anbieten. Ist dies nicht der Fall, kann der Lastwechsel sofort stattfinden.

5.3 Zentrale Steuer- und Beobachtbarkeit

Ein Logistik- bzw. Materialflusssystem erfüllt im Normalfall eine eher untergeordnete Aufgabe im Tätigkeitsbereich eines Unternehmens. Die Verwaltung der Aufträge und Lagerbestände, die Steuerung der Produktion sowie mittel- bis langfristige strategische und organisatorische Entscheidungen werden von Menschen oder von übergeordneten Systemen, wie z.B. ERP oder WMS getroffen. Da eine Dezentralisierung von Steuerungskompetenzen jenseits der Materialflusssteuerung derzeit nicht realistisch erscheint und nicht Gegenstand dieser Arbeit ist, muss auch das hier vorgestellte Netzwerk aus Modulen, Transporteinheiten und Diensten in der Lage sein, Befehle und Vorgaben solcher übergeordneter, zentraler Instanzen umzusetzen und auch die

für die übergeordnete Entscheidungsfindung notwendigen Daten zur Verfügung zu stellen.

Die wichtigsten Aufgaben und Vorgänge, bei denen ein Austausch von Befehlen und Informationen zwischen der Materialflusssteuerung und den übergeordneten Planungs-, Verwaltungs- und Kontrollebenen stattfinden müssen, sind:

- Auftragseinlastung
- Visualisierung
- Manuelles Erteilen von Befehlen bzw. indirektes Beeinflussen des Verhaltens von Entitäten.

5.3.1 Auftragseinlastung

Das Einlasten von Aufträgen in das Materialflusssystem bzw. die Vorgabe von Transportzielen oder komplexeren Workflows für Transporteinheiten geschieht auf Grundlage übergeordneter Entscheidungen, wie z.B. einer Produktionsplanung oder als Folge externer Faktoren wie dem Empfang einer Kundenbestellung. Diese Aufträge müssen nun an die dezentrale Materialflusssteuerung übergeben werden, was durch einen (oder mehrere) Dienste geschehen kann. Dabei muss ein solcher Dienst die beispielsweise vom LVS erhaltenen Aufträge als einen den TE-Agenten des Internet der Dinge verständlichen Workflow aufbereiten und die Transporteinheiten anschließend mit der Bearbeitung des Workflows beauftragen. Ebenso sind von den TEs erhaltene Erfolgs- oder Fehlermeldungen an das übergeordnete System weiterzuleiten. Ein Dienst zum Management bzw. der Einlastung von Transportaufträgen benötigt also eine recht einfache Funktionalität und ist im Grunde nur eine Schnittstelle zwischen der Materialflusssteuerung und übergeordneten Systemen.

5.3.2 Visualisierung

Komplexe Systeme müssen für den Anlagenbetreiber oder auch einen Inbetriebnehmer bzgl. ihrer Struktur und ihrem aktuellen Zustand nachvollziehbar sein. Denn nur so lassen sich beispielsweise Fehler oder Schwachstellen eines Systems schnell erkennen und beheben. Dabei ist gerade in einem dezentral gesteuer-

ten System auch die Aufzeichnung von Historien besonders wichtig, um allgemeine Vorgänge oder auch Störungen nachvollziehen zu können.

Visualisierungen oder Logger für einzelne Entitäten stellen keine besondere Herausforderung dar und sind heutzutage auf Microcontrollern, die als Steuerungshardware für Fördertechnikmodule einsetzbar wären, oftmals in Form eines parametrierbaren Webservers vorinstalliert.

Das Sammeln von Daten und deren grafische Aufbereitung für das gesamte System sind hingegen als komplexere Aufgabe anzusehen. Dabei müssen Informationen aus einem verteilten Netzwerk autonomer Einheiten zeitnah gesammelt, zeitlich geordnet und auf verschiedene Arten, z.B. textuell oder grafisch, dargestellt werden können. Eine zusätzliche Anforderung ergibt sich aus einem der wichtigsten Ziele des Internet der Dinge: der Wiederverwendung von Standardkomponenten. Dies bedeutet, dass auch eine Visualisierungsumgebung bzw. ein Visualisierungsdienst so universell gestaltet sein muss, dass möglichst ohne Änderungen am Programmcode verschiedenste Materialflusssysteme dargestellt werden können.

Das entwickelte Kommunikationskonzept (siehe Abschnitt 4.2 und 4.4) sieht bereits vor, dass systemweit relevante oder zu visualisierende Informationen, wie z.B. Status- oder Topologiedaten auf einem Blackboard gespeichert werden. Ein Visualisierungsagent kann nun über alle Veränderungen dieser Dateninhalte automatisch benachrichtigt werden und kennt auf diese Weise immer den aktuellen Zustand des Systems. Durch die dezentrale Architektur und die funktionale Entkopplung der Entitäten ist es dabei ohne weiteres möglich, mehrere Visualisierungsagenten an ein (oder mehrere) Blackboards zu koppeln. Auf diese Weise können mehrere identische oder auch verschiedene Visualisierungsumgebungen in einem System gestartet werden.

Da die vom Blackboard abgesetzten Benachrichtigungen mit einem Zeitstempel versehen werden, kann ein Visualisierungsdienst die empfangenen Nachrichten zeitlich sortieren und auf diese Weise eine Historie der auf dem Blackboard durchgeführten Veränderungen aufzeichnen, ohne dass eine Synchronisation der Uhrzeiten aller im System agierenden Entitäten notwendig ist. Die maximale Länge der Historie, beispielweise in Form eines Zeitfensters oder einer maximal zu speichernden

Anzahl an Nachrichten, kann wiederum in der Visualisierung eingestellt werden, ohne dass andere Agenten davon wissen müssen.

Der Visualisierungsdienst erzeugt für jede vom Blackboard erhaltene Information ein lokales Datenobjekt mit der entsprechenden Typbezeichnung (z.B. „Auftrag“, „Modulstatus“, „Wegpunkt“, etc.) und eindeutigen ID sowie dem aktuellen Zeitstempel. Alle weiteren Attribute dieses Datenobjektes werden als beliebig lange Liste von Schlüssel-Wert-Paaren gespeichert (z.B. Attribut[„Erfüllt“] = „False“, Attribut[„Ziel“] = „Modul_X“), wobei an dieser Stelle nur reine Zeichenketten vorgehalten werden und keine Konvertierung in beispielsweise ganzzahlige oder boolsche Werte erfolgt. Auf diese Weise ist die Speicherung und Verarbeitung fast beliebiger Dateninhalte möglich. Darüber hinaus besitzt jedes Datenobjekt weitere, für die grafische Darstellung relevante Eigenschaften wie Farbe oder Position (siehe Abbildung 5-17).

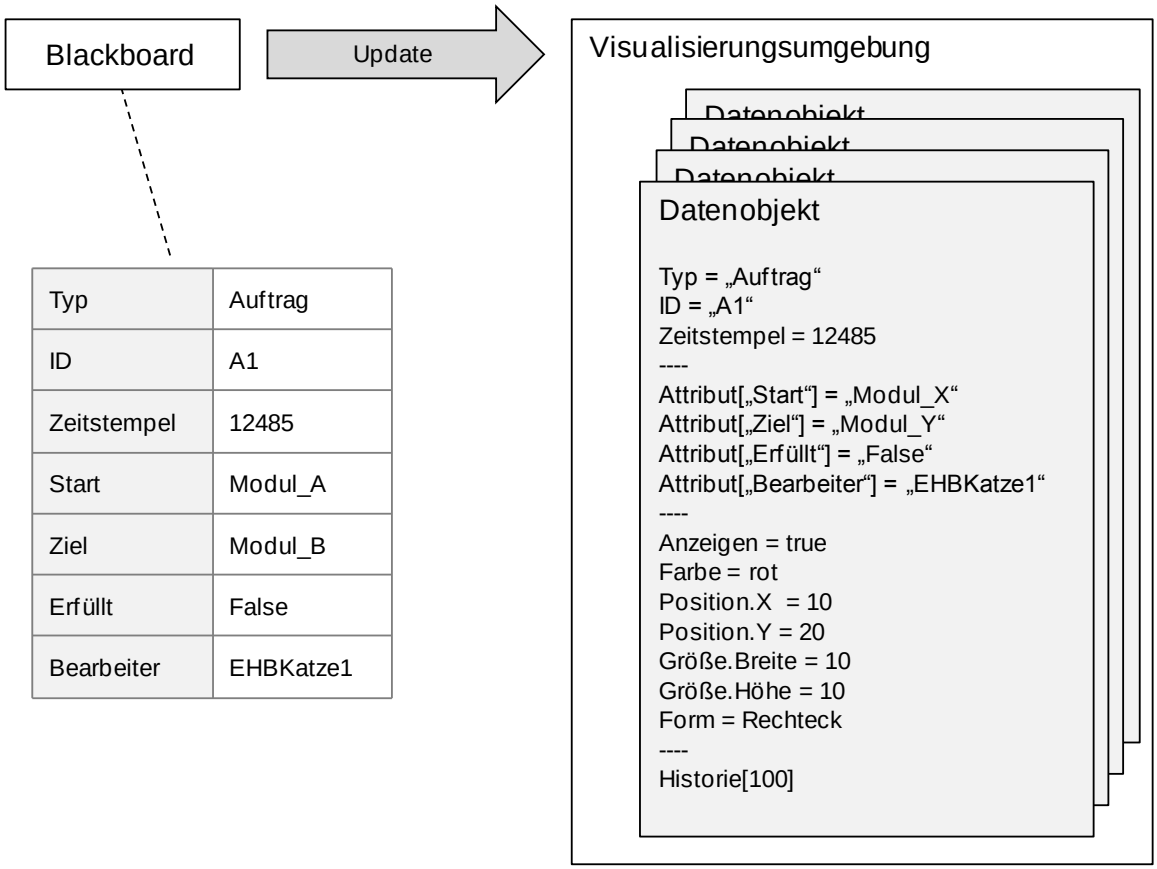


Abbildung 5-17 Speicherung der Daten in der Visualisierungsumgebung

Je nach Aufgabe sind für die Anzeige der so aufgezeichneten Daten verschiedene Visualisierungsstrategien sinnvoll, wie z.B.

- **Textausgabe** – hauptsächlich für die detaillierte Analyse umfangreicher Informationen.
- **Diagramme** – beispielsweise Linien-, Kreis- oder Balkendiagramme für die Anzeige von Veränderungen entlang der Zeitachse oder zum direkte Vergleich verschiedener Werte, wie z.B. der Auslastung verschiedener Module.
- **2D-/3D-Grafiken** – beispielsweise für die Anzeige des Gesamtsystems bzw. der Topologie und eines groben Überblicks der Zustände einzelner Module oder Transporteinheiten, wie z.B. des Auftretens eines Fehlerzustands, des aktuell durchgeführten Auftrags oder der Position.

Neben der grafischen Ausgabe sind auch Möglichkeiten für ein Versenden von Nachrichten an einzelne Entitäten sinnvoll, beispielweise um den Betriebszustand eines Moduls manuell zu ändern oder Fehler zu quittieren. Im Rahmen dieser Arbeit wurde eine Visualisierungsumgebung entwickelt, die die oben aufgeführten Anforderungen erfüllt.

Die Darstellung im Textformat zeigt einzelne Datenobjekte und deren Unterelemente in einer Baumstruktur an (siehe Abbildung 5-18). Ebenfalls kann die gesamte Änderungshistorie verfolgt werden, wobei für jede Veränderung der Daten der Absender, der genaue Inhalt und der Zeitpunkt, an dem diese Änderung auf dem Blackboard vorgenommen wurde, bekannt sind. Diese Darstellungsform ist vor allem für das Auffinden von Synchronisationsproblemen hilfreich, z.B. wenn verschiedene Module gleichzeitig eine Strecke oder einen Auftrag zu reservieren versuchen und auf diese Weise den voran gegangenen Dateninhalt überschreiben.

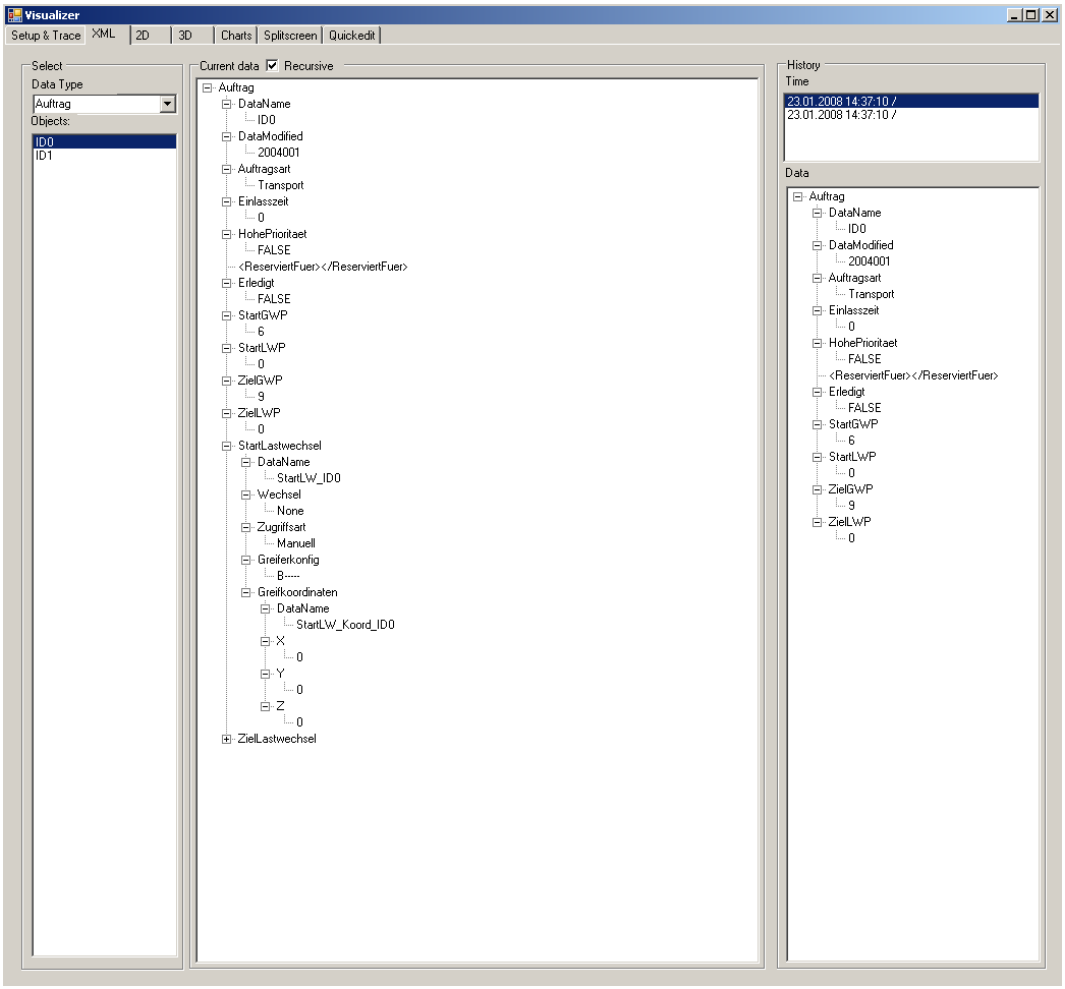


Abbildung 5-18 Visualisierung - Baumansicht mit Historie für einzelne Datenobjekte

Mittels verschiedener Kreis- und Liniendiagramme können sowohl bestimmte Zusammenhänge im aktuellen Systemzustand, beispielweise der Anteil freier oder mit der Bearbeitung von Aufträgen beschäftigter Module als auch Veränderungen entlang der Zeit dargestellt werden (siehe Abbildung 5-19), so z.B. dynamische Veränderungen von Streckenkosten oder die Zeitanteile, die ein Modul mit verschiedenen Aufgaben (wie z.B. Transportieren, Warten, Lastübergabe) verbracht hat. Der Benutzer kann dabei für verschiedene Dateninhalte unterschiedliche Diagrammtypen frei definieren, ähnlich zu gängigen Tabellenkalkulationsprogrammen.

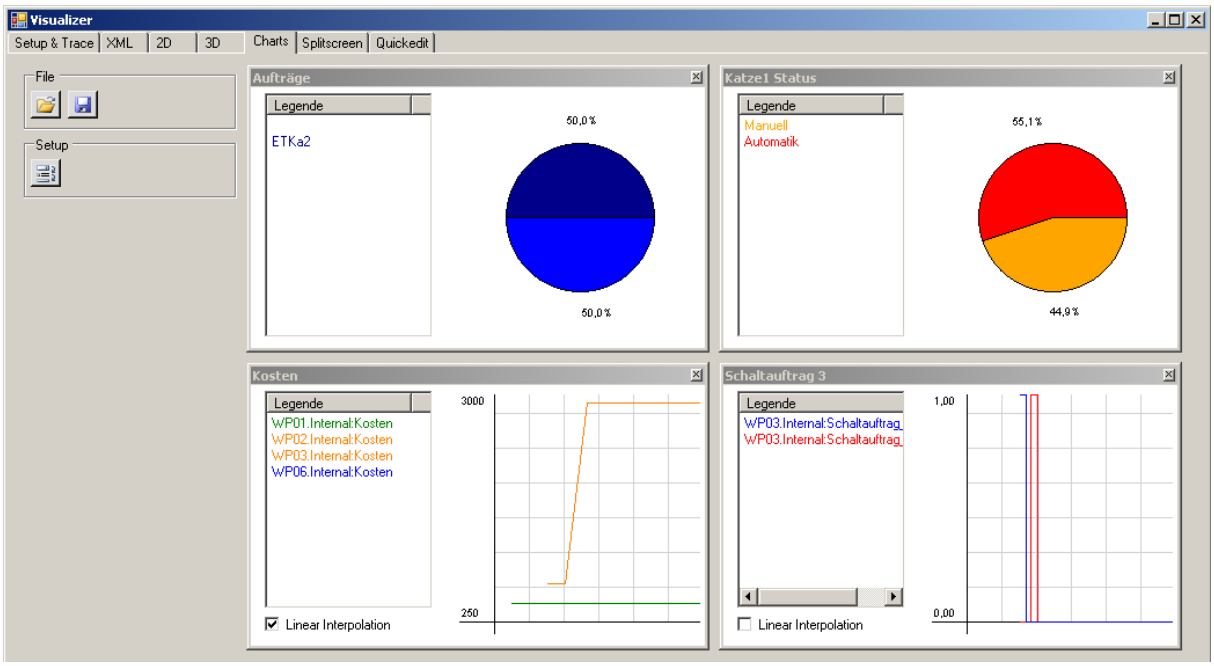


Abbildung 5-19 Visualisierung - Diagramme

Die grafische 2D- und 3D-Darstellung der Anlage erfolgt regelbasiert. Dabei kann der Benutzer des Programms jedem Datenobjekt beliebig viele Visualisierungsregeln zuweisen, die festlegen, wie das Objekt angezeigt wird und wie sich die Anzeige in Abhängigkeit seiner Attribute verändert. Bei der Interpretation der Attribute wird an dieser Stelle auch eine eventuell notwendige Typumwandlung durchgeführt, um beispielsweise die Zeichenkette „1234“ in den ganzzahligen Wert 1234 umzuwandeln und so für weitere Berechnungen oder Vergleiche nutzen zu können. Es stehen folgende Visualisierungsregeln bzw. -funktionen zur Verfügung:

- Statische Anzeige eines Datenobjekts als geometrische Form (Linie, Kurve, Quadrat oder Kreis), Text oder 2D-Bild bzw. 3D-Modell an einer festen Position im 2D-/3D-Raum und mit einer festen Größe und Farbe.
- Dynamisches Anzeigen bzw. Verstecken eines Datenobjekts in Abhängigkeit der Werte seiner Attribute.

  Beispiel:

  WENN Datenobjekt.Typ = „TE_Status“ UND Datenobjekt.Attribut [„Workflow-Erfüllt“] = „true“ DANN Datenobjekt.Anzeigen = false
- Dynamisches Ändern der Farbe, der Größe oder des Bildes eines Datenobjekts in Abhängigkeit seiner Attribute.

  Beispiel 1:

WENN Datenobjekt.Typ = „Auftrag“ UND Datenobjekt.Attribut [„Erfüllt“] = „true“ DANN Datenobjekt.Farbe = grün

Beispiel 2:

WENN Datenobjekt.Typ = „Stauplatz_Status“ UND Datenobjekt.Attribut [„Status“] = „belegt“ DANN Datenobjekt.Bitmap = „StauplatzMitPalette.bmp“ SONST Datenobjekt.Bitmap = „StauplatzFrei.bmp“

- Neuberechnung der Position eines Datenobjekts im 2D-/3D-Raum auf Grund seiner Datenfelder, relativ zum Ursprung des Koordinatensystems oder relativ zu einem anderen Datenobjekt.

  Beispiel 1:

  WENN Datenobjekt.Typ = „EHB-Kran“ DANN Datenobjekt.Position.X = integer (Datenobjekt.Attribut[„Position_Fahrwerk“]) / 100

  Beispiel 2:

  WENN (Datenobjekt.Typ = „EHB_Katze“ UND integer(Datenobjekt.Attribut [„Position_Fahrwerk“]) >= 0 UND integer (Datenobjekt.Attribut [„Position_Fahrwerk“]) < 9500) DANN Datenobjekt.Position.X = Datenobjekt („EHB_Kran“).Position.X; Datenobjekt.Position.Y = Datenobjekt („EHB_Kran“). Position.Y + integer(Datenobjekt.Attribut[„Position_Fahrwerk“]) / 100

Die vom Benutzer definierten Regeln werden zyklisch auf die vom Blackboard erhaltenen Daten bzw. gespeicherten Datenobjekte angewendet und die Anzeige so ständig aktualisiert (siehe Abbildung 5-20).

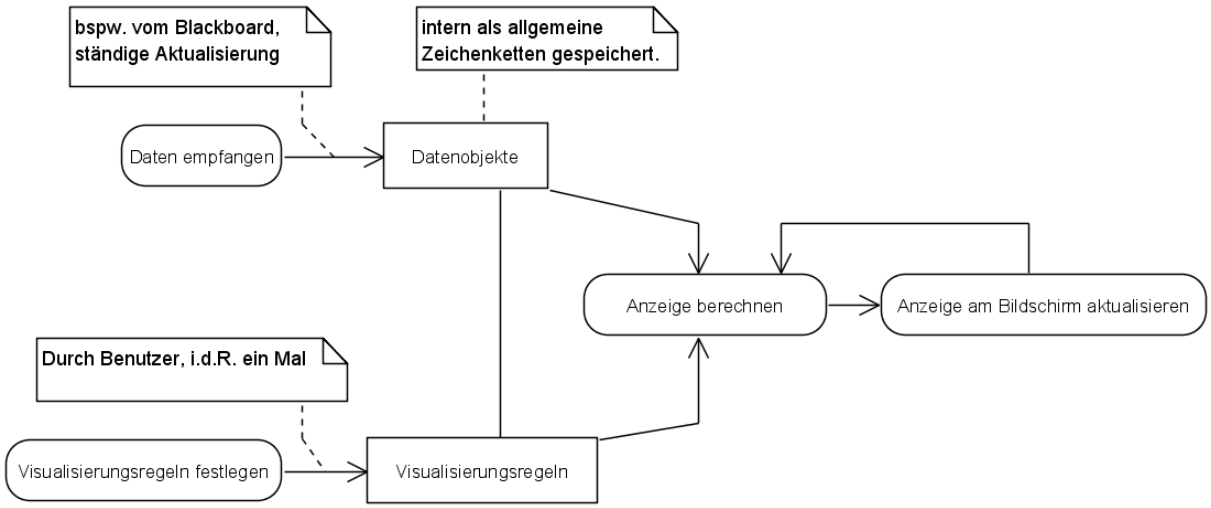


Abbildung 5-20 Interne Funktionsweise der Visualisierungsumgebung

Der Visualisierungsagent ist bzgl. der Regeldefinition und der Verarbeitung von Dateninhalten, ähnlich wie das Blackboard selbst, vollkommen flexibel und frei von jeder projekt- oder anlagenspezifischen Logik oder auch Ontologie. Auf diese Weise lassen sich verschiedenste Systeme mit einem einheitlichen Tool visualisieren. Abbildung 5-21 zeigt eine 2D-Darstellung einer EHB-Versuchsanlage mit mehreren Streckenabschnitten, Weichen, Fahrzeugen und Kranfeldern und -brücken. Die vorgestellten Visualisierungsregeln reichen dabei aus, um die gesamte Anlage inkl. der Bewegung der Fahrzeuge und Krane sowie der aktuellen Weichenstellung anzuzeigen. Die Darstellung einer Flughafengepäckförderanlage mit demselben Visualisierungstool ist in Abbildung 6-14 zu sehen.

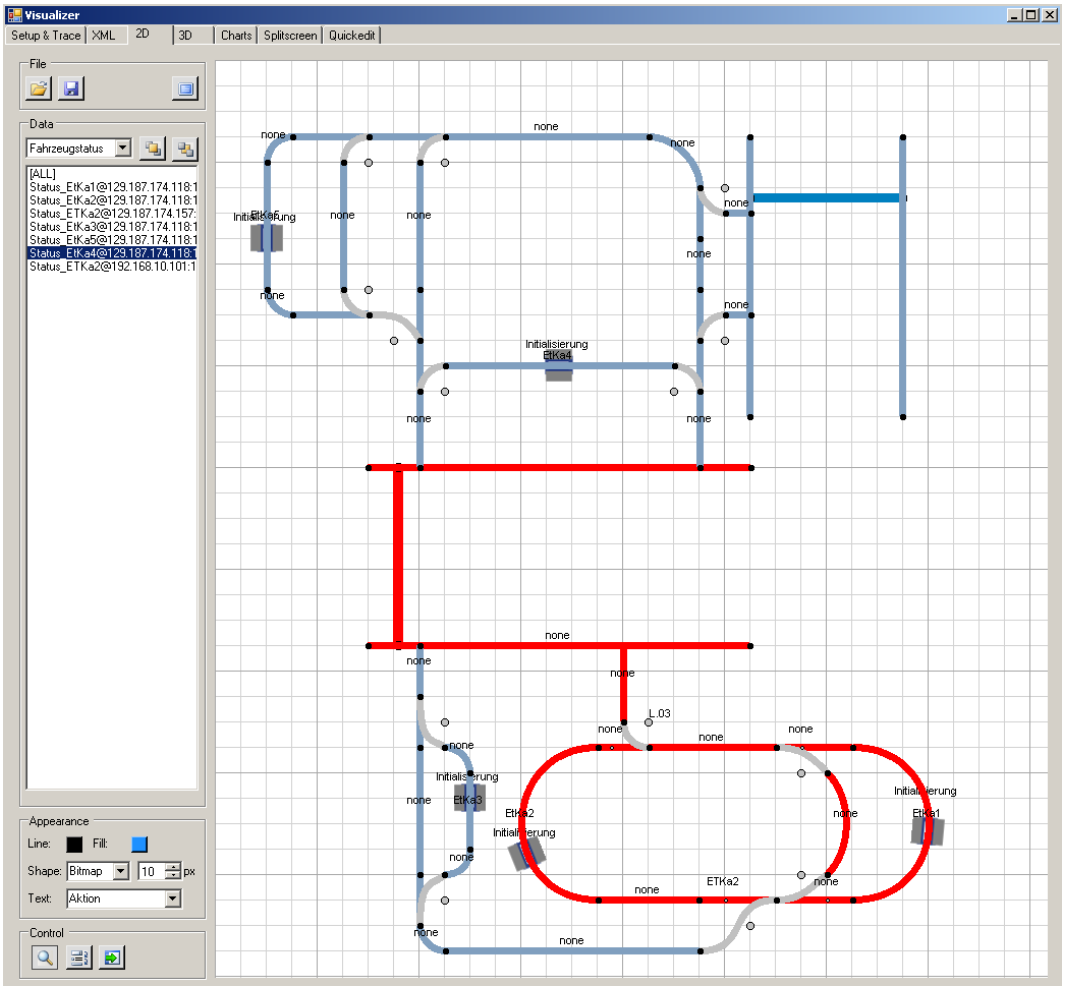


Abbildung 5-21 Visualisierung - 2D Ansicht einer EHB-Anlage

Im 2D-Modus definierte Visualisierungsstrategien lassen sich direkt für eine 3D-Anzeige übernehmen (siehe Abbildung 5-22). Die 3D-Anzeige basiert direkt auf dem OpenGL-Standard und besitzt dieselbe Visualisierungsfunktionalität wie der 2D-Modus.

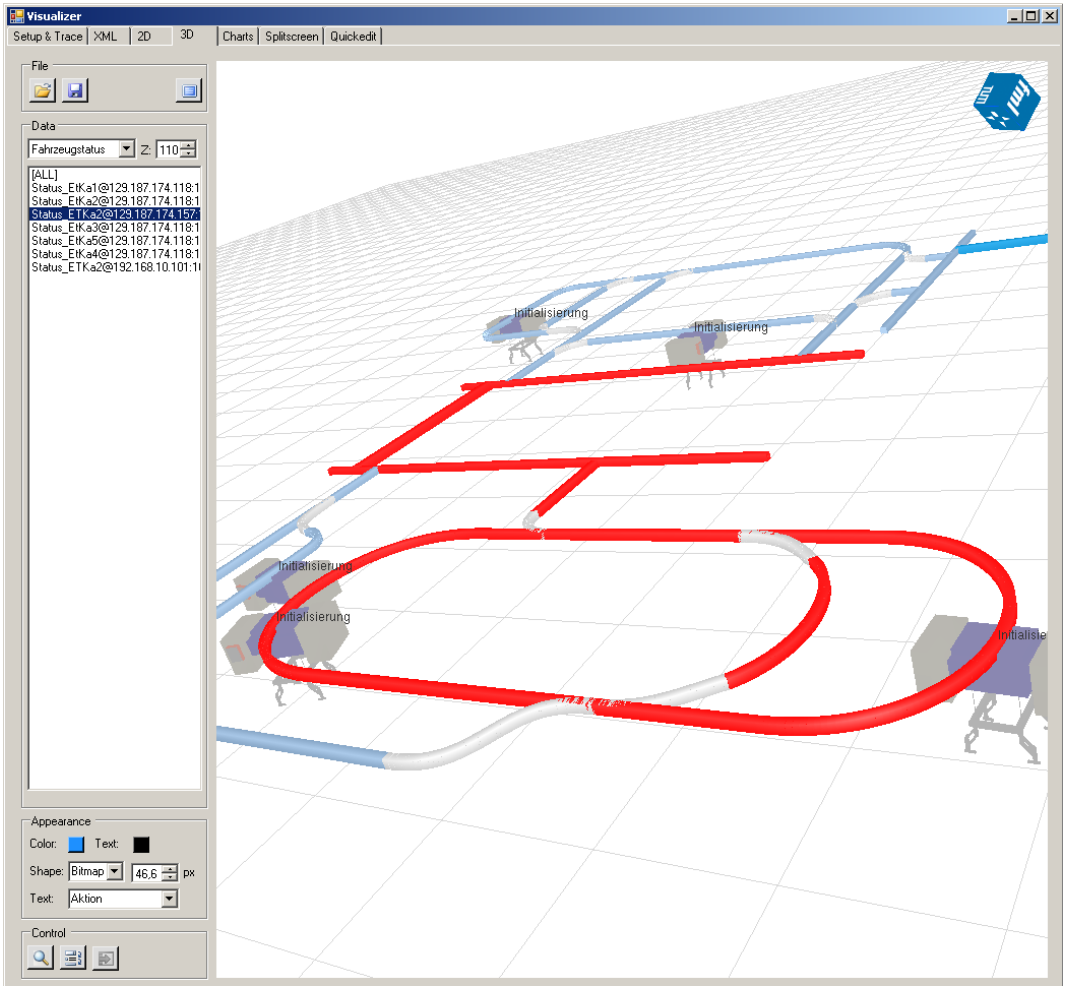


Abbildung 5-22 Visualisierung - 3D Ansicht einer EHB-Anlage

Neben der reinen Anzeige lassen sich aber auch interaktive grafische Elemente definieren, über die ein Benutzer mit wenigen Klicks beliebige Nachrichten ans Blackboard versenden kann, um so beispielsweise Streckenabschnitte manuell sperren oder Fehler quittieren zu können.

5.3.3 Manuelle Befehle, Editieren des Blackboards

Das manuelle Versenden von Befehlen an Entitäten im Internet der Dinge kann dann notwendig werden, wenn ein Anlagenbetreiber beispielweise auf Grundlage seiner Erfahrung oder seiner Kenntnis über zukünftige Ereignisse, wie z.B. dem baldigen Eintreffen einer großen Anzahl an Aufträgen, das autonome Verhalten von Modulen oder Transporteinheiten beeinflussen möchte. Da das Internet der Dinge für den automatisierten Betrieb angedacht ist, dürften manuelle Eingriffe in das System eher einen Ausnahmefall darstellen. Trotzdem müssen diese technisch machbar sein und von der Gesamtsystemarchitektur unterstützt werden. Eine andere Situation, in der

das manuelle Erteilen von Befehlen hilfreich ist, ist während der Entwicklungsphase bzw. bei der Emulation eines Systems. So können Meldungen von noch nicht vorhandenen Geräten, wie z.B. über das Identifizieren einer TE durch einen RFID-Reader, manuell vom Entwickler ausgelöst werden. Abbildung 5-23 zeigt einen solchen Dienst, der zwei Möglichkeiten zum Versenden von Befehlen, Strategievorgaben oder anderen Nachrichten bietet:

- durch direkte Nachrichten an einen oder mehrere Empfänger, z.B. zur Fehlerquittierung, zum Wechsel des Betriebsmodus oder zur Emulation von Ereignissen, also zur direkten Beeinflussung einer oder mehrerer Entitäten,
- durch Veränderung der Daten auf dem Blackboard, z.B. zur manuellen Veränderung von Streckenkosten oder zum Sperren von Wegen, also durch die Manipulation der „Umwelt“, in der die Entitäten des Internet der Dinge agieren, ohne auf die autonome Entscheidungsfindung der Einheiten einzuwirken.

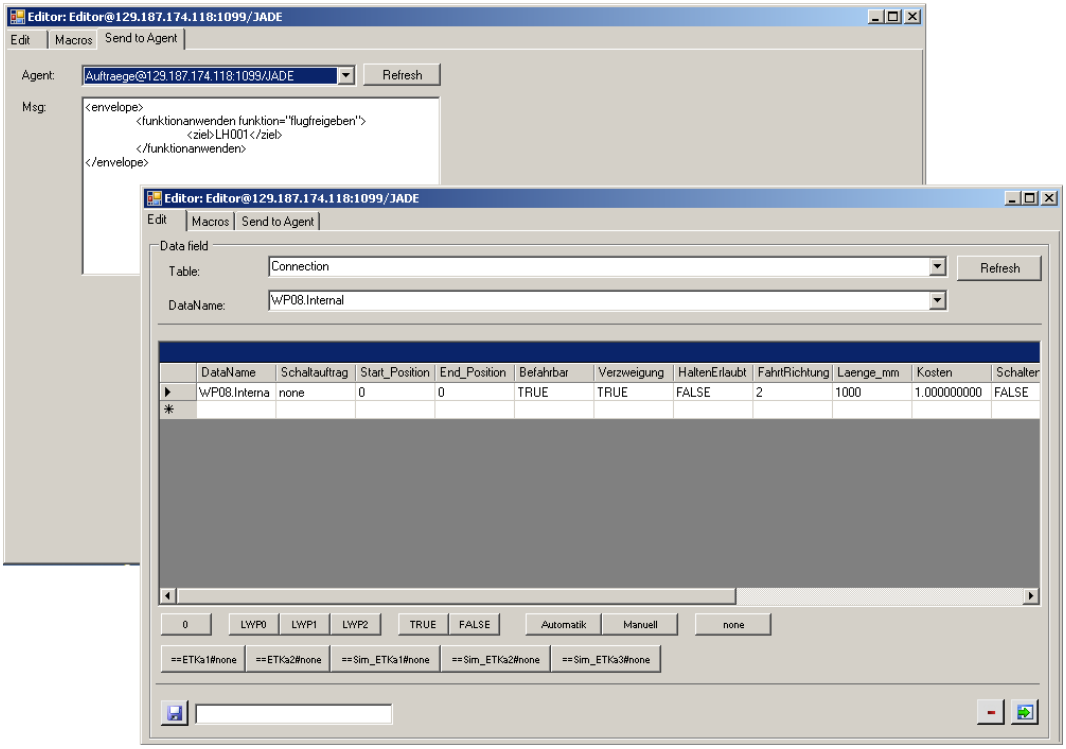


Abbildung 5-23 Dienst zum Versenden von Nachrichten und Bearbeiten der Blackboard-Daten

6 Realisierung

Als Grundlage für die Entwicklung und auch zur Validierung der hier vorgestellten Kommunikations- und Steuerungskonzepte diente zum einen die Versuchsanlage des Lehrstuhls für Fördertechnik Materialfluss Logistik fml der TU München. Aspekte wie z.B. die Steuerung von Frühgepäckspeichern oder Sortern, die in der Versuchsanlage nicht umgesetzt werden konnten, wurden in Form einer Emulation ohne mechanische Komponenten untersucht.

6.1 Umsetzung in der Versuchsanlage des Lehrstuhls fml

Für unterschiedliche thematische Schwerpunkte im Internet der Dinge wurden folgende Versuchsstände in der Anlage des Lehrstuhls genutzt:

- Roboterzelle mit Sechs-Achs-Knickarmroboter für die Identifikation von Paletten und Steuerung von Palettier- / Depalettiervorgängen mit RFID (siehe Abschnitt 6.1.2) sowie zur Demonstration hochflexibler Greiftechnik.
- Stetigförderer für Kleinladungsträger für die Entwicklung von Stetigfördereragenten und Mechanismen für die Lastübergabekoordination sowie die Übergabe von Transporteinheiten zwischen stetig und unstetig fördernden Subsystemen einer Anlage (siehe Abschnitt 6.1.3).
- Elektrohängebahnanlage (EHB) für die Entwicklung von Unstetigförderer- und Weichenagenten sowie Auktions- und Koordinationsmechanismen (siehe Abschnitt 6.1.4).

Diese Bereiche sind miteinander verbunden, sodass ein durchgehender Materialfluss realisiert werden konnte. Die gehandhabten Transporteinheiten – Europaletten, Kartons und VDA-Kleinladungsträger – wurden mit jeweils einem RFID-Chip gekennzeichnet, der sowohl eine eindeutige ID als auch weitere für die Materialflusssteuerung notwendigen Daten enthielt.

6.1.1 Einsatz von RFID

Die Transporteinheiten wurden mit für den UHF-Frequenzbereich ausgelegten passiven RFID-Transpondern gekennzeichnet. Die eingesetzten Transponder haben eine Speicherkapazität von 12 Byte bzw. 96 Bit und sind für die Speicherung des Elektronischen Produktcodes (EPC-96) geeignet.

Für den EPC sind mehrere Versionen bzw. Nutzungsmöglichkeiten vorgesehen. Diese für Deutschland von der GS1 Germany GmbH spezifizierten und verwalteten Identifikationssysteme erlauben die weltweit eindeutige Identifikation von Unternehmen (Globale Lokationsnummer GLN), Artikeln (Globale Artikelidentnummer GTIN), wiederverwendbaren Verpackungen (Global Returnable Asset Identifier GRAI) oder Versandeinheiten (Nummer der Versandeinheit NVE bzw. Serial Shipping Container Code SSCC) [GS1].

Im Internet der Dinge steht die eindeutige Identifikation einer Transporteinheit im Vordergrund, was die Verwendung einer NVE nahe legt. Wird diese als 96-Bit-EPC codiert, spricht man von EPC-NVE-96. Diese besteht im Wesentlichen aus einer unternehmensspezifischen Basisnummer und einer versandeinheitsspezifischen Seriennummer. Diese belegen zusammen 58 Bit, sodass, abzüglich der EPC-typischen Zusatzinformationen wie Header oder Filter, 24 der 96 Bit ungenutzt bleiben. Der genaue Aufbau der EPC-NVE-96 wird in Abbildung 6-1 dargestellt.

	EPC-NVE-96						
	Header	Filter	Partition	Basisnr.	Seriennr.	Unbesetzt	∑
Länge (Bit)	8	3	3	20-40	38-18	24	96

Abbildung 6-1 Aufbau des EPC-NVE-96

Standardisierte Vorgaben für die Codierung weiterer Daten, wie sie zur Steuerung des Materialflusses notwendig sind, existieren derzeit nicht. Obwohl damit keine strikte Konformität mit den EPC-NVE-96-Vorgaben mehr gegeben ist, wurde entschieden, die bisher ungenutzten 24 Bit für die Speicherung des Zieles sowie einer einfachen Beschreibung des Inhalts der Transporteinheit zu verwenden (siehe Abbildung 6-2). Die 3 Byte werden folgendermaßen verwendet:

- *Zielbereich* (4 Bit, entspricht einem Wertebereich von 0 bis 15) beschreibt den Bereich der Versuchsanlage, in dem sich das Ziel befindet. Dabei entsprechen die Werte 1, 2 und 3 jeweils dem Hochregallager, dem EHB-Bereich und dem Warenausgang.
- *Zielmodul* (8 Bit, entspricht einem Wertebereich von 0 bis 255) bezieht sich auf ein Modul im angegebenen Bereich. Dabei wird auch in der Anlagentopologie jedem Modul, z.B. einer EHB-Strecke, Weiche oder Kranfeld, eine eindeutige ID in Form einer ganzen Zahl zugewiesen, sodass eine eindeutige Bestimmung des Zieles anhand einer einzigen Zahl möglich ist.
- *Zielort* (4 Bit, entspricht einem Wertebereich von 0 bis 15) bezieht sich auf einen bestimmten Lastübergabe- oder auch Kommissionierplatz, der zum Zielmodul gehört bzw. von diesem direkt bedient werden kann.
- *Inhalt Anzahl* (6 Bit, entspricht einem Wertebereich von 0 bis 63) gibt an, wie viele weitere Transporteinheiten sich in bzw. auf der aktuellen Transporteinheit befinden. Diese Zahl gibt beispielsweise an, wie viele Kartons oder Behälter auf einer Europalette liegen oder wie viele Kartons in einem VDA-Behälter untergebracht sind.
- *Inhalt* (2 Bit, entspricht einem Wertebereich von 0 bis 3) gibt an, von welchem Typ der Inhalt der aktuellen Transporteinheit ist. Im Demonstrationsszenario wurden für eine Palette Kartons (Inhalt = 1), VDA-Behälter (Inhalt = 2) oder eine Mischung der Beiden (Inhalt = 3) vorgesehen, während sich in einem VDA-Behälter wiederum kleinere Kartons (Inhalt = 1) befinden können.

	Header	Filter	Partition	Basisnr.	Seriennr.	Internet der Dinge	∑
Länge (Bit)	8	3	3	20-40	38-18	24	96

	Zielbereich	Zielmodul	Zielort	Inhalt Anz.	Inhalt
Länge (Bit)	4	8	4	6	2

Abbildung 6-2 Auf EPC-NVE-96 basierendes Datenformat für RFID-Transponder im Internet der Dinge

Entlang des Materialflusses von der Roboterzelle, über den Kleinteileförderer und über die EHB-Anlage sind insgesamt vier RFID-Lesepunkte angebracht, an denen diese Informationen ausgelesen werden. Dies bildet die Grundlage für die dezentrale Wegplanung bzw. Depalettierung.

6.1.2 Roboterzelle

6.1.2.1 Beschreibung des Versuchsfeldes

Die Roboterzelle ist in Form einer automatisierten Lagervorzone gestaltet und dient dem Palettieren, Depalettieren, Einlagern, Auslagern und Kommissionieren von VDA-Behältern und Kartons. Der Sechs-Achs-Knickarm-Roboter der Firma Kuka (Typ KR210 L150) ist mit einem von J. Schmalz GmbH speziell für das Internet der Dinge entwickelten, hochflexiblen Greifer (siehe Abbildung 6-4) ausgestattet und führt neben den bereits erwähnten Aufgaben auch die Handhabung und Verwaltung der Leerpaletten innerhalb der Zelle durch (siehe Abbildung 6-3). Die Roboterzelle wird über eine Schwerlastrollenbahn mit Paletten versorgt, eine zweite Schwerlastrollenbahn entfernt die kommissionierten Europaletten aus der Zelle. Ein Kleinteileförderer kann VDA-KLTs von der Roboterzelle zur EHB-Anlage fördern.

Abbildung 6-3 Roboterzelle in der Versuchsanlage des Lehrstuhls fml

Der Greifer wurde so gestaltet, dass er mehrere Greifprinzipien (Vakuum, Klemmen) vereint und unterschiedliche Aufgaben erfüllen kann:

- Palettieren und Depalettieren von VDA-Kleinladungsträgern und Kartons auf eine Europapellte bzw. von einer Europalette,
- Ein- / Auslagern von VDA-KLT und Kartons in das anliegende Regal,
- Greifen von Kartons aus einem VDA-Behälter bzw. Befüllen eines KLTs mit Kartons,
- Greifen von Leerpaletten.

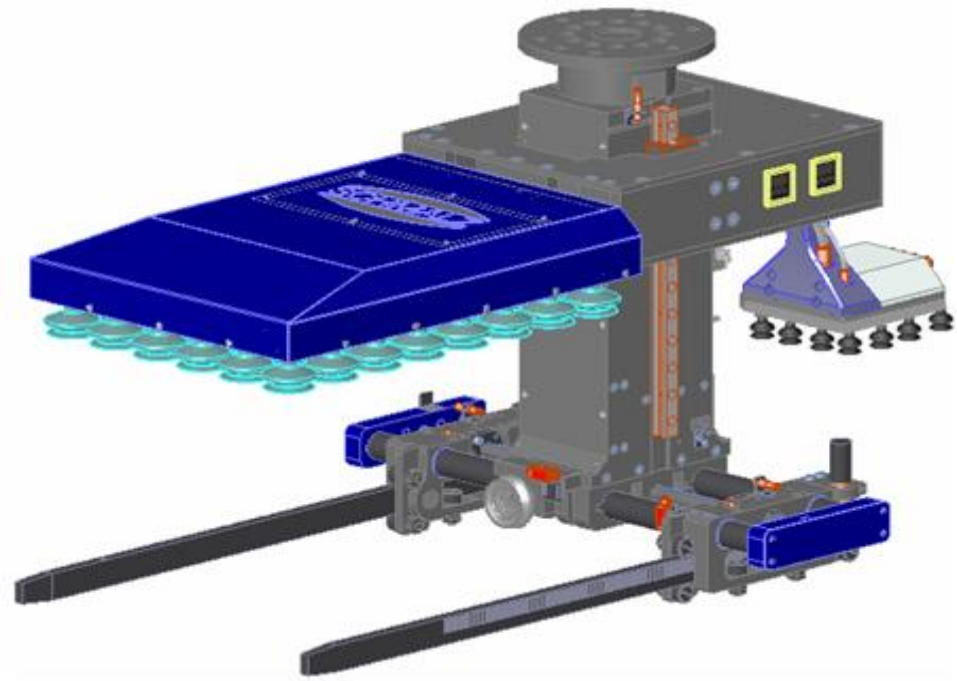


Abbildung 6-4 Hochflexibles LAM zum Greifen von KLTs, Kartons und Leerpaletten, entwickelt von der J. Schmalz GmbH

6.1.2.2 Steuerungsarchitektur

Der Fokus dieses Versuchsaufbaus liegt allgemein auf der Steuerung automatisierter (De-)Palettier- und Kommissionierprozesse durch RFID. Um den Aufwand für die Erstellung des Demonstrators relativ gering zu halten, wurde entschieden, das bereits aus früheren Projekten existierende Steuerungskonzept zu verwenden und lediglich durch neue Funktionen zu erweitern. Zwar beruht das Konzept ebenfalls auf der in Abschnitt 3.3.3 dargestellten Zweiteilung zwischen höherwertiger Logik und Maschinensteuerung, wurde aber nicht in Form eines kommunizierenden und koope-

rierenden Softwareagenten, sondern als Stand-Alone-Applikation ohne Schnittstellen nach außen implementiert.

Die Verwaltung der Zellenperipherie, der zwei RFID-Reader, der Aufträge sowie die Schnittstelle zum Benutzer wurden als Windowsanwendung programmiert, die über einen DeviceNet-Bus mit der Kuka-Robotersteuerung, also der Maschinensteuerungsebene kommuniziert. Diese implementiert eine Reihe von KRL-Programmen (Kuka Robot Language) zur Bewegungssteuerung, E/A-Überwachung und Ansteuerung des Greifers und ist für die Abarbeitung jeweils eines von der Zellensteuerung vorgegebenen Auftrags verantwortlich. Die Robotersteuerung kommuniziert weiterhin über einen ProfiBus mit der Steuerung des Greifers, die ihrerseits verschiedene Greiferkonfigurationen umsetzt, das Vakuum überwacht und einen Webserver zur Überwachung des Greifprozesses bereitstellt (siehe Abbildung 6-5).

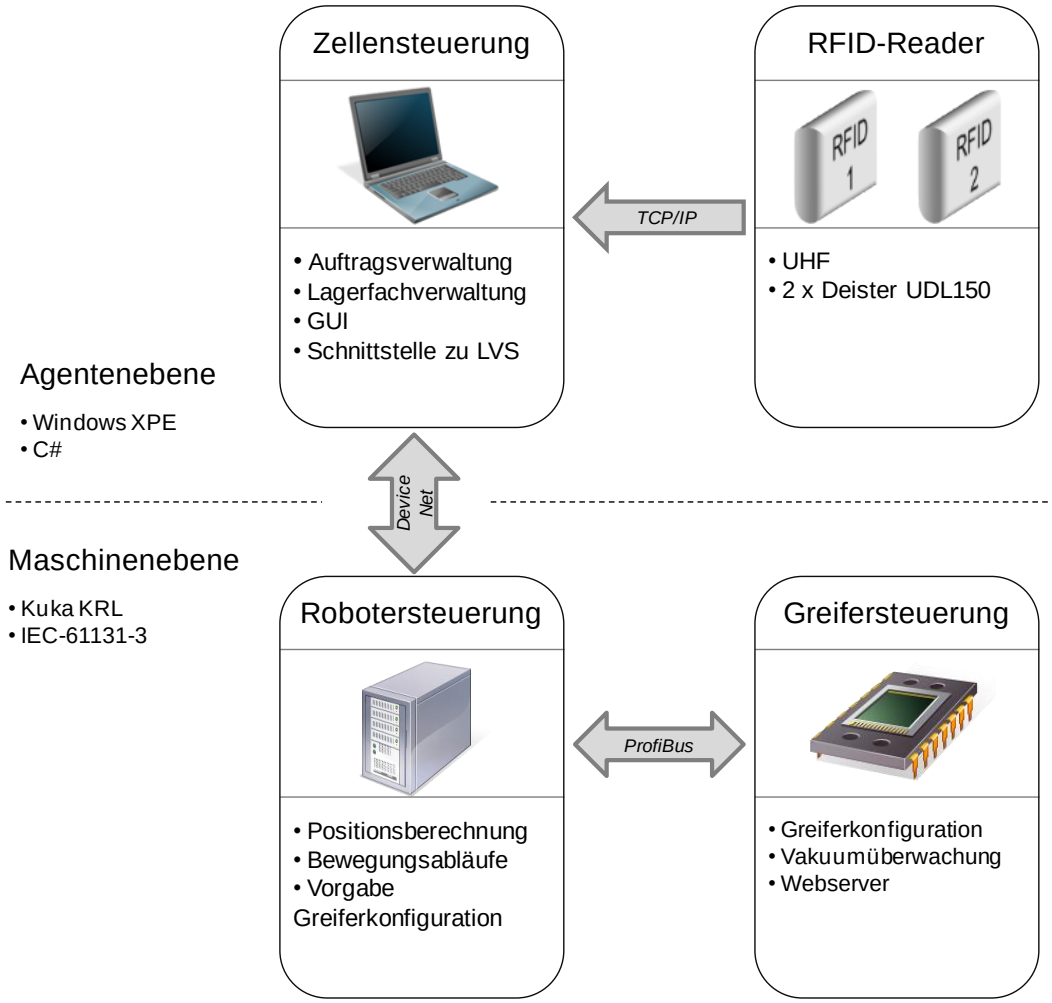


Abbildung 6-5 Steuerungsarchitektur der Roboterzelle

6.1.2.3 Funktionalität und Abläufe

Wird eine neue Palette in die Roboterzelle gefördert, liest ein unter der Rollenbahn angebrachter RFID-Reader die Daten der Palette aus und stellt die Daten einer auf einem Embedded PC laufenden Auftragsverwaltungssoftware zur Verfügung. Diese Software ist in der Lage, aus den gelesenen RFID-Informationen entsprechende Depalettieraufträge zu generieren, die dann einzeln an die Robotersteuerung übergeben werden. Die Auftragsdaten enthalten zum einen die Position der nächsten zu greifenden Transporteinheit (Palettenlage in Z-Richtung sowie die Angabe „vorne“/„hinten“ und „links“/„rechts“) sowie deren Typ (Karton oder VDA-KLT). Der Roboter führt nach Anfahren der übermittelten Grobposition selbständig eine Feinpositionierung durch und greift die Ladung. Das nun vereinzelte Gut wird an einer zweiten, am Zaun der Roboterzelle angebrachten RFID-Antenne vorbei geführt. Aus den nun gelesenen Daten wird ein Folgeauftrag generiert:

- Entspricht der Zielbereich dem Hochregallager, wird ein freies Fach ausgewählt und die Transporteinheit eingelagert. Handelt es sich dabei allerdings um einen VDA-Behälter, der mit kleinen Kartons befüllt ist, wird der Behälter zuerst auf einen Kommissioniertisch gelegt, die Kartons durch den Roboter aus dem KLT entfernt und der leere KLT anschließend eingelagert.
- Entspricht der Zielbereich der EHB-Anlage und handelt es sich um einen VDA-Behälter, wird dieser auf den Kleinteileförderer gelegt.
- Entspricht der Zielbereich dem Warenausgang, wird die Transporteinheit auf die auf der Abtransportstrecke befindlichen Palette gelegt.
- Konnte ein RFID-Tag nicht gelesen werden, wird die Transporteinheit eingelagert, eine entsprechende Fehlermeldung ausgegeben und mit dem nächsten Depalettierauftrag fortgefahren.

Dieser Vorgang wird in Abbildung 6-6 als UML-Aktivitätsdiagramm zusammenfassend dargestellt.

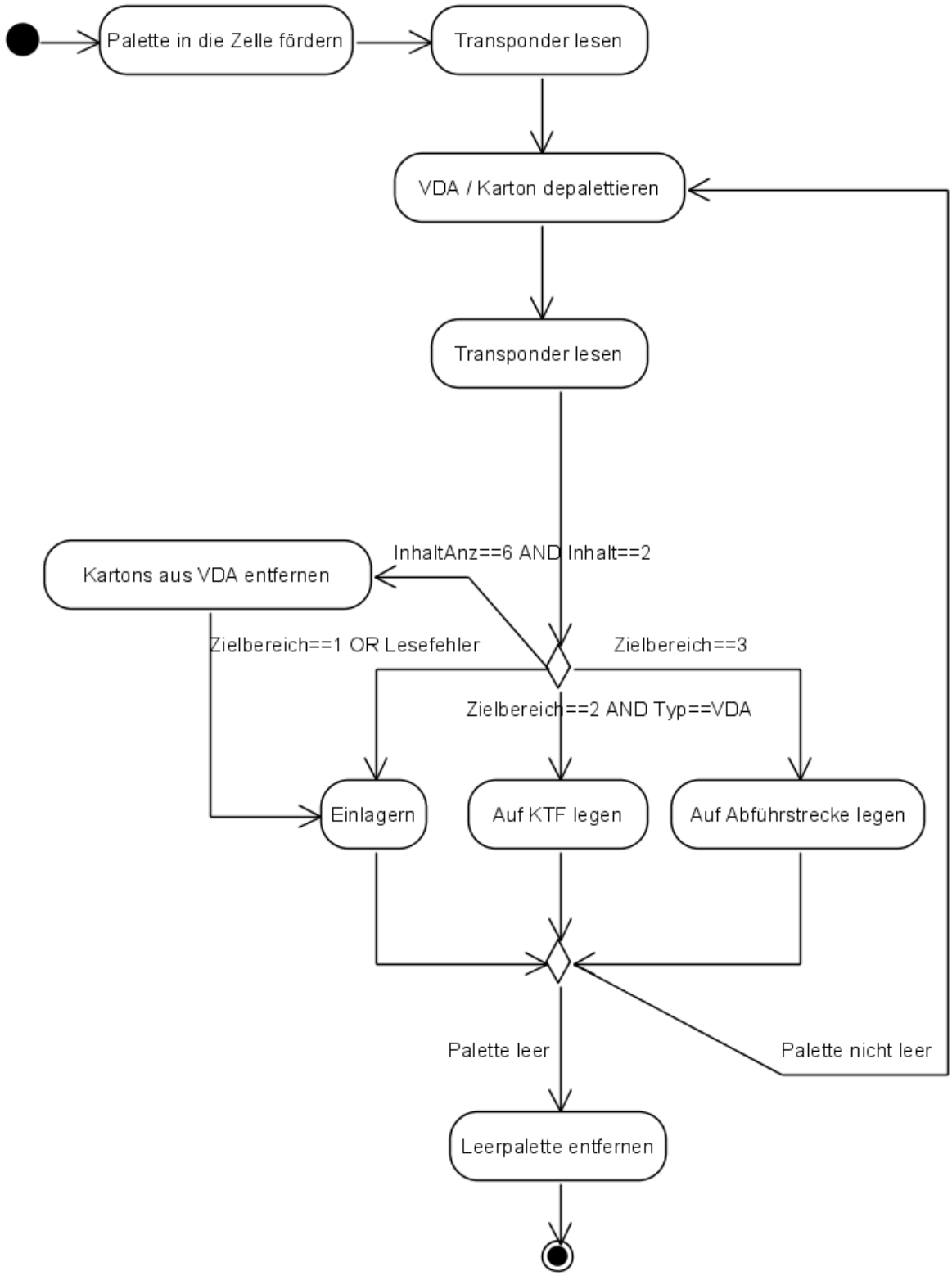


Abbildung 6-6 Automatische, mit RFID gesteuerte Depalettierung

Auslager- und Kommissionieraufträge werden von einem Bediener über eine GUI manuell eingegeben, wobei eine Anbindung an ein externes System, z.B. ein LVS, ebenso denkbar ist.

6.1.2.4 Erkenntnisse

Das Versuchsfeld „Roboterzelle“ demonstriert die technische Machbarkeit der Steuerung automatisierter Prozesse in der Lagervorzone mit RFID. Beim Greifen und Palettieren von Kartons und Kleinladungsträgern sowie beim Handhaben von Leerpaletten treten äußerst selten Fehler auf – beim Greifen kleiner Kartons aus Behältern heraus ist dies aber recht häufig der Fall.

Das Problem liegt dabei jedoch nicht an dem Steuerungskonzept an sich oder an der RFID-Technologie, sondern ergibt sich durch die ungewisse Position der Ladungsträger. Zwar findet bei Kartons und VDA-KLTs eine Feinpositionierung mittels zweier Lichttaster statt, diese fängt aber nur translatorische Verschiebungen entlang zweier Achsen ab. Auf Drehungen der Ladeeinheiten, so wie sie beim Transport der Palette über die Schwerlastrollenbahn oder bei der Handhabung durch das Regalbediengerät entstehen können, kann nur in einem kleinen Umfang reagiert werden. Bei der Handhabung von Leerpaletten findet keinerlei Feinpositionierung statt, was in Anbetracht der recht großen Toleranzen bei der Lastaufnahme keinen nennenswerten Nachteil darstellt. Anders gestaltet sich der Griff in Behälter hinein: hier wäre ein System zur digitalen Bilderkennung, Identifikation und Positionsbestimmung einzelner Kartons im Behälter dringend erforderlich, um die Genauigkeit und Zuverlässigkeit des Greifprozesses zu erhöhen.

6.1.3 Stetigförderer

6.1.3.1 Beschreibung des Versuchsfeldes

Eine Rollenbahn für den Transport von Kleinladungsträgern beginnt in der Roboterzelle, verläuft zu einem großen Teil unter dem Schienennetz der Elektronhängebahnanlage und stellt somit eine Verbindung zwischen den beiden Bereichen her. Diese Verbindung besteht aus zwei mechanisch unabhängigen Förderern, wobei der erste aus einer Geraden mit insgesamt vier Stauplätzen und der zweite aus zwei Geraden, einer 90-Grad-Kurve und drei Stauplätzen besteht (siehe Abbildung 6-7). Am jeweils letzten Stauplatz kann ein VDA-KLT an eine EHB-Katze abgegeben werden, wobei an diesen Stellen auch je ein RFID-Lesepunkt angebracht wurde. Die Lesepunkte bestehen aus jeweils einer Sender- und einer Empfängerantenne, die an einem gemeinsamen UHF-RFID-Reader der Firma Siemens angeschlossen sind.

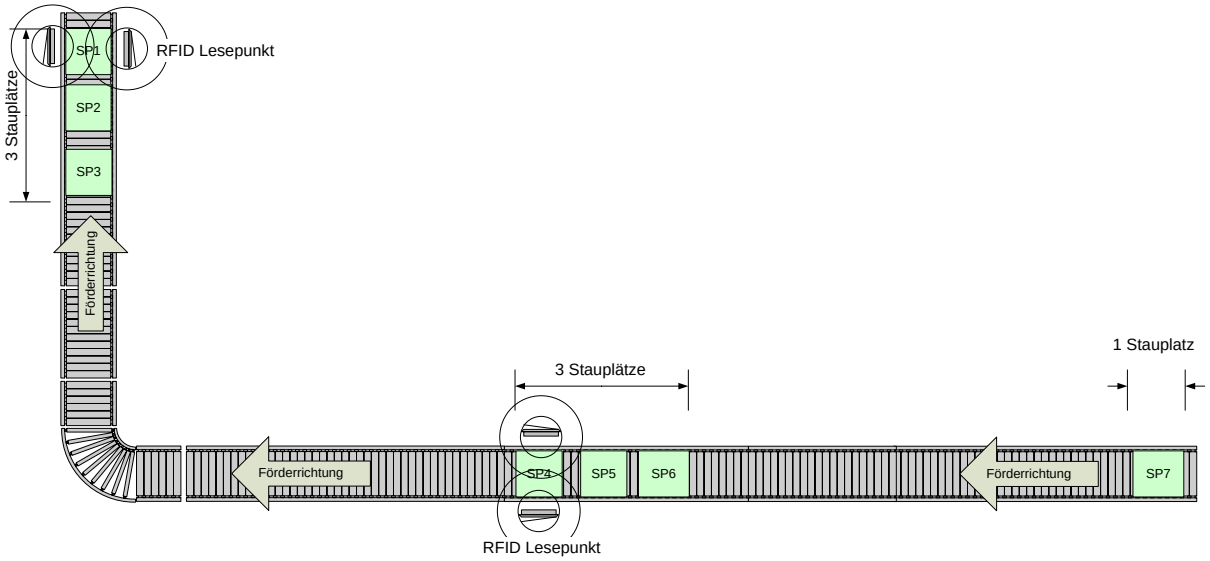


Abbildung 6-7 Kleinteileförderer in der Versuchsanlage des Lehrstuhls fml

6.1.3.2 Steuerungsarchitektur

Die zwei Förderer wurden gemäß der Modularisierungssystematik des Internet der Dinge als zwei unabhängige Module aufgefasst. Die Modullogik wurde als zweischichtige Software realisiert, wobei die Softwareagenten als Java-Programm ausgeführt wurde und auf dem JADE/LEAP-Framework basieren, während die Maschinensteuerung als IEC-61131-3 Programm implementiert wurde. Die dazwischen vermittelnde Middleware wurde in Form einer Sammlung von Java-Klassen realisiert, die in jedem Agenten instanziiert werden und entsprechend den Angaben in einer XML-Konfigurationsdatei Informationen zwischen dem Agenten und der Maschinensteuerung oder zwischen dem Agenten und einem Emulator überträgt. Die gesamte zur Steuerung eines Moduls notwendige Software wird auf einem am Gerät angebrachten Controller ausgeführt (siehe Abbildung 6-8).

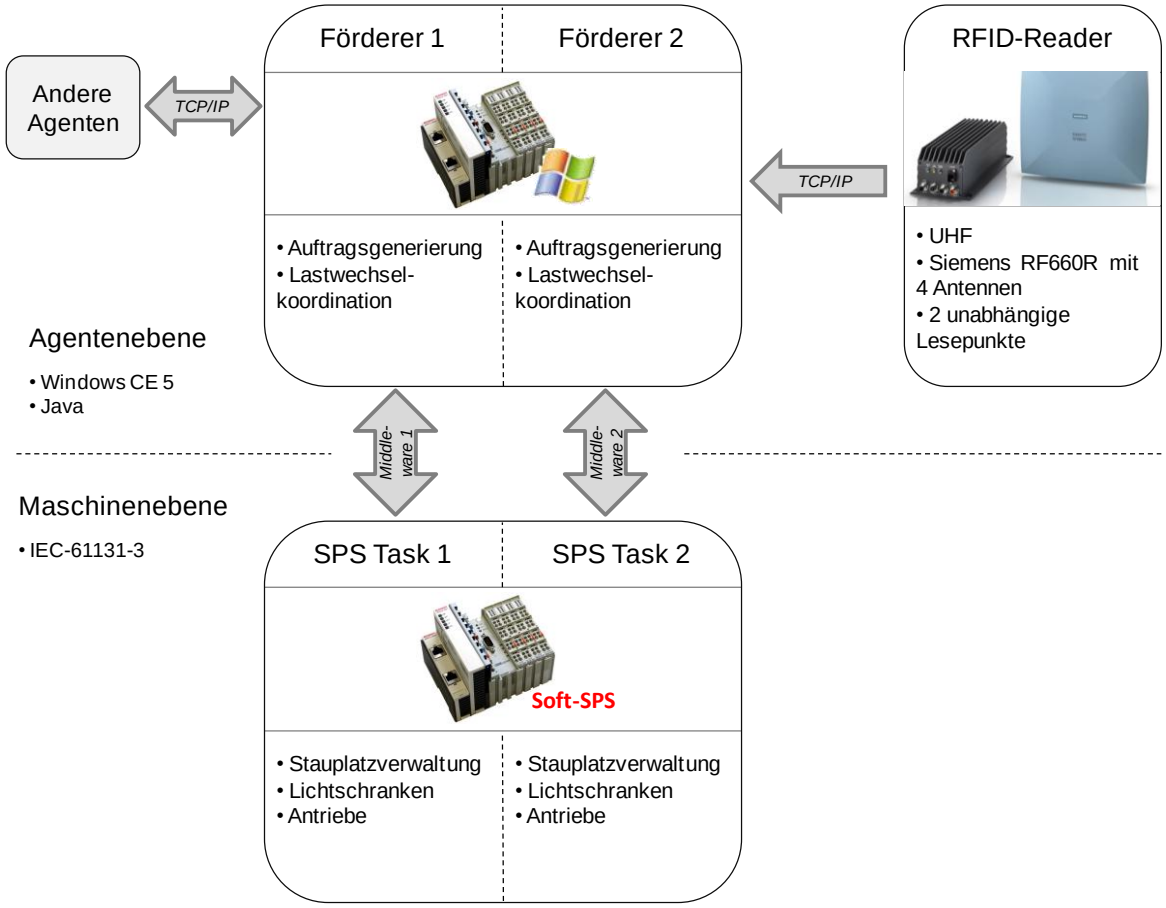


Abbildung 6-8 Steuerungsarchitektur der Stetigförderer

Als Steuerungshardware für die Module werden Embedded PCs der Firma Beckhoff (Modell CX9010) eingesetzt, die mit Windows® CE 5 und Soft-SPS sowie einem Intel® IXP 420 Prozessor mit 533MHz ausgerüstet sind (siehe Abbildung 6-9). Für die Kommunikation zwischen Entitäten wird Ethernet- bzw. WLAN eingesetzt.

Abbildung 6-9 Embedded PC mit CANopen Buskoppler und E/A-Klemmen

Dabei werden aus Kostengründen die Steuerungsprogramme der zwei Rollenförderermodule auf demselben Embedded PC mit Soft-SPS ausgeführt, zudem greifen beide Module auf denselben RFID-Reader zu.

6.1.3.3 Funktionalität und Abläufe

Wird ein Kleinladungsträger vom Roboter auf den Rollenförderer gelegt, wird er bis zum letzten freien Stauplatz gefördert. Die Verwaltung der Pufferplätze findet dabei auf der Maschinensteuerungsebene statt. Kommt ein KLT auf dem letzten Stauplatz des Förderers an, gelangt er in das Lesefeld des RFID-Readers. Die gelesenen Daten werden dem Agenten des Rollenförderers zur Verfügung gestellt, der anhand der Zielinformationen auf dem Transponder entscheidet, ob der Behälter an den nächsten Rollenförderer oder an eine EHB-Katze abgegeben werden muss. Wird der Behälter an den zweiten Rollenförderer übergeben, wiederholt sich der gerade beschriebene Ablauf, bis die Transporteinheit wiederum am letzten Stauplatz ankommt. Von dort aus ist eine Übergabe zur EHB die einzig mögliche Alternative. Der Rollenförderer erzeugt einen Transportauftrag, der als Startpunkt die Position des eigenen Übergabeplatzes und als Ziel den auf dem RFID-Tag angegebenen Ort hat. Dieser Auftrag wird in Form einer XML-Nachricht auf das Blackboard geschrieben, von wo aus er von den EHB-Katzen abgerufen wird. Abbildung 6-10 zeigt diesen Vorgang als UML-Aktivitätsdiagramm.

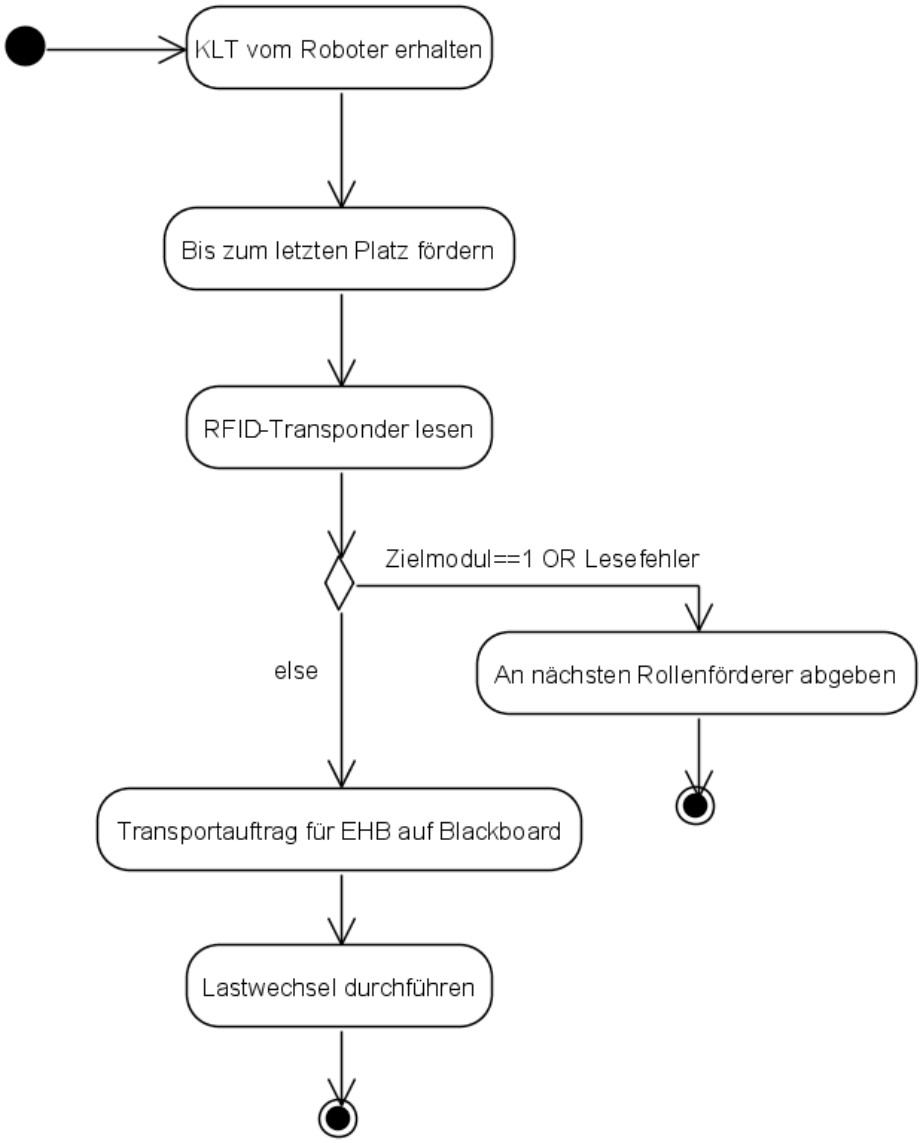


Abbildung 6-10 Funktion eines Stetigförderers in der Versuchsanlage des Lehrstuhls fml

6.1.3.4 Erkenntnisse

Das Szenario deckt mehrere wesentliche Aspekte einer dezentralen Steuerung ab. Zum einen findet eine Kopplung mit einem nicht von Agenten gesteuerten und damit nicht direkt ansprechbaren System ab, nämlich der Roboterzelle. Der Informationsfluss zwischen Lagervorzone und Förderer wird hierbei lediglich über die an den VDA-KLTs angebrachten RFID-Tags realisiert. Zum anderen findet eine abgestimmte Lastübergabe zu einer EHB-Anlage statt, einem mechanisch völlig unterschiedlichen aber nach demselben Prinzip gesteuerten Subsystem.

Weiterhin wird die Möglichkeit demonstriert, mehrere unabhängige Softwareagenten inkl. der zugeordneten Maschinensteuerung auf einer einzigen Rechenplattform

auszuführen und auch an einen einzigen RFID-Reader zu koppeln. Dieses Vorgehen entspricht zwar nicht mehr einer rein mechatronischen Systematik und hat somit Einbußen in Bezug auf die Robustheit und Transparenz des Systems zu verzeichnen, kann aber in einem Internet der Dinge System zu Kosteneinsparungen führen. Anzumerken ist hierbei, dass eine Verteilung der zwei Steuerungsprogramme auf unabhängige Rechner jederzeit und ohne Programmieraufwand möglich ist – lediglich eine Umkonfiguration der Middleware wird unter Umständen notwendig, wenn sich bei der Migration der Software die Adressen der SPS-Variablen oder beispielsweise die IP des RFID-Readers ändern.

6.1.4 EHB-Anlage

6.1.4.1 Beschreibung des Versuchsfeldes

Die Elektrohängebahnanlage (siehe Abbildung 6-11) besteht aus folgenden Komponenten, die als jeweils ein Modul aufgefasst werden:

- zwei EHB-Katzen, wobei eine mit einem automatischen und die andere mit einem manuellen Lastaufnahmemittel (LAM) ausgestattet ist,
- drei EHB-Weichen und
- ein Einträgerkran.

Abbildung 6-11 EHB-Anlage in der Versuchsanlage des Lehrstuhls fml

6.1.4.2 Steuerungsarchitektur

Genau wie im Fall der Rollenförderer wurde jedem Modul ein JADE/LEAP-Softwareagent und ein IEC-61131-3 Steuerungsprogramm zugeordnet, welche auf einem Embedded-PC ausgeführt werden. Dabei verfügen die zwei Katzen und der Kran über einen jeweils eigenen Rechner, die drei Weichen teilen sich eine einzige Hardwareplattform. RFID-Reader zum Identifizieren von Transporteinheiten wurden nicht verbaut.

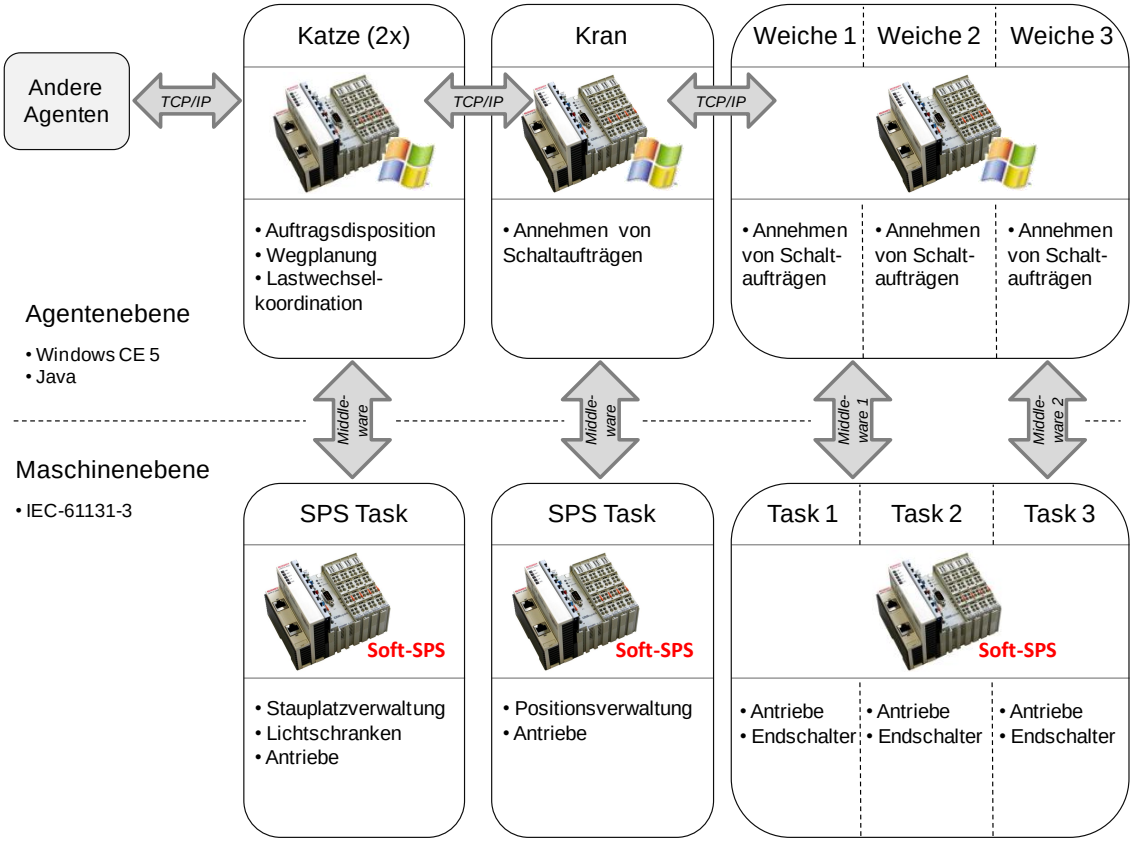


Abbildung 6-12 Steuerungsarchitektur der EHB-Anlage

6.1.4.3 Funktionalität und Abläufe

Die Katzen rufen in periodischen Abständen offene Transportaufträge vom Blackboard ab. Erhält eine Katze einen neuen Auftrag, dessen Start- und Zielpunkt sie erreichen kann, prüft sie zuerst, ob andere freie Katzen im System existieren und somit eine Verhandlung notwendig ist, um den Auftrag zuzuteilen. Diese Information wird über den Directory Facilitator abgerufen. Ist kein anderes freies Fahrzeug verfügbar, nimmt die Katze den Auftrag an und trägt auf dem Blackboard einen entsprechenden Vermerk ein. Ist eine Verhandlung notwendig, berechnen die beteiligten Katzen im ersten Schritt ein Gebot, das sich aus der Länge des kürzesten

reservierbaren Weges zum Startpunkt ergibt. Beide Katzen schreiben ihr eigenes Angebot auf das Blackboard und rufen nach einer fest definierten Zeit von fünf Sekunden alle vorhandenen Gebote ab – damit besitzt jedes Fahrzeug einen Gesamtüberblick der Auktion und kann somit erkennen, ob es die Verhandlung gewonnen hat. Da die Transporteinheiten im vorliegenden Szenario nicht mit eigenen Softwareagenten ausgestattet sind, muss die Auftragsdisposition in Form einer unmoderierten Auktion nur zwischen den Katzen und unter Zuhilfenahme des Blackboards geregelt werden. Der Gewinner reserviert den Auftrag auf dem Blackboard in seinem Namen, womit doppelte Reservierungen verhindert werden, und beginnt anschließend mit der Auftragsbearbeitung.

Die Berechnung eines Weges vom aktuellen Standort der Katze zum Rollenförderer, von dem der Behälter abzuholen ist, wird mit einem Dijkstra-Algorithmus [Dij-59] durchgeführt. Dieser berücksichtigt zum einen die (statische) Anlagentopologie, die bei der Initialisierung abgerufen wurde. Zum anderen wird aber auch der dynamische Zustand der Anlage – so z.B. der Aufenthaltsort anderer Fahrzeuge, Wegreservierungen oder manuell eingetragene Streckensperrungen – betrachtet. All diese Informationen sind auf einem Blackboard verfügbar, was die Aktualität und Korrektheit der Daten garantiert. Hat die Katze einen oder mehrere Wege gefunden und sich für den kürzesten entschieden, trägt sie Reservierungen für die zu befahrenden Streckenstücke und Weichen in die Topologie ein. Diese sind notwendig, um Kollisionen mit anderen Fahrzeugen zu vermeiden, vor allem unter dem Gesichtspunkt, dass das Streckennetz in beide Richtungen befahren wird. Nachdem ein Streckenstück oder eine Weiche verlassen wurde, wird die entsprechende Reservierung sofort gelöscht. Während der Fahrt muss die Katze evtl. Weichen oder den Einträgerkran damit beauftragen, bestimmte Wegverbindungen herzustellen. Details zum Wegplanungs- und Fahrzeugsteuerungskonzept finden sich auch in [Wil-06].

Am Startpunkt des Auftrags angekommen prüft die Katze zuerst, ob an dieser Stelle eine Lastwechselkoordination notwendig ist (siehe Abschnitt 5.2.4). Wird im Directory Facilitator ein Rollenförderer für die entsprechenden Koordinaten aufgeführt, so wird über einen einfachen Nachrichtenaustausch zwischen Katze und Rollenbahnmodul vor dem Lastwechsel sichergestellt, dass sich beide Module in einem für den Lasttransfer sicheren Zustand befinden. Dies berücksichtigt u.a. das Erreichen der korrekten Position, das Ausschalten der Antriebe, das Vorhandensein eines KLT auf

dem Rollenförderer und die Bereitschaft der Katze, den Behälter aufzunehmen. Das Lastaufnahmemittel der Katze wird gesenkt und im Fall des automatischen Lastaufnahmemittels nach dem Greifen des Behälters wieder angehoben. Besitzt die Katze ein manuelles Lastaufnahmemittel, muss ein Bediener den Greifvorgang durchführen und das LAM anschließend hochfahren. Die erfolgreiche Beendigung der Lastübergabe wird über einen weiteren Nachrichtenaustausch zwischen Katze und Förderer quittiert, nach dem die zwei Module mit ihren jeweiligen Transportaufträgen fortfahren können.

Die Katze kann nun zum endgültigen Ziel fahren. Dabei wird zuerst überprüft, ob der Zielpunkt von der zweiten Katze reserviert wird – ist dies der Fall, wird versucht, dieser zweiten Katze einen Ausweichauftrag zu erteilen, damit diese den Wegpunkt räumt. Ist der Weg frei, wird der bereits beschriebene Vorgang der Wegplanung und -reservierung, des Transports und des Lastwechsels wiederholt. Wurde der Auftrag erfüllt, wird auch dies auf dem Blackboard eingetragen, sodass sich ein Bediener über den Auftragsverwaltungs- oder die Visualisierungsdienst jederzeit einen Überblick der abgearbeiteten oder noch freien Aufträge machen kann.

In Abbildung 6-13 wird dieser Vorgang in Form einer UML-Aktivitätsdiagramms zusammengefasst.

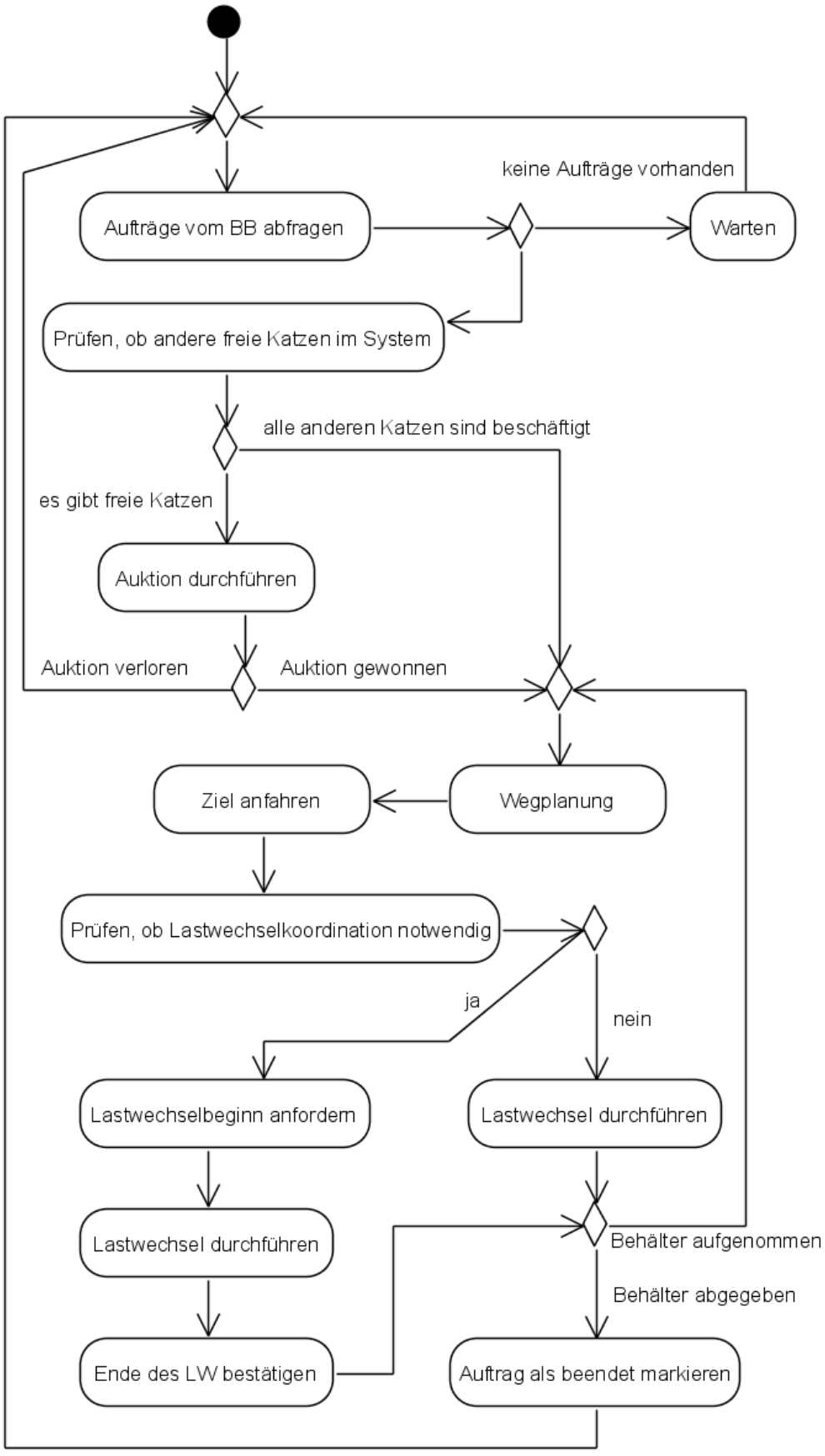


Abbildung 6-13 Funktion einer EHB-Katze in der Versuchsanlage des Lehrstuhls fml

6.1.4.4 Erkenntisse

Die Umsetzung einer komplett dezentralen EHB-Anlagen-Steuerung mittels eines Agentensystems demonstriert aus softwaretechnischer Sicht zunächst einmal die einfache Interoperabilität und Integration von Programmen in verschiedenen Laufzeitumgebungen. So kommunizieren die in der Anlage auf Windows-CE-Rechnern befindlichen Java-Modulagenten mit auf einem Windows-XP-PC laufenden .Net-Dienstagenten (Blackboard, Visualisierung) und mit der ihnen zugeordneten IEC-61131-3-Maschinensteuerung.

Die Materialflusssteuerung implementiert einige komplexe Funktionen: die Auftragsdisposition, die Wegplanung und -reservierung und die Möglichkeit, Ausweichaufträge automatisch zu generieren und somit einen deadlockfreien Betrieb zu gewährleisten. All diese Algorithmen sind unabhängig von der Topologie, der Auftragsstruktur oder der Anzahl Fahrzeuge in einer Anlage gestaltet. Somit ist eine Erweiterung oder ein Umbau des Systems jederzeit möglich, fehlerhafte oder gesperrte Streckenstücke werden von den Katzen erkannt und automatisch umfahren, sofern Alternativrouten vorhanden sind. Das Ergebnis ist ein sehr robustes und flexibles System.

6.2 Emulationen

Nicht alle Aspekte der Steuerung funktional komplexer Materialflusssysteme konnten in der Versuchshalle des Lehrstuhls fml untersucht werden. Daher wurden Emulationen realisiert, die zum einen der Realisierung einer Steuerung für Frühgepäckspeicher an Flughäfen und einer Steuerung für Sorter mit Paarbildung dienten. Zum anderen wurde auch ein Großteil der im Abschnitt 6.1 beschriebenen Steuerungsagenten für Stetigförderer und EHB-Anlagen zunächst in einer virtuellen Umgebung entwickelt und getestet.

Bei der Emulation wurde die Maschinensteuerungsschicht der Module durch einen in Java programmierten Agenten ersetzt, der die Funktionalität der Maschinsteuerung – z.B. Veränderung der Position des Moduls oder der beförderten Ladungsträger oder das Generieren von Sensorsignalen – nachahmt. Auf diese Weise kann die (Software-)Funktionalität eines Moduls in einer virtuellen Umgebung ohne die mechanischen oder elektrischen Komponenten der Fördertechnik getestet werden. Die Kopp-

lung zwischen Agent und Emulator bzw. Agent und Maschinensteuerung wurde über die in Abschnitt 3.3.3 beschriebene Middleware realisiert.

Da Java-Programme im Normalfall nicht echtzeitfähig sind und somit ein nicht vorhersehbares und nicht reproduzierbares Zeitverhalten aufweisen, kann die hier eingesetzte Emulationsmethode lediglich funktionale Aspekte einer Anlage, nicht aber das Zeitverhalten oder davon abhängige Kennzahlen wie z.B. den Durchsatz zuverlässig abbilden.

6.2.1 Emulation eines Frühgepäckspeichers

6.2.1.1 Beschreibung des Szenarios

Das erste emulierte Szenario befasst sich mit der Thematik von Frühgepäckspeichern an Flughäfen (siehe auch Abschnitt 5.2.3.1). Dabei müssen von Passagieren eingecheckte Gepäckstücke auf einer automatisierten Stetigförderanlage vom Check-In-Schalter zum Flugzeug gefördert werden. Oftmals kommt es dabei aber vor, dass der entsprechende Flug noch nicht verfügbar ist und die Koffer, die somit nicht verladen werden können, zuerst in einem so genannten Frühgepäckspeicher (auch Early Baggage Store oder EBS) gelagert werden müssen. Dieser besteht im untersuchten Fall aus mehreren Bahnen, auf denen Trays mit jeweils einem Gepäckstück gepuffert werden. Dabei werden Bahnen entweder einem bestimmten Flug zugeordnet und dürfen somit nur Gepäck aufnehmen, das für den entsprechenden Flug bestimmt ist, oder sie nehmen alle Gepäckstücke auf, die innerhalb eines bestimmten Zeitfensters in beliebige Flugzeuge verladen werden müssen.

Zur Entwicklung und dem Test der in Abschnitt 5.2.3.1 vorgestellten dezentralen Strategien für diesen Einsatzfall wurde eine Emulation eines stark vereinfachten Flughafenlayouts mit drei Verlade-Gates, sechs EBS-Bahnen und zwölf weiteren Stetigförderern realisiert. Dabei besitzt jedes dieser 21 Module einen eigenen Agenten und Emulator (dieser entfällt für die Verladegates, da diese nur einen einfachen Status verwalten – „Gate offen für Flug XYZ“ oder „Gate geschlossen“). Jede Transporteinheit im System besitzt ebenfalls einen Agenten, der für die Abarbeitung des eigenen Workflows zuständig ist.

Abbildung 6-14 zeigt die beschriebene Topologie, wobei die drei links vom Hauptförderstrang befindlichen EBS-Bahnen jeweils einem Flug zugeordnet und die drei

Bahnen auf der rechten Seite pro Zeitscheibe zugeteilt werden. Der aus einer Visualisierungsumgebung stammende Screenshot zeigt den Fall, dass einige Koffer nach Öffnen des gesuchten Fluges den EBS bereits verlassen haben und sich zu ihrem Verladeziel fördern lassen. Das Suchen des Ziels, die Reservierung von EBS-Bahnen und das Anstoßen der Ausschleusung aus dem EBS fallen in den Verantwortungsbereich der TEs. Die Routenfindung, der Transport und die Verwaltung der eigenen Belegung obliegen den Förderern bzw. EBS-Bahnen.

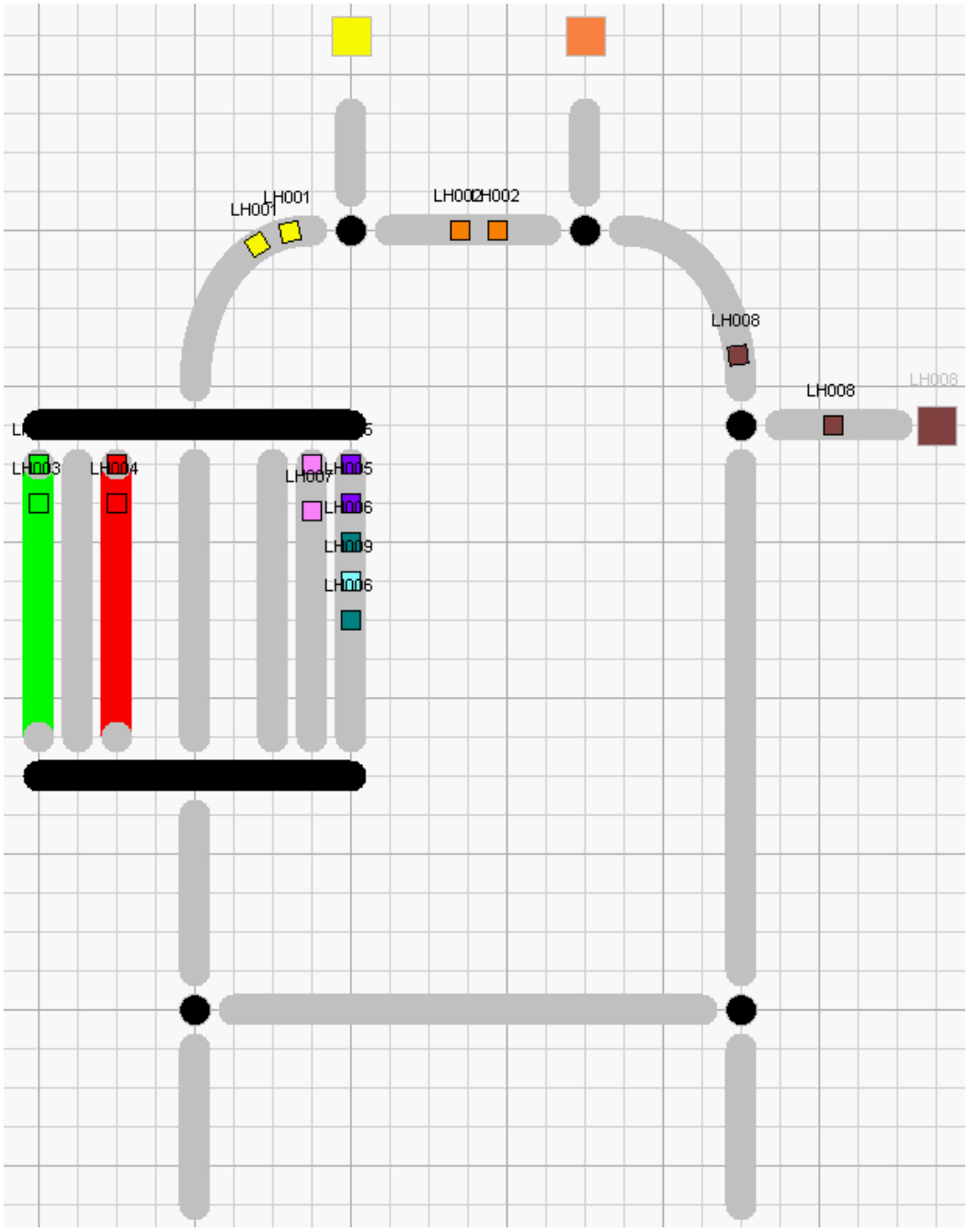


Abbildung 6-14 Emulation eines Frühgepäckspeichers

6.2.1.2 Erkenntnisse

Hier ist zu erwähnen, dass die Agenten zur Steuerung der Stetigförderer in diesem Szenario mit denen in der fml-Versuchsanlage eingesetzten (siehe Abschnitt 6.1.3) völlig identisch sind. Lediglich ein einfacher Emulator für die Realisierung des Förderverhaltens – also der Veränderung der Position der TEs – musste implementiert werden. Die EBS-Bahnen werden ebenfalls vom genannten Stetigfördereragenten abgeleitet und nur durch die Funktionalität zum Verwalten von Belegungen und Flug- bzw. Zeitscheibenreservierungen ergänzt. Zudem verwenden auch diese denselben Emulatoragenten wie die sonstigen Förderer im System. Dies demonstriert den hohen Grad an Wiederverwendbarkeit und auch die einfache Erweiterbarkeit und Anpassbarkeit von Agentenlogik durch den Vererbungsmechanismus.

6.2.2 Emulation eines Sorters mit Paarbildung

6.2.2.1 Beschreibung des Szenarios

Die zweite Emulation befasst sich mit einer funktionalen Erweiterung des beschriebenen EBS-Szenarios. Ähnlich wie bei einem Frühgepäckspeicher soll auch hier eine Sortierung von Transporteinheiten in Abhängigkeit ihres Zieles stattfinden, wobei bei der Belegung der einzelnen Sorterbahnen zusätzliche Einschränkungen bestehen. Ein typisches Szenario aus der Lagerlogistik aufgreifend wird hier die gleichzeitige Übergabe zweier Behälter von zwei Bahnen auf einen Doppelverschiebewagen betrachtet. Dabei müssen, um eine optimale Auslastung des Verschiebewagens zu ermöglichen, die Sorterbahnen so belegt sein, dass eine gleichzeitige Aufnahme *und* Abgabe zweier TEs durch den Verteilwagen möglich ist. Die Transporteinheiten müssen also in Abhängigkeit ihres Zieles paarweise vorsortiert werden.

Das dafür in Abschnitt 5.2.3.3 vorgestellte, dezentrale Steuerungsverfahren wurde anhand der folgenden Emulation entwickelt und validiert: Über eine Zuführstrecke angelieferte Behälter gelangen zu einem Verteilelement, z.B. einem Roboter. Die Transporteinheit bestimmt nun in Zusammenarbeit mit den sechs Sorterbahnen, welche der Pufferstrecken am besten geeignet ist. Grundlage für diese Entscheidung sind das Ziel der TE, die Nachbarschaftsbeziehungen der Sorterbahnen und vom Anlagenbediener vorgegebene, umzusetzende Kombinationen von Ladeeinheiten.

Abbildung 6-15 zeigt einen Screenshot aus einem Emulationslauf. Zu sehen ist die korrekte Bildung von Paaren im Sorter in Abhängigkeit der anzufahrenden Endziele.

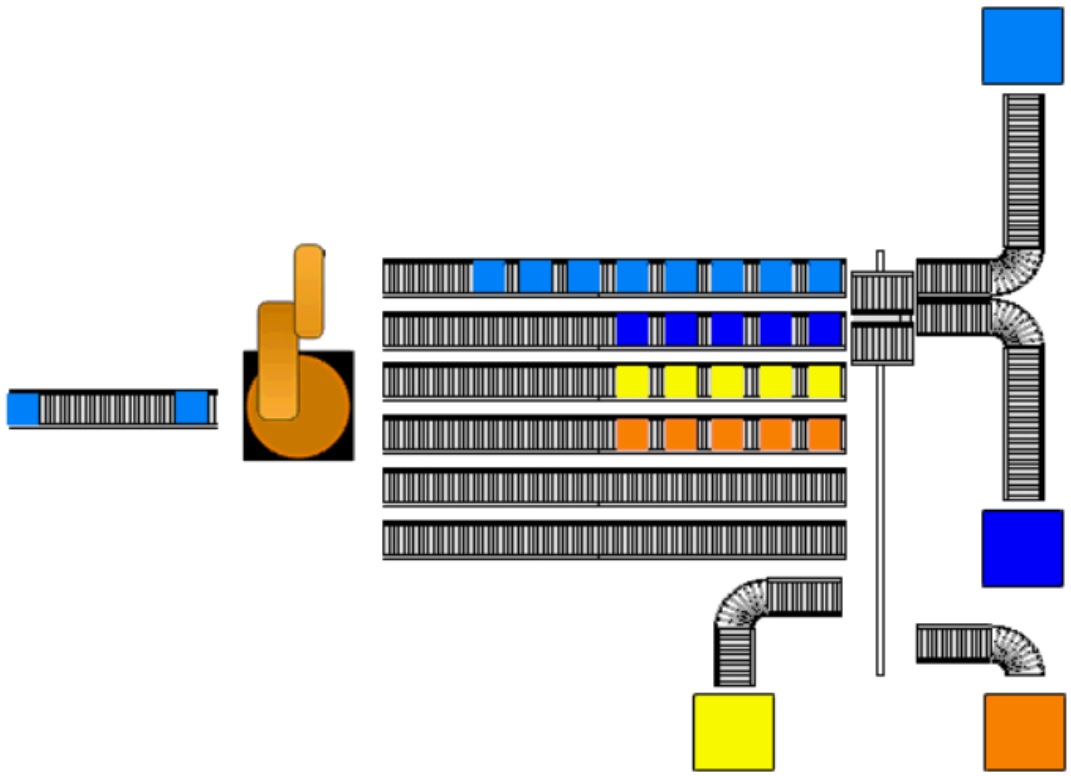


Abbildung 6-15 Emulation eines Sorters mit Paarbildung

6.2.2.2 Erkenntnisse

Auch hier wurden wie bei der EBS-Emulation dieselben Agenten wie in der fml-Versuchsanlage eingesetzt und lediglich durch die neue Logik zur Realisierung von Paaren ergänzt.

Gerade dieser Anwendungsfall zeigt, dass auch komplexe Strategien, die das geordnete Zusammenwirken mehrerer Module erfordert, auf dezentrale Weise mit geringem Aufwand umgesetzt werden können. Die notwendigen Algorithmen sind unabhängig von der Anzahl der Bahnen im Sorter oder den zu gruppierenden TEs. Diese Parameter können vom Anlagenprogrammierer oder -betreiber für die Realisierung beliebiger Szenarien konfiguriert werden und erfordern keinerlei Programmierarbeit.

6.2.3 Emulation der EHB-Versuchsanlage

6.2.3.1 Beschreibung des Szenarios

Die Agenten für die Steuerung des in Abschnitt 6.1.4 beschriebenen Elektrohängebahnsystems wurden eingangs ebenfalls anhand einer Emulation entwickelt und vorgetestet. Zu diesem Zweck wurden Emulatoren für EHB-Katzen, -Weichen und -Krane implementiert.

Die erstellte Emulation wurde anschließend um einen weiteren Bereich mit 18 Weichen, einem Kranfeld und fünf Katzen erweitert (siehe Abbildung 6-16). Diese Erweiterung diente zum einen dem Überprüfen der korrekten Funktion der Wegplanungs-, Reservierungs- und Ausweichalgorithmen anhand eines größeren Systems. Zum anderen wurde diese „virtuelle Anlagenerweiterung“ herangezogen, um einen gleichzeitigen Betrieb emulierter bzw. „virtueller“ und realer Module innerhalb eines Systems zu demonstrieren. Dies ist durch die Softwarearchitektur des Internet der Dinge, vor allem durch die Trennung der höheren Logik von der Maschinensteuerung sowie die plattformübergreifende Einsetzbarkeit des hier verwendeten Agentensystems ohne weiteres möglich. Da die eigentliche Mechanik vor dem Agenten durch eine Middleware versteckt ist und Agenten immer miteinander kommunizieren können, können beispielsweise emulierte Katzen sowohl im virtuell abgebildeten als auch im realen Anlagenteil fahren und dort Weichen oder Krane schalten. Auch die realen Katzen reagieren dabei auf die Wegreservierungen virtueller Katzen und verhalten sich genau so, wie sie es in einer tatsächlich größeren Anlage mit mehreren Teilnehmern tun würden. Da sich aber reale Fahrzeuge selbstverständlich nicht im virtuellen Schienennetz bewegen können, müssen die entsprechenden Punkte in der erweiterten Topologie mit einem Attribut versehen werden, das sie als real oder virtuell kennzeichnet. Den Katzenagenten wird über eine Konfigurationsdatei ebenfalls mitgeteilt, welche Wegpunkte sie bei der Wegplanung berücksichtigen dürfen.

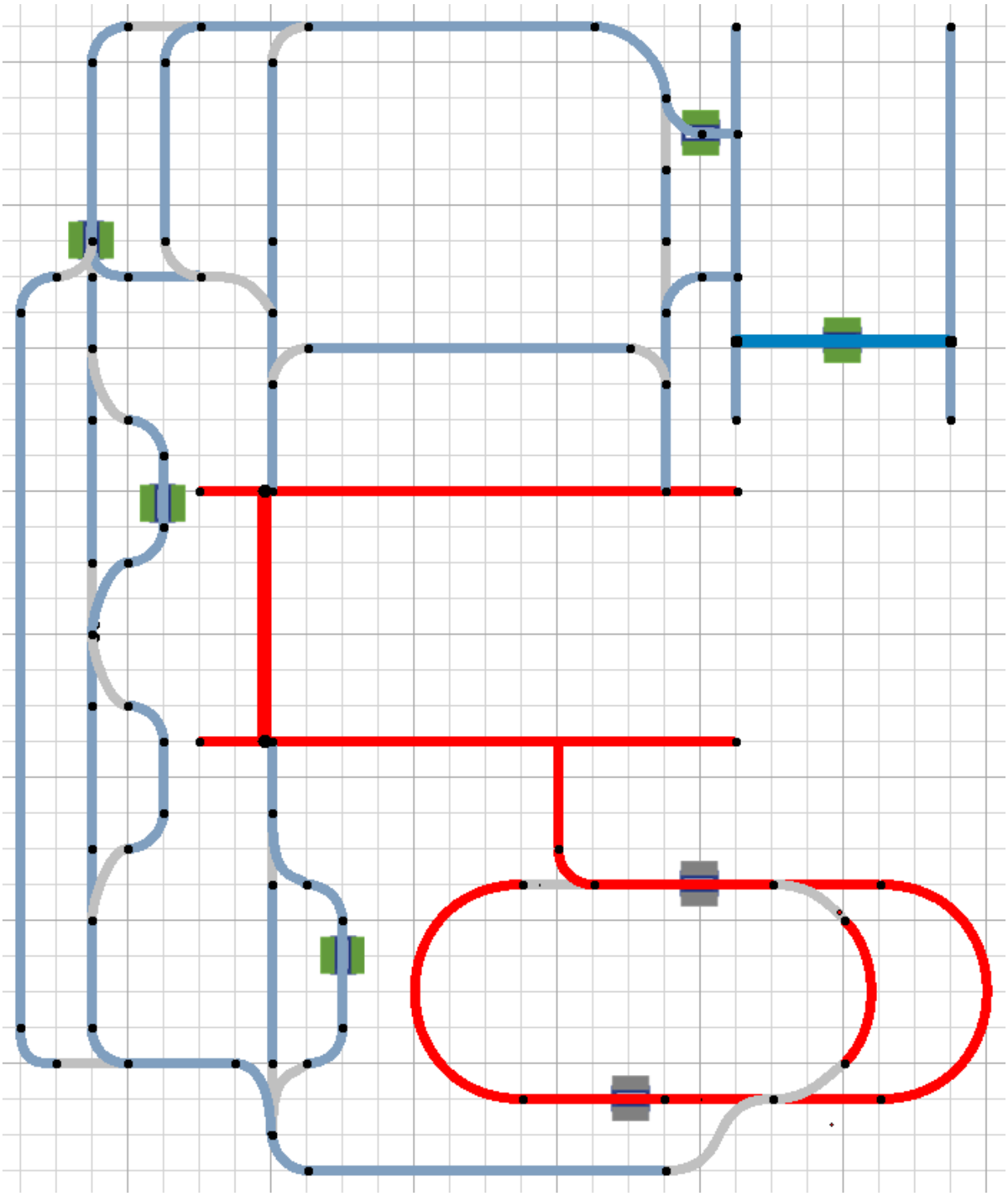


Abbildung 6-16 Emulation der EHB-Versuchsanlage inklusive virtueller Erweiterung

6.2.3.2 Erkenntnisse

Die Möglichkeit, eine komplexe Steuerungslogik wie die einer EHB-Anlage innerhalb einer Emulationsumgebung zu entwickeln bietet die Grundlage zur Parallelisierung von Arbeiten bei der Realisierung einer Materialflussanlage bzw. eines Agentenbaukastens. Die Implementierung der Agenten- und Maschinensteuerungsebene sowie

der mechanische Aufbau der Komponenten können gleichzeitig und parallel durchgeführt werden. Einzige Voraussetzung dafür ist die Definition der – im Allgemeinen sehr einfachen – Schnittstelle zwischen Agent und SPS-Programm. Die Erfahrungen aus dem vorliegenden Szenario zeigen zudem, dass die anschließende Integration der zwei Steuerungsebenen eines Moduls sehr wenig Aufwand verursacht und im Normalfall nur eine Umkonfiguration der Middleware erfordert.

Weiterhin ermöglicht ein Mischbetrieb realer und virtueller Module bzw. Anlagenteile eine bessere Abschätzung der Folgen eines geplanten Umbaus, ohne dass die Anfertigung einer eigenen Simulation notwendig wäre. Allein durch die Modifikation der Topologie und die Instanziierung der neuen Entitäten kann das zukünftige Systemverhalten sofort realisiert und analysiert werden.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung der Ergebnisse

Bei der Planung und Auslegung herkömmlicher Materialflusssysteme sehen sich Anlagenplaner und -betreiber oft einer schwierigen Wahl gegenübergestellt: Automatisierte Systeme sind sehr effizient und günstig im Betrieb, dafür aber meist sehr unflexibel. Manuelle Systeme hingegen bieten eine viel größere – und heutzutage immer wichtiger werdende – Flexibilität, haben aber dafür durch den intensiven Mitarbeitereinsatz wirtschaftliche Nachteile. Die heutige Marktsituation und die sich daraus ergebenden Anforderungen an die Logistik erlauben aber immer weniger eine Entscheidung zwischen Flexibilität und Wirtschaftlichkeit, denn um langfristig am Markt bestehen zu können, müssen Unternehmen sowohl flexibel reagieren als auch kostengünstig operieren. Dies erfordert eine neue Herangehensweise an die Gestaltung automatisierter Systeme und neue Architekturen, die sowohl die selbständige Anpassung einer Materialflussanlage an veränderte Anforderungen als auch deren aufwandsminimalen Umbau ermöglichen.

Dieses Ziel kann durch eine durchgehende Modularisierung und starke Standardisierung von Komponenten erreicht werden, die in fast beliebigen Szenarien eingesetzt und miteinander kombiniert werden können. Diese Vision von einer dezentralen, sich selbst anpassenden Materialflusssteuerung aus autonomen Komponenten wird in den letzten Jahren immer mehr durch das Schlagwort „Internet der Dinge“ verkörpert. Die Materialflusstechnik wird in intelligente Module gegliedert, die untereinander kommunizieren und so ein ständig aktuelles Bild ihrer Umwelt besitzen. Transporteinheiten wie Paletten, Behälter oder Kartons werden selbst intelligent und verhalten sich wie Datenpakete im Internet – sie beauftragen in Eigeninitiative die Fördertechnik mit dem Transport zu ihrem Ziel oder der Durchführung anderer Funktionen. Dieses Vorgehen erlaubt eine starke Reduzierung des Aufwands für die Programmierung und Inbetriebnahme einer Materialflussanlage und ist gegenüber Schwankungen während des Betriebs und auch gegenüber Umbauten viel flexibler als herkömmliche zentral gesteuerte Systeme.

Diese Vision kann ohne Frage begeistern. Allerdings blieb ein Großteil der technischen Fragestellungen zur Realisierung eines solchen Systems, vor allem im Hinblick auf spezielle logistische Anforderungen und die Frage, ob eine dezentrale Steuerung für bestimmte Prozesse überhaupt möglich ist, bislang ungeklärt. Vor allem trifft man dabei auf einen anderen scheinbaren Widerspruch in der Logistik: der Wunsch, möglichst immer standardisierte Module einzusetzen auf der einen Seite und die Tatsache, dass jede Materialflussanlage ganz individuelle, nicht standardisierbare Abläufe umsetzen muss, auf der anderen.

Die vorliegende Arbeit adressiert daher die wichtigsten Aspekte der Gestaltung kundenspezifischer Prozesse sowie der Kommunikation und Steuerung in dezentralen Materialflusssystemen. Ausgehend von grundlegenden Funktionsprinzipien des Internets und theoretischen Aspekten verteilten Problemlösens, vornehmlich aus dem Bereich der Multiagentensysteme, werden die elementaren Bausteine oder Entitäten des Internet der Dinge definiert. Eine Analyse ihrer jeweiligen Funktionalität führt zur Ableitung einer Softwarearchitektur, die sich stark an den Prinzipien objektorientierter Programmierung orientiert und damit die Vorteile der Kapselung und Vererbung nutzbar macht. Dies ermöglicht einen hohen Wiederverwendungsgrad von Code und minimiert so den Programmieraufwand bei gleichzeitig steigender Qualität und Zuverlässigkeit der Software. Die Berücksichtigung besonderer Anforderungen an Steuerungsprogramme für den industriellen Einsatz mündet in einen Überblick möglicher und empfohlener Rechenhardware für verschiedene Arten von Entitäten.

Die Verteilung der Entscheidungskompetenzen auf Fördertechnikmodule, Transporteinheiten und Softwaredienste findet vor dem Hintergrund der angestrebten hohen Wandelbarkeit in der Intralogistik statt. Dabei wird die Prozessflexibilität – die Fähigkeit eines Systems, neue Abläufe und funktionale Verkettungen umzusetzen – als ein entscheidender Faktor identifiziert. Durch die Spezifikation bzw. Auswahl einer geeigneten, formalen Sprache für die freie Definition von Arbeitsabläufen werden Anlagenprogrammierer und -betreiber in die Lage versetzt, die von einzelnen, standardisierten Modulen angebotene Funktionalität auf fast beliebige Weise zu verketten und zu kombinieren. Transporteinheiten sind dann in der Lage, solche Workflows automatisiert zu interpretieren und abzuarbeiten und werden somit zum Träger der Geschäftslogik eines Systems. Die Fördertechnik setzt dann einzelne Aufträge und

Funktionen um, ohne einen Überblick der Gesamtfunktionalität der Anlage zu besitzen oder zu benötigen.

Ein weiterer wichtiger Aspekt bei der Modellierung dezentraler Systeme ist die Definition der auszutauschenden Informationen sowie grundlegender Abläufe bei der Kommunikation. Kapitel 4.1 beschreibt zu diesem Zweck eine Basis- und eine Kommunikationsontologie für das Internet der Dinge. Diese stellt eine erweiterbare Grundlage für die Modellierung des Datenaustauschs in Materialflussanlagen dar. Das bewusst sehr abstrakt gewählte Niveau gewährleistet dabei die Allgemeingültigkeit der definierten Begriffe und Abläufe und ermöglicht somit ihre breite Einsetzbarkeit.

Der Datenaustausch wird in einem weiteren Schritt auch aus technischer Sicht betrachtet. Dabei wird die Kommunikation als neuer und im Internet der Dinge wichtigster Komplexitätstreiber identifiziert: Obwohl durch die Strukturierung und Dezentralisierung der Steuerungslogik eines herkömmlichen Materialflussrechners die Komplexität der einzelnen Funktionskomponenten bzw. Entitäten stark abnimmt, ist das Zusammenspiel, der Nachrichtenaustausch und die Nebenläufigkeit zahlreicher autonomer Einheiten nur schwierig nachzuvollziehen und sehr intransparent. Dies führt zur Betrachtung zweier unterschiedlicher Mechanismen zum Datenaustausch und deren Kombination in einen hybriden Ansatz. Zusätzlich werden auch Aspekte wie die Minimierung des Nachrichtenaufkommens, die Gewährleistung der Datenkonsistenz sowie die Ausfallsicherheit des Systems berücksichtigt.

Nach der Erarbeitung der architektonischen und kommunikationstechnischen Grundlagen werden diese zur Lösung typischer Materialflussprobleme herangezogen. Dabei werden sowohl die Auftragsdisposition und der Transport an sich als auch der Materialflusssteuerung beigeordnete Funktionen wie die Visualisierung oder Möglichkeiten zur manuellen Beeinflussung des Systems behandelt. Weiterhin wird aufgezeigt, wie häufig eingesetzte, komplexe Materialflussstrategien aus verschiedenen Bereichen der stetig und unstetig fördernden Systeme durch dezentrale Entscheidungs- und Kommunikationsprozesse realisiert werden können. Einfache und allgemeingültige Methoden wie Service Discovery und Auktionen bilden dabei die Grundlage zur Entwicklung von Lösungen für sehr vielfältige Szenarien.

Die vorgestellten Konzepte wurden zum einen in der Versuchsanlage des Lehrsuhls fml in drei verschiedenen, miteinander verbundenen Demonstratoren umgesetzt. Diese bestehen aus einem Industrieroboter zur RFID-gesteuerten (De-)Palettierung und Kommissionierung, zwei agentengesteuerten Rollenförderern sowie einer Elektrohängebahnanlage mit insgesamt sechs autonom agierenden Katzen, Weichen und Kranen. Weitere Szenarien aus dem Bereich der Flughafen- und Lagerlogistik wurden in Form einer agentenbasierten Emulation umgesetzt und boten so eine Entwicklungs- und Testumgebung für die vorgestellten Materialflussstrategien.

7.2 Ausblick

Die beschriebenen Konzepte und Lösungen bilden die Grundlage für die Modellierung dezentraler, hochautomatisierter Systeme nach dem Prinzip des Internet der Dinge und für die Entwicklung von Materialflussstrategien. Dabei wurde ganz bewusst auf die Vorgabe oder Empfehlung konkreter Implementierungsmöglichkeiten verzichtet und die erarbeiteten Lösungen so allgemeingültig wie möglich gehalten. In Vorbereitung einer Implementierung für den industriellen Einsatz ist jedoch festzulegen, welche Programmierumgebungen und vor allem welche Agentenstandards zur Steuerung des Materialflusses geeignet sind oder ggf. neu geschaffen werden müssen. In diesem Rahmen wäre auch eine Entscheidungssystematik bzw. ein Leitfaden für die Wahl einer bestimmten Implementierungsform wünschenswert. So muss das grundlegende Paradigma einer „Agentensteuerung“ nicht zwangsläufig mit einem Agentenframework oder nach den FIPA-Richtlinien implementiert werden. Eine objektorientierte Vorgehensweise ist dabei genauso gut denkbar, obwohl in diesem Fall mit einer Verschlechterung der Integrations- und Kombinationsfähigkeit verschiedener Entitäten zu rechnen ist.

Allgemein ist die Sicherstellung der Interoperabilität von Agenten von großer Relevanz, aber bislang ungelöst. Damit Entitäten oder Module verschiedener Hersteller ohne Weiteres miteinander interagieren können, müssen Schnittstellen und auszutauschende Daten, sowie die dabei eingesetzten Protokolle sehr genau definiert werden. Die beschriebene Ontologie macht hier einen ersten Schritt, ist aber noch zu allgemein und abstrakt, um als Grundlage für eine Implementierung zu dienen. Soll ein solcher Standardisierungsprozess für Agentenschnittstellen erfolgreich sein und

sich die erarbeiteten Spezifikationen im realen Einsatz, und nicht nur in der Forschungslandschaft durchsetzen, ist hier eine intensive Mitarbeit der Industrie notwendig. Nur so kann die sehr große Datenvielfalt materialflusstechnischer Anlagen analysiert und strukturiert und ein allgemeingültiges Format definiert werden.

Ein weiterer wichtiger Punkt ist die interne Gestaltung und die einfache Anpassbarkeit von Agenten. Objektorientierte Programmiertechniken bieten hier bereits wichtige Vorteile im Vergleich zur klassischen SPS- oder funktionalen Programmierung. Trotzdem müssen gerade Fördertechnikmodule oftmals sehr speziellen Anforderungen genügen und werden zu diesem Zweck häufig mit kunden- oder projektspezifischen Erweiterungen ausgestattet – so z.B. eine zusätzliche Lichtschranke oder eine Gewichts- oder Profilkontrolle. Die technische Kopplung und der reine Datenaustausch mit solchen Bauteilen werden zwar durch das vorgestellte Middlewarekonzept erleichtert; dabei unbetrachtet bleibt jedoch die funktionale Integration in die Entscheidungs- und Steuerungsprozesse des Agenten. Zum derzeitigen Stand erfordert jede Veränderung der Sensoren und Aktoren eine manuelle Anpassung der Logik, entweder auf Agenten- oder auf Maschinensteuerungsebene. Um dies zu vermeiden, wäre es beispielsweise denkbar, auch die internen Abläufe eines Agenten mittels eines Workflows, einer Regelmaschine oder ähnlichen Tools frei zu konfigurieren.

8 Literaturverzeichnis

[Aal-94] van der Aalst, W.M.P.; van Hee, K.M.; Houben, G.J.: Modelling and analysing workflow using a Petri-net based approach, 1994, Eindhoven University of Technology, Dept. of Mathematics and Computing Science, 1994

[Aal-96] van der Aalst, W.M.P.: Three Good Reasons for Using a Petri-net-based Workflow Management System, In Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96), 1996

[Abe-90] Abel, D.: Petri-Netze für Ingenieure − Modellbildung und Analyse diskret gesteuerter Systeme, Springer Verlag, Berlin, 1990, ISBN 3-540-51814-2

[App-99] Appelrath, H.J.; Freese, T.; Sauer, J.; Teschke, T.: Ein Multiagentensystem für die verteilte Ablaufplanung In: S. Kirn, M. Petsch (Hrsg.): Workshop Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien, Arbeitsbericht Nr. 14, Technische Universität Ilmenau, 1999, S.195-200

[App-00] Appelrath, H.J.; Sauer, J.; Freese, T.; Teschke, T.: Strukturelle Abbildung von Produktionsnetzwerken auf Multiagentensysteme In: KI - Künstliche Intelligenz, 3/00, arenDTaP Verlag, Bremen, 2000, S.64-70

[Arn-95] Arnold, D.: Materialflußlehre, Vieweg, 1995, ISBN 3-528-03033-X

[Bal-06] Balbach, U.: Interdisziplinäre Zusammenarbeit in der Intralogistik erschließt Innovationspotentiale, In: Intralogistik: Potentiale, Perspektiven, Prognosen; Dieter Arnold (Hrsg.), Springer Verlag, Berlin, Oktober 2006, ISBN 3540296573, S. 31-44

[Bau-96] Baumgarten, B.: Petri-Netze - Grundlagen und Anwendungen, Heidelberg: Spektrum Akademischer Verlag, 1996, ISBN 3-8274-0175-5

[Bel-58] Bellmann, R.E.: On a Routing Problem, Quart. Appl. Math. 16, 1958, S. 87-90

[Bel-07] Bellifemine, F.L.; Caire, G.; Greenwood, D.: Developing Multi-Agent Systems with JADE, Wiley, 2007, ISBN: 978-0-470-05747-6

[Beu-06] Das BEUMER autover®-System – ein aktives DCV System, Prospekt der Beumer Maschinenfabrik GmbH & Co. KG Deutschland, http://www.beumer.com/htcms/get_file.php4?pid=346&ref=pdf&lang=de (Abgerufen am 11.11.2008)

[Bil-03] Billington , J.; Christensen, S.; van Hee, K.; Kindler, E.; Kummer, O.; Petrucci, L.; Post, R.; Stehno, C.; Weber, M.: The Petri Net Markup Language: Concepts, Technology, and Tools, In: Proceedings of the ICATPN 2003, Eindhoven, Netherlands, 2003

[Bru-98] van Brussel, H.; Wyns, J.; Valckenaers, P.; Bongaerts, L.; Peeters; P: Reference Architecture for Holonic Manufacturing Systems: PROSA; Katholieke Universiteit Leuven, Mechanical Engineering Department, Leuven (Belgium), http://www.irit.fr/TFGSO/DOCS/TFG2/TFGIISO_Valckenaers.pdf (Abgerufen am 15.10.07)

[Büc-00] Büchter, H.: Steuerungen für Stückgutfördersysteme: Entwicklungspotenziale und Trends, F+H Fördern und Heben Nr.3, 2000, s. 156

[Bus-04] Bussmann, S.; Jennings, N.R.; Woolridge, M.: Multiagent Systems for Manufacturing Control, Springer Verlag, Berlin, 2004, ISBN 978-3-540-20924-9

[Cai-04] Caira, G.; Cabanillas, D.: Jade Tutorial Application-Defined Content Languages and Ontologies, TILAB, November 2004

[Dah-07] Dahl, S.; Derigs, U.: Ein Decision Support System zur kooperativen Tourenplanung in Verbünden unabhängiger Transportdienstleister, In: Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V., Bremen, 2007

[Dem-07] Dematic: Multishuttle fehlerfrei mit dem Schinken auf Achse; In: Logistik & Fördertechnik Nr. 12, Dezember 2007, S. 22-23

[Dij-59] Dijkstra, E.W.: A note in two problems in connexion with graphs, Nummer. Math. 1, 1959, S. 269-271

[Dru-88] Drucker, P.F.: The Coming of the New Organization, Harvard, Business Review, Januar-Februar, 1988, S. 45–53

[Fer-01] Ferber, J.: Multiagentensysteme, Eine Einführung in die verteilte künstliche Intelligenz, Deutsche Übersetzung von Stefan Kirn, Addison-Wesley Verlag, München, 2001

[FIPA-SC00023] FIPA Agent Management Specification, SC00023K, 2003 http://www.fipa.org/specs/fipa00023/SC00023K.html, Abgerufen am 18.09.2008

[FIPA-05] FIPA Standards Committee (FIPA SC), Policies and Procedures, Institute of Electrical and Electronics Engineers (IEEE), 2005

[FIPA-SC00037J] FIPA Communicative Act Library Specification, SC00037J, 2002, http://www.fipa.org/specs/fipa00037/SC00037J.html, Abgerufen am 18.09.2008

[For-56] Ford, L.R.: Network Flow Theory, Rand Copr., Santa Monica, California, USA, 1956

[Gar-04] Garofalakis, J.;, Panagis, Y.;, Sakkopoulos, E.; Tsakalidis, A.: Web Service Discovery Mechanisms: Looking for a Needle in a Haystack? In proceedings of The Fifteenth International ACM Conference on Hypertext and Hypermedia, Santa Cruz, California, USA, August 2004

[Gol-93] Goldman, S.L.; Nagel, R.N.: Management, technology and agility: the emergence of a new era in manufacturing, International Journal of Technology Management 8 (1/2), 1993, S. 18–38

[Gou-98] Gou, L.; Luh, P. B.; Kyoya, Y.: Holonic Manufacturing Scheduling: Architecture, Cooperation Mechanism and Implementation, In: Special Issue of Computers in Industry on Intelligent Manufacturing Systems, Vol. 37, 1998, S. 213-231

[Gru-93] Gruber, T. R.: A Translation Approach to Portable Ontology Specifications, Academic Press, Juni 1993

[GS1] GS1 Germany GmbH: http://www.gs1-germany.de/internet/content/e4/e38/, Abgerufen am 20.05.2009

[Guj-07] Gujo, O.; Schwind, M.; Vykoukal, J.; Wendt, O.: Mehrrundige kombinatorische Auktionen beim innerbetrieblichen Austausch von Logistikdienstleistungen, In: Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V., Bremen, 2007

[Gün-98] Günthner, W.A.: Dynamik im Griff, In: Logistik heute (1998) 12

[Gün-02a] Günthner, W. A.; Heinecker, M.; Wilke, M.: Materialflusssysteme für wandelbare Fabrikstrukturen, Industrie Management 18 (2002) 5, GITO mbH Verlag für Industrielle Informationstechnik und Organisation, Berlin, 2002

[Gün-02b] Günthner, W. A.; Wilke M.: Anforderungen an automatisierte Materialflusssysteme für wandelbare Logistikstrukturen, In: Tagungsband Wissenschaftssymposium Logistik der BVL 2002, Huss-Verlag GmbH, München, 2002, ISBN-Nr. 3-931724-61-1, S. 335-345

[Gün-04] Günthner, W. A.; Wilke, M.: Wandelbare Materialflusssysteme in Minifabriken, In: Tagungsband, Industriekolloquium des Sonderforschungsbereichs 582, München, April 2004, ISBN 3-8316-0378-2;

[Gün-08a] Günthner, W.A.; Chisu, R.; Kuzmany, F.: Internet der Dinge - Steuern ohne Hierarchie, In: F+H Fördern und Heben, Ausgabe September 2008, S. 494-497

[Gün-08b] Günthner, W.A.; Chisu R.; Kuzmany F.: Internet der Dinge - Intelligent verteilt, In: F+H Fördern und Heben, Ausgabe Juli, August 2008, S. 422-425

[Gün-08c] Günthner, W.A.; Chisu, R.; Kuzmany, F.: Internet der Dinge – Zukunftstechnologie mit Kostenvorteil, In: F+H Fördern und Heben, Ausgabe Oktober 2008, S. 556-558

[Guz-97] Guzzoni, D; Cheyer, A.; Julia, L.; Konolige, K.: Many Robots Make Short Work, In: Report of the SRI International mobile robot team at AAAI95, AI Magazine, 18 (1), 1996, S. 55-64

[Han-01] Handrich, W.: Flexible, flurfreie Materialflusstechnik für dynamische Produktionsstrukturen (Dissertation), Herbert Utz Verlag, München, 2001

[Hee-94] van Hee , K.M.: Information System Engineering: a Formal Approach. Cambridge University Press, 1994

[Hom-04a] ten Hompel, M.; Stuer, P.: Realtime logistics, In: Fördertechnik Jahressonderheft, 2004

[Hom-04b] ten Hompel, M.; Stuer, P.: Echtzeitfähigkeit gefordert, In: Fördertechnik, 7-8 (2004), S. 52-54

[Hom-05] ten Hompel, M.; Liekenbrock, D.; Stuer, P.: Realtime Logistics: echtzeitnahe Steuerung von Materialflusssystemen auf Basis autonomer Agenten und Entitäten, In: Logistics Journal: Nicht referierte Veröffentlichungen, 2005, ISSN 1860-5923

[Hom-07] ten Hompel, M.; Libert, S.; Liekenbrock, D.: Analyse der Echtzeitproblematik bei der Steuerung von Stetigfördersystemen, In: Crostack (Hrsg.), ten Hompel (Hrsg.): 1. Kolloquium des Sonderforschungsbereichs SFB 696, Universität Dortmund, 2007, S. 235-255

[Hom-08] ten Hompel, M.: „Logistics by Design“ – ein Ansatz für zukunftsfähige Intralogistik, Serviceorientierte Architektur als Grundlage, Hebezeuge und Fördermittel, Ausgabe 5/2008

[Hop-07] Hoppe, R.: Serviceorientierte Architekturen in Transport und Logistik: Warum kein Weg daran vorbei führt, In: Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V., Bremen, 2007

[Gui-08] Guizzo, E. (Hrsg): Three Engineers, Hundreds of Robots, One Warehouse, In: IEEE Spectrum, Juli 2008

[ISO/IEC-13250] ISO/IEC 13250: Topic Maps, Dezember 1999

[ISO/IEC-15909-1] ISO/IEC 15909-1: High-level Petri nets - Part 1: Concepts, definitions and graphical notation, März 2008

[ISO/IEC-15909-2] ISO/IEC 15909-2: High-level Petri nets - Part 2: Transfer Format, Working Draft, 2005

[IEC-61131-3] IEC 61131-3: Programmable controllers - Part 3: Programming languages, 2003

[IEC-61499-1] IEC 61499-1: Function blocks - Part 1: Architecture, 2005

[IdD] Homepage des BMBF-Forschungsprojektes “Internet der Dinge”, http://www.internet-der-dinge.de/ (Abgerufen am 26.08.2009)

[Irr-07] Irrgang, R.: Standards für erfolgreiche Logistikprojekte, Projekt Systemarchitektur Intralogistik (SAIL) durch VDMA und VDI initiiert; In: Logistik für Unternehmen Sonderdruck, Springer VDI Verlag, März 2007, http://www.sail-modell.de/download/sd-lfu07-03.pdf, Abgerufen am 5.06.2009

[IWS] Internet World Stats, http://www.internetworldstats.com/, Abgerufen am 10.07.2008

[JADE] Java Agent Development Framework – an open source platform for peer-to-peer agent based applications, http://jade.tilab.com/ (Abgerufen am 04.02.2009)

[Jen-91] Jensen, K.; Rosenberg, G.. High-level Petri-Nets: Theory and Application. Springer Verlag, Berlin, 1991, ISBN 038754125X

[Jen-92] Jensen, K.: Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Springer Verlag, Berlin. 1992. ISBN 0387555978

[Joh-94] Johnson, D.B.: Routing in Ad Hoc Networks of Mobile Hosts, In: Proceedings of the Workshop on Mobile Computing Systems and Applications, IEEE Computer Society, Santa Cruz, CA, Dezember 1994, S. 158–163

[Jun-04] Jungbluth, V.: MultiShuttle als Alternative in der Behälterlagertechnik, In: Logistik für Unternehmen, Juni 2004, S. 35-37

[Kan-97] Kanchanasevee, P.; Biswas, G.; Kawamura, K.; Tamura, S.: Contract-Net Based Scheduling for Holonic Manufacturing Systems, In: Proceedings of SPIE, “Architectures, Networks, and Intelligent Systems for Manufacturing Integration”, Pittsburgh, Pennsylvania, Oktober 1997, S. 108-115

[Kee-03] McKeeth, J.: A Guide to Peer-2-Peer, In: Proceedings of BorCon 2003

[Kir-07] Kirschke, C.: Erstellung von Dispositionsvorschlägen durch ein Multiagenten-System, In: Proceedings „INFORMATIK 2007 – Informatik trifft Logistik“, Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V. (GDI), 24.-27. Bremen, September 2007

[Kop-07] Kopfer, H.; Krajewska, M.A.: Praktische Aspekte der kollaborativen Auftragsdisposition, In: Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V., Bremen, 2007

[Koe-84] Koestler, A.: Die Wurzeln des Zufalls. Scherz Verlag, München 1984, ISBN 3502153868

[Koe-76] Koestler, A.: The Ghost in the machine, New York, 1976

[Kla-06] Klaus, P.; Kille, C.: Die TOP 100 der Logistik. Marktgrößen, Marktsegmente und Marktführer in der Logistikdienstleistungswirtschaft, 4. Auflage Deutscher Verkehrsverlag, 2006

[Knu-94] Knudsen, J.K. ; MacConaill, P.A. ; Bastos, J. ; European Commission, Directorate-General for Industry: Sharing CIM Solutions. Linking Innovation with Growth, Amsterdam: IOS Press, 1994 (Advances in Design and Manufacturing 5). ISBN: 90-5199-194-0, S.84-93

[KS-08] Kiva Systems Deploys 1,000th Autonomous Mobile Robot, Press Release, http://www.kivasystems.com/news_PR_1000RB.html (Abgerufen am 11.11.2008)

[Küv-07] Küveler, G.; Schwoch, D: Informatik für Ingenieure und Naturwissenschaftler 2, 5. Auflage, Vieweg, 2007, ISBN 978-3-8348-0187-6

[Kwa-04] Kwangyeol, R.: Fractal-based Reference Model for Self-reconfigurable Manufacturing Systems. Ph.D. Thesis of Pohang University of Science and Technology. Pohang Korea 2004. http://organ.postech.ac.kr/papers/Dissertation_FrMS%20(Kwangyeol%20Ryu).pdf, (Abgerufen am 30.10.2007)

[Lan-97] Lang, M: Effizienz von Verfahren zur adaptiven und verteilten Verkehrslenkung in Paketvermittlungsnetzen, Universität Stuttgart, Institut für Kommunikationsnetze und Rechnersysteme, 1997

[Log-07] Logistik für Unternehmen: Standards für erfolgreiche Logistik-projekte, Springer-VDI-Verlag Düsseldorf, Sonderdruck, 2007

[Lib-10] Libert, S.; Chisu, R.; Keutner, K.: Eine Ontologie für das Internet der Dinge, In: Günthner, W.A.; ten Hompel, M. (Hrsg.): Internet der Dinge in der Intralogistik, Springer, Berlin, Januar 2010, ISBN 3642048951

[Lüt-98] Lüth, T.: Technische Multi-Agenten-Systeme: verteilte autonome Roboter- und Fertigungssysteme, Carl Hanser Verlag, München, Wien, 1998

[Oes-06] Oesterreich, B.: Analyse und Design mit UML 2.1 8. Auflage, Oldenbourg Verlag, 2006, ISBN 3-486-57926-6

[Mat-88] Mattern, F.: Virtual Time and Global States of Distributed Systems in Proceedings of the Workshop on Parallel and Distributed Algorithms, Chateau de Bonas, France, Elsevier, October 1988, S. 215–226

[May-09] Mayer, S. H.: Development of a completely decentralized control system for continuous conveyors, Dissertation, Universität Karlsruhe (TH), April 2009

[Mei-96] Meinber, U.: Chancen mit dezentralen Steuerungen, In: Frankfurter Allgemeine Zeitung, 22.08.1996

[Mei-04] Meinel, C.; Sack, H.: WWW - Kommunikation, Internetworking, Web-Technologien, Springer-Verlag, Berlin, Heidelberg, New York, 2004

[Mön-99] Mönch, G.; Podewils, M. Von; Schmalfuß, V.; Aron, J.; Frauenhoffer, F.; Sturm, R.: Objektorientierte Modellierung des Materialflusses für eine flexible Produktion, In: Logistik im Unternehmen 13 (1999) 10, S. 44-47

[Mur-89] Murata, T.: Petri Nets: Properties, Analysis and Applications, In: Proceedings of the IEEE, 77(4):541-580, 1989

[Nag-08] Nagel, L.; Roidl, M.; Follert, G.: The Internet of Things: On Standardisation in the Domain of Intralogistics, In: Proceedings of the First International Conference on The Internet of Things, Zürich, März 2008, S. 16-21

[Nwa-96] Nwana, H.S.: Software Agents: An Overview, In: Knowledge Engineering Review, Vol. 11, Nr 3, September 1996, S. 1-40

[PNK] Petri Net Kernel, Humboldt Universität Berlin, http://www2.informatik.hu-berlin.de/top/pnk/, Abgerufen am 29.08.2008

[RFC-1058] RFC1058 Routing Information Protocol, 1988 http://www.faqs.org/rfcs/rfc1058.html, Abgerufen am 06.08.2009

[RFC-1723] RFC1723 RIP Version 2 - Carrying Additional Information http://www.faqs.org/rfcs/rfc1723.html, Abgerufen am 06.08.2009

[RFC-4728] RFC4728 The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4 http://www.faqs.org/rfcs/rfc4728.html, Abgerufen am 06.08.2009

[Rit-00] Ritter, A.; Braatz, A.; Einz, G.: Agentensysteme in der Produktion, In: Tagungsband / SPS IPC Drives Nürnberg 2001, 11. Fachmesse und Kongress, Hüthig, Heidelberg, 28-30 November 2000

[Rit-03] Ritter, A.: Ein Multi-Agenten-System für mobile Einrichtungen in Produktionssystemen, Dissertation, Universität Stuttgart, Jost Jetter Verlag Heimsheim, 2003, ISBN 3-936947-07-4

[Roi-07] Roidl, M.; Follert, G.: Simulation von multiagentenbasierten Materialflusssteuerungen, In: Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V., Bremen, 2007

[Rus-04] Russell, S. J.; Norvig, P.: Künstliche Intelligenz. Pearson Studium, München, 2004, ISBN 3-8273-7089-2, S. 55-84

[San-96] Sandholm, T.: Negotiation Among Self-Interested Computationally Limited Agents, Dissertation, Department of Computer Science, University of Massachusetts, 1996

[Sau-00] Sauer, J.; Freese, T.; Teschke, T.: Towards agent-based multi-site scheduling; In: Sauer J., Köhler J. (Hrsg.): Proceedings of the ECAI 2000 Workshop on the New Results in Planning, Scheduling, and Design, 200, S.123-130

[Sau-03] Sauer, J.; Appelrath, H.-J.: Scheduling the Supply Chain by Teams of Agents, In: Sprague R.H. (Hrsg.): Proc. of the 36rd Hawaii International Conference on System Sciences (HICSS-36), 2003

[Sca-05] Schanz, M.: Einführung in die Verteilte Künstliche Intelligenz; Universität Stuttgart, 2005

[Sch-01] Schulte, H.: Fabrikplanung bewegt sich, VDI-Nachrichten, 15. Juni 2001, Nr. 24, S. 27

[Sch-04] Schulte, W.: Routing – Wegewahl durch die Netze, In: Funkschau 1/2004, S. 51-52

[Sch-05] Schroer, W.: MultiShuttle – universell einsetzbar, In: Hebezeuge Fördermittel, Februar 2005, S. 34-36

[SFB582-04] SFB 582: Marktnahe Produktion individualisierter Produkte, TU München, 2004, http://www.sfb582.de, Abgerufen am 10.12.2008

[Ski-99] Skiera, B.; Revensdorff, I.: Auktionen als Instrument zur Erhebung von Zahlungsbereitschaften, Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung, 1999, S.224-242

[Sun-05] Sun Microsystems: Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI), Sun Developer Network, 2005

[Tse-03] Tseng, M.; Piller, F.T. (Hrsg.): The Customer Centric Enterprise: Advances in Mass Customization and Personalization, New York, Berlin, 2003

[Ung-99] Ungerer, T: Parallelrechner und parallele Programmierung, Spektrum Akademischer Verlag, November 1999, ISBN 382740231X

[VDMA-15276] Verband Deutscher Maschinen- und Anlagenbau (VDMA): VDMA-Einheitsblatt 15276: Datenschnittstellen in Materialflusssystemen, Beuth-Verlag, Berlin, 1994

[VDI/VDMA-5100] Verein Deutscher Ingenieure (VDI) und Verband Deutscher Maschinen- und Anlagenbau (VDMA): VDI/VDMA 5100 Blatt 1, Systemarchitektur für die Intralogistik (SAIL) – Grundlagen, Richtlinienentwurf, 2008-03

[W3C-04] World Wide Web Consortium: OWL Web Ontology Language Overview, W2c Recommendation 10 February 2004; http://www.w3.org/TR/owl-features/, Abgerufen am 12.05.2009

[Wan-00] Wang, Z.; Pineros, C.; Takahashi, T.; Kimura, Y.; Nakano, E.: Arrangement and Control of a Cooperative Multi-Robot System for Object Transportation, In: Intelligent Autonomous Systems 6, 2000, S. 196-203

[War-93] Warnecke, H.-J.: Revolution der Unternehmenskultur Das Fraktale Unternehmen, Springer Verlag, Berlin, 1993

[War-99] Warnecke, H.-J.: Vom Fraktal zum Unternehmensnetzwerk, Springer Verlag, Berlin, 1999

[Web-02] Weber, M.; Kindler, E.: The Petri Net Markup Language, In: "Petri Net Technology for Communication Based Systems" in the LNCS series "Advances in Petri Nets", 2002

[Weh-04] Wehking, K.-H.: Das materielle Internet - der Treiber für neue Logistikstrukturen und -komponenten, 2. Wissenschaftssymposium Logistik der BVL, Berlin, 2004

[Wen-07] Wenger, W.; Geiger, M. J.: Ein agentenbasiertes Konzept zur interaktiven Lösung multikriterieller Tourenplanungsprobleme, In: Beiträge der 37. Jahrestagung der Gesellschaft für Informatik e.V., Bremen, 2007

[Wil-06] Wilke, M.: Wandelbare automatisierte Materialflusssysteme für dynamische Produktionsstrukturen, Dissertation, TU München, 2006

[Wip-06] Wippler, D.: Algorithmen und Grafik mit Java, Books on Demand GmbH, Norderstedt, 2006, ISBN 3-8334-4452-5

[Wir-01] Wirth, S.; Baumann, A.: Wertschöpfung durch vernetzte Kompetenz – Schlanke Kompetenzkooperation, München, Huss-Verlag, 2001

[Woo-02] Wooldridge, M.: An introduction to multi agent systems, 2002, ISBN 0 47149691X

[Yam-00] Yamamura, M.: A Case Study of Indircet Control on Large Scale Evolutionary Multi-agent Systems in Dynamic Environment, In: Intelligent Autonomous Systems 6, 2000, S. 219-224