Projektgruppe FAISE

Endbericht
im Rahmen des Masterstudiums

Betreuer: Prof. Dr.-Ing. Jürgen Sauer
Dipl.-Ing. (FH) Arne Stasch
Dipl.-Inform. Jan-Hinrich Kämper
Prof. Dr.-Ing. Axel Hahn

Vorgelegt von: Berthe Pulcherie Ongnomo
Chancelle Merveille Tematio Yme
Christopher Schwarz
Jan Paul Vox
Jan-Gerd Meß
Jannik Fleßner
Malte Falk
Matthias Aden
Michael Goldenstein
Nagihan Aydin
Raschid Alkhatib
Simon Jakubowski

Abgabetermin: 30. September 2014

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In diesem Kapitel wird ein einführender Überblick über die Projektgruppe Fully Autonomous Intralogistic Swarm Experiments gegeben, die im Rahmen der Masterstudiengänge Informatik und Wirtschaftsinformatik in der Abteilung Systemanalyse und -optimierung der Carl von Ossietzky Universität Oldenburg stattgefunden hat. Das Projekt lief über einen Zeitraum von zwei Semestern: Wintersemester 2013/2014 und Sommersemester 2014.

1.1 Motivation

Im Zeitalter der Globalisierung werden hohe Anforderungen an die Leistungsfähigkeit von modernen Intralogistiksystemen gestellt. Neben einem hohen Automatisierungsgrad wird gleichzeitig auch eine möglichst hohe Flexibilität gefordert, da sich Anforderungen im logistischen Umfeld häufig ändern (Vgl.[26]).

Stetigförderer bieten die Möglichkeit einen automatisierten Materialfluss einzurichten. Es handelt sich dabei um Transportsysteme, die Güter kontinuierlich und automatisiert entlang eines festgelegten Transportwegs befördern (Vgl.[6]). Ein solches System könnte beispielsweise ein Netz von Schienen sein. Nachteile dieser Systeme sind insbesondere Unflexibilität und schlechte Skalierbarkeit. Ändern sich Anforderungen in einem Logistiksystem, dann stoßen Stetigförderer schnell an ihre Grenzen. Transportwege sind festgelegt und können nicht ohne einen gewissen Aufwand geändert werden. Auch kann die Anzahl an Gütern, die pro Zeiteinheit befördert werden kann, nicht ohne eine Änderung am Transportnetz maximiert werden.

Eine Alternative zu Stetigförderern sind Fahrerlose Transportsysteme (FTS). FTS sind ein Gesamtsystem aus Fahrerlosen Transportfahrzeugen, die Ware automatisiert befördern, und der Infrastruktur, die zum Betrieb der Transporteinheiten notwendig ist (Vgl.[19]). Fahrerlose Transportsysteme sind wesentlich flexibler als Stetigförderer. Müssen mehr Güter befördert werden, so können zusätzliche Transporteinheiten aktiviert werden. Folglich sind FTS problemlos skalierbar und können schnell auf veränderte Anforderungen in einem Intralogistiksystem eingestellt werden. FTS bieten einen automatisierten Warenfluss bei gleichzeitig hoher Flexibilität und entsprechen somit den Anforderungen, die an moderne Intralogistiksysteme gestellt werden.

Es bietet sich an ein System zu entwickeln, das basierend auf FTS, einen vollautomatisierten Warenfluss implementiert, um verschiedene Fragestellungen zu untersuchen. Wie muss ein solches System aufgebaut sein, welche Kommunikationsabläufe sind zwischen

den verschiedenen Akteuren notwendig, welche Anforderungen werden an Hard- und Software gestellt und wie flexibel ist ein solches System?

1.2 Zielsetzung

Im Rahmen der Projektgruppe FAISE soll ein System entwickelt werden, das den vollautomatisierten Warenfluss in einem Lager auf Basis von Fahrerlosen Transportsystemen simuliert. Dabei sollen die Transporteinheiten nicht zentral gesteuert werden, sondern dezentral als Schwarm agieren. Das Gesamtsystem besteht aus zwei Teilsystemen, einem physisch vorhandenem System und einer softwarebasierten Simulation.

Das physische System beinhaltet Fahrerlose Transporteinheiten und Lagerrampen, die miteinander über ein Sensornetzwerk kommunizieren und deren Steuerung auf Basis von Mikrocontrollern erfolgt. Ziel ist es den Materialfluss von den Transporteinheiten und Rampen vollständig autonom und ohne dezentrale Steuerung durchzuführen.

Das rein softwarebasierte System implementiert ebenfalls einen automatisierten Warenfluss. Die Software soll als Abbild des physischen Systems realisiert werden. Die Akteure, ihre physikalischen Eigenschaften (Geschwindigkeit etc.) und ihr Verhalten im Einzelnen sowie als Schwarm sollen in der Software abgebildet werden. Beide Systeme laufen unabhängig voneinander und sollen in einem festen Einsatzszenario erprobt werden.

1.3 Einsatzszenario

Das Einsatzszenario besteht aus n fahrerlosen Transporteinheiten in einem Umschlagslager. Zusätzlich sind m Rampen verfügbar an denen Pakete zwischengelagert werden können. Im Gegensatz zu einem herkömmlichen Lager, werden Waren in einem Umschlagslager nur kurzfristig gelagert, um anschließend weitertransportiert zu werden. Es herrscht ein kontinuierlicher Materialfluss. Jedes Paket, das ins Lager gebracht wird, ist eindeutig identifizierbar und wird zu einem definierten Zeitpunkt ins Lager gebracht und wieder abgeholt. Die Rampen im Lager sollen drei unterschiedliche Zwecke erfüllen. Eingangsrampen dienen der Warenannahme, Zwischenrampen der Zwischenlagerung. Pakete werden zum Ausgangslager gebracht und zum Zwecke des Weitertransports dort abgeholt. Auf Basis des Einsatzszenarios wird im Rahmen der Anforderungen ein Ablaufszenario erstellt, das die Interaktionen zwischen den Akteuren beschreibt auf deren Basis eine automatisierte, dezentrale Abwicklung des Materialflusses erfolgen kann.

1.4 Komponenten

2 Stand der Technik

Die Fahrerlosen Transportsysteme und die Materialflusssysteme sind Prozesse der Logistik. In den vergangenen Jahren hat die Verbreitung Fahrerloser Transportsysteme (FTS) stark zugenommen. Beim Einsatz von FTS stellen sich vielfältige Konfigurierungs- und Planungsprobleme, so auch die Einsatzplanung für die einzelnen Fahrerlosen Transportfahrzeuge. [vgl. 5, S. 2]. Der innerbetriebliche Materialfluss von Industrieunternehmen bietet fahrerlosen Transportsystemen (FTS) zahlreiche Einsatzgebiete: Sie verketten Produktionsprozesse, verknüpfen Fertigungsstationen oder ganze Betriebsbereiche und beschicken Montageplätze. Darüber hinaus dienen sie als mobile Werkbank oder versorgen und entsorgen Lager unterschiedlicher Art. Um die Systemvorteile von Fahrerlosen Transportsystemen und Materialflusssystemen zu optimieren, braucht man ein maßgerechtes Wissen auf Ihr spezifisches Anlagekonzept abzustimmen. Wichtige Kriterien sind allerdings z. B. die Einbindung der Fahrerlosen Transportsysteme in den gesamtbetrieblichen Materialfluss, die Anpassung an die vorhandenen Steuerungshierarchien und die optimale Auslegung der Technik in Bezug auf Fahrzeugbauart, Lastaufnahmemittel, Energiekonzept, Kommunikation und Leitsystem [vgl. 17, S. 2]. Ziele von Fahrerlosen Transportsystemen und Materialflusssystemen sind Kostensenkung durch Personaleinsparung, Verringerung von Transportschäden, hohe Zuverlässigkeit in Vorgängen und bessere Materialflussplanung. Dieses Kapitel wird in drei Teile gegliedert. Der erste Teil wird die Fahrerlosen Transportsysteme bzw die Orientierungs- und die Steuerungssysteme vorstellen und erklären; der zweite Teil ist eine Darstellung der Materialflusssysteme und ihrer verschiedenen Funktionen und der dritte Teil wird erklären, wie fahrerlose Transport- und Materialflusssysteme in großen Firmen wie Volkswagen und BMW Anwendung finden.

2.1 Fahrerlose Transportsysteme

Nach dem Verein Deutscher Ingenieure 2510 bestehen FTS im Wesentlichen aus „einem oder mehreren Fahrerlosen Transportfahrzeugen (FTF), einer Leitsteuerung, Einrichtung zur Standortbestimmung und Lagererfassung, Einrichtungen zur Datenübertragung sowie Infrastruktur und peripheren Einrichtungen“. In seinem Buch Transport und Lagerlogistik fasst Martin die Definition von VDI 2510 eines FTS zusammen. Er beschreibt ein FTS als mit FTF ausgestattete rechnergesteuerte Materialflussanlagen zum automatischen Transport von Gütern im innerbetrieblichen Materialfluss[vgl. 14, S. 262f]. Bei FTS handelt es sich um flurgebundene Fördersysteme mit automatisch geführten FTF. Die einzelnen FTF befördern Ladungsträger zwischen zwei oder mehrere Stationen innerhalb eines Gebietes. Die Fahrzeugsteuerung erfolgt automatisch und rechnergestützt. Der Einsatzbereich von FTS ist generell überwiegend innerbetrieblich ausgerichtet. In diesen Rahmen übernehmen FTS

sowohl reine Förderaufgaben, wie Verkettung von Fertigungs- und Montageeinrichtungen als auch Aufgaben der Lagerbedienung und Kommissionierung[vgl. 5, S. 3]. Das FTS ist eine Technik, die im Vergleich gegenüber Stetigfördersystemen eine hohe Anpassungsfähigkeit an die sich ändernden Marktsituationen zum Vorteil hat. Daher konzentrieren die Forschungs- und Entwicklungsaktivitäten sich heutzutage auf die sog. „Zellulären Fördersysteme“, in welchen stetige Förderanlagen zur Verknüpfung von Logistischen Funktionen durch individuelle, autonom arbeitende FTF ersetzt werden (vgl. Ten Hompel; Heidenblut, 2008). Die Haupteinsatzgebiete des FTS liegen nun in der Intralogistik. Also bei der Organisation, der Steuerung, der Durchführung und der Optimierung des innerbetrieblichen Waren- und Materialflusses und Logistik, der Informationsströme sowie des Warenumschlags in Industrie, Handel und öffentlichen Einrichtungen. Z. B. Automobil- und Zulieferindustrie, Papiererzeugung und –verarbeitung, Elektroindustrie, Getränke-, Lebensmittelindustrie, Baustoffe, Stahlindustrie, Kliniklogistik[vgl. 18, S. 13]. FTS bestehen im Wesentlichen aus drei Systemkomponenten: Die Fahrerlosen Transportfahrzeuge, das Orientierungssystem, das Steuerungssystem.

2.2 Fahrerlose Transportfahrzeuge

Die FTF sind flurgebundene Fördermittel mit eigenem Fahrantrieb, die automatisch geführt, gesteuert und berührungslos geführt werden. Sie dienen dem Materialtransport, und zwar zum Ziehen und/oder Tragen von Fördergut mit aktiven oder passiven (FTF mit passiver Lastaufnahme werden von anderen Fördermitteln gezogen oder manuell mit den Gütern bestückt) Lastaufnahmemittel [10](VDI 2510). Da das FTS mit fahrerlosen Aspekten systematisiert ist, ergeben sich dann auf der funktionalen Ebene Unterschiede zu fahrerbedienten Fahrzeugen, wie z. B. den klassischen Gabelstaplern und FTF. Im Rahmen dieser Arbeit wird nur auf eine Kategorie von FTF tiefer eingegangen: das Mini-FTF. Die Mini-FTF sind kleine, schnelle, intelligente und flexible Fahrzeuge, die extrem schnell Bedürfnisse befriedigen können. Heutzutage arbeiten viele Universitäten in der ganzen Welt im Bereich der Schwarm-Experimente. Hier sollen die kleinen FTF intelligent miteinander arbeiten. Die Fahrzeuge sollen sich ohne eine eigene separate FTS-Leitsteuerung untereinander verständigen, Strategien entwickeln und gemeinsam Arbeiten ausführen. Die Forschungsgebiete heißen Agentensysteme und Schwarmtheorie. Die Mini-FTF können nur intralogistische Aufgaben ausführen. Dennoch sind viele unkonventionelle Einsatzfälle denkbar. Die Kommissionierung (eine ausführliche Begriffserklärung wird im Teil Materialfluss gegeben) ist die verbreitetste Anwendungsmöglichkeit von Mini-FTF[vgl. 18, S. 105]. Als Zusammenfassung kann man sagen, dass die Fahrzeugsteuerung die Systemsicherheit, das Energiemanagement, das Lastaufnahmemittel und die Lenkung eines FTF gewährleistet. Ein FTF kann ohne Energie nicht funktionieren. Damit ein FTF seine Aufgabe erfüllen kann, ist eine Ener-

gieversorgung notwendig. Die FTF können durch Akkus oder Traktionsbatterien oder mit Hilfe eines Induktionssystems oder einer Stromschiene mit Energie versorgt werden. Jedoch können die beiden Versorgungsarten gekoppelt werden, um ein Hybridsystem zu bekommen. Die Notwendigkeit der Existenz einer Ladestation in einem FTS ist unumstritten. Die FTF müssen immer mit Energie versorgt werden. Je nachdem wie die FTF programmiert sind, kann ein FTF bei Energiebedarf selber zur Ladestation fahren oder von einem Auftraggeber (Mensch) zur Ladestation geführt werden.

2.2.1 Orientierungssystem bzw. Navigation

Das Orientierungssystem bzw. die Navigation dient zur Lokalisierung des Fahrzeugs. Sie ist ein Hilfsmittel zur Berechnung des sichersten Wegs, um das Ziel zu erreichen. Außerdem dient die Navigation auch zur Vermeidung von eventuellen Kollisionen. Sie gilt sowohl für die Orientierung als auch für die Sicherheit des Fahrzeuges und seines Umfeldes. Während seiner Bewegung bzw. Orientierung folgt das FTF einer physischen oder virtuellen Linie (Spur), damit es sein Ziel gefahrlos erreichen kann. Aufgrund eines Sicherheitssystems s ollte das FTF bei Kollisionsgefahr oder wenn Hindernisse vor ihm stehen, sofort anhalten. Unter Navigationshilfe versteht man nicht nur die Positionierung und Orientierung des Fahrzeuges sondern auch, wohin das Fahrzeug gelangen würde, wenn keine auf seine Bewegung verändernden Maßnahmen ergriffen würden. Die Steuerung sagt, was zu tun ist, und die Navigation bestimmt, auf welchem Weg das gewünschte Ziel sicher zu erreichen ist bzw. ob das FTF einen vorgegebenen Weg weiter verfolgen oder einen alternative Weg nehmen soll. Die Steuerung von fahrerlosen Transportfahrzeugen, deren Grundfunktionen und der Umgang mit diesen werden in den VDI-Richtlinien [VDI92], [VDI94], [VDI04] vorgestellt. Für das Konstrukt der fahrerlosen Transportsysteme werden verschiedene Ansätze verfolgt, die abhängig vom System verschiedene Konstruktionsbemühungen auf das Fahrzeug oder auf der Strecke erfordern. Es gibt mehrere Navigationsverfahren: die physische Leitlinie, die Orientierung durch Magnetmarken, das Global Positioning System (GPS) und die Lasernavigation[vgl. 18, S. 112].

- **Die physische Leitlinie:** Fahrerlose Transportsysteme, die auf physischen Leitlinien navigieren bzw. fahren, benutzen Einrichtungen am oder im Fußboden. Die verschiedenen Varianten sind:
- **Orientierung durch optische Leitspur:** Bei dieser Methode wird ein Farbstrich mit deutlichem Farbkontrast zum umgebenden Boden entweder lackiert oder mit einem speziellen Gewebeband aufgebracht. Eine geeignete Kamerasensorik unter dem Fahrzeug nutzt Kantendetektions-Algorithmen und errechnet so die Ansteuerungssignale

für den Lenkmotor[vgl. 18, S. 112]. Optische Verfahren dienen durch eine ständige Kurskorrektur dazu, eine hohe Fahrgenauigkeit zu erreichen.

- **Orientierung durch induktive Leitspur:** Diese Methode der Navigation fahrerloser Transportfahrzeuge ist profitabel aufgrund der permanenten Kurskorrektur und außerdem besonders zuverlässig und fahrzeugseitig durch die Nutzung einfacher Komponente zu realisieren. Es ist möglich, die Stromversorgung der Fahrzeuge fahrbahnseitig zu realisieren, sodass die Nutzung schwerer Akkumulatoren entfällt. Jedoch sind Systeme mit Leitdrahtsteuerung nicht flexibel und in der Konstruktion sehr teuer.

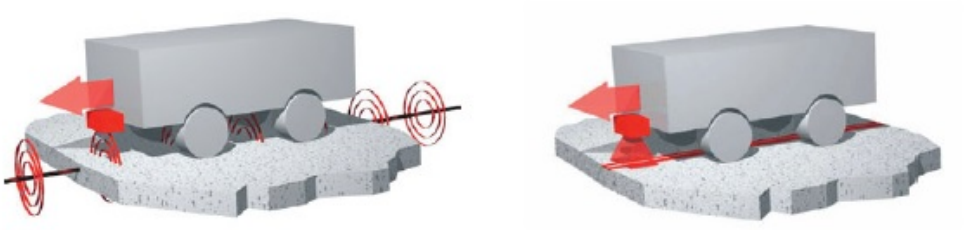

Abbildung 1: Prinzipskizze zur induktiven und optischen Spurführung (Quelle: Günter Ullrich, 2011 S. 79)

- **Orientierung durch Magnetmarken:** Eine weitere Möglichkeit der Steuerung ist die Abtastung von Magnetstreifen oder magnetischen Markierungen auf der Straßenoberfläche. Dabei bedarf es zur Berechnung der Leitlinie entweder der Koppelnavigation, oder der Peilung von in regelmäßigen Abständen in den Boden eingelassenen Marken. Diese Marken können rein passive Dauermagnete oder aber quasi-aktive Transponder sein [vgl. 18, S. 80]. Das Bild 2 ist eine Repräsentation der Navigation durch Magnetstreifen.

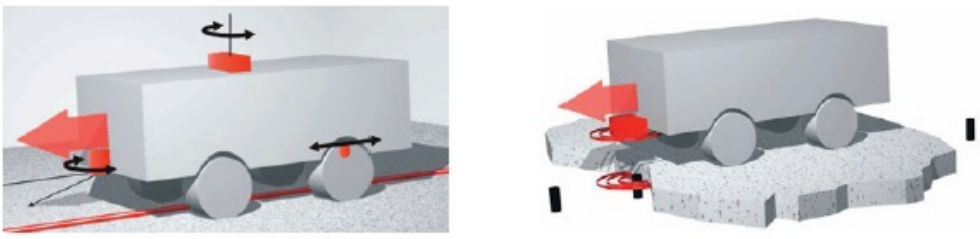

Abbildung 2: Prinzipskizze zur Koppelnavigation (links) und zur Magnet- bzw. Transpondernavigation (rechts) (Quelle: Günter Ullrich, 2011 S. 79)

- Bei der Lasernavigation bestimmt der Laserscanner die Position des FTF, dazu kommen noch optische Sensoren für die Erkennung von Hindernissen wie z. B. Menschen. Lasergeführte FTS bieten einen hohen Wert an Flexibilität, da sie ohne Bodeninstallation funktionieren. Nur bei engerem Raum kann die Lasernavigation nicht so effizient wie z. B. eine induktive Spurführung sein, wenn viele Fahrzeuge zum Einsatz kommen. Um die Systemvorteile einer Lasernavigation optimal zu nutzen, benötigt man allerdings ein passendes Anlagenkonzept. Die wichtigsten Kriterien sind: die Einbindung in das gesamtbetriebliche Materialflusssystem, die Anpassung an die vorhandenen Steuerungshierarchien und die optimale Auslegung der Technik in Bezug auf Fahrzeugbauart, Lastaufnahmemittel, Energiekonzept, Kommunikation und Leitsystem. Ein Aspekt, der für das Laser-geführte FTS spricht, ist die Wirtschaftlichkeit. Und dies trotz der Alternativen Elektro-, Low-Cost- sowie induktiv geführten FTS. Letztere lassen sich so einrichten, dass sie auch auf leitdrahtlosen, rein rechnergeführten Teilstrecken verkehren können. Keinerlei kostenintensive Bodeninstallation benötigt dagegen das über Lasersensor gesteuerte, völlig frei navigierende Laser-FTS. Die Fahrzeuge orientieren sich lediglich an im Raum verteilten Reflektoren und mit Hilfe der Kombination von Winkel- und Distanzmessung[17, S. 2]. Das Bild 3 ist eine Visualisierung der Lasernavigation.

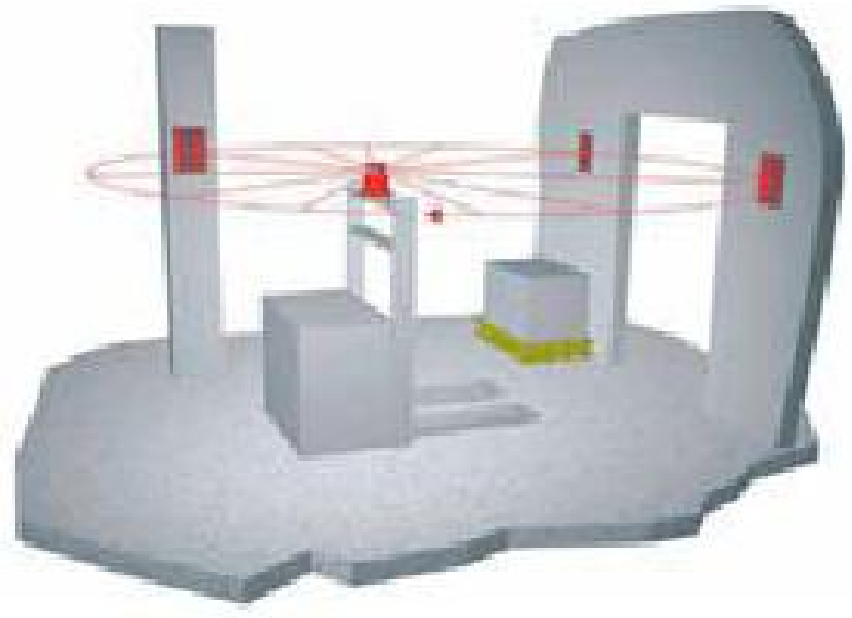

Abbildung 3: Prinzipskizze zur Koppelnavigation (links) und zur Magnet- bzw. Transpondernavigation (rechts) (Quelle: Günter Ullrich, 2011 S. 79)

- **Orientierung durch GPS:** Seine Anwendung im Bereich der Fahrzeugsteuerung wird in Form des DGPS eingesetzt. DGPS bedeutet differential GPS und meint die Verwen-

dung eines zusätzlichen GPS-Empfängers, der nicht auf dem FTF, sondern stationär fest installiert ist. Mit Hilfe dieses ortsfesten GPS-Empfängers wird der sich zeitlich ändernde Fehler ermittelt, der dem GPS-System eigen ist. Mit Hilfe dieser Kenntnis können zeitgleich die fahrenden GPS-Empfänger auf den FTF exakte Positionen ermitteln[18, S. 112]. Diese Navigationstechnik benötigt einen freie Sichtkegel von 15 Grad nach oben (siehe Bild 4), um zuverlässig arbeiten zu können. Die Schritte zur Erlangung der erforderlichen Fahr- und Positioniergenauigkeit sind:

- Prüfung der örtlichen Gegebenheiten, insb. der Empfangsstärken der Satelliten
- Einsatz des Differential-GPS
- Real Time Kinematic Differential GPS.

Abbildung 4: Prinzipskizze zur Koppelnavigation (links) und zur Magnet- bzw. Transpondernavigation (rechts) (Quelle: Günter Ullrich, 2011 S. 79)

Im Rahmen des Projekt FAISE wird die Navigation durch den Laser durchgeführt. Es kann hier kein Global Positioning System (GPS) verwendet werden, da das ganze Experiment in einem geschlossenen Raum gemacht wird. Weiterhin wird auch keine Navigation durch die physische Leitlinie oder durch die Stützpunkte im Boden erzielt, weil dazu der Boden aufgebrochen werden müsste.

2.2.2 Steuerungstechnik

Die interne Materialflusssteuerung ist eine Vorstufe der Transportauftragsabwicklung und wird nur dann benötigt, wenn die Transportaufträge nicht klar dezidiert übertragen, sondern aufbereitet werden müssen. Eine Anforderung wie z. B. benötige Ware A an Maschine B erfordert eine Umsetzung in einen oder mehrere Transportaufträge nach dem klassischen Muster. Hole von C und bringe nach D. Die FTS-interne Materialflusssteuerung kombiniert also Quelle und Senke über die in ihr hinterlegten Transportbeziehungen zu einem Transportauftrag und schickt diesen zur Durchführung an die Transportauftragsverwaltung. Diese ganze Transportauftragsverwaltung ist in der FTS-Leitsteuerung geregelt. Die FTS-Leitsteuerung ist die Kommandozentrale, um das FTS in das Umfeld zu integrieren.

Außerdem steuert es die FTF, die sich im System befinden. Damit ist das FTS dann in der Lage, die ihm übertragenen Aufträge zu erfüllen. „Eine FTS-Leitsteuerung besteht aus Hard- und Software. Kern ist ein Computerprogramm, das auf einem oder mehreren Rechnern abläuft. Sie dient der Koordination mehrerer Fahrerloser Transportfahrzeuge und/oder übernimmt die Integration des FTS in die innerbetrieblichen Abläufe.“ [11](VDI 4451). Die Leitsteuerung bringt das FTS in seinem Umfeld zusammen, bietet seinen Bedienern vielfältige Service-Möglichkeiten und nimmt Transportaufträge entgegen. Weiterhin stellt sie den Aufgaben entsprechende Funktionsblöcke zur Verfügung. Die FTS-Leitsteuerung ist der Kern des FTS. In Rahmen des Projekt FAISE, wird auch eine Leisteuerung benötigt. Eine Leitsteuerung ist nur mit Hilfe einer Systemarchitektur zu implementieren und zu verstehen. In seinem Buch Fahrerlose Transportsysteme, hat Günter Ulrich zwei verschiedene Systemarchitekturen dargestellt. Eine für ein einfaches FTS und eine andere für ein komplexes FTS. Da bei FAISE nur mit vier FTF gearbeitet wird, ist es sinnvoll mit einer einfachen Systemarchitektur zu arbeiten. Das Bild 3 ist eine Repräsentation einer einfachen Systemarchitektur.

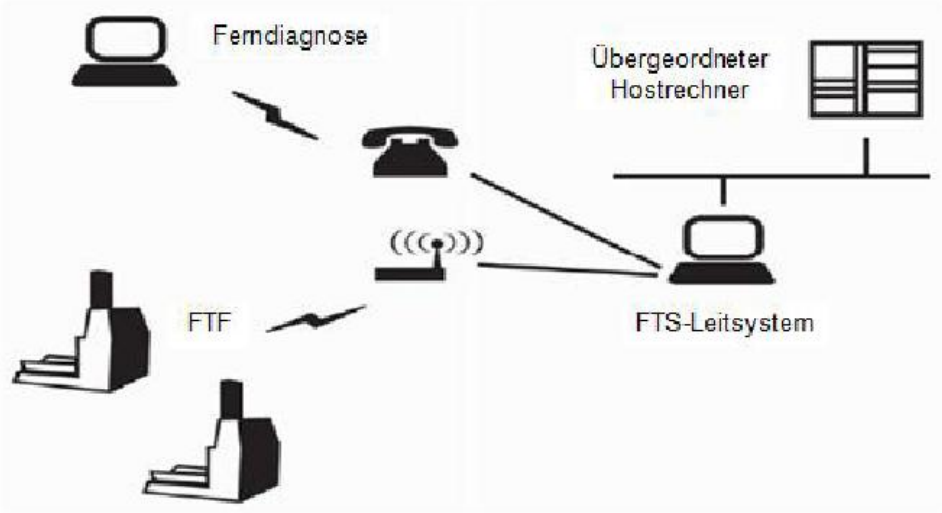

Abbildung 5: Die Systemarchitektur eines einfachen FTS (Quelle: Günter Ullrich, 2011 S. 93)

Es gibt eine geringe Anzahl von FTF, mit denen die Leitsteuerung per WLAN in Verbindung ist. Außerdem gibt es ein LAN, über das es eine direkte Verbindung mit einem übergeordneten Rechner gibt, von dem die Transportaufträge kommen. Über die angedeutete Telefonleitung ist eine VPN-Verbindung zur Ferndiagnose eingerichtet. Die Datenübertragung zu den übergeordneten Host-Rechnern erfolgt meist über lokale, Ethernet basierte Netzwerke mit dem Protokoll TCP/IP. Solche Host-Rechner können beispielweise Materialflusssteuerungssysteme zur Produktionssteuerung (z. B. SAP) Produktionsplanungssysteme (PPS) Lagerverwaltungssysteme (LVS) sein[vgl. 18, S. 96]. Außerdem gehören nach

der VDI 4451(Blatt 3) „zum internen Umfeld der FTF-Steuerung [...] das Lastaufnahmemittel (LAM), Sensoren und Aktoren, Bedienfeld am Fahrzeug und das Sicherheitssystem. Das externe Umfeld besteht aus der FTS-Leisteuerung, anderen FTF, automatischen Stationen und Gebäudeeinrichtungen“. Die Abbildung 1 stellt eine Darstellung eine FTF-Steuerung und ihr Steuerungsumfeld dar. Die administrative Ebene, die häufig über einen stationären

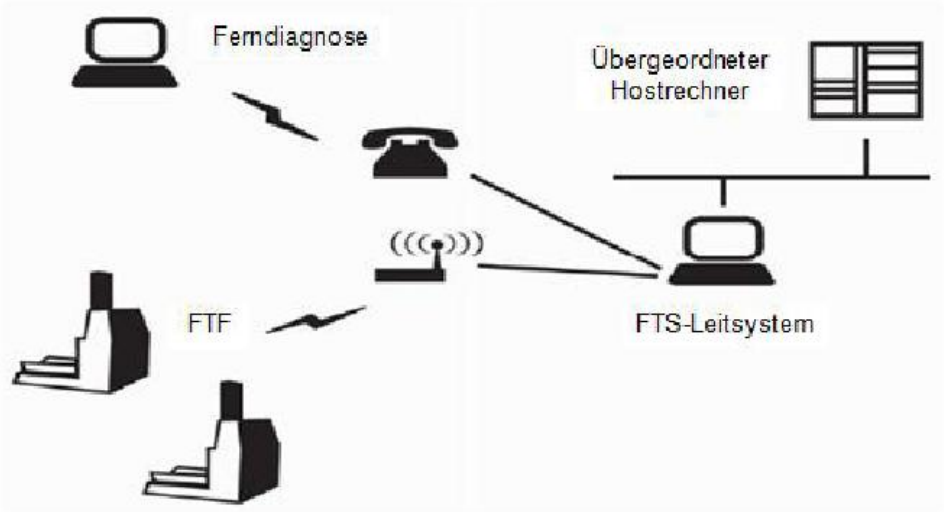

Abbildung 6: Allgemeine Darstellung einer FTF-Steuerung mit Datenschnittstellen (vgl. VDI 4451)

Leitrechner realisiert wird, verwaltet die Transportaufträge der ganzen Materialflusssteuerung. Die operative Ebene, die auch als Fahrzeugsteuerung bezeichnet wird, erhält ihre Informationen über die Fahrzeugdisposition der administrativen Ebene. Der Funktionsblock Kommunikation leitet den stattgefundenen Datenaustausch zum Manager weiter. Dieser sorgt für die Koordination, indem er die Fahraufträge in einzelne Befehle aufteilt, sowie für ein reibungsloses Zusammenwirken der einzelnen Funktionsblöcke. Neben dem Block Kommunikation sind weitere Blöcke vorhanden. Dazu gehört für die gesamte Lastübergabe inklusive der Lastlagererfassung verantwortliche Lastaufnahme, das Energiemanagement, welches den Lade- und Allgemeinzustand der Batterien überwacht, und der Block Überwachung/Sicherheitsschnittstelle, welcher zum Schutz der Personen und Sachgegenstände dient. Der Funktionsblock Fahren und die damit verbundene Sensorik bzw. Aktorik koordinieren die Ablaufsteuerung der Funktionen des Orientierungssystems[13, S. 33].

2.3 Materialflusssysteme

Damit ein Produkt auf den Markt kommen kann, muss man es durchdenken, erstellen und dann vermarkten. Die Produkterstellung und -vermarktung sind Prozesse des Wirtschaf-

tens. Vorprodukte oder Materialen werden von Beschaffungsmärkten in die Unternehmen geführt und dort durch besondere Produktionsprozesse transformiert. Am Ende der Produktion steht ein Endprodukt, das für den Konsum bereits ist. Die Produktion und Logistik von Gütern sind daher sehr wichtige Bereiche für den Unternehmenserfolg. Allerdings führen heute die unterschiedlichen Ausprägungen der Logistik z. B. in Produktions-, Handels- oder Verkehrsunternehmen zu einer terminologischen Differenzierung der Logistik. Der Materialflussbegriff leitet sich einfach von dem logistischen Konzept ab, in anderen Worten führt das Materialflusssystem in die Logistik zurück. Die Abbildung 2. dient zur Erläuterung einer konventionellen Wertschöpfungskette.

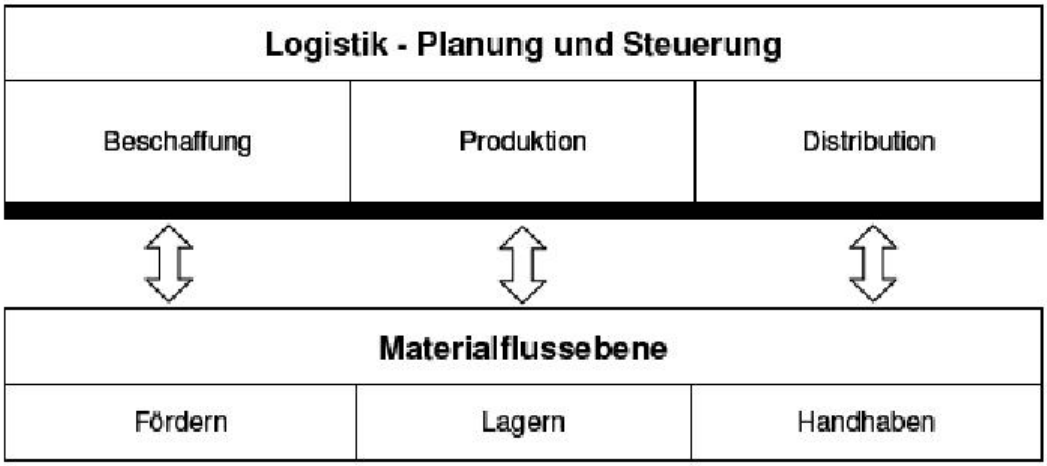

Abbildung 7: Elemente einer Wertschöpfungskette (vgl. Wulz, J, 2008, S. 7)

Der Begriff Materialfluss bedeutet die Verkettung aller Prozesse bei der Beschaffung, Bearbeitung, Verarbeitung sowie bei der Distribution von Gütern innerhalb festgelegter Bereiche. Deswegen lässt sich der Materialfluss in vier Stufen unterordnen: externer Transport, betriebsinterner Materialfluss, gebäudeinterner Materialfluss und Materialfluss am Arbeitsplatz. Nach dem Verein Deutscher Ingenieure bzw. VDI-241 beinhaltet die Logistik fünf Hauptfunktionen. Diese Funktionen sind Bearbeiten, Prüfen, Handhaben, Fördern, Lagern und Aufenthalten. Neben diesen Hauptfunktionen zählen auch Nebenfunktionen wie z. B. Montieren, Umschlagen, Kommissionieren, Palettieren und Verpacken[9]. Jedoch sind auf der Ebene des Materialflusssystems nur drei Funktionen zu berücksichtigen: Fördern, Lagern, Handhaben. Die anderen Funktionen setzen sich normalerweise aus den erläuterten Funktionen zusammen. Dieser Arbeitsteil wird in zwei Teile gegliedert. Im ersten Teil werden die drei Funktionen der Materialflusssysteme vorgestellt Im zweiten Teil wird eine Planung von Materialflusssystemen dargestellt.

Funktionen von Materialflusssystemen

- **Funktion Fördern**
  Fördern bedeutet Transportieren und ist einer der wichtigsten Aspekte innerhalb des Materialflusssystems. Nach der VDI 2411 ist Fördern das Fortbewegen von Arbeitsgegenständen in einem System. „Die Fortbewegung oder Ortveränderung von Gütern oder Personen mit technischen Mitteln wird allgemein als Transport bezeichnet. Findet diese Ortsveränderung in einem räumlich begrenzten Gebiet wie beispielsweise innerhalb eines Betriebes oder Werkes statt, so wird dieser Vorgang durch den Begriff Fördern präzisiert. Das Fördern bzw. die Fördertechnik umfasst also das Bewegen von Gütern und Personen über relativ kurze Entfernungen einschließlich der dazu notwendigen technischen organisatorischen und personellen Mittel“[7, S. 119]. Das Fördermittel (technisches Transportmittel zur Ortsveränderung von Gütern oder Personen) und das Förderelement bilden den physikalischen Bestandteil eines Fördervorgangs. Der Ablauf und die Steuerung werden durch den Fördervorgang dargestellt. In Punkto Fördermittel kann auf verschiedenste Elemente der Materialflusstechnik zurückgegriffen werden. Dies umfasst unter anderen Rollenbahnen und FTS. Neben der Möglichkeit auf automatisierte Fördermittel zurückzugreifen, kommen auch manuell mechanisierte bzw. rein manuelle Systeme zum Einsatz. In diesem Fall ist der Mensch oder der Bediener eines Fördermittels wesentlich für den Ablauf eines reibungslosen Materialflusses in Zusammenspiel mit den physikalischen Elementen sowie dem Prozessablauf verantwortlich[12, S. 8](Wulz, J, 2008, S. 8). Das Bild 4 gilt als Beispiel eines Fördersystems.

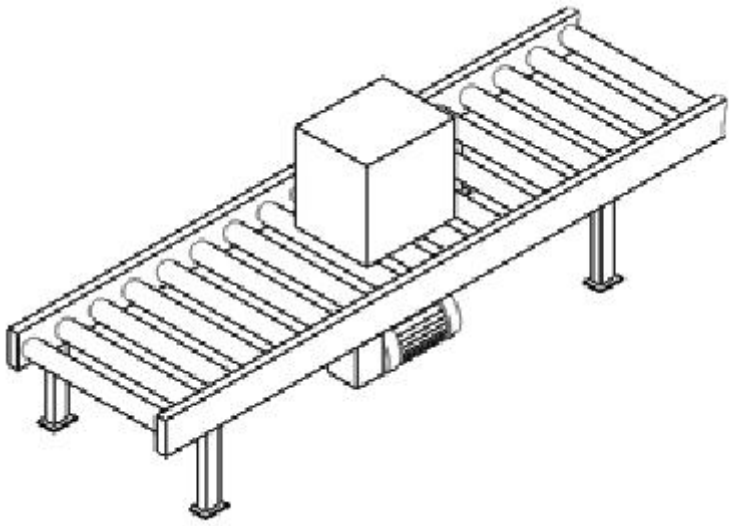

Abbildung 8: Beispiel eines Stetigförderers (entnommen aus Ten Hompel, Schmidt, Nagel, 2007, S. 131)

- **Funktion Lagern**

Das Lagern ist jedes geplante Liegen des Arbeitsgegenstandes im Materialfluss. Das Lager ist ein räumlich abgegrenzter Bereich bzw. eine Fläche zum Aufbewahren von Stück- und/oder Schüttgütern in Form von Rohmaterialien, Zwischenprodukten oder Endprodukten, das mengenmäßig erfasst wird[9]. Die Einlagerung von Lagereinheiten, die Aufbewahrung und Bereithaltung von Lagereinheiten auf Lagerplätzen und die Auslagerung einer Lagereinheit, sind die grundlegenden Prozesse in einem Lager. Aufgrund der starken Veränderungen im Markt, müssen auch die unternehmerischen Abläufe an Lagersysteme schnell angepasst werden. In einem Lagersystem werden im Verlauf des Materialflusses Speicher- bzw. Lagerfunktionen sowie Förderfunktionen wahrgenommen. Aufgabe eines Lagers ist das Bevorraten, Puffern und Verteilen von Gütern. Während Vorratslager lang- und mittelfristige und Pufferlager kurzfristige Bedarfsschwankungen ausgleichen sollen, erfüllen Verteillager neben der Bevorratungs- noch eine Kommissionierfunktion. Daher können die Aufgaben eines Lagers anhand folgender Ausgleichsmaßnahmen beschrieben werden: Zeitausgleich, Mengenausgleich, Raumausgleich und Sortimentsausgleich[16]. Ein Zeitausgleich ist immer dann erforderlich, wenn die Zeitfunktion der Nachfrage nicht der Zeitfunktion der Produktion entspricht. Beispielsweise steht eine losgrößenoptimierte Fertigung einer saisonalen Nachfrage gegenüber. Gerade in Bereichen mit Serienfertigung, in denen aus Kostengründen in der Regel größere Mengen als die Nachfragemengen produziert werden, muss Mengenausgleich vollzogen werden. Sobald der Produktionsort nicht mit dem des Produktabnehmers übereinstimmt, findet mit Hilfe von Verkehrsträgern ein Raumausgleich statt. Mit zunehmender Sortimentsbreite steigt die Wahrscheinlichkeit, dass die Anzahl der Produktionsstandardorte steigt[13, S. 14].

- **Funktion Handhaben**
  Der Begriff Handhaben wurde gedanklich von der menschlichen Hand abgeleitet, wird aber auch für automatisch ablaufende Vorgänge zur Manipulation von Objekten gebraucht. Handhaben bedeutet etwas greifen, bewegen und an einem bestimmten Ort ablegen. Das heißt, durch Handhaben wird die Lage oder Position von Objekten geändert. Im übertragenen Sinne bedeutet handhaben auch bewerkstelligen bzw. praktisch ausüben. Von Handhabungstechnik spricht man, wenn für die Handhabung Geräte eingesetzt werden. Die Richtline VDI 2860 definiert die Funktion Handhaben als „das Schaffen, definiertes Verändern oder vorübergehendes Aufrechterhalten einer vorgegebenen räumlichen Anordnung von geometrisch bestimmten Körpern." Die Teilfunktionen des Handhabens stellen das Speichern, das Bewegen, das Sichern, das Kontrollieren und das Verändern von Gütern dar. Das Handhaben kann sowohl als eine Funktion als auch eine Fertigung des Materialflusses betrachtet werden. Eine mögliche Handhabungsfunktion im Materialfluss ist z. B. das Palettieren, worunter die Stapelung von Stückgütern zu einem Stückgutstapel nach einem gewissen Mus-

ter verstanden wird. Handhabungsfunktionen können entweder von Automaten z. B. Roboter oder von Menschen durchgeführt werden. Auf Grund der Greifflexibilität ist der Mensch jedoch meist unübertroffen in der Handhabung.

2.4 Fallbeispiele

2.4.1 FTS in der Gläsernen Manufaktur Dresden (Volkswagen)

Volkswagen AG montiert das neue Modell der Luxusklasse „Phaeton“ in der „Gläsernen Manufaktur“ in Dresden. Die Materialversorgung übernimmt ein fahrerloses Transportsystem mit 56 frei navigierenden Fahrzeugen. Die gesamte Steuerungs- und Navigationstechnik stammt von FROG Navigation Systems, dem Projektpartner des Generalunternehmers AFT (Mechanik). Die Produktion ist auf drei Ebenen unterteilt: Die eigentliche Montage findet auf den beiden oberen Montageebenen statt: Die Rohkarosse befindet sich auf einer Montageplattform, die Teil des Schuppenbandes ist, das sich sicher in den Hallenboden einfügt und mit konstanter Geschwindigkeit durch die Montagezyklen bewegt. Danach erfolgt die Übergabe an eine schwere Elektrohängebahn (EHB) zur Hängemontage. Während der Hängemontage erfolgt die Hochzeit, d. h. das Zusammenfügen von Karosse und Triebsatz, wobei der Triebsatz von einem Fahrerlosen Transportfahrzeug (FTF) herangebracht wird. Anschließend wird die Karosse wieder auf eine Schubplattform, die sogenannte Schuppe, zur Komplettierung und Qualitätskontrolle gestellt.

Im Untergeschoss, der Logistikebene, wird die verbauende Ausrüstung zur Verfügung gestellt und in Betrieb genommen. Die FTS übernimmt die Versorgungsleitungen der Materialien und damit eine erhebliche logistische Funktion. Um zwischen den Ebenen zu wechseln, nutzen die automatischen Fahrzeuge Hebebühnen. Das FTS hat die grundsätzliche Aufgabe, die Montagelinien (Schuppenband oder EHB) zu versorgen. Dabei wird allerdings zwischen folgenden sechs Gewerken unterschieden:

1. Anlieferung von Warenkörben auf die Schuppe
2. Anlieferung von Schalttafeln (Cockpits)
3. Anlieferung von Kabelsträngen
4. Anlieferung des Triebwerks mit Fahrwerk und Ausführung der Hochzeit
5. Anlieferung von Warenkörben zur Hängemontage
6. Anlieferung der Türen plus Warenkörbe

2.4.2 FTS beim Automobilhersteller BMW im Werk Leipzig

Das BMW-Werk in Leipzig hat im Jahre 2005 mit der Produktion der 3er Reihe (E90) gestartet. Im Bereich der Teileversorgung übernimmt erstmals in der Geschichte der Automobilindustrie ein Fahrerloses Transportsystem (FTS) umfangreiche Logistikfunktionen. Folgende Prozesse wurde für die Teilversorgung im Leipzig-Werk definiert:

- Direktanlieferung per LKW: Große Teile mit geringer Komplexität (z. B. Bodenmatte oder Kofferraumverkleidung) werden per LKW zeitnah und in unmittelbarer Nähe des Verbauortes angeliefert.
- Modulanlieferung per EHB8: Große und komplexe Baugruppen (z. B. Cockpit) werden direkt auf dem Werksgelände von externen Lieferanten oder BMW Mitarbeitern montiert.
- Lagerware per FTS: Die Mehrzahl der Teile wird in einem Versorgungszentrum gelagert, kommissioniert und mit Fahrerlosen Transportfahrzeugen (FTF) an die jeweiligen Verbauorte in der Montage gebracht[18, S. 36].

Es sind 74 FTF im Einsatz, als Ladehilfsmittel werden mehr als 2.000 Rollwagen in zwei unterschiedlichen Ausführungen eingesetzt. Je FTF werden entweder zwei kleine Rollwagen, zur Aufnahme von Behältern bis DIN-Größe, oder ein sogenannter übergroßer Rollwagen zur Aufnahme von Großbehältern eingesetzt. Zusätzlich gibt es noch die Sequenziergestelle mit Sonderaufbauten[18, S. 37]. Durch einen Laser-Scanner auf dem FTF werden der Personenschutz und die Hinderniserkennung übernommen.

Die Fahrerlosen Transportfahrzeuge finden ihren Weg mit Hilfe der so genannten freien Navigation. Damit ist gemeint, dass die Fahrzeuge ohne physikalische Leitspuren und nach einem kombinierten Prinzip aus Kopplung und Peilung arbeiten. Kopplung bedeutet die Auswertung von fahrzeuginternen Sensoren (Messräder und ein faseroptischer Kreisel), wodurch der zurückgelegte Weg samt Kurven bestimmt wird [18, S. 37]. Bei jeder Peilung werden aufgetretene Fahrfehler, die durch Schlupf der Räder oder durch Veränderungen des Raddurchmessers auftreten können, korrigiert. Die Vorteile dieses, auch Magnet Navigation genannten, Verfahrens liegen in der Zuverlässigkeit und der Flexibilität bei zukünftigen Layoutanpassungen [18, S. 37].

3 Projektorganisation

Die nachfolgenden Abschnitte beschreiben den organisatorischen Ablauf innerhalb der Projektgruppe. Dazu gehört die Aufteilung in Gruppen, sowie die zeitliche Planung und Rollenverteilung. Des weiteren werden hier das Vorgehen für das gesamte Projekt und die verwendeten Werkzeuge beschrieben.

3.1 Organisation der Teilgruppen (Materialfluss, Fahrzeuge, Simulation)

Die Projektgruppe ist in die drei Teilgruppen unterteilt, da das Gesamtprojekt in eindeutig abgrenzbare Aufgabenfelder gegliedert ist und so Kompetenzen und Verantwortlichkeiten klar definiert werden können.

- **Materialfluss**
  Die Teilgruppe Materialfluss befasst sich mit Programmierung der Sensorik und Aktorik für die Rampen, sowie dem Aufbau eines Sensornetzwerkes zur Kommunikation zwischen den verschiedenen Akteuren der Simulation.
- **Fahrzeuge**
  Die Teilgruppe Fahrzeuge befasst sich mit allen Aspekten, die für das Funktionieren der Fahrzeuge verantwortlich sind. Dazu zählen unter anderem die Navigation, Odometrie und Lokalisierung. Auch ist die Einrichtung der Versorgungsinfrastruktur für die Fahrzeuge in Form von Ladestationen fällt in den Aufgabenbereich der Fahrzeuggruppe. Zusammen entwickeln die Teilgruppen Fahrzeuge und Materialfluss das physische System, so dass Kommunikation und Abstimmung zwischen diesen beiden Gruppen besonders wichtig sind.
- **Simualtion**
  Die Teilgruppe Simulation entwickelt die Software mit der eine virtuelle Simulation erstellt wird. Diese Software beinhaltet einen hybriden Modus, in dem das physische System auf die Software abgebildet wird und beide Teilsysteme ein Gesamtsystem bilden. Für die Entwicklung des Hybridmodus muss ein funktionierendes physisches System vorliegen.

3.2 Rollenverteilung

Um organisatorische Aspekte innerhalb des Projektes besser umsetzen zu können, wurden unterschiedliche Rollen definiert, die jeweils einen Bereich des Projektes abdecken sollen. Dazu gehören folgende Aufgaben:

- **Administrator**

  Der Administrator ist für die Einrichtung und Betreuung der Server und Tools zuständig, dazu zählt auch die Einrichtung der Webseite.

- **Aussendarstellung**

  Um das Projekt vernünftig zu Repräsentieren, verwalten die zuständigen der Aussendarstellung den Inhalt der Webseite. Ausserdem sind sie für die externen Kontakte und Events / Präsentationen verantwortlich.

- **Dokumentenbeauftragte**

  Damit am Ende ein einheitliches Format für den Endbericht gilt, organisieren, sammeln und verwalten die Beauftragten jegliche Quellen und Berichte (dazu zählt auch das Repository). Des weiteren sind sie Ansprechpartner bei fragen zur Literatur.

- **Gruppenleiter**

  Da das Projekt aus drei Teilgruppen besteht, besitzt jede Gruppe einen eigenen Gruppenleiter, der die Prozesse innerhalb der Gruppe lenkt und gemeinsam mit den anderen Gruppenleitern das gesamte Projekt koordiniert.

- **Qualitätsmanagement**

  Damit der Projektplan eingehalten wird, ist es Aufgabe des Qualitätsmanagements, dass die Prozesse (Scrums) eingehalten und korrekt ausgefürt werden. Zusätzlich sind sie für die System und Integrationstests verantwortlich.

- **Werkzeugbeauftragte**

  Die Werkzeugbeauftragen verwalten die benötigten Geräte und Schlüssel für die gesamte Projektgruppe. Weiterhin regeln sie, mit Rücksprache mit den Betreuern, den Einkauf der Hardware und Software.

3.3 Vorgehensmodell

Für die Durchführung der Projektgruppe muss ein Vorgehensmodell, sowohl für die Gesamt- als auch für die Teilgruppen, festgelegt werden. Durch ein Vorgehensmodell wird die Arbeit im Team strukturiert und es wird festgelegt, wie bestimmte Aufgaben, wie z.B. Abgleich mit Kunden und Anwendern, umgesetzt werden sollen. Sowohl für die Teilgruppen als auch für die Gesamtgruppe wurde ein Scrummodell als Vorgehensmodell gewählt (siehe Abbildung 9).

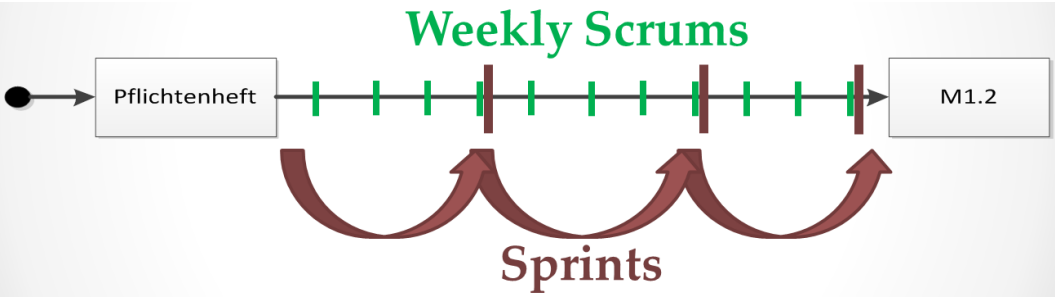

Abbildung 9: Scrumplanung für einen Meilenstein

Da es für ein sehr komplexes Projekt, wie das vorliegende, schwierig ist nur von einer groben Vision, sowie von User Stories auszugehen, wurden zunächst auf Basis des vorliegenden Lastenhefts in jeder Teilgruppe Pflichtenhefte erstellt. Anschließend wurden die dazugehörigen User Stories definiert, die die Anforderungen aus dem Pflichtenheft berücksichtigen und aus Anwendersicht darstellen. Die Sprints in den Teilgruppen sind mit einem Monat bemessen und werden zur Durchführung der User Stories genutzt. Die Rolle des Product Owners wird von den beiden Betreuern eingenommen, die sowohl für die Teil- als auch für die Gesamtgruppen zur Verfügung stehen, um die entwickelten Funktionalitäten abzugleichen. Die Durchführung von Daily Scrums ist zeitlich nicht möglich, da es sich um eine studentische Projektgruppe handelt, deren Stundenplan keine täglichen Treffen ermöglicht. Deshalb wurde das Scrum Vorgehensmodell dahingehend angepasst, dass die Daily Scrums in Weekly Scrums abgewandelt wurden. Die Weekly Scrums finden sowohl in den Teilgruppen als auch in der Gesamtgruppe statt. In den Weekly Scrums der Gesamtgruppe wird zunächst von jeder Person berichtet, welche Aufgaben in der vorherigen Woche erledigt wurden, damit entstandene Probleme und Hindernisse direkt in der Gruppe besprochen und eventuell beseitigt werden können. Die Ergebnisse aus den Teilgruppen werden ebenfalls vorgestellt und mit den Product Ownern abgeglichen. Das Scrum Vorgehensmodell wird mit Prototyping kombiniert (siehe Abbildung 10).

Durch das Prototyping sollen zu bestimmten Meilensteinen die kombinierten Ergebnisse aus den Teilgruppen vorgestellt werden, um den Stand des Gesamtsystems begutachten zu können. Betrachtet man das gesamte Projekt, so besteht es aus 2 Prototyping Phasen. Diese trennen sich im zweiten Meilenstein. Bis zu diesem Zeitpunkt wird mit horizontalen Prototyping die Basis des Projektes geschaffen. Das bedeutet, dass alle Grundfunktionalitäten implementiert und umgesetzt werden. Danach folgt das vertikale Prototyping in dem die Funktionalitäten um weitere Aspekte und Feinheiten eränzt werden. Innerhalb dieser beiden Phasen kommt es immer wieder zu Aufgabenbereichen, die jede Teilgruppe für sich umsetzt. Ebenfalls sind Phasen vorhanden, in denen die Ergebnisse der einzelnen Gruppen zusammengeführt werden müssen, wodurch das Zusammenspiel zwischen dem Material-

fluss und der Volksbots, sowie die Darstellung in der Simulation, entsteht.

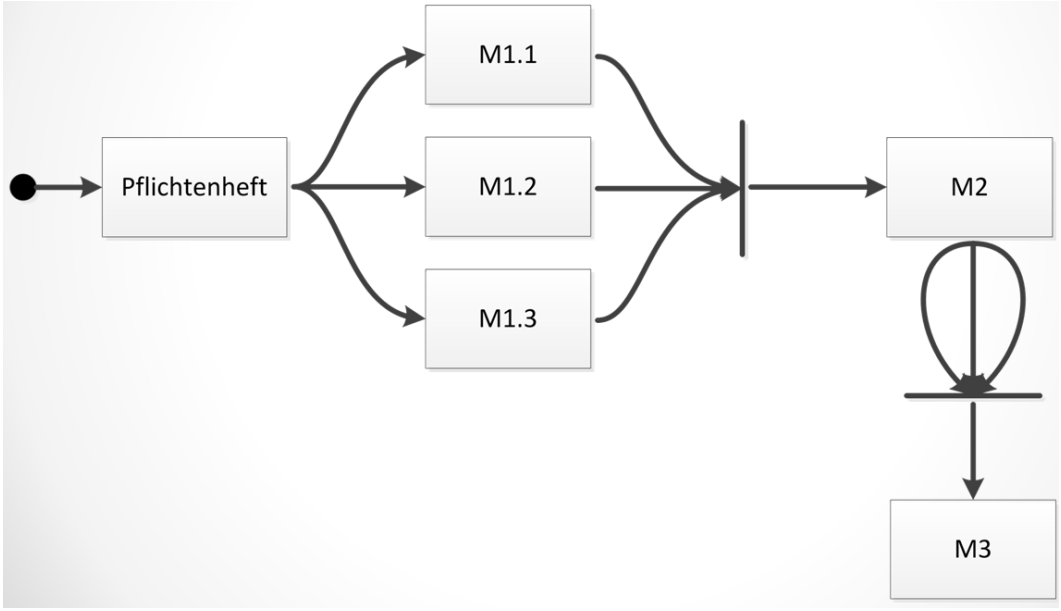

Abbildung 10: Modell für die Darstellung der einzelnen Projektphasen

3.4 Werkzeuge

Zur Unterstützung der Zusammenarbeit in der Projektgruppe wurden verschiedene Werkzeuge ausgewählt um die Kollaboration zu vereinfachen. Um eine gute Verknüpfung der Werkzeuge zu gewährleisten, wurde versucht die Produkte aus einer Hand zu beziehen. Nach einiger Recherche, stachen zwei Möglichkeiten heraus. Entweder die Open-Source Software Redmine oder eine kommerzielle Lösungen von der Firma Atlassian.

Die beiden Lösungen wurden daraufhin auf Basis verschiedener Kriterien verglichen. Zu diesen Kriterien zählten mitunter Usability, Verbreitung am Markt, Stabilität und Wartbarkeit.

Die Evaluation ergab Atlassian als klaren Sieger und die Projektgruppe einigte sich im speziellen auf die Werkzeuge **Jira** und **Confluence**.

Abbildung 11: Hersteller der Projektwerkzeuge

3.4.1 Jira

Jira ist ein Projektmanagement-Tool. Neben dem Verwalten von Vorgängen und Fehlern, lässt sich Jira mit einer *Agile* Erweiterung sehr gut zum planen, steuern und verwalten von Sprints. Eine einfache Übersicht über die verschiedenen, anstehenden Aufgaben können die Projektmitglieder über die sogenannten Boards erhalten. Hier wird genau für die verschiedenen User Stories deren Fortschritt angezeigt und welches Projektmitglied gerade welche Aufgabe bearbeitet.

3.4.2 Confluence

Confluence ist ein Wiki welches sich sehr gut in Jira integriert. So muss die Benutzerverwaltung nur einfach in Jira eingerichtet werden. Confluence greift dann auf das Benutzerverzeichnis von Jira zurück.

4 Allgemeine Anforderungen

In diesem Abschnitt sollen allgemeine Anforderungen beschrieben werden, die für das Gesamtsystem gelten. Zunächst soll ein Ablaufszenario beschrieben werden, das beschreibt, wie Rampen und Volksbots im Lager miteinander kommunizieren.

4.1 Ablaufszenario

Das Ablaufszenario beschreibt die logischen Schritte, die von den Akteuren ausgeführt werden, um das Ziel zu erreichen, ohne dabei auf die technischen Implementierungsdetails einzugehen. Es dient als Basis für die Implementierung sowohl des physischen Systems als auch der Simulationssoftware. In der Software soll der Ablauf komplett umgesetzt werden. Aus zeitlichen Gründen ist es nicht möglich, den gesamten Ablauf im physischen System umzusetzen, weshalb dort eine Anpassung erfolgt. Der Ablauf lässt sich in folgende Schritte unterteilen:

- Nachdem ein Paket am Eingang angekommen ist, wird ihm eine ID zugewiesen, so dass es eindeutig anhand seiner ID identifiziert werden kann.
- Die Eingangsrampe, auf der sich das Paket befindet, fragt alle Ausgänge, ob dieses Paket benötigt wird und zeitgleich werden die Zwischenrampen kontaktiert, um zu überprüfen, ob dort Platz frei ist. Falls ein Ausgang antwortet, wird dem Paket die ID des Ausgangs als Ziel zugewiesen. Falls nicht, wird dem Paket die ID eines der freien Zwischenlager zugeordnet.
- Falls der Ausgang eines oder mehrere Pakete benötigt, fragt der Ausgang die Zwischenrampen, ob ein Paket mit der vorhandenen ID verfügbar ist. Falls dies der Fall ist, wird dem Paket aus dem Zwischenlager die Ausgangs ID als Ziel zugewiesen.
- Sowohl Eingang als auch Ausgang stellen ihre Anfragen zyklisch, da bei einer einzigen Anfrage keine Garantie besteht, dass eine Zielrampe gefunden wird.
- Nachdem einem Paket ein Ziel zugewiesen wurde, versucht die Rampe, ein Transportmittel zu finden. Dazu werden die Volksbots kontaktiert, die anhand des Energie- und Zeitaufwands ein Angebot abgeben. Die Rampe wählt den Bot mit dem besten Angebot aus.
- Der ausgewählte Volksbot bewegt sich zur Rampe mit dem Paket und meldet sich an der Rampe, wenn er seine Zielposition erreicht hat. Die Rampe übergibt dem Volksbot das Paket mit allen notwendigen Informationen (ID und Ziel).
- Der Volksbot fährt zur Zielrampe, meldet sich dort an und übergibt das Paket.

5 Teilbericht Simulation

Der Teilbericht der Simulationsgruppe beschreibt die entwickelte Simulationssoftware von der Anforderungsanalyse über die Implementierung hin zum Testen und Validieren.

5.1 Lastenheft

Mit der Komponente Simulation soll auf Basis des Ablaufkonzepts eine Software erstellt werden, die es erlaubt, einen automatisierten Materialfluss auf Basis von FTS zu simulieren, ohne dabei an die zahlenmäßigen Beschränkungen des physischen Systems gebunden zu sein. Vonseiten des Auftraggebers wurde ein Lastenheft vorgegeben, das die gewünschten Kernfunktionalitäten der Simulationssoftware beschreibt. Es enthält folgende Anforderungen:

1. **Akteure**: Die virtuellen Akteure sind in ihrem Verhalten und Eigenschaften (Geschwindigkeit, Dauer einer Paketübergabe etc.) den echten Objekten aus dem physischen System nachempfunden (Volksbots und passive Rampen).
2. **Ablauf**: Der in Abschnitt 4.1 beschriebene Ablauf wird in der Simulation umgesetzt.
3. **Visualisierung**: Die Zustände der Akteure werden dynamisch visualisiert. Wird beispielsweise die Anzahl der Pakete auf einer Rampe um eins erhöht, dann soll dies unmittelbar in der Anzeige visualisiert werden.
4. **Generierung von Aufträgen**: Eingehende und ausgehende Transportaufträge können erstellt und simuliert werden.
5. **Einstellungen**: Verschiedene Parameter der Simulation (Anzahl und Art der Akteure, Anzahl der Aufträge etc.) können vom Nutzer vor dem Starten der Simulation angepasst werden.
6. **Statistiken**: Es werden wichtige Daten geloggt, um am Ende eines Simulationslaufs aussagekräftige Analysen über Stromverbrauch, gefahrene Strecken, Vergabe von Aufträgen usw. machen zu können.

5.2 Grundlegende Designentscheidungen

Vor der Entwicklung der Simulationssoftware mussten grundlegende Designentscheidung getroffen werden. Zum einen musste entschieden werden, ob die Simulation von einem autonomen Lager auf Basis eines vorhanden Tools oder komplett neu entwickelt werden sollte. Auch musste zwischen Desktop- und Webanwendung entschieden werden und ob die

jeweilige Alternative mit oder ohne Zuhilfenahme eines Frameworks implementiert wird. In den nachfolgenden Abschnitten werden die getroffenen Designentscheidungen begründet.

5.2.1 Eigenentwicklung

Im Vorfeld der Entwicklung wurde der Teilgruppe Simulation das Player/Stage Tool als Alternative zu einer kompletten Neuentwicklung einer Software vorgeschlagen. Das Tool beinhaltet zum einen die Komponente Player, die eine Hardware Abstraktionsschicht darstellt. Mit dieser Komponente kann mit Robotern, wie beispielsweise einem Volksbot, interagiert werden, ohne dass technische Details der Komponenten (Laserscanner, Motor etc.) bekannt sein müssen. Auf Basis von selbstgeschriebenem Code können Roboter gesteuert werden. Die Komponente Stage horcht auf die Befehle, die Player ausführt und visualisiert diese in einem eigenen Graphical User Interface. Jedoch kann Stage auch ohne Hardware benutzt werden, indem man über Konfigurationsdateien ein eigenes Szenario erstellt und die virtuellen Roboter über den eigenen Code steuert. Somit bietet das Tool die Möglichkeit, eine Simulation mit Robotern zu erstellen und das gewünschte Verhalten der Roboter über eigenen Code abzubilden. Auch muss die Visualisierung nicht selbst entwickelt werden (Vgl.[15]).

Dennoch wurde eine Eigenentwicklung der Nutzung des Tools vorgezogen. Die Benutzeroberfläche von Stage bietet die Möglichkeit, die Anzahl der Roboter und das Layout eines Szenarios über die entsprechenden Konfigurationsdateien einzustellen (Vgl.[15]). Jedoch gibt es beispielsweise keine Möglichkeit Aufträge zu erstellen bzw. zu simulieren oder Statistiken anzuzeigen. Somit ist die Entwicklung einer eigenen Benutzeroberfläche unumgänglich. Das bedeutet, dass die Stage Oberfläche über eine eigene Benutzeroberfläche gesteuert werden muss. Somit hätte man eine Trennung zwischen Visualisierung und Konfiguration eines Szenarios, was die Benutzerfreundlichkeit erheblich beeinträchtigt, da ein Nutzer den Durchlauf einer Simulation über zwei Benutzeroberflächen hinweg verfolgen müsste. Die Entwicklung eines eigenen Systems bietet somit erheblich mehr Benutzerfreundlichkeit und ermöglicht es, alle Anforderungen an das Interface in einer Benutzeroberfläche zu integrieren.

5.2.2 Entwicklung einer Webanwendung

Die Software soll als Webanwendung implementiert werden. Gegenüber einer Desktopanwendung bietet eine Webapplikation folgende Vorteile:

- Das System ist plattformunabhängig und kann somit auf jedem Rechner, der über einen Webbrowser verfügt, ausgeführt werden.
- Die Software muss nicht lokal installiert werden und kann direkt genutzt werden.
- Werden Änderungen an der Software vorgenommen, sind diese direkt verfügbar, da Updates über den Webserver eingespeist werden. Die Software ist somit immer auf dem aktuellsten Stand.

5.2.3 Umsetzung durch GWT

Die Entwicklung der Webanwendung sollte mithilfe eines Frameworks erfolgen, das es erlaubt, den Code sowohl für die Client- als auch für die Serverseite in einer Programmiersprache zu entwickeln. Außerdem sollte das Framework Schnittstellen bieten, um asynchrone Kommunikation und Push-Dienste zu nutzen, ohne sich um die exakten Details kümmern zu müssen. Ausgewählt wurde das Google Web Toolkit (GWT). GWT ist ein von Google entwickeltes Framework zur Erstellung von Webanwendungen. Der Java-Code für den Client wird von dem GWT Compiler in den entsprechenden Javascript- und HTML-Code übersetzt. Somit kann die Entwicklung sowohl für Client als auch für den Server auf Basis von Java erfolgen. Zudem entfällt die Anpassung des Javascript-Codes für die verschiedenen Browser, da GWT beim Kompilieren automatisch für jeden Browser eine lauffähige Version erzeugt. Weiterhin besitzt GWT eine RCP-Schnittstelle für die asynchrone Kommunikation zwischen Client und Server und lässt sich um Komponenten erweitern, um Daten vom Server zum Client zu pushen (Vgl.[24]). Somit erfüllt GWT sämtliche an ein Framework gestellte Anforderungen. Weitere Alternativen wurden nicht in Betracht gezogen, da drei von fünf Mitgliedern der Teilgruppe Simulation bereits positive Erfahrungen mit GWT gemacht haben und die anderen Mitglieder somit schnell einarbeiten konnten.

5.3 Konzeption der Systemkomponenten

In diesem Abschnitt wird die Konzeption der Gesamtarchitektur als auch der einzelnen Systemkomponenten beschrieben, die im Rahmen der Sprints erarbeitet wurde. Die Konzeption beinhaltet zum einen die Anforderungen, als auch die daraus abgeleiteten Implementierungsvorgaben.

5.3.1 Gesamtarchitektur

Wie in Kapitel 5.2.2 beschrieben, soll die Software als Webanwendung realisiert werden. Eine Webanwendung erfordert eine Client-Server Architektur. Abbildung 12 beschreibt die wesentlichen Komponenten des Systems und wie diese sich auf die Client- und Serverseite verteilen. Basis der Anwendung ist der Webserver, der die Basiskomponenten des Systems hosted. Zum einen stellt er die Laufzeitumgebung für Webanwendung und Datenbank bereit. Die Datenbank wird benötigt, um Daten, wie beispielsweise erstellte Szenarien, persistent zu speichern. Aus der Webanwendung heraus kann auf die Datenbank lesend und schreibend zugegriffen werden. In die Webanwendung soll ein Multiagentensystem (MAS) eingebettet werden. Ein MAS ist ein Netzwerk aus Softwareagenten. Softwareagenten sind Softwareeinheiten, die in in der Lage sind, Aufgaben selbstständig durchzuführen (Vgl.[3]). Mithilfe des MAS kann ein Lager simuliert werden, in dem der Warenfluss durch vollständig autonome Akteure durchgeführt wird.

Der Webserver beinhaltet die Logik des Systems. Auf dem Client soll die Visualisierung erfolgen und der Nutzer soll das Starten einer Simulation initiieren können. Das System soll von einem Webbrowser aus aufrufbar sein, in dem die Ergebnisse der serverseitigen Prozesse dargestellt werden. Mehrere Nutzer sollen das System gleichzeitig nutzen können, ohne Login und Registrierung.

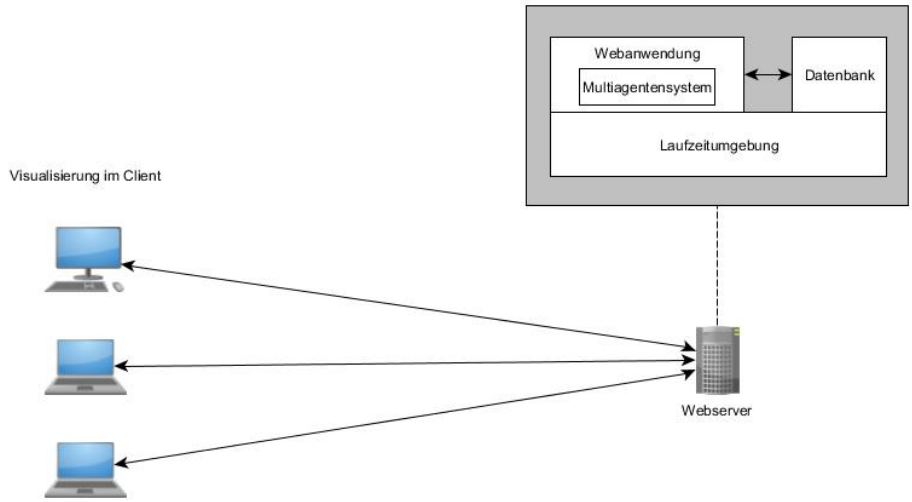

Abbildung 12: Gesamtarchitektur

5.3.2 Konzeption der Benutzeroberfläche

Die Benutzeroberfläche soll einem Nutzer die Möglichkeit bieten, auf sämtliche Funktionalitäten, die für die Durchführung eines Simulationsdurchlaufs relevant sind, zuzugreifen. Abbildung 13 zeigt den schematischen Aufbau der Gui anhand eines Mockups. Die Menüleiste beinhaltet drei Menu Items: Simulation, Auftragsliste und Statistiken. Über die Items Simulation und Auftragsliste sollen erstellte Szenarien und Auftragslisten geladen und gespeichert werden können. Außerdem soll eine Simulation gestartet werden können. Das Statistik Item erlaubt den Zugriff auf Statistiken, die für einen Durchlauf generiert wurden. Links unter der Menüleiste befindet sich die Auftragsliste, über die Aufträge generiert und angezeigt werden können. Darunter befindet sich der Bereich, der für die Modellierung eines Szenarios relevant ist. Es sollen Rampen, Fahrzeuge und Wände als Modellelemente auswählbar sein und in der Zeichenfläche platziert werden können. Die Zeichenfläche selber befindet sich rechts unter der Menüleiste. Dort werden die Aktionen der Akteure, wie z. B. Aufladen eines Pakets, visualisiert. Das unterste Element enthält eine Debug-Konsole, in der serverseitige Aktionen dargestellt werden können, die nicht in der Zeichenfläche dargestellt werden sollen.

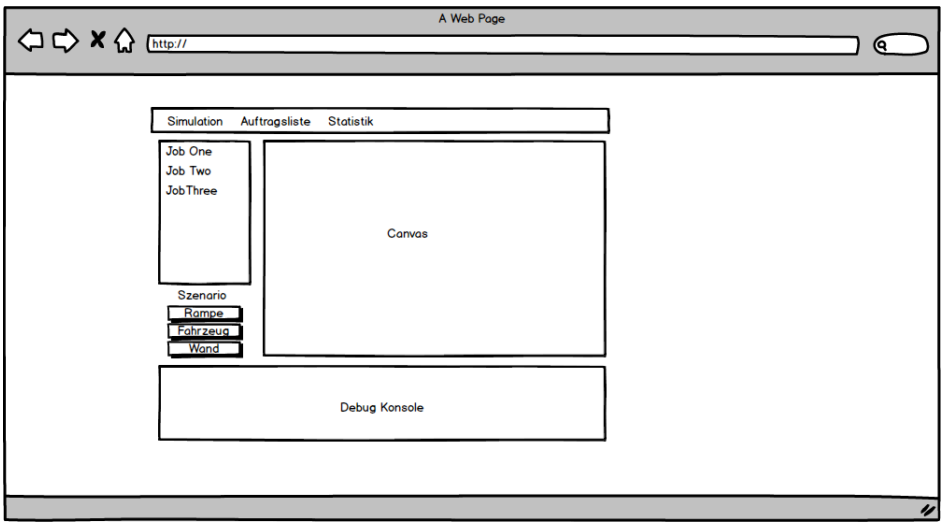

Abbildung 13: Schematischer Aufbau der Benutzeroberfläche

5.3.3 Generierung von Aufträgen

Die Simulationssoftware soll ein Umschlagslager simulieren. Das bedeutet, dass Pakete in das Lager geliefert, zwischengelagert und an den Ausgangsrampen wieder abgeholt werden, wenn ein bestimmtes Paket angefragt wird (Vgl.1.3). Das bedeutet, dass eine Unterscheidung getroffen werden muss zwischen einem physischen Paket und einer Nachfrage

nach einem bestimmten Paket, die ein Ausgang stellt. Deshalb soll zwischen eingehenden und ausgehenden Aufträgen unterschieden werden. Es soll möglich sein, eine festgelegte Anzahl an Aufträgen zufällig über die Gui zu generieren. Außerdem muss sichergestellt werden, dass die Menge an generierten eingehenden und ausgehenden Aufträgen in einem bestimmten Verhältnis zueinander stehen, um zu verhindern, dass nur eingehende oder nur ausgehende Aufträge generiert werden.

5.3.4 Anforderungen an ein Multiagenten-Framework

Um ein Umschlagslager und die darin enthaltenden Akteure (Rampen und Fahrzeuge) zu simulieren, wird ein Multiagentensystem benötigt (Vgl.5.3.1). Zur Erstellung eines MAS soll ein Framework verwendet werden, dass die nachfolgenden Anforderungen erfüllt: Das Framework muss das Erstellen von verschiedenen Agententypen ermöglichen, um die Akteure und ihre spezifischen Aufgabenstellungen umzusetzen. Agenten müssen in der Lage sein, untereinander Nachrichten auszutauschen und auf bestimmte Nachrichten oder Ereignisse mit definierten Verhaltensweisen zu reagieren. Weiterhin sollen die Aktionen der Agenten denen der realen Akteure hinsichtlich der Dauer ähneln. Außerdem muss das Framework Aktionen von verschiedenen Agenten parallel ausführen können, damit beispielsweise das gleichzeitige Fahren mehrerer Fahrzeuge möglich ist.

5.3.5 Benötigte Agententypen

Sowohl Rampen als auch Fahrzeuge müssen verschiedene Aktionen durchführen. Dazu gehören u. a. das Befördern von Paketen, die Vergabe von Aufträge, Durchführung von Auktionen usw. Würde man alle Aufgaben, die ein Akteur durchführen muss, in einem Agenten bündeln, so wäre ein solcher Agent nur schwer wartbar und es könnten abhängig vom Agenten-Framework Probleme bei der Parallelisierung von Aktionen auftreten. Es bietet sich an, die erforderlichen Aufgaben eines Akteurs auf mehreren Agenten zu verteilen. Durch die Modularisierung kann die Entwicklung des Systems parallelisiert werden und Änderungen an einem Agenten haben geringere Auswirkungen auf das Gesamtsystem. Die Wartbarkeit des Systems erhöht sich. Für das zu entwickelnde System wurden die folgenden vier Agententypen konzipiert:

- Paketagent: Verwaltung der Paketdaten
- Orderagent: Ermittlung von Zielrampen und Zuweisung von Zielen (Wird nur bei Rampen benötigt)
- Routingagent: Durchführung von Auktionen und Berechnung von möglichen Pfaden

- Plattformagent: Durchführung physischer Aktionen (Fahren, Aufladen von Paketen etc.)

5.3.6 Kommunikation zwischen den Agenten

Die zu entwickelnde Software soll die in Abschnitt 4.1 beschriebenen Abläufe umsetzen. Die Aufgaben der verschiedenen Akteure müssen auf die Agenten verteilt werden. Die entworfenen Kommunikationsschritte der Agenten sollen für den Fall dass ein Eingang Ausgänge und Zwischenlager fragt, beschrieben werden, um die Aufgaben der einzelnen Agenten genauer abzugrenzen. Im Rahmen der Implementierung können sich noch Änderungen ergeben, die technisch notwendig sind.

1. Trifft ein Paket ein, verlangt der Paketagent vom Orderagenten eine Destination für das entsprechende Paket.
2. Der Orderagent einer Eingangsrampe fragt die Orderagenten der Ausgangs- und Zwischenrampen.
3. Die Orderagenten prüfen im Abgleich mit ihren Paketagenten, ob Platz frei ist (Zwischenrampe) oder die Paket-ID benötigt wird (Ausgang). Anschließend antworten sie dem Orderagenten am Eingang.
4. Wurde ein Ziel für das Paket gefunden, soll der Orderagent den Start einer Auktion initiieren, indem er den Routingagenten benachrichtigt.
5. Der Routingagent verlangt eine Aufwandsschätzung von allen Routingagenten der Fahrzeuge.
6. Der Routingagent eines Fahrzeugs berechnet eine Aufwandsschätzung anhand seiner Position und antwortet dem Routingagenten der Rampe. Fährt der Volksbot oder nimmt er an einer anderen Auktion teil, so wird -1 als Aufwandsschätzung zurückgeschickt.
7. Der Routingagent einer Rampe wählt, sofern vorhanden, den Bot aus, der den geringsten Aufwand benötigt, um ein Paket abzuholen und weist ihm den Auftrag zu.
8. Der Plattformagent fährt zu der jeweiligen Eingangsrampe und lädt das Paket auf. Dies geschieht durch einen Nachrichtenaustausch mit dem jeweiligen Plattformagenten der Rampe, der die Paketdaten übergibt. Das Fahren zur Zielrampe und das Abladen des Pakets erfolgt analog.

5.3.7 Konzeption des Pathfindings

5.3.8 Konzeption der Statistiken

5.3.9 Interaktion der Komponenten

Durch das Zusammenspiel der Komponenten der verschiedenen Systemkomponenten soll eine Simulation durchgeführt werden können. Die Aufträge müssen entsprechend ihrer Zeiten an den Server geschickt werden. Dies soll clientseitig durch einen Timer durchgeführt werden, um das MAS zu entlasten. Abbildung 14 zeigt den Ablauf und die Komponenten, die für das Starten einer Simulation erforderlich sind:

1. Ein potenzieller Nutzer startet eine Simulation über die Benutzeroberfläche.
2. Der Server wird durch die GUI informiert, dass die Simulation gestartet werden soll.
3. Der Server startet das Multiagentensystem.
4. Der Server meldet dem Client den Start des MAS.
5. Der Client weiß nun, dass die Agenten bereit sind, Aufträge entgegenzunehmen und startet den Timer für die Jobliste.
6. Die Aufträge werden gemäß ihrer Startzeit an den Server geschickt.
7. Der Server leitet die Aufträge an das MAS weiter
8. Die Daten über sichtbare Zustandsveränderungen werden an den Client geschickt und dort in der GUI visualisiert.

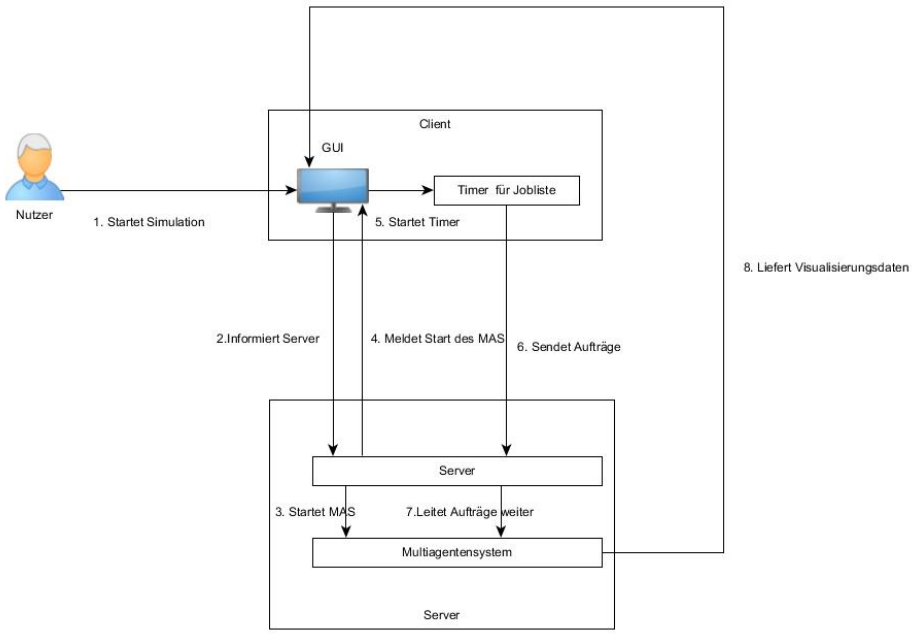

Abbildung 14: Interaktion der Systemkomponenten

5.4 Implementierung der Systemkomponenten

In diesem Abschnitt soll die Implementierung der zuvor konzipierten Systemkomponenten beschrieben werden. Zum einen soll die Auswahl von Technologien und Frameworks zur Umsetzung beschrieben sowie die Funktionalität des Systems auf technischer Ebene dargestellt werden.

5.4.1 Implementierte Gesamtarchitektur

Abbildung 15 zeigt die konkrete Gesamtarchitektur des Systems, die durch die Auswahl von Umsetzungstechnologien entstanden ist. Die Webanwendung soll, wie bereits in Abschnitt 5.2.3 beschrieben, durch das GWT Framework implementiert werden. Der Code für Server und Client, der für Visualisierung, Client-Server Kommunikation u. ä., benötigt wird, wird durch GWT-Bilbiotheken bereitgestellt. Eingebettet in die GWT-Klassen wird das Multiagentensystem, das mithilfe des Java Agent Development Framework (JADE)[1] implementiert wurde. Als Datenbankmanagementsystem wurde PostgreSQL ausgewählt. Die Applikation wird durch einen Jetty Server gehostet, der die Java Laufzeitumgebung und das GWT Software Development Kit ebenfalls bereitstellt.

[1] [1]

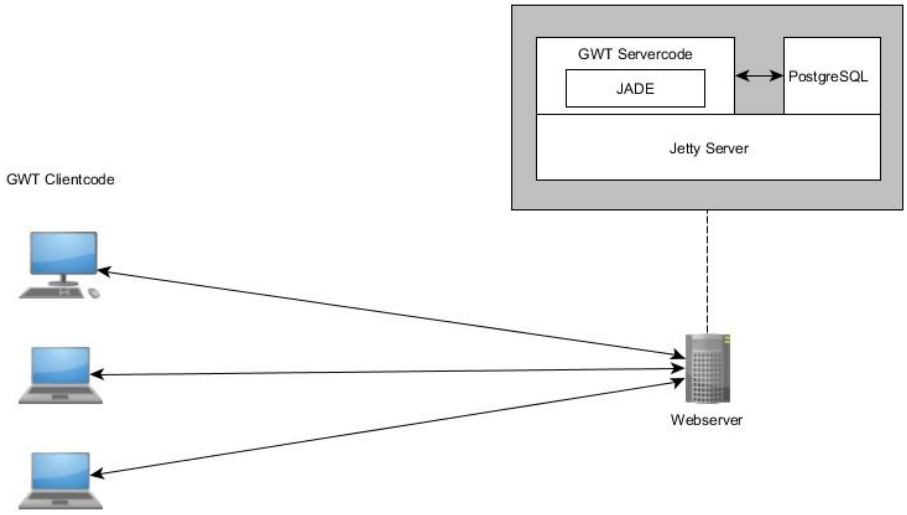

Abbildung 15: Implementierte Gesamtarchitektur

5.4.2 Benutzeroberfläche

5.4.3 Generierung von Aufträgen

5.4.4 Auswahl eines Multiagenten-Frameworks

Das Java Agent Development Framework wurde genutzt, um ein Netzwerk aus verschiedenen Agenten zu implementieren. Jade erlaubt das Erstellen von verschiedenen Agenten. Agenten werden als Java-Klasse implementiert, die von der vordefinierten Klasse Agent erbt. Die Aktionen, die ein Agent ausführen soll, werden durch Behaviours implementiert, die ebenfalls Klassen sind. Um die Anzahl an Klassen zu reduzieren, wurden die Behaviours als interne Klassen in die Agenten eingefügt. Jade ermöglicht die Kommunikation zwischen den Agenten durch ein vordefiniertes Nachrichtensystem. Nachrichten können sowohl gezielt an einen Agenten adressiert oder an alle Agenten verschickt werden. Die Agenten können in Echtzeit miteinander kommunizieren und mehrere Agenten können parallel Aktionen ausführen, da jeder Agent in einem eigenen Thread läuft (Vgl.[**jadetwo**]).

5.4.5 Implementierte Agententypen

Implementiert wurden sieben verschiedene Agententypen. Jede Rampe besitzt jeweils einen Paket-, Order-, Routing- und Plattformagenten. Ein Fahrzeug besitzt einen Paket-, Routing- und Plattformagenten. Die Routing- und Plattformagenten wurden für Fahrzeuge und Rampen durch unterschiedliche Agenten implementiert, um die Anzahl an Behaviours pro

Agent zu reduzieren und den Code übersichtlicher zu gestalten und unnötigen Speicherverbrauch zu vermeiden. Damit ein Agent weiß, welchem Szenario und welchem Akteur er zugeordnet ist, werden Szenario und Conveyor als Referenz übergeben [2]. Neben den genannten Agenten war es notwendig, einen Jobagent zu implementieren, der die Übergabe eines Pakets, die im phyischen System durch einen Menschen geschieht, zu simulieren. Außerdem werden ausgehende Aufträge durch den Jobagent an die Ausgangsrampen übermittelt.

5.4.6 Kommunikation zwischen den Agenten

In den nachfolgenden Abschnitten soll die Kommunikation zwischen den Agenten, die implementiert wurde, anhand von Sequenzdiagrammen beschrieben werden. Dabei werden folgende Anwendungsszenarien durchlaufen:

- Jobzuweisung an Ein- und Ausgänge.

Jobzuweisung an Ein- und Ausgänge Abbildung 16 zeigt die Jobzuweisung an die Ein- und Ausgangsrampen durch den Jobagent. Bevor die Jobzuweisung beginnt, wird die Simulation durch den Client gestartet. Neben dem Jobagent, den Plattformagenten der Rampen und dem Paketagenten ist das Servlet AgentPlattformServiceImpl an der Kommunikation beteiligt. Auf die Aspekte der Client-Server Kommunikation wird in Abschnitt 5.3.9 genauer eingegangen. Das Diagramm zeigt folgenden Ablauf:

- Die Simulation wird aus dem Browser heraus gestartet und das Servlet wird benachrichtigt.
- Das Servlet startet die Agentenplattform und initialisiert die Agenten, einschließlich dem Jobagent.
- Der Jobagent startet einmalig eine Anfrage an die Plattformagenten der Rampen, um die IDs und die Anzahl der Ein- und Ausgänge zu erfragen, die für die Jobzuweisung nötig sind.
- Die Plattformagenten senden ihren Rampentyp an den Jobagent.
- Der Jobagent besitzt zwei Listen in denen er die IDs, der Ein- und Ausgänge speichert. Anhand des empfangenen Rampentyps, speichert er die ID des Senders in der entsprechenden Liste. Anschließend wird der Client benachrichtigt, dass die Simulation gestartet wurde.

[2] Der genaue Ablauf der Initialisierung wird in Abschnitt 5.3.9 beschrieben

- Der Client startet den Jobtimer und sendet die Aufträge entsprechend ihrer zeitlichen Reihenfolge an das Servlet.
- Das Servlet leitet die Aufträge an den Jobagent weiter.
- Der Jobagent empfängt den Auftrag und sendet eine Nachricht an sich selbst.
- Je nach Art des Auftrags werden entweder die Eingangs- oder Ausgangsrampen gefragt, ob der Auftrag entgegengenommen werden kann.
- Die Plattformagenten senden eine Nachricht an ihre Paketagenten, um zu prüfen, ob der Auftrag aufgenommen werden kann. Der Plattformagent wartet auf die Antwort des Paketagenten.
- Der Paketagent prüft, ob der Auftrag entgegengenommen werden kann und antwortet dem Plattformagent.
- Der Plattformagent verarbeitet die Anfrage und antwortet dem Jobagent. Läuft der Plattformagent auf einer Ausgangsrampe, so wird automatisch geantwortet, dass Platz verfügbar ist, da ausgehende Aufträge nur IDs repräsentieren, die der Ausgang anfragt. Ein Ausgang kann unbegrenzt IDs anfragen.
- Der Jobagent wählt unter den Rampen, die einen Auftrag entgegennehmen können, zufällig eine aus und sendet die Auftragsdaten an den Plattformagenten.
- Der Plattformagent initialisiert mit den Auftragsdaten die Paketdaten und schickt diese an den Paketagenten. Zu beachten ist, dass die Klasse PackageDate sowohl eingehende als auch ausgehende Aufträge repräsentiert.
- Der Paketagent fügt das Paket einer Liste hinzu.

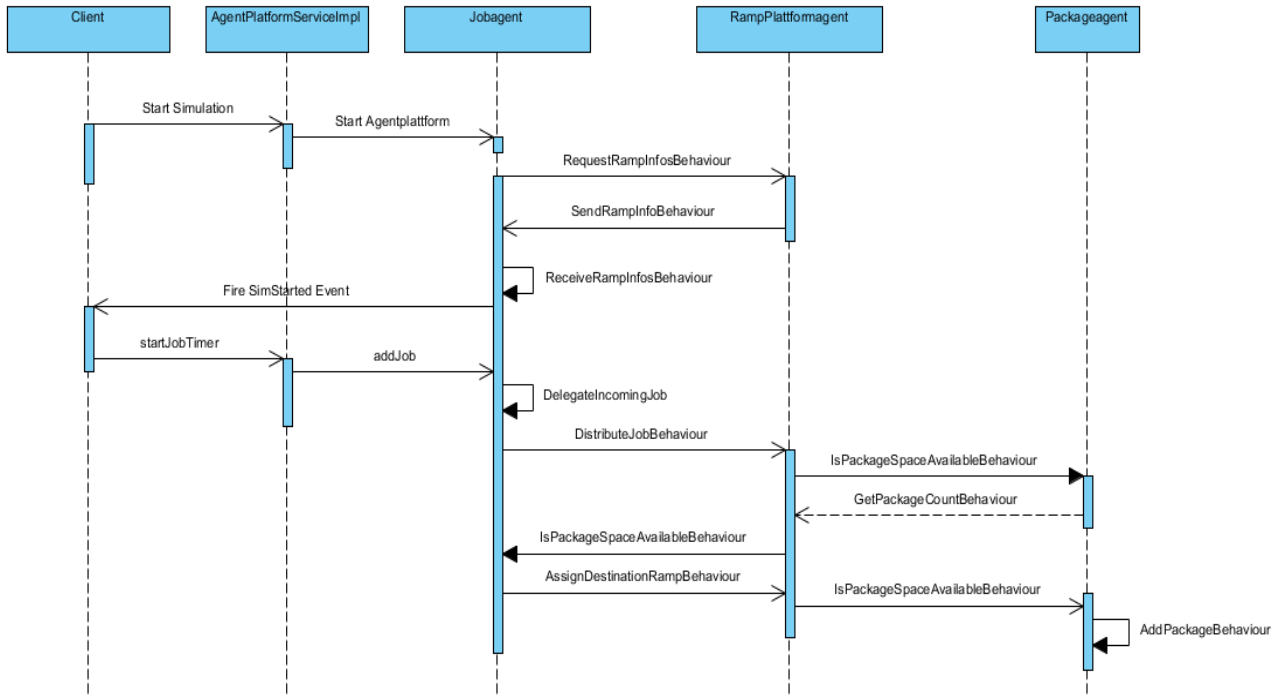

Abbildung 16: Jobzuweisung an Ein- und Ausgänge

Eingang fragt Ausgang und Zwischenlager Pakete, die am Eingang eintreffen, werden weitergeleitet. Besteht ein Bedarf am Ausgang, so soll das Paket zum Ausgang befördert werden. Ansonsten wird es ins Zwischenlager gebracht. Abbildung 17 zeigt den Nachrichtenaustausch der Agenten, der zur Zielfindung nötig ist:

- Der Paketagent stellt mithilfe einer Tickerbehaviour zyklisch Anfragen an seinen Orderagenten, um ein Ziel für das vorderste Paket zu bekommen. Bevor der Orderagent benachrichtigt wird, prüft der Paketagent mithilfe der Variable OutgoingJobFlag, ob für das Paket bereits ein Ziel gefunden wurde. Falls nicht, sendet er die Anfrage an den Orderagenten.
- Der Orderagent sendet eine Nachricht an alle Orderagenten der Ausgangs- und Zwischenrampen.
- Der jeweilige Orderagent fragt seinen Paketagenten, ob bereits ein Paket erwartet wird. Dies ist nötig, damit sich nicht mehrere Bots vor einer Rampe blockieren.
- Der Paketagent prüft anhand der Variable IncomingJobFlag, ob ein Paket erwartet wird und antwortet seinem Orderagenten.
- Sofern kein Paket erwartet wird, fährt der Orderagent fort und stellt eine erneute Anfrage an seinen Paketagenten, um zu fragen, ob das entsprechende Paket benötigt wird

(Ausgang) oder ob Platz frei ist (Zwischenlager).

- Der jeweilige Orderagent antwortet dem Eingang, ob das Paket entgegengenommen werden kann oder nicht.
- Der Orderagent des Eingangs speichert die IDs der Rampen, die ein Paket aufnehmen können, wählt unter diesen zufällig eine aus und übermittelt die Daten für Start- und Zielrampe an den Routingagent, damit dieser die Auktion starten kann. Der Orderagent wartet bis die Auktion beendet ist.

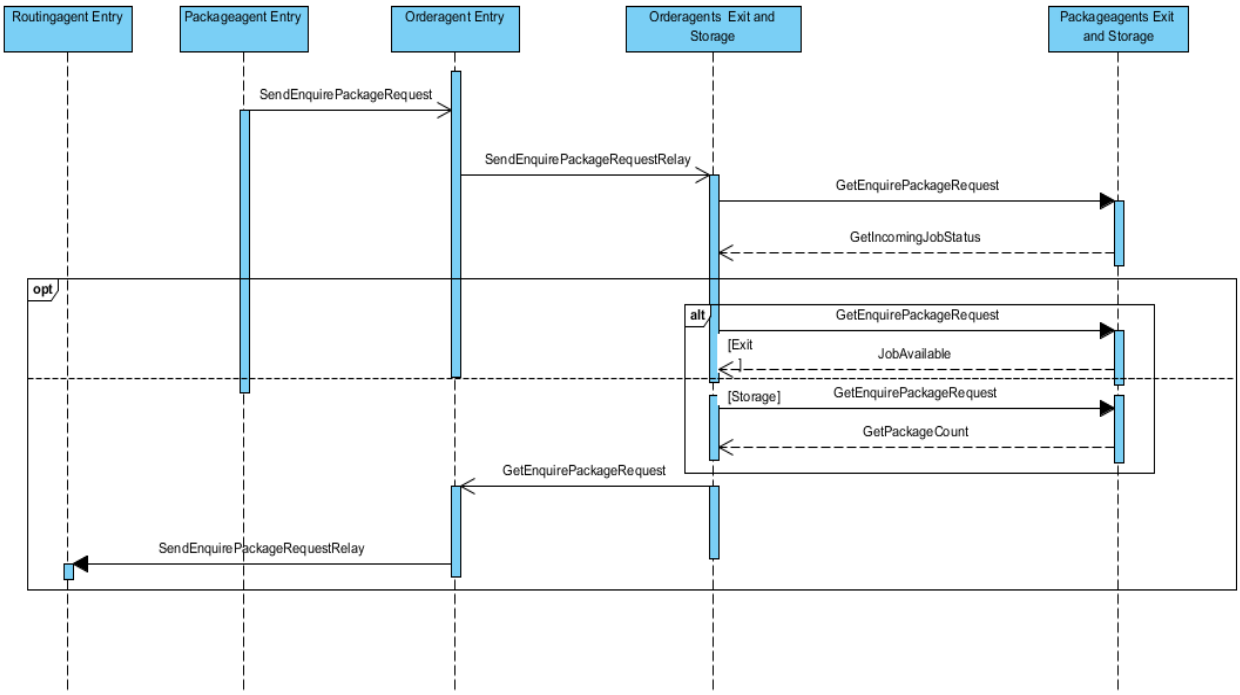

Abbildung 17: Eingang fragt Ausgang und Zwischenlager

Ausgang fragt Zwischenlager Pakete, die am Eingang eintreffen und für die noch keine Nachfrage an den Ausgangsrampen besteht, werden ins Zwischenlager gebracht. Damit auch diese Pakete zum Ausgang gelangen, fragt der Ausgang zyklisch im Zwischenlager für die entsprechenden Pakete nach. Abbildung 18 zeigt den Ablauf im Detail:

- Der Paketagent des Ausgangs stellt mithilfe einer Tickerbehaviour zyklisch Anfragen an seinen Orderagenten, damit im Zwischenlager geprüft wird, ob ein Paket vorhanden ist, für das eine Nachfrage besteht. Bevor der Orderagent benachrichtigt wird, prüft der Paketagent mithilfe der Variable IncomingJobFlag, ob bereits ein Paket erwartet wird. Falls nicht, sendet er die Anfrage an den Orderagenten.
- Der Orderagent sendet eine Nachricht an alle Orderagenten der Zwischenrampen.

- Der jeweilige Orderagent fragt seinen Paketagenten, ob bereits ein Paket abgeholt wird. Dies ist nötig, damit sich nicht mehrere Bots vor einer Rampe blockieren und damit ein Bot nicht das falsche Paket abholt.
- Der Paketagent prüft anhand der Variable OutgoingJobFlag, ob ein Paket erwartet wird und antwortet seinem Orderagenten.
- Sofern kein Paket erwartet wird, fährt der Orderagent fort und stellt eine erneute Anfrage an seinen Paketagenten, um zu fragen, ob das Paket, das der Ausgang verlangt, an vorderster Stelle der Rampe ist.
- Der jeweilige Orderagent antwortet dem Ausgang, ob das Paket vorhanden ist oder nicht.
- Sofern das Paket in einem der Zwischenlager ist, benachrichtigt der Orderagent des Ausgangs den Routingagenten des Zwischenlagers, damit das Paket von einem Bot abgeholt wird.

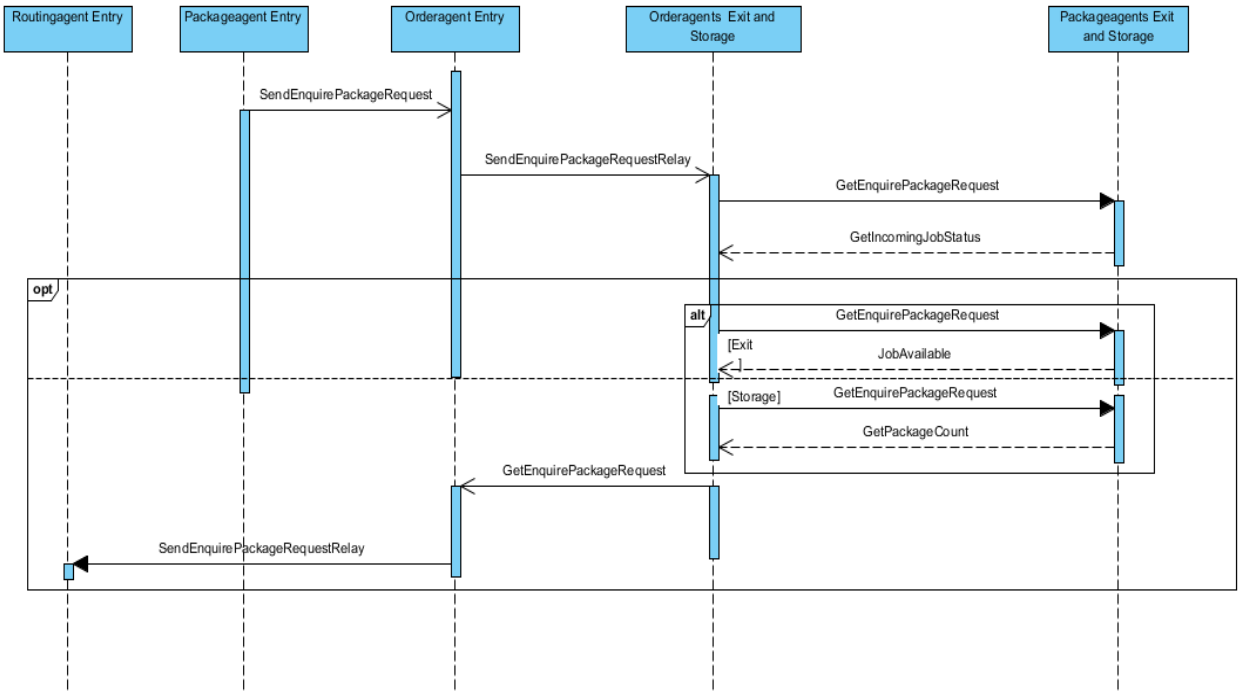

Abbildung 18: Ausgang fragt Zwischenlager

Auktion Nachdem ein Paket ein Ziel zugewiesen bekommen hat, startet der Routingagent der jeweiligen Eingangs- oder Zwischenrampe eine Auktion, um ein Fahrzeug zu finden, das das Paket befördert. Abbildung 19 zeigt den Ablauf der Auktion:

- Der Routingagent schickt eine Nachricht an alle Routingagenten der Fahrzeuge.
- Die Routingagenten der Fahrzeuge fragen ihre Paketagenten, ob bereits ein Auftrag durchgeführt wird. Der Routingagent wartet bis der Paketagent geantwortet hat.
- Der Paketagent antwortet dem Routingagenten.
- Falls das Fahrzeug nicht belegt ist und der Bot auch an keiner anderen Auktion teilnimmt, erfragt der Routingagent seine aktuelle Position von seinem Plattformagenten.
- Der Plattformagent schickt die aktuelle Position an den Routingagenten.
- Der Routingagent berechnet mithilfe des Pathfindings eine Aufwandsabschätzung und schickt Sie dem Routingagenten der Rampe. Falls er nicht in der Lage ist, ein Paket zu befördern, teilt er dies der Rampe über die Nachricht mit.
- Der Routingagent der Rampe speichert alle Estimations nacheinander ab. Haben alle Fahrzeuge geantwortet oder ist der Timeout für die Auktion abgelaufen, wird, sofern vorhanden, der Bot mit der besten Estimation ausgewählt. Falls es einen Bot gibt, der das Paket abholen kann, wird der Paketagent des Zwischenlagers oder Ausgangs benachrichtigt, dass ein Paket in Kürze geliefert wird, damit das IncomingJobFlag gesetzt wird. Dadurch wird verhindert, dass die Bots sich gegenseitig vor einer Rampe blockieren.
- Der Paketagent antwortet dem Routingagenten nach dem Setzen des Flags.
- Der Routingagent teilt dem Routingagenten des ausgewählten Fahrzeugs mit, dass es ausgewählt wurde.
- Der Routinagent des Fahrzeugs sendet eine Nachricht an seinen Paketagenten, um Platz für das Paket zu reservieren.
- Der Fahrzeug Routingagent leitet die Information an seinen Plattformagenten weiter, damit die Fahrt beginnen kann.

Paket von Startrampe abholen und zur Zielrampe bringen Nachdem das Fahren initiiert wurde, führt der Plattformagent des entsprechenden Fahrzeugs den Transport durch. Abbildung 20 zeigt den Ablauf auf Agentenebene:

- Das Fahrzeug fährt zur Startrampe und benachrichtigt den Plattformagenten der jeweiligen Rampe.
- Dieser benachrichtigt seinen Paketagenten, dass das Paket transferiert werden kann.

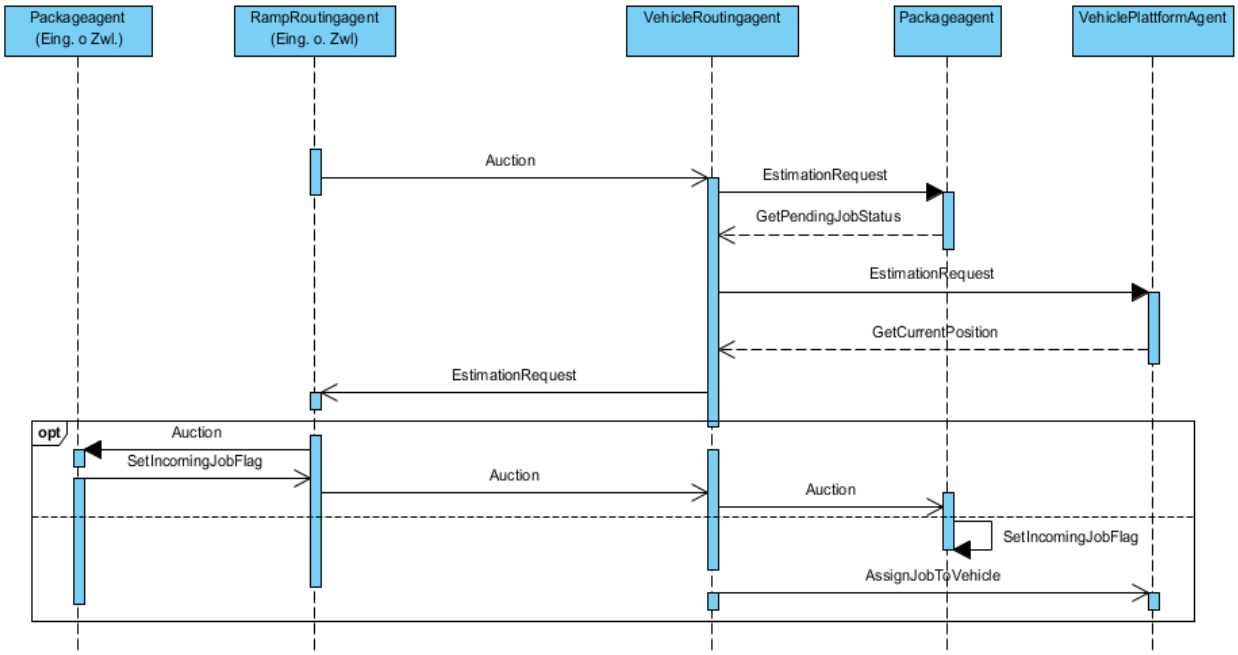

Abbildung 19: Auktion

- Der Paketagent sendet eine Nachricht an den Paketagenten des Fahrzeugs, um das Paket zu übergeben und entfernt es von der Rampe.
- Der Paketagent des Fahrzeugs fügt das Paket hinzu und antwortet dem anderen Paketagenten.
- Der Paketagent der Startrampe benachrichtigt den Plattformagenten, dass das Paket übergeben wurde.
- Dieser benachrichtigt wiederum den Plattformagenten des Fahrzeugs, damit dieser weiß, dass er weiterfahren kann.
- Der Plattformagent des Fahrzeugs fährt zur Zielrampe und benachrichtigt seinen Paketagenten, dass das Paket übergeben werden kann.
- Der Paketagent entfernt das Paket und sendet eine Nachricht an den Paketagenten der Rampe, dass das Paket aufgenommen werden soll.
- Dieser fügt das Paket hinzu und gibt dem Paketagent des Fahrzeugs Bescheid, dass das Aufladen beendet ist.
- Der Fahrzeug Paketagent informiert seinen Plattformagent, dass der Auftrag erledigt ist.

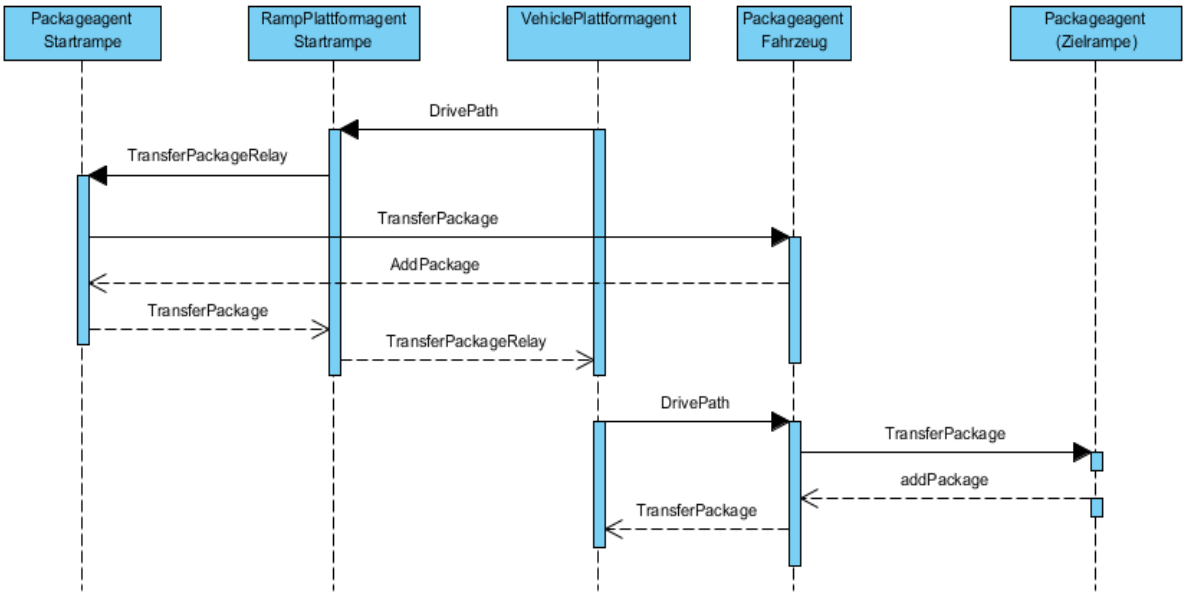

Abbildung 20: Paket abholen und zur Zielrampe bringen

5.4.7 Implementierung des Pathfindings

5.4.8 Implementierung der Statistiken

5.4.9 Interaktion der Komponenten

In Abschnitt 5.3.9 wurden die Interaktionen der Systemkomponenten auf logischer Ebene beschrieben, die für das Starten einer Simulation bis hin zur Visualisierung notwendig sind. Im Folgenden Abschnitt soll beschrieben werden, wie und mit welchen Mitteln die einzelnen Schritte implementiert wurden.

Client-Server Kommunikation Das Starten einer Simulation wird über das Menuitem „Starten/Anhalten“ausgelöst (s. Abbildung21). Wird das Menuitem gedrückt, dann wird die Methode execute ausgeführt. In der Methode wird geprüft, ob die Simulation bereits läuft, um festzustellen, ob die Simulation gestartet oder angehalten werden soll. Wenn die Simulation noch nicht gestartet wurde, wird zunächst geprüft, ob das erstellte Szenario konsistent ist (Mindestens eine Eingangs- und Ausgangsrampe). Wenn dies der Fall ist, wird die Simulation gestartet. Dies erfordert eine Client-Server Kommunikation. Die Implementierung erfolgt durch den von GWT bereitgestellten Remote Procedure Call (RPC)

Mechanismus (Vgl. [25]).

Die Variable agentPlatformService referenziert das Interface AgentPlatformServiceAsync. Serverseitig gibt es ein Servlet „AgentPlatformService“, das verschiedene Methoden bereitstellt, die über das Interface AgentPlatformServiceAsync aufgerufen werden können. Die Variable ist als AgentPlatformServiceAsync Interface deklariert, wird aber mit einer automatisch generierten Proxy-Klasse instantiiert (s. Quellcode/Konstruktor der Klasse MainframePresenter). In der Methode execute wird über den Aufruf der Methode startSimulation das Szenario übergeben und zum Server geschickt. Für den Aufruf ist ein Objekt notwendig, das das Interface AsyncCalback implementiert, welches wiederum der Methode als interne Klasse übergeben wird. Der Typparameter, in diesem Fall ein Integer, definiert, welcher Rückgabewert vom Server erwartet wird. Außerdem wird in den Methoden onFailure und onSuccess definiert, was passieren soll, wenn der RPC fehlschlägt bzw. erfolgreich war. Wenn die Simulation serverseitig erfolgreich gestartet wurde, dann soll die auf dem Server generierte ID dem aktuellen Szenario zugewiesen werden und der Status der Simulation auf gestartet gesetzt werden.

```
/**
 * start / stop simulation
 *
 * @author Matthias, Nagi
 */
menuName = "Starten/Anhalten";
mapSimMenuItems.put(menuName, this.display.getSimulationMenuBar().addItem(menuName, new Command() {
    public void execute() {
        // doesn't run yet?
        if (!isSimulationRunning()) {
            //Only start simulation if szenario is consisten
            if(checkIfSzenarioIsConsistent()){
            // start simulation
            agentPlatformService.startSimulation(MainFramePresenter.this.currentSzenario, new AsyncCallback<Integer>() {
                public void onFailure(Throwable caught) {
                    Window.alert("Simulation error: An error occured during start of simulation!");
                }

                public void onSuccess(Integer id) {
                    MainFramePresenter.this.currentSzenario.setID(id);
                    MainFramePresenter.this.setSimulationState(true);
                }
            });
            }else {
                Window.alert("Szenario cannot be started, because it is inconsisten. You need at least from each Ramptype one exemplar");
            }
        }
        else {
            // stop simulation
            agentPlatformService.stopSimulation(new EmptyAsyncCallback());
            MainFramePresenter.this.setSimulationState(false);
        }
    }
}));
```

Abbildung 21: Starten einer Simulation

Start des Multiagenten-Systems Nachdem die Daten zum Server geschickt wurden, wird die Methode startSimulation des Servlets AgentPlatformServiceImpl aufgerufen. Für

das Szenario wird eine ID generiert und über das Object Array wird das Szenario allen Agenten als Parameter zur Verfügung gestellt. Danach wird eine Agentenplattform für das jeweilige Szenario gestartet. Die Liste der Conveyor aus dem Szenario wird durchlaufen und für jeden Conveyor werden die benötigten Agenten erzeugt. Die Methode addAgentToSimulation erzeugt anhand der Conveyor- und Szenario-ID einen eindeutigen Namen für jeden Agenten, übergibt die Parameter und fügt ihn der Simulation hinzu. Anschließend wird der Jobagent initialisiert und die Agenten werden gestartet. Die letzte Anweisung schickt die Szenario ID zum Client zurück.

```
public int startSimulation(Szenario szenario) {
    // async function call and static variable, so remember,
    // in case someone else starts before function is finished
    szenario.setID(++szenarioID);

    Object[] argsJobAgent = new Object[1];
    argsJobAgent[0] = szenario;

    List<Conveyor> lstConveyor = szenario.getConveyorList();

    if (lstConveyor.size() < 1)
        return -1;

    startAgentPlatform(szenario);

    for (Conveyor myConveyor : lstConveyor) {
        Object[] argsAgent = new Object[2];
        argsAgent[0] = szenario;
        argsAgent[1] = myConveyor;

        int id = myConveyor.getID();

        if (myConveyor instanceof ConveyorRamp) {
            addAgentToSimulation(id, szenario.getId(), argsAgent, PackageAgent.NAME          , new PackageAgent());
            addAgentToSimulation(id, szenario.getId(), argsAgent, RampOrderAgent.NAME        , new RampOrderAgent());
            addAgentToSimulation(id, szenario.getId(), argsAgent, RampPlattformAgent.NAME    , new RampPlattformAgent());
            addAgentToSimulation(id, szenario.getId(), argsAgent, RampRoutingAgent.NAME      , new RampRoutingAgent());
        } else if (myConveyor instanceof ConveyorVehicle) {
            addAgentToSimulation(id, szenario.getId(), argsAgent, PackageAgent.NAME          , new PackageAgent());
            addAgentToSimulation(id, szenario.getId(), argsAgent, VehiclePlattformAgent.NAME , new VehiclePlattformAgent());
            addAgentToSimulation(id, szenario.getId(), argsAgent, VehicleRoutingAgent.NAME   , new VehicleRoutingAgent());
        }
    }

    addAgentToSimulation(0, szenario.getId(), argsJobAgent, JobAgent.NAME, new JobAgent());

    startAgents();

    return szenario.getId();
}
```

Abbildung 22: Starten des MAS

Start des Jobtimers Nachdem der Jobagent alle Informationen erhalten hat, die er für die Zuweisung von Aufträgen benötigt, schickt er eine Nachricht an den Client, dass die Aufträge zum Server geschickt werden können (Vgl.Abschnitt 5.4.6). Der Client startet den Jobtimer mit der Methode startJobTimer des MainframePresenters (s.Abbildung 23). Die Integer Variable elapsedTimeSec repräsentiert die Zeit. Der Timer wird gestartet und die run-Methode wird kontinuierlich ausgeführt, bis der Jobtimer gestoppt wird. Die Zeitvariable wird bei jedem Durchlauf um eins hochgezählt, was bedeutet, dass eine Sekunde vergangen ist. Die einzelnen Jobs aus der Liste werden in der for-Schleife abgefragt und ihr Zeitstempel wird mit der elapsedTimeSec Variable abgeglichen. Ist die vergangene Zeit größer oder gleich der Zeit zu der ein Job ausgeführt werden soll, dann wird der Job mit

der Methode addJob zum Server geschickt. Mit der vorletzten Anweisung wird festgelegt, in welchen Zeitabständen der Timer ausgeführt werden soll. Die letzte Anweisung dient dazu, den Timer initial zu starten.

```
public void startJobTimer() {
    if (tmrJobStarter != null)
        return;

    elapsedTimeSec = 0;

    tmrJobStarter = new Timer() {
        public void run() {
            elapsedTimeSec += 1;

            if (lstJobs.size() < 1)
                return;

            //Job pendingJob = lstJobs.getJob(0);

            for (Job pendingJob : lstJobs.getJoblist()) {
                if (elapsedTimeSec >= pendingJob.getTimestamp()) {
                    agentPlatformService.addJob(currentSzenario.getId(), pendingJob, new EmptyAsyncCallback());
                    break;
                }
            }
        }
    };

    tmrJobStarter.scheduleRepeating(1000);
    tmrJobStarter.run();
}
```

Abbildung 23: Start des Jobtimers

Abbildung 24 zeigt die serverseitige Weiterleitung des empfangenen Auftrags. In der Methode addJob des AgentPlatformServiceImpl Servlets wird der jeweilige Jobagent anhand der Szenario ID aus einer Hashmap geholt. Es wird ein ACLMessageObjekt erzeugt und der Nachrichtentyp wird im Konstruktor festgelegt, um gezielt die notwendige Behaviour zu adressieren. Als Empfänger wird der Jobagent selber festgelegt. Anschließend wird der Job der Nachricht als ContentObject hinzugefügt und versendet.

Visualisierung von Zustandsveränderungen Die einzelnen Agenten führen Aktionen durch. Werden dadurch Zustände verändert (z. B. Hinzufügen eines Pakets etc.), müssen diese visualisiert werden. Das MAS läuft auf dem Server. Damit ein Client die Zustandsveränderungen visualisieren kann, muss er vom Server über die Art der Zustandsveränderung informiert werden. Das Schicken von Nachrichten vom Server zum Client wurde mithilfe des GWT Eventservices implementiert. Der Eventservice ist ein event-basiertes Kommunikationsframework, dass auf dem GWT-RPC Mechanismus und der Comet Server-Push Technologie basiert (Vgl. [8]). Abbildung 25 zeigt die Methode AddPackage des Packa-

```
public void addJob(int szenarioID, Job myJob) {
    Agent myAgent = mapJobAgent.get(szenarioID);

    ACLMessage msg = new ACLMessage(MessageType.ASSIGN_JOB);
    msg.addReceiver(myAgent.getAID());

    try {
        msg.setContentObject(myJob);
        myAgent.send(msg);
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}
```

Abbildung 24: Weiterleitung des Auftrags

geagents, die für das Hinzufügen eines Pakets benötigt wird. Nachdem das Paket in die Liste eingefügt wurde, wird mithilfe der Klasse EventHelper ein Event zum Client gefeuert. Der EventHelper erbt von dem Servlet RemoteEventServiceServlet, das vom EventService Framework bereitgestellt wird und das GWT-Servlet, um die Funktionalität zum Versenden von Events erweitert (Vgl.[8]).

Die Klasse PackageAddedEvent ist eine selbst definierte Event-Klasse, die das Interface Event des Eventservices implementiert (Vgl.[8]). Mithilfe dieser Klasse können Informationen für den Client transportiert werden. Die Informationen werden im Konstruktor der Klasse PackageAddedEvent übergeben. In diesem Fall werden die Conveyor- und Package-ID übergeben, damit der Client weiß, welchen Conveyor er neu zeichnen soll und welches Paket mit welcher ID hinzugefügt wurde.

Abbildung 26 zeigt die clientseitige Verarbeitung der Events, die vom Server empfangen werden. In der Methode HandleEvents des Mainframepresenters , wird zunächst eine Instanz der Klasse RemoteEventService über die RemoteEventServiceFactory geliefert. Alle vorhandenen Listener werden entfernt und es wird ein neuer Listener hinzugefügt. Beim Hinzufügen des Listeners wird eine Domain mit einer Zeichenkette übergeben. Durch die Domain kann gezielt festgelegt werden, für wen eine Nachricht bestimmt ist (Vgl.[8]). Die

```
private class AddPackage extends CyclicReceiverBehaviour {
    protected AddPackage(int msgType) {
        super(MessageTemplate.MatchPerformative(msgType));
    }

    public void onMessage(ACLMessage msg) throws UnreadableException {
        PackageAgent currentAgent = (PackageAgent) myAgent;
        PackageData myPackage = (PackageData) msg.getContentObject();

        currentAgent.lstPackage.add(myPackage);

        EventHelper.addEvent(new PackageAddedEvent(myConveyor.getID(), myPackage.getPackageID()));
```

Abbildung 25: Feuern eines Events

Zuordnung einer Nachricht zu einer Domain, wird beim Versenden der Nachricht festgelegt (S. Quellcode Klasse EventHelper). In der Methode apply des Listeners wird für alle definierten Events festgelegt, welche Aktion beim Empfang auf dem Client ausgeführt werden soll. Beispielsweise wird durch das SimStartedEvent das Starten des Jobtimers durch die Methode startJobTimer initialisiert.

```
private void handleEvents() {
    myRES = RemoteEventServiceFactory.getInstance().getRemoteEventService();
    myRES.removeListeners();

    myRES.addListener(DomainFactory.getDomain(DOMAIN_NAME), new RemoteEventListener() {
        public void apply(Event anEvent) {
            if (anEvent instanceof SimStartedEvent) {
                bSimulationRunning = true;
                display.log("Simulation started!");

                startJobTimer();

                return;
            }

            if (anEvent instanceof SimStoppedEvent) {
                bSimulationRunning = false;
                display.log("Simulation stopped!");

                stopJobTimer();

                return;
            }

            if (anEvent instanceof JobAssignedEvent) {
                JobAssignedEvent myEvent = (JobAssignedEvent)anEvent;

                lstJobs.removeJob(myEvent.getJob().getPackageId(), myEvent.getJob().getType());

                setupJobTable();

                return;
            }

            if (anEvent instanceof JobUnassignableEvent) {
                JobUnassignableEvent myEvent = (JobUnassignableEvent)anEvent;
```

Abbildung 26: Clientseitige Verarbeitung des Events

6 Teilbericht Materialfluss

Das Ziel der Materialfgruppe ist das dezentrale Management der Fördereinheiten auf der Rampen und auf dem Volksbots.

6.1 Lastenheft

Aus einem neutzwerk drathlosen Sensorknoten müssen die Rampen auf mikrocontrollerebene miteiander kommunizieren können und die Fördereinheiten steuern. Die Steuerung und Überwachung muss dezentral durch softwareagenten ablaufen. Die Mikrocontrollern dienen als Agentenplattform und bei der Entwicklung muss auf Leistungsressourcen (CPU, Kommunikationsbandbreite, Speicher) geachtet werden.

Produktfunktionen

1. **Aktorik/sensorik**: Die Rampen verfügen über Magnetstreifen zum Vereinzeln der Pakete und Lichtschranken zum Erkennen von ein- bzw. ausgehenden Pakete. An den ATMega128 Mikrocontroller der Rampen werden die Lichtschranken und Magnetstifte angeschlossen.
2. **Echtzeitsteuerung**: An den ATMega128 Mikrocontroller der Rampen werden die Lichtschranken und Magnetstifte angeschlossen. Der Controller führt die Befehle für die rampe aus.
3. **Kommunikation**:Die Micaz-Module kommunizieren drahtlos mit anderen Rampen und Volksbot
4. **Synchronisation**: Die Simulation wird über ein Micaz-Modul an die drahtlose Kommunikation angebunden. Die Synchronisation der Zustand erfolgt mittels vorhandenem Ice Interface.
5. **Disposition**:Die Controller kennen den Belegungszustand der Rampe und generieren entsprechend Aufträge, die sie an die Fahrzeuge vergeben. Das ganze soll nach dem FIFO-Verfahren ablaufen.
6. **Übergabe**: Wenn eine Ein- oder Auslagerung an einer Rampe ausgeführt werden soll, so übernimmt der Controller der Rampe die Kontrolle über die Fördereinheit des Fahrzeugs und sorgt dafür, dass das Paket verladen wird
7. **Kooperation**: Einsatz Kooperativer Lösungsstrategien für die Materialflusssteuerung. Überwachung und Steuerung mittels Multi-Agentensystem

6.2 Komponentenbeschreibung

6.2.1 Rampe

Eine rampe dient zur Aufnahme und Ausgabe von paketen. Es kann bis zu 4 paketen aufnehmen. Es gibt eine Eingangsseite und eine Ausgangsseite. Die Pakete bewegen sich nur in eine Richtung. Jeder Slot wird mit einer Lichtschranke überwacht und ausfahrbare Bolzen vereinzeln die Pakete. Ein Micaz Controller ist die Agentenplattform. Bild

6.2.2 Mikrocontroller

Ein Mikrocontroller ist ein kleiner Computer auf einem einzelnen Halbleiter-Chip. Dazu gehört ein Prozessor, der Programme ausführen kann, Arbeits- und Programmspeicher sowie Schnittstellen, die eine Kommunikation mit der Umgebung ermöglichen sog. Peripheriefunktionen [22]. Mit ihnen lassen sich komplexe Aufgaben lösen, für die sonst ein aufwändiger Schaltungsaufbau notwendig wäre. Standardmäßig sind folgende Bestandteile in Mikrocontrollern integriert: CPU, SRAM und Flash-Speicher für den Programmcode. Weiterhin bieten MCs analoge und digitale Ports, mehrere AD/DA-Wandler, Timer und Schnittstellen zur Kommunikation mit der Aussenwelt [20, vgl.]. In der nachstehenden Abbildung ist der allgemeine schematische Aufbau eines Mikrocontrollers in folgender Abbildung dargestellt.

- Prozessor (CPU)
  - Arithmetic Logic Unit kurz ALU (Rechenwerk)
  - 32 GPIO-Register (Arbeitsregister für ALU)
  - Programmcounter (Programmposition)
  - Statusregister (Status der aktuellen Operation)
- Speicher
  - SRAM Datenspeicher (Static Random-Access Memory)
  - Flash ROM Programmspeicher (Read Only Memory)
  - EEPROM Festspeicher (Electrically Erasable Programmable Read-Only Memory)
- Peripheriekomponenten
  - I/O-Ports Primärfunktion der Pins (Ein- und Ausgänge)
  - A/D-Wandler (Einlesen von analogen Spannungen)

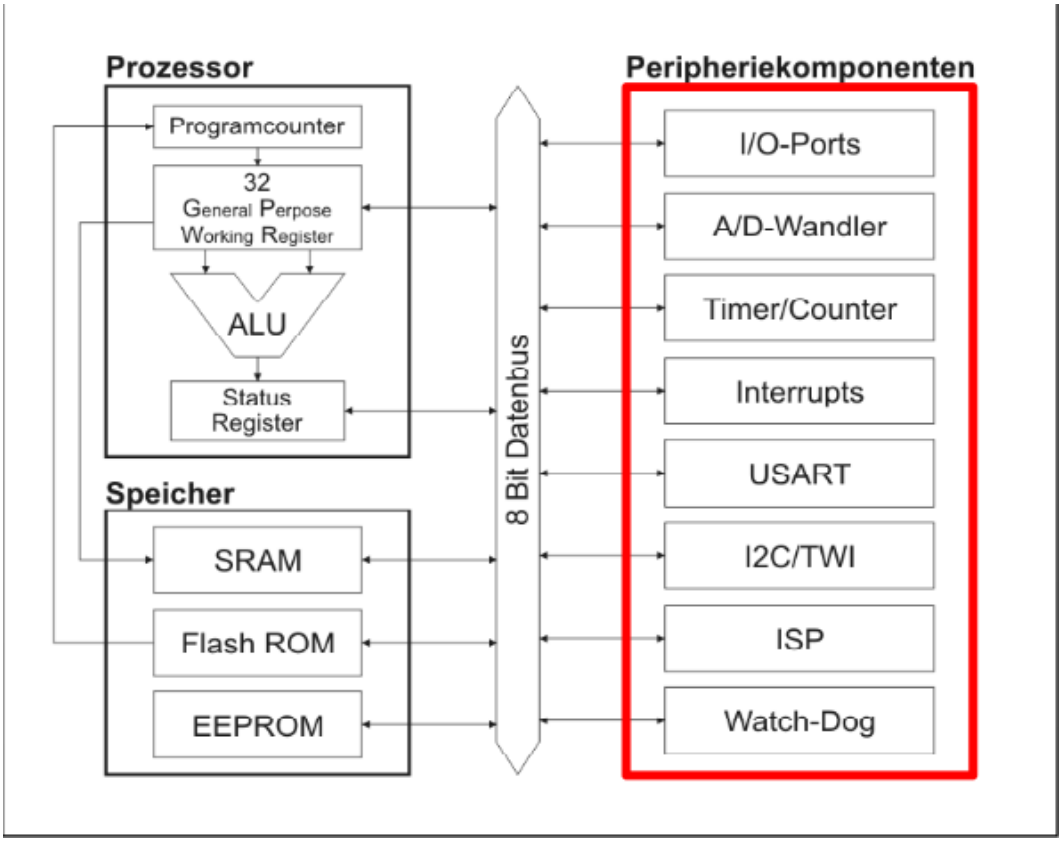

Abbildung 27: schematischer Aufbau eines Mikrocontrollers [2]

- Timer/Counter (Zeitintervall-/PBM-Generator)
- Interrupts (Programmunterbrechungsroutinen)
- USART, I2C/TWI und SPI (Kommunikationsschnittstellen)
- Watch-Dog (Absicherung gegen Systemfehler)
- ISP (Schnittstelle zum Übertragen des kompilierten Programms)

Mikrocontroller sind im heutigen Leben weit verbreitet und es gibt eine viele Anzahl von Herstellern, die mikrocontroller anbieten. Im folgenden werden einige Hersteller mit ihren MC-Familien beispielhaft aufgefürt:

- Intel (8051-Serie)
- Renesas (H8)
- Zilog (Z8)
- Microchip (Pic)

- Freescale (früher Motorola) (68HC08 bzw. 68HCS08)
- Atmel (AVR, 8051-Serie)

Für das Projekt FAISE wurde das Atmel-Serie eingesetzt. Es sind Mikrocontroller mit erweiterten Peripherien und Funktionen, die auf der 8-Bit-AVR-Architektur basieren. Bei AVR handelt es sich um einen RISC-Kern, der an der Universität von Trondheim in Norwegen entwickelt und von Atmel aufgekauft wurde. Die CPU besitzt 32 allgemeine 8-Bit Register (general purpose registers) und ist in der Lage in einem einzigen Taktzyklus Daten aus zwei beliebigen Registern in die ALU zu laden, diese zu verarbeiten und das Ergebnis in einem beliebigen Register zu speichern [20, vgl.]. Die Konfiguration eines Atmega 8 der Firma Atmel sieht so aus:

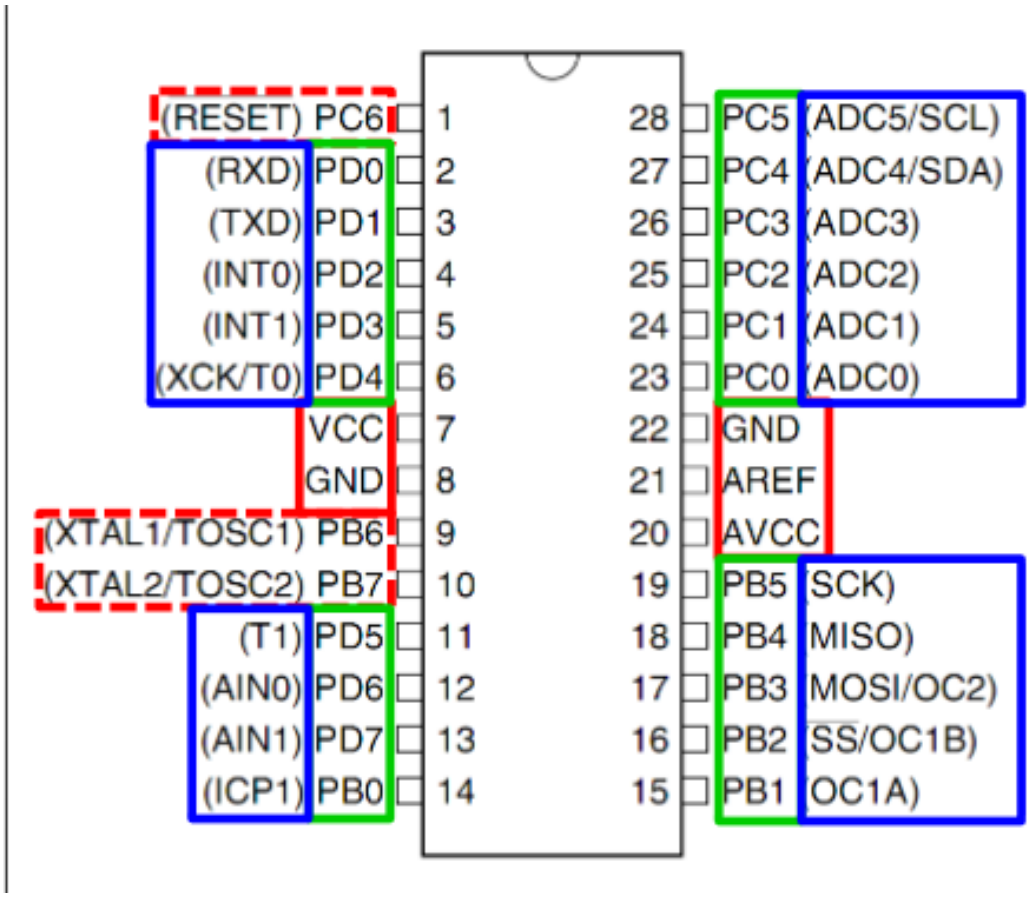

Abbildung 28: Atmel 8
(http://www.ids.tu-bs.de/tl_files/Lehre/Vorlesungen/Simulation2/Einfuehrung_in_die_MC_Programmierung_Teil1.pdf)

- Pins für die Minimalbeschaltung

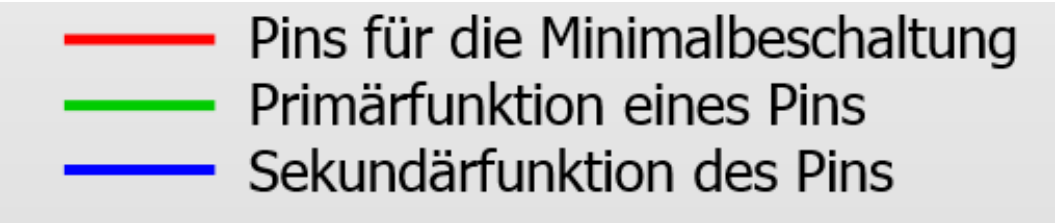

Abbildung 29: Atmel 8 (http://www.ids.tu-bs.de/tl_files/Lehre/Vorlesungen/Simulation2/Einfuehrung_in_die_MC_Programmierung_Teil1.pdf)

  - Spannungsversorgung
  - ReferenzspannungTaktgeber
  - Reset
- Primärfunktion eines Pins
  - Ein- bzw. Ausgang
- Sekundärfunktion des Pins
  - A/D-Wandlereingang
  - Ext. Interrupt
  - PBM-Ausgang

Für die Programmierung der AVR-Controller gibt es eine kostenlose Entwicklungsumgebung AVR-Studio, die das Einbinden des Compilers problemlos erlaubt.

6.2.3 Sensorik/ Aktorik

Hauptziel der Teilgruppe Materialfluss ist das Management von Paketen auf einer Rampe. Die Aufgabe der Sensorik ist dabei, die mit Lichtschranken ausgestatteten Rampen Paketen zu detektieren und auf Änderung der Position der Paketen zu reagieren. Die Lichtschranken bestehen aus einer Lichtstrahlenquelle (dem Sender) und einem Sensor (dem Empfänger) für diese Strahlung. Als Lichtquelle kommt Infrarotlicht zum Einsatz und der Vorteil besteht in der einfachen Einstellung des Sensorsystems durch den sichtbaren Lichtfleck. Das Funktionsprinzip der Lichtschranke besteht darin, der zu ändernden Zustand durch die Lichtintensität mit dem Sensor zu registrieren. Die Rampen werden auf Hardwareebene um eine Aktorik zum Arretieren der Kisten ergänzt. Diese Aktoren (in unserem Fall die eingesetzte Bolzenpaar) sind für das Ausführen von Bewegungen zuständig. Sie sind aktive

Stellelemente, die in der Antriebs - und Steuerungstechnik, die vom Mikrorechner angesteuert werden und das Verhalten des Prozesses durch das vom Sensor kommende Signal in einer gewünschten Weise zu ermöglichen. In dieser allgemeinen Darstellung stehen die Ausgangssignale eines Sensors und die Stellsignale der Aktoren mit einem Informationsverarbeitungssystem (IVS) in Verbindung.

6.2.4 Contiki

Contiki ist ein Open Source Echtzeitbetriebssystem, das bei uns in der PG auf den MICAz-Modulen eingesetzt wird. Contiki bietet einen einfachen ereignisgesteuerten Betriebssystemkern mit sogenannten Protothreads, optionalem präemptiven Multiprogramming, Interprozesskommunikation via Messagepassing durch Events, eine dynamische Prozessstruktur mit Unterstützung für das Laden und Entladen von Programmen, nativen TCP/IP-Support ̈ber den uIP TCP/IP-Stack und eine grafische Benutzerschnittstelle, welche direkt auf einem Bildschirm oder als virtuelle Anzeige über Telnet oder VNC genutzt werden kann [23].

Systemarchitektur Ein laufendes Contiki System besteht aus dem Kernel, Bibliotheken, Prozessen und dem Programm-Lader, mit dem Anwendun- gen zur Laufzeit aus dem Speicher oder über ein Funkmodul geladen werden können. Die unter stehende Abbildung zeigt die Aufteilung des Betriebssystems in zwei Teile. Der Core ist ein Basissystem und besteht aus dem Kernel, Bibliotheken, Gerätetreiber und der Programm-Lader. Im allgemeinen sind Änderungen am Core nicht vorgesehen und nur unter Verwendung eines speziellen Bootloader möglich. Die konkrete Aufteilung des Systems in Core und ladbare Programme wird beim Kompilieren des Systems entschieden und hängt von der Hardware-Plattform ab [vgl. 21, S. 7]. Gerätetreiber werden als Bibliotheken implementiert.

Events In Contiki kommunizieren Prozesse über Events. Auch der Kernel versendet Events, um Prozesse über ihren Status (Init, Continue, Exit) oder über abgelaufene Timer zu Informieren. Zur Identifikation stehen dabei Event IDs zur Verfügung. Die Event IDs 0-127 können vom Benutzer frei vergeben werden, während die Prozess IDs ab 128 vom System genutzt werden. Grundsätzlich unterscheidet Contiki zwischen synchronen und asynchronen Events.

- Asynchronen Events

Asynchrone Events sind eine Form der Deferred Procedure Call: asynchrone Events werden vom Kernel in einer Warteschlange gespeichert. Die Scheduling-Funktion des Kernels

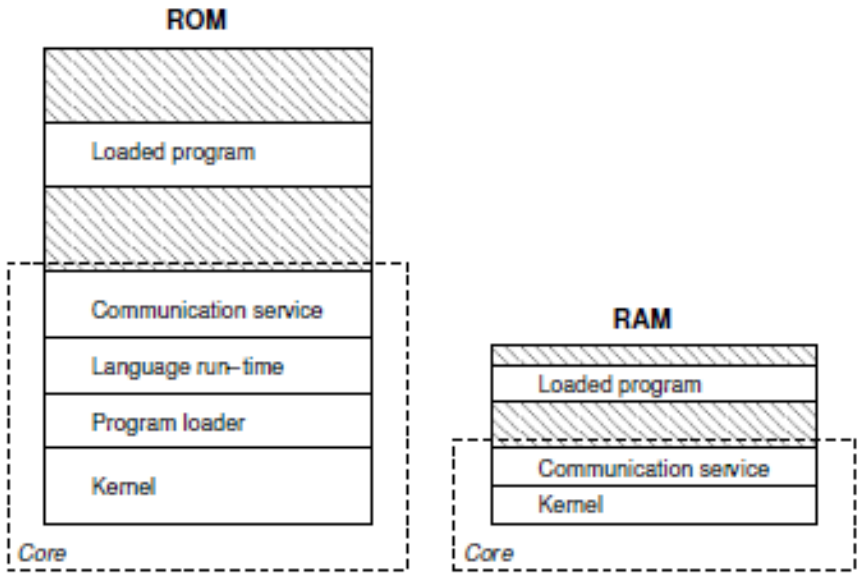

Abbildung 30: Adam Dunkels, Björn Grönvall, Thiemo Voigt: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. (Quelle:http://www.up.edu.ps/ocw/upinar/moodledata/872/moddata/assignment/970/1287/10.1.1.59.2303.pdf)

läuft nach Systemstart in einer Endlosschleife. In jedem Durchlauf wird ein Event aus der Schlange entnommen und wird einige Zeit später an den Zielprozess weitergeleitet.

- Synchronen Events

Synchrone Events gleichen einem Funktionsaufruf. Die werden ohne Umweg über die Warteschlange direkt an den Empfänger-Prozess zugestellt [vgl. 21, S. 7]. Mit der Funktion process_post_synch(&example_process, EVENT_ID, msg) wird gezielt ein Prozess aufgerufen (ein Broadcast ist nicht möglich). Während der aufgerufene Prozess aktiv ist, blockiert der Aufrufer und setzt seine Ausführung erst fort, wenn der aufgerufene Prozess die Kontrolle wieder abgibt.

Prozesse Prozesse in Contiki implementieren ein Konzept namens Protothreads. Dies erlaubt es Prozessen, ohne den Overhead und die langen Prozesswechselzeiten von normalen Threads auszukommen. Gleichzeitig können trotzdem andere Prozesse ausgeführt werden, falls ein Prozess auf ein Event (Timer, Nachricht von anderem Prozess...) warten muss. Für die Entwicklung mit Prozessen ist wichtig, dass nicht-statische Variablen nicht zwischen zwei Aufraufen erhalten bleiben. Der relevante Status eines Prozesses sollte daher mithilfe von statischen Variablen abgelegt werden (siehe Variable i im folgenden Beispiel:

Beispielprozess

```
1  PROCESS(example_process, "Example process");
2  AUTOSTART_PROCESSES(&example_process);
3
4  PROCESS_THREAD(example_process, ev, data)
5  {
6    PROCESS_BEGIN();
7    static uint8_t i = 1;
8    while(1) {
9      PROCESS_WAIT_EVENT();
10     printf("Got event number %d\n", i);
11     i++;
12   }
13   PROCESS_END();
14 }
```

In Zeile 1 wird der Prozess initialisiert und in Zeile 2 automatisch beim Boot von Contiki gestartet. Zeile 4 beinhaltet die Deklaration. So können andere Prozesse diesem Prozess Events (mit oder ohne Daten) schicken, auf die unser Beispielprozess mit ev und data zugreifen kann. Zeile 6 kennzeichnet den Beginn der tatsächlichen Ablauflogik. Code über dieser Zeile wird bei jedem Prozessaufruf ausgeführt, dies wird jedoch in den meisten Fällen nicht benötigt. Zeile 13 schließlich beendet den Prozess und entfernt ihn aus der Prozess-Liste des Kernels. In diesem Beispiel wird die Zeile jedoch nie erreicht, sodass er Prozess immer wieder aufgerufen wird, bis er von einem anderen Prozess beendet wird. Wichtige Funktionen in Prozessen:

- PROCESS_WAIT_EVENT()- Wartet auf ein beliebiges Event, bevor die Ausführung fortgesetzt wird
- PROCESS_WAIT_EVENT_UNTIL(condition) - Wartet auf ein beliebiges Event, setzt die Ausführung aber nur fort, wenn die Bedingung erfüllt ist
- PROCESS_WAIT_UNTIL() - Wartet, bis die Bedingung erfüllt ist. Muss den Prozess nicht zwangsläufig anhalten

Prozesse können über Events (siehe Events) oder Polling-Anfragen kommunizieren. Polls sind Events mit hoher Priorität und können genutzt werden, um den angerufenen Prozess so schnell wie möglich auszuführen. Sie sind besonders bei der Abarbeitung von Hardware-

Interrupts wichtig, da Interrupts-Handler keine Events, sondern nur Polling-Anfragen absetzten dürfen [vgl. 21, S. 7].

6.3 Gesamtarchitektur

Zur Umsetzung der dezentralen Steuerung werden Softwareagenten eingesetzt. Bei Software-Agenten handelt es sich um Prozesse, die lose gekoppelt und leicht austauschbar sind [**GH:2010**]. Es existieren verschiedene Definitionen eines Agenten, von denen sich keine als Standard etablieren konnte. Die hier verwendete Definition stammt von Brenner, Zarnekow und Wittig. Sie definieren einen Agenten als „ ... ein Softwareprogramm, das für einen Benutzer bestimmte Aufgaben erledigen kann und dabei einen Grad an Intelligenz besitzt, der es befähigt, seine Aufgaben in Teilen autonom durchzuführen und mit seiner Umwelt auf sinnvolle Art und Weise zu interagieren“[**BZW:1998**]. Die Fähigkeit von Agenten, miteinander zu kommunizieren und zu interagieren, ermöglicht das Erstellen eines Multiagentensystems (MAS). Ein wesentlicher Vorteil von MAS bzw. von verteilten Steuerungssystemen ist die Fähigkeit, dynamisch auf Veränderungen zu reagieren. Ein Beispiel für eine solche Veränderung ist der Ausfall einer Steuerungseinheit bzw. eines Agenten. Der Ausfall einer Einheit hat nicht unbedingt zur Folge, dass das gesamte System ausfällt. Die restlichen Einheiten können sich eigenständig auf eine solche Veränderung einstellen und diese beim weiteren Ablauf berücksichtigen[**Roidl:2012**]. Diese Eigenschaft bringt eine ganze Reihe von Vorteilen für ein dezentral gesteuertes Materialflusssystem mit sich.
Die Systemarchitektur der Materialfgruppe lehnt sich an der AUTOSAR Softwarearchitektur. Die nächste Abbildung zeigt den Aufbau der Softwarearchitektur:

6.3.1 Hardware Level

Auf der Hardwareebene werden die Treiber für die Steuerung der Lichtschranke und Bolzen implementiert. Die Treiber bekommen ein allgemeines Interface für die Agenten RTE.

6.3.2 BackgroundLevel

Auf dieser Ebene werden der Agenten RTE alle Funktionalitäten zur Verfügung gestellt. Das RTOS stellt Infrastruktur wie z.B. Task Management, Timing, Events usw. Die Driver Ebene kümmert sich auf um die Schnittstellen/Pins des Controllers wie z.B. Protokolle angeschlossener Devices[**Stasch:Hahn**]. Die aufgabe des service ist es, Funktionen für spezielle Anwendungen zur Verfügung zu stellen. Darüber kommen die Interfaces, diese sind nötig damit bestimmte Funktionen immer gleich der Agenten RTE zur Verfügung gestellt werden

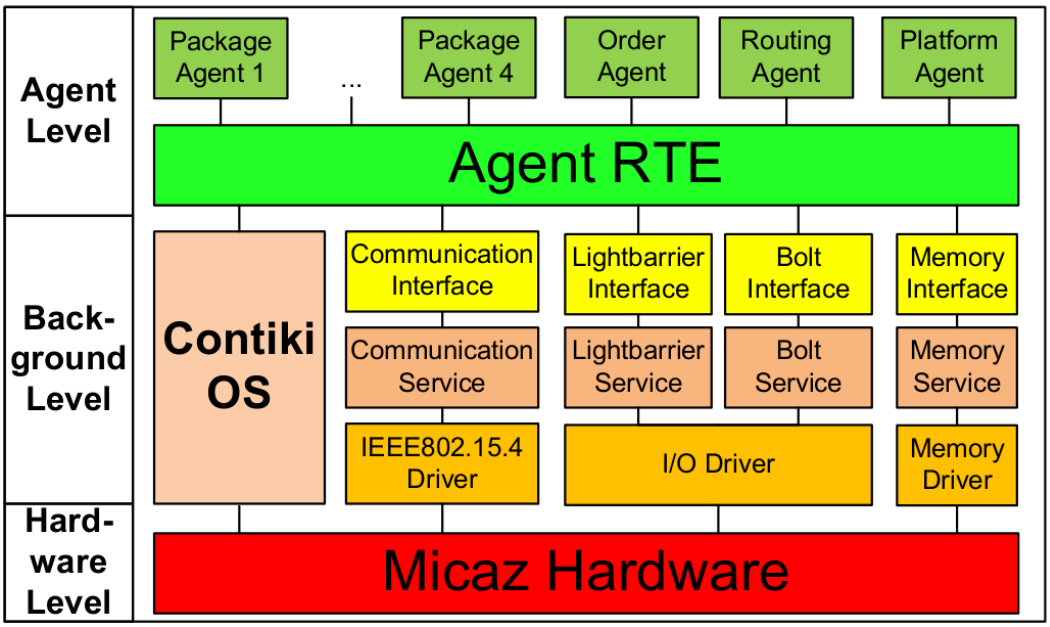

Abbildung 31: Architektur Micaz Rampe[**Stasch:Hahn**]

auch wenn der Service anders ist oder wenn ein Service verschiedene Interfaces bedienen soll[**Stasch:Hahn**].

6.3.3 Agent Level

Auf diesem Level werden die Verschiedene Agenten implementiert. Die Agenten RTE soll den Agenten ihnen alle benötigten Schnittstellen zur Verfügung stellen. Sie soll die Agenten aktivieren und deaktivieren können und die Kommunikation zwischen den Agenten managen (Messages innerhalb und außerhalb der Plattform trennen)[**Stasch:Hahn**].

Plattformagent Jeder Plattformagent ist für die Steuerung von der verantwortlichen Rampe zuständig. Der Plattform Agent kümmert sich um die Annahme und Abgabe der Pakete mit den Volksbots. Er hat die alleinige Zugriffsrechte auf die Lichtschranken und ausfahrbaren Stifte. Er überwacht auch die reale Position der Pakete und damit auch deren Reihenfolge.

Orderagent Der Orderagent spielt die Rolle der alten Materialflussrechner im Materialflusssystem. Seine aufgabe liegt bei der Bearbeitung der aufträge wo und wann die ware zu sein hat. Sie befinden sich auf jeder Plattform und können den Paketen ihre Destination (Zielmodul)mit Prioritäten (z.B. eine Ausgangszuweisung zu einem näheren Zeitpunkt hat

höhere Priorität als eine Spätere und diese ist höher als eine Zwischeneinlagerung usw.) und ihre Agent ID zuweisen[**Stasch:Hahn**].

Paketagent Der Paket Agent repräsentiert die physische Fördereinheit. Beim Eintritt im System kümmert sich der Paketagent um eine destination und schon bei vorhandener Destination gibt er dem Routingagent sein ziel mit Priorität zur weiteren Planung durch. Er aktualisiert regelmäßg seinen Status und gibt neue Routinganfragen wenn sich seine Destination ändert. Wenn das physische Paket das Modul wechselt, wandert der paketagent auch von der jeweiligen Plattform zur nächsten. Da jeder Paket Agent gleich ist wird der Plattformwechsel realisiert, in dem seine Parameter weitergegeben werden, ein neuer Paket Agent auf der nächsten Plattform aktiviert wird und auf der vorigen Plattform deaktiviert[**Stasch:Hahn**].

Routingagent Der Routingagent kümmern sich um die Wegplanung der Agenten im Materialfluss. Der Weg bestimmt die Hops über welche Module das Paket geleitet wird. Das Ziel der Paket-Agents ist es, an ihrem Ziel in der effizientesten Art und Weise durch die Wahl optimaler Routen mit minimalen Abständen und kürzesten Fahrzeiten anzukommen. Das heisst, dass Die Planungsanfragen von Paketagent kommen und bei erfolgter Planung oder neu Planung werden die betroffenen Paket Agenten informiert. Für die Planung der Route wurde ein Routingalgorithmus entwickelt und implementiert.

Literatur

[1] F. Bellifemine, G. Caire und et.al. *JAVA Agent DEvelopment Framework*. `http://jade.tilab.com/`. 2014.

[2] Prof. Dr.-Ing. habil. G.-P. Ostermeye. *Einführung in die Programmierung von Mikrocontrollern*. 2014. URL: `http://www.ids.tu-bs.de/tl\_files/Lehre/Vorlesungen/Simulation2/Einfuehrung\_in\_die\_MC\_Programmierung\_Teil1.pdf`.

[3] S. Ghanbari. *Multiagentensysteme zur Analyse und Verbesserung von vernetztem, kooperativem Lernen*. Waxmann Verlag, 2006.

[4] M. (Hrsg.) Günthner W. A. (Hrsg.) ; ten Hompel. *Internet der Dinge in der Intralogistik*. Berlin : Springer, 2010 (VDI-Buch).

[5] Günther. *Einsatzplanung Fahrerloser Transportsysteme*. Berlin: Logistik Management, Heft 1, 2000.

[6] M. Heinrich. *Transport- und Lagerlogistik: Planung, Struktur, Steuerung und Kosten von Systemen der Intralogistik*. Vieweg + Teubner Verlag, 2009.

[7] Ten Hompel und Heidenblut. *Taschenlexikon Logistik - Abkürzungen, Definitionen und Erläuterungen der wichtigsten Begriffe aus Materialfluss und Logistik*. Berlin: Springer, 2. Auflage, 2008.

[8] https://code.google.com/p/gwteventservice/. *gwteventservice*. `https://code.google.com/p/gwteventservice/`. 2014.

[9] Verein Deutscher Ingenieure. *Begriffe und Erläuterungen im Förderwesen, VDI 2411*. Berlin: VDI- Richtlinien, 1978.

[10] Verein Deutscher Ingenieure. *Fahrerlose Transportsysteme, VDI 2510*. Düsseldorf: Beuth Verlag, 1992.

[11] Verein Deutscher Ingenieure. *Kompatibilität von Fahrerlosen Transportsystemen (FTS) –Energieversorgung und Ladetechnik, VDI 4451*. Düsseldorf: Beuth Verlag, 1994.

[12] Wulz Johannes. *Menschintegrierte Simulation in der Logistik mit Hilfe der Virtuellen Realität*. 2008.

[13] Langenbach Maik. *Beitrag zur Systemfindung von Shuttle-Lagersystemen mit horizontaler Bedienebene*. Dortmund, 2012.

[14] H. Martin. *Transport und Lagerlogistik – Planung, Struktur, Steuerung und Kosten von Systemen der Intralogistik*. Wiesbaden: Springer, 6. Auflage, 2006.

[15] J. Owen. *How to Use Player/Stage 2nd Edition*. Techn. Ber. Player Community, 2010.

[16] A. Stich V.; Bruckner. *Industrielle Logistik*. Mainz, Aachen, 7. Auflage, 2002.

[17] Werner Swoboda. *Lasernavigation sorgt für flexiblere Einsatzmöglichkeiten*. 2014. URL: http://www.industrieanzeiger.de/home/-/article/12503/29025343/Sp%C3%A4tere-Kurskorrektur-macht-kaum-Probleme/art_co_INSTANCE_0000/maximized/.

[18] Günter Ullrich. *Fahrerlose Transportsysteme*. Wiesbaden: Vieweg+Teubner Verlag, 2011.

[19] G. Ulrich. *Fahrerlose Transportsysteme*. Vieweg + Teubner Verlag, 2011.

[20] Seib Viktor. *Einführung in MCU*. 2014. URL: http://www.uni-koblenz.de/~physik/informatik/MCU/Einfuehrung.pdf.

[21] Florian Walter. *Sensorknoten: Betrieb, Netze und Anwendungen*. München, 2010.

[22] Die freie Bibliothek Wikibooks. *Mikrocontroller Einleitung und Grundlagen*. 1999. URL: http://de.wikibooks.org/wiki/Mikrocontroller/_Einleitung_und_Grundlagen.

[23] Die freie Bibliothek Wikipedia. *Contiki*. 2013. URL: http://de.wikipedia.org/wiki/Contiki.

[24] www.gwtproject.org. *GWT*. http://www.gwtproject.org/. 2014.

[25] www.gwtproject.org. *What is GWT RPC?* http://www.gwtproject.org/doc/latest/tutorial/RPC.html. 2014.

[26] www.transportlogistic.de. *Intralogistik & modulare, flexible Systeme für effiziente, transparente und zukunftsfähige Prozesse*. http://www.transportlogistic.de/link/de/26917546. 2014.