Dual Reality Framework

Basistechnologien zum Monitoring und Steuern von Cyber-Physischen Umgebungen

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten I
der Universität des Saarlandes

vorgelegt von
Gerrit Matthias Kahl, M.Sc.

Saarbrücken
26. November 2014

Dekan:
Prof. Dr. Markus Bläser
Berichterstatter:
Prof. Dr. Antonio Krüger
Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster
Prof. Dr. Max Mühlhäuser
Vorsitzender des Prüfungsausschusses:
Prof. Dr. Philipp Slusallek
Promovierter akademischer Mitarbeiter der Fakultät:
Dr. Boris Brandherm
Tag des Kolloquiums:
26.11.2014

Danksagung

Die vorliegende Arbeit wurde vorwiegend im Rahmen der beiden Software-Cluster-Projekte “EMERGENT – Grundlagen emergenter Software“ und „SINNODIUM - Softwareinnovationen für das digitale Unternehmen“ am Deutschen Forschungszentrum für Künstliche Intelligenz entwickelt, welche vom Bundesministerium für Bildung und Forschung (BMBF) gefördert wurden. Grundlagen zur praktischen Umsetzung sind zudem im Rahmen des Software Campus entstanden.

Von der Entstehung bis zur Finalisierung der Arbeit haben mich viele Personen unterstützt und inspiriert. Ihnen möchte ich an dieser Stelle vielmals für die konstruktiven Ratschläge, die interessanten Diskussionen und das hilfreiche Feedback danken.

An erster Stelle möchte ich meinem Doktorvater Antonio Krüger danken, der mich auf dem Weg der Arbeit begleitet und stets mit Feedback unterstützt hat. Ohne die Aufnahme in seine Arbeitsgruppe und seine Unterstützung wäre diese Arbeit nicht möglich gewesen. Ebenso möchte ich Wolfgang Wahlster danken, der es mir durch das Stellenangebot ermöglicht hat, am DFKI zu arbeiten. Gleichzeitig möchte ich ihm danken, dass er sich als Gutachter der Arbeit bereit erklärt hat. Des Weiteren geht mein Dank an Max Mühlhäuser, der kurzfristig einer Begutachtung der Arbeit zugestimmt hat.

Ein großer Dank geht auch an meine Arbeitskollegen, die mir in einer harmonischen Arbeitsatmosphäre konstruktives Feedback in jedem Stadium meiner Arbeit gegeben haben. Neben den Mitarbeitern danke ich insbesondere auch den Bachelor- und Masterstudenten sowie den Hilfswissenschaftlern, deren Arbeiten ich betreuen durfte und die einen Großteil der Umsetzungen meiner Ideen verwirklichten und mit eigenen Ansätzen anreicherten.

Mira Spassova und Ralf Jung möchte ich einen besonderen Dank aussprechen, da sie mir mit viel Mühe und Aufwand Feedback zur schriftlichen Ausarbeitung gegeben haben, um deren Qualität zu steigern.

Schließlich möchte ich meiner Familie und meinen Freunden danken, die insbesondere in der Zeit der Verfassung der schriftlichen Arbeit viel Toleranz und Verständnis aufbringen mussten, wenn ich aus Zeitgründen persönliche Ereignisse absagen musste. Insbesondere möchte ich meiner Freundin, Denise Paradowski, für ihre Unterstützung, ihr Mitgefühl und ihre Motivation danken.

Gerrit Kahl, September 2014

Zusammenfassung

Diese Promotionsarbeit untersucht die Thematik des Monitoring und der Steuerung von Cyber-Physischen Umgebungen (CPE). In diesem Zusammenhang wird das Konzept und die Umsetzung eines *Dual Reality (DR) Frameworks* präsentiert, welches sich aus zwei Komponenten zusammensetzt: dem *Dual Reality Management Dashboard (*DURMAD*)* zur interaktiven dreidimensionalen Visualisierung von CPE und dem *Event Broadcasting Service (*EBS*)*, einer modularen Kommunikationsinfrastruktur. Hierbei stellt DURMAD basierend auf dem DR-Konzept den aktuellen Status der Umgebung in einem 3D-Modell visuell dar. Gleichzeitig umfasst es weitere Auswertungs- und Darstellungsformen, welche verschiedene Formen der Entscheidungsunterstützung für Manager der Umgebung bieten. Speziell entwickelte Filtermechanismen für den EBS ermöglichen eine Vorverarbeitung der Informationen vor dem Versenden bzw. nach dem Empfangen von Events. Durch offene Strukturen können externe Applikationen an das DR-Framework angeschlossen werden. Dies wird anhand von Objektgedächtnissen, semantischen Beschreibungen und Prozessmodellen präsentiert. Basierend auf einer Formalisierung von Dual Reality wird der Begriff *Erweiterte Dual Reality (DR++)* definiert, welcher auch die Auswirkungen von Simulationen in DR-Applikationen umfasst. Durch eine Integration des DR-Frameworks in das Innovative Retail Laboratory werden die Potenziale der erarbeiteten Konzepte anhand einer beispielhaften Implementierung in der Einzelhandelsdomäne aufgezeigt.

Abstract

Within the scope of this dissertation, the issues of monitoring and control of *Cyber-Physical Environments (CPE)* have been investigated. In this context, the concept and implementation of a *Dual Reality (DR) framework* is presented, consisting of two components: the *Dual Reality Management Dashboard (*DURMAD*)* for interactive three-dimensional visualization of instrumented environments and the *Event Broadcasting Service (*EBS*)*, a modular communication infrastructure. DURMAD is based on the DR-concept and thus visually represents the current status of the environment in a 3D model. Simultaneously, it includes more analysis and presentation tools providing various forms of decision-makting support for managers of these environments. Specially developed filter mechanisms for the EBS allow preprocessing of the information before sending or after receiving events. By means of open structures external applications can be connected to the DR framework. This is pointed out by digital object memories, semantic descriptions and process models. Based on a formalization of Dual Reality, the term *Advanced Dual Reality (DR ++)* is defined, which includes the impact of simulations in DR applications. By integrating the DR framework in the Innovative Retail Laboratory, the potential of the developed concepts on the basis of an exemplary implementation in the retail domain are shown.

Erfolg hat nur, wer etwas tut,

während er auf den Erfolg wartet.

Thomas A. Edison

Inhaltsverzeichnis

I Einleitung und Grundlagen **1**

1 Einleitung **3**
1.1 Herausforderungen und Themenmotivation 5
1.1.1 Monitoring und Steuerung 7
1.1.2 Virtualisierung und Visualisierung 8
1.1.3 Dual Reality Management Dashboard 11
1.2 Forschungsfragen 13
1.3 Aufbau der Promotionsarbeit 15

2 Theoretische Grundlagen und Basiskonzepte **17**
2.1 Datenakquise und -verarbeitung 17
2.1.1 Modalität 18
2.1.2 Internet der Dinge und Dienste 19
2.1.3 Digitale Objektgedächtnisse 20
2.1.4 Kontext 21
2.1.5 Agenten 23
2.2 Smarte Umgebungen 25
2.2.1 Sensoren und Aktuatoren 25
2.2.2 Datenverarbeitung in Smarten Umgebungen 28
2.2.3 Unterscheidungskriterien von Smarten Umgebungen 29
2.2.4 Cyber-Physische Umgebungen 32
2.3 Kommunikation 35
2.3.1 Blackboard und Multiblackboard Architekturen 39
2.3.2 Tupelräume 40
2.3.3 Publish/Subscribe 41
2.3.4 Komplexe Eventverarbeitungssysteme 43
2.3.5 Datenstrom Managementsysteme 44
2.4 Visualisierung, Virtualisierung und Simulation 45
2.4.1 Reale und virtuelle Welt 46
2.4.2 3D im Web 49
2.4.3 Simulation und Vorhersagen 51
2.5 Monitoring und Steuerung von Smarten Umgebungen 52

2.5.1 Monitoring von Smarten Umgebungen . . . 53
2.5.2 Erfassung von Menschenansammlungen . . . 54
2.5.3 Business Intelligence . . . 54
2.5.4 Dashboard . . . 55
2.5.5 Leitstände zur Verkehrsflussüberwachung . . . 56
2.5.6 Planungstools . . . 57
2.6 Zusammenfassung . . . 58

II Stand der Wissenschaft und Technik 61

3 Verwandte Arbeiten 63
3.1 Kommunikationsinfrastrukturen . . . 63
3.1.1 T Spaces . . . 63
3.1.2 Event Heap / iROS . . . 64
3.1.3 MundoCore . . . 65
3.1.4 MQTT / MQTT-SN . . . 66
3.1.5 Snoop / Sentinel . . . 68
3.1.6 Aurora / SQuAl . . . 69
3.1.7 TelegraphCQ . . . 70
3.1.8 CQL / STREAM . . . 72
3.1.9 GEM . . . 74
3.1.10 Cayuga . . . 75
3.1.11 RACED . . . 76
3.1.12 T-REX / TESLA . . . 77
3.2 Dual Reality Visualisierungskomponenten & Managementtools . . . 78
3.2.1 Repräsentation von Kontextfaktoren . . . 78
3.2.2 Twin-World Mediator . . . 80
3.2.3 Eolus One . . . 81
3.2.4 Virtuelle Schokoladenfabrik . . . 82
3.2.5 iWorlds . . . 83
3.2.6 YAMAMOTO . . . 84
3.3 Zusammenfassung . . . 87
3.3.1 Kommunikationsinfrastrukturen im Vergleich . . . 87
3.3.2 Visualisierungskonzepte im Vergleich . . . 91

III Dual Reality Framework für Cyber-Physische Umgebungen 95

4 Virtualisierung und Visualisierung von Cyber-Physischen Umgebungen 97
4.1 Modellierung . . . 97
4.1.1 Objektmodellierung . . . 99
4.1.2 Grundrissmodellierung . . . 101
4.2 Virtualisierung . . . 104
4.2.1 Virtualisierung von Cyber-Physischen Umgebungen . . . 105

4.2.2 Formalisierte Definition von Virtualisierung . . . 106
4.2.3 Dual Reality in Cyber-Physischen Umgebungen . . . 108
4.2.4 Formalisierte Definition von Dual Reality . . . 109
4.3 Visualisierung . . . 113
4.4 Zusammenfassung . . . 116

5 Dual Reality Management Dashboard DuRMaD 119
5.1 Datenquellen . . . 120
5.1.1 Objektmodelle . . . 121
5.1.2 Grundrissplan . . . 121
5.1.3 Semantik . . . 122
5.2 Schnittstellen . . . 124
5.2.1 Datenbankschnittstelle . . . 124
5.2.2 OMM . . . 127
5.3 Kernmodule . . . 128
5.3.1 Modell . . . 128
5.3.2 Darstellung des Hauptfensters . . . 130
5.3.3 Textuelle Darstellungen . . . 131
5.3.4 Grafische Darstellungen . . . 132
5.3.5 Informationsrepräsentation im 3D-Modell . . . 134
5.3.6 Steuerung & Verarbeitung in DuRMaD . . . 138
5.4 Verarbeitungsautomatismen . . . 139
5.4.1 Regelsystem . . . 140
5.4.2 Agentensystem . . . 141
5.4.3 BI-Dienste . . . 143
5.5 Kontroll- und Steuerungssysteme . . . 144
5.6 Zusammenfassung . . . 146

6 Event Broadcasting Service (EBS) 149
6.1 Events . . . 151
6.2 Filter und Vorverarbeitung . . . 155
6.2.1 Filter . . . 156
6.2.2 Vorverarbeitung . . . 158
6.2.3 Filter- und Verarbeitungsketten . . . 161
6.3 EBS Architektur . . . 162
6.4 Evaluation . . . 167
6.4.1 Geschwindigkeit . . . 167
6.4.2 Durchsatz . . . 170
6.4.3 Filtereinsatz . . . 171
6.5 Dual Reality Framework . . . 172
6.5.1 Informationsflussvisualisierung . . . 173
6.5.2 Verschmelzung von Real und Virtuell . . . 173
6.5.3 Semantik und Verarbeitung . . . 177
6.6 Zusammenfassung . . . 178

7 Anwendungsszenario in der Einzelhandelsdomäne 181
7.1 Innovative Retail Laboratory (IRL) - Instrumentierte Einzelhandelsumgebung 184
7.1.1 Sensoren und Aktuatoren im IRL 185
7.1.2 Dienste im IRL 187
7.2 Integration des DR-Frameworks in das IRL 195
7.2.1 DuRMAD im IRL 197
7.2.2 Spezialisierung des EBS 202
7.2.3 Integration von Geschäftsprozessen und BI-Diensten 205
7.2.4 Beispielhafte Integration eines Produktberaterdienstes in das DR-Framework 207
7.3 Zusammenfassung 210

IV Diskussion 213

8 Resümee und Ausblick 215
8.1 Zusammenfassung 215
8.2 Wissenschaftliche Beiträge 216
8.2.1 Theoretische Beiträge 216
8.2.2 Ingenieurwissenschaftliche Beiträge 218
8.2.3 Publikationen 220
8.3 Potentielle Anwendungsbereiche 222
8.4 Möglichkeiten für zukünftige Forschungsthemen 225

V Anhang 227

Teil I

Einleitung und Grundlagen

1 EINLEITUNG

Die kommerzielle Verwendung von Sensoren und Aktuatoren in privaten und öffentlichen Umgebungen schreitet rasant voran. Dies ist unter anderem der Miniaturisierung und steigender preislicher Attraktivität geschuldet. Durch den intelligenten Einsatz dieser Technologien sollen primär Menschen in ihrem Alltag unterstützt und Prozesse soweit wie möglich automatisiert werden. Sensoren und Aktuatoren werden dabei miteinander verknüpft und durch Dienste angesteuert, welche außerdem die erfassten Informationen auswerten. Beispielsweise sind in vielen öffentlichen Räumen (z.B. Büroräume) Klimaanlagen installiert, die sich abhängig von der Raumtemperatur selbstständig an- oder abschalten, um ein angenehmes Raumklima zu schaffen. Diese Verbindung von Sensor (Temperatursensor) und Aktuator (Klimagerät) ist verhältnismäßig einfach, da eine direkte Beziehung zwischen ihnen besteht. Betrachtet man jedoch den Fall, dass die Klimaanlage nur angeschaltet wird, wenn sich Personen im Raum befinden, wird ein weiterer Sensor benötigt, der vom Klimasteuerungsdienst mit ausgewertet werden muss. In der Regel würde dieser Sensor von einem anderen, dem Klimageräte-Hersteller womöglich unbekannten, Anbieter produziert werden, so dass sich eine weitere Partei mit der Verknüpfung zwischen Sensor und Klimasteuerungsdienst befassen müsste. Dieser Sensor könnte dabei auch gleichzeitig dazu verwendet werden, das Licht im Raum automatisch an- und auszuschalten. Somit wären die Sensordaten für mehr als nur einen Dienst notwendig und müssten entsprechend verteilt werden. Geht man weiter davon aus, dass es sich um einen Seminarraum handelt, in welchem eine Präsentation gezeigt werden soll, können weitere Faktoren berücksichtigt werden. Soll beispielsweise, sobald eine Präsentation gestartet wird, das Licht ausgeschaltet und die Jalousien der Fenster geschlossen werden, spielen noch weitere Sensoren und Aktuatoren eine Rolle, die über entsprechende Schnittstellen und Dienste miteinander verbunden werden müssten. Eine solche Umgebung, welche mit Sensoren und Aktuatoren instrumentiert ist, wird als *Smarte Umgebung* bezeichnet. Die erfassten Sensordaten werden hierbei mittels entsprechender Dienste verarbeitet, welche daraufhin die jeweiligen Aktuatoren ansteuern.

Aus diesem geschilderten Szenario kann man ersehen, dass bereits in kleineren Umgebungen die Verbindung zwischen Sensoren, Diensten und Aktuatoren beliebig komplex werden kann. Zudem wird für diese Verknüpfung eine Möglichkeit der Datenübertragung benötigt, welche die Kommunikation und den Informationsaustausch zwischen den einzelnen Komponenten regelt. Bei einer Eins-zu-Eins-Beziehung, wie bei der zuerst geschilderten

Situation mit dem sich automatisch anpassenden Klimagerät, ist eine harte Verdrahtung in Form einer direkten Verbindung zwischen dem Sensor und dem Aktuator im ersten Schritt sinnvoll, da durch diese vereinfachte Bauweise ein System mit geringerer Störanfälligkeit entsteht. Zudem kann solch ein System von einer Firma hergestellt werden, die die reibungslose Verbindung über eine geeignete Schnittstelle sicherstellen kann. Werden die erfassten Sensordaten jedoch von weiteren Diensten benötigt oder soll von einem externen Dienst eine Ansteuerung des Aktuators erfolgen, sind harte Verbindungen zwischen Sensoren und Aktuatoren nicht vorteilhaft. Vielmehr sollten die Komponenten im Hinblick auf eine offene Kommunikationsstruktur, welche möglichst standardisiert sein sollte, strikt voneinander getrennt werden, um eine mögliche Erweiterbarkeit der Systeme zu vereinfachen. So entstehen Umgebungen, die mit Sensoren und Aktuatoren instrumentiert sind. Ein weiterer wichtiger Bestandteil solcher Smarter Umgebungen stellen ad-hoc-Verbindungen von mobilen Endgeräten dar, die in die Kommunikationsinfrastruktur integriert werden müssen. Im Unterschied zu stationären Komponenten sind diese nicht ständig erreichbar, sondern melden sich spontan bei der Kommunikationsinfrastruktur an und wieder ab. Entwickelt man in diesem Zusammenhang das vorherige Szenario weiter und geht davon aus, dass während Besprechungen die Mobiltelefone der Mitarbeiter automatisch auf lautlos geschaltet werden sollen, sobald eine Präsentation gezeigt wird, könnte dies auch durch ein System unterstützt werden. Beispielsweise könnte ein entsprechender Dienst auf den Mobiltelefonen installiert werden, so dass diese Informationen aus der Umgebung empfangen und verarbeiten können. Die Schwierigkeit hierbei ist, dass sich diese Geräte ad hoc mit der Infrastruktur des Raums verbinden und mit dieser kommunizieren müssen, ohne die existierenden Kommunikationskanäle zu beeinträchtigen. Dadurch entstehen Verbindungskonstellationen, die bei der Entwicklung solcher Kommunikationsinfrastrukturen weder vorherbestimmt noch getestet werden können. Da die Anzahl der vorhandenen Mobiltelefone a priori nicht bekannt ist und sich jederzeit ändern kann, erfordert dies eine gewisse Agilität der Kommunikationsinfrastruktur.

Geht man nun davon aus, dass es aufgrund der Erweiterung des zuvor beschriebenen Systems um die Kommunikation mit den Mobiltelefonen zu einem unvorhergesehenen Phänomen kommt, wodurch die gesamte Vernetzung nicht mehr funktioniert, zeigen sich weitere Problemstellungen von solch komplexen Umgebungen. Besteht die fehlerhafte Funktionalität beispielsweise darin, dass trotz des Startens einer Präsentation das Licht nicht mehr automatisch ausgeschaltet und die Jalousien nicht mehr geschlossen werden, kann es mehrere Ursachen für die Funktionsstörung geben. Die Fehlersuche kann in diesem Fall aufgrund der Vernetzung der einzelnen Systeme und Komponenten relativ schwierig werden. Es könnten Fehler in der Kommunikationsschnittstelle aufgetreten sein, aber auch Sensoren nicht mehr ordnungsgemäß funktionieren. Oftmals können potentielle Ursachen fehlgedeutet und dadurch die eigentliche Fehlerbehebung erschwert werden. Wird etwa zufälligerweise ein spezieller Sensor parallel zu einem Update der Kommunikationsinfrastruktur defekt, würde zuerst vermutet, die fehlerhafte Funktionalität hinge mit dem Softwareupdate zusammen und dieses würde dahingehend untersucht. Dieses Szenario zeigt die Notwendigkeit, solche Fehlerquellen in Smarten Umgebungen schneller ausfindig machen zu können. Deshalb müssen geeignete Ansätze konzipiert und umgesetzt werden, die

eine Detektion unterstützen und vereinfachen. Gerade aufgrund heterogener und komplexer Infrastrukturen muss besonderes Augenmerk auf eine einfache und schnell verständliche Analysemöglichkeit gelegt werden.

In den zuvor dargestellten Beispielszenarien sind mehrere Problemstellungen bei der Installation und Verwendung Smarter Umgebungen aufgeführt. Sensordaten können für mehrere Dienste von Nutzen sein, so dass eine geeignete Übertragungsmöglichkeit verwendet werden muss. Dasselbe gilt für die Ansteuerung von Aktuatoren, welche ebenfalls durch mehrere Dienste möglich sein muss. Die Anzahl der Sensoren und Aktuatoren kann hierbei beim Aufbau noch nicht bekannt sein, da durch die Dynamik solcher Umgebungen neue hinzukommen können. Diese müssen sich ad hoc mit der Infrastruktur verbinden können, ohne die existierenden Kommunikationswege zu stören. Durch diese Dynamik ist eine Fehlersuche komplex und muss entsprechend unterstützt werden. Des Weiteren werden Darstellungsmöglichkeiten für die erfassten Sensorwerte und möglichen Fehlerquellen benötigt. Gleichzeitig sollte eine solche Darstellung auch die Ansteuerung von Aktuatoren in der Umgebung ermöglichen.

1.1 Herausforderungen und Themenmotivation

Weiser prognostizierte bereits 1991, dass Computer durch Miniaturisierung mit gleichzeitiger Leistungssteigerung der Systeme immer unsichtbarer werden. In diesem Zusammenhang prägte er den Begriff der *ubiquitären Verarbeitung (Ubiquitous Computing)* [Weiser, 1991]. Während in der Anfangszeit der Computerentwicklung in der Regel Zentralrechner existierten, auf die mehrere Personen gleichzeitig zugegriffen haben, wurde im weiteren Verlauf durch die Entwicklung der PCs eine Beziehung geschaffen, in der ein Benutzer Zugriff auf einen Computer hat. Durch die Verbreitung von Smartphones und tragbaren Computern kam es dazu, dass eine Person auf mehrere computerisierte Objekte Zugriff hatte. Entsprechend der Vorhersage werden heutzutage immer mehr Sensoren und Aktuatoren sowohl in öffentlichen als auch in privaten Umgebungen eingebettet. Dabei sind diese oftmals nicht mehr als solche zu erkennen, sondern „verschwinden“ in alltäglichen Gegenständen. Der Verlauf dieser Computerisierung ist schematisch in Abbildung 1.1 dargestellt. Durch die Vernetzung mehrerer Sensoren und Aktuatoren mit Verarbeitungslogiken und Diensten entstehen *Smarte Umgebungen (Smart Spaces)*, die auch unter dem Begriff *Instrumentierte Umgebungen (Instrumented Environments)* bekannt sind. Eine Smarte Umgebung zeichnet sich dabei dadurch aus, dass sie die Fähigkeit hat, Wissen über die Umgebung und deren Einwohner zu erwerben und dieses anzuwenden, um deren Erlebnis in dieser Umgebung zu steigern [Cook und Das, 2004, Seite 3].

Definition 1.1 (Smarte Umgebung) *Eine smarte Umgebung hat die Fähigkeit, Wissen über die Umgebung und deren Einwohner zu erwerben und dieses anzuwenden, um deren Erlebnis in dieser Umgebung zu steigern [Cook und Das, 2004, Seite 3].*

Eine Illustration einer solchen Smarten Umgebung ist in Abbildung 1.2 am Beispiel einer Supermarktumgebung zu sehen. In der Grafik sind diverse Sensoren und Aktuatoren, welche bereits heute oder in naher Zukunft eingesetzt werden, dargestellt. Als Sensoren in

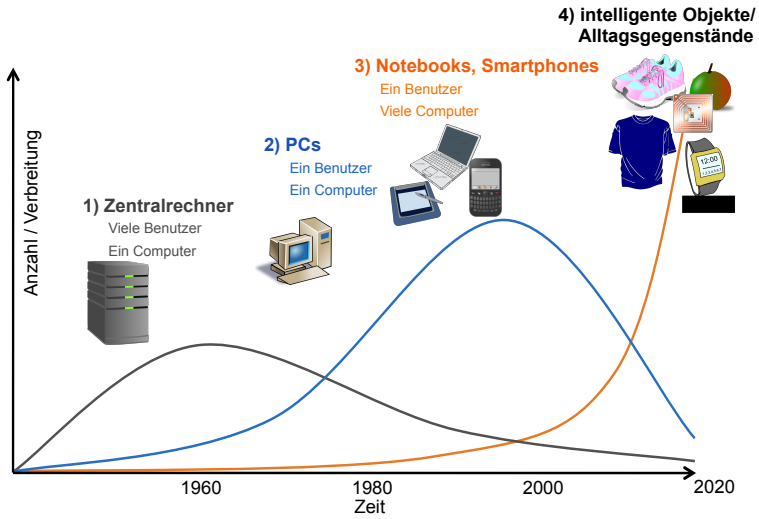


***Abbildung 1.1**: Darstellung der Verteilung von Computersystemen nach der Vorlage von Mark Weiser*

der Umgebung sind hierbei exemplarisch Kamerasysteme und Schranken, basierend auf Funktechnologie, zu nennen. Beide Sensoren werden beispielsweise zum Diebstahlschutz eingesetzt. Ein instrumentierter Einkaufswagen kann alle Produkte erfassen, welche sich in seinem Korb befinden sowie seine Position ermitteln und weiter kommunizieren. Für die Informationsübertragung kann hierbei auf WLAN oder andere Funkprotokolle, wie Bluetooth Low Energy (BLE) zurückgegriffen werden. Eine spezielle Form der Informationsübertragung via BLE stellen iBeacons dar, welche durch Apple entwickelt wurden [Newman, 2014]. Diese ermöglichen eine Notifikation der Geräte, die sich in ihrem Umkreis befinden und ermöglichen dadurch eine Positionierung oder können als Informationstrigger dienen. Des Weiteren sind auf Seiten der Aktuatoren Licht, Lautsprecher, Bildschirme, Drucker, funkende Produkte und elektronische Preisauszeichnung zu nennen. Die Displays werden oftmals als Werbemöglichkeit verwendet. Funkende Produkte können beispielsweise mit einem RFID-Tag oder einem Temperatursensor ausgestattet sein. Neben der Instrumentierung der Umgebung können auch Kunden Sensoren und Aktuatoren mit in den Markt einbringen. Neben Smartphones, welche über mehrere Sensoren und Aktuatoren verfügen sind auch Smart Watches und Smart Glasses immer mehr im Kommen.

Als Erweiterung von Smarten Umgebungen durch die Verknüpfung der Sensor-Aktuator-Netzwerke mit dem Cyberspace entstehen *Cyber-Physische Systeme (CPS)*. Dies bedeutet, dass von den Sensoren erfasste Daten in Kombination mit weltweit verfügbaren Informationen und Diensten ausgewertet und gespeichert werden. Basierend hierauf interagieren die CPS aktiv oder reaktiv mit der physischen sowie mit der digitalen Welt [acatech, 2011]. Unter anderem werden CPS als Weiterentwicklung im industriellen Umfeld gesehen, was unter dem Begriff *Industrie 4.0* bekannt ist [Kagermann et al., 2011, 2012]. Die Möglichkeit des Datenaustauschs zwischen den Maschinen, die Einbeziehung von weltweiten Informationen und die neuartigen Mensch-Maschine-Kommunikationsschnittstellen erlauben eine gezielte

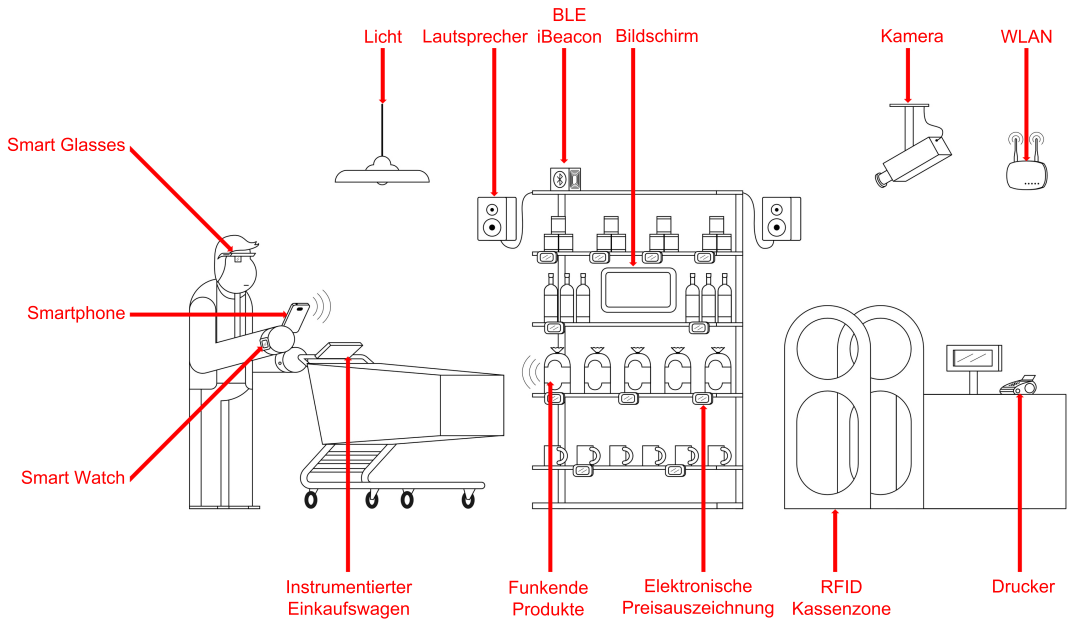


***Abbildung 1.2**: Illustration einer instrumentierten Supermarktumgebung*

Steuerung von Produktionsanlagen. CPS sind daher Technologien, welche innovative Anwendungen möglich machen und beispielsweise zur Ressourcenschonung beitragen und eine individuelle Produktion auf Losgröße 1 ermöglichen. Durch Zusammenschluss aller in der Umgebung befindlichen CPS entstehen sogenannte *Cyper-Physische Umgebungen (CPE, Cyber-Physical Environments)*.

Durch die verteilte heterogene Struktur in CPE können Fehler über mehrere Systeme hinweg propagiert werden. Dies in Verbindung mit mehreren potentiellen Fehlerquellen erschwert die Detektion der eigentlichen Ursache. Aus diesem Grund werden Monitoringmechanismen benötigt, die eine schnelle Fehlerquellenerkennung ermöglichen.

1.1.1 Monitoring und Steuerung

Im Zeitalter von Smarten Umgebungen, in denen mehrere Komponenten miteinander kommunizieren, ist ein Monitoring deutlich schwieriger geworden. Durch die Verbreitung von sehr vielen und unterschiedlichsten Sensoren und damit von immer mehr technologischen Geräten, die in die Kommunikationsinfrastrukturen eingebettet werden sollen, erhöht sich das Risiko für Fehler, welche einen vorhersehbaren, reibungslosen Ablauf der Kommunikation stören und damit die Verarbeitung der übersendeten Daten deutlich verkomplizieren können. Bei Programmen, die auf einem zentralen System laufen, können auftretende Fehler oftmals mit einfachen Mitteln erfasst werden. Anschließend kann entsprechend darauf reagiert werden, da nur ein System mit einer fest vorgegebenen Konfiguration gegeben ist. Bei verteilten Systemen hingegen ist eine solche Überwachung und dadurch Fehler- und Ursachenerkennung erheblich komplizierter. Oftmals werden Fehler aus einem

System über mehrere weitere Systeme propagiert, wodurch die Suche nach der eigentlichen Ursache erschwert wird. Ist beispielsweise ein Sensor defekt, führt dies zu einer fehlerhaften Verarbeitung auf den Folgesystemen und hat womöglich daher keine Ansteuerung von Aktuatoren zur Folge. Als Resultat kann man jedoch nur feststellen, dass keine Veränderung in der Umgebung erfolgt. Als potentielle Ursache dafür kann jede Station in Frage kommen, also Aktuator, verarbeitende Dienste, Kommunikationsschnittstelle und Sensor. Gerade bei heterogenen, sich ständig verändernden Konstellationen von Komponenten in Smarten Umgebungen ist die Fehleranfälligkeit höher als bei starren, standardisierten Systemen.

Neben der Überwachung der korrekten Funktionsweise von Smarten Umgebungen werden Sensoren eingesetzt, um Veränderungen zu erfassen, die durch externe Einflüsse zustande kommen, wie zum Beispiel Benutzerinteraktionen oder veränderte Umweltfaktoren. Basierend auf diesen Daten können und sollen entsprechende Handlungsanweisungen für Menschen oder Maschinen generiert werden. Während der Mensch hierauf eigenständig agieren muss, reagiert die Maschine durch gezielte Ansteuerung der Aktuatoren auf die Anweisung. Durch die Sensoren kann somit der aktuelle Zustand der Umgebung erfasst und mittels der Aktuatoren eine Veränderung dieser initiiert werden. Die Kommunikation bzw. die Ansteuerung der Smarten Komponenten wird dabei aufgrund heterogener Schnittstellen und unterschiedlicher Interaktionsmedien erschwert.

Betrachtet man exemplarisch Systeme der Unterhaltungselekronik in Privathaushalten, wie Fernseher, Receiver und Stereoanlagen, ist festzustellen, dass für jedes einzelne dieser Geräte in der Regel eine separate Kommunikationsschnittstelle in Form einer individuellen Fernbedienung vorliegt. Aufgrund der stetig ansteigenden Anzahl solcher Komponenten in einem Haushalt gibt es heutzutage bereits kommerzielle Systeme, die eine verbesserte Kontrolle und einfachere Bedienung ermöglichen, indem sie eine zentrale Steuerkomponente für mehrere Geräte zur Verfügung stellen[1]. Diese multifunktionalen Fernbedienungen oder auch mobilen Applikationen für Smartphones und Tablets sind auf das Einsatzgebiet der Steuerung von Unterhaltungselektronik angepasst, so dass für eine Ansteuerung weiterer Komponenten, wie beispielsweise elektrische Rollläden, ein zusätzliches Interaktionsmedium verwendet werden muss. Zudem beinhalten sie keine Darstellung von Sensorinformationen, die es ermöglicht,Sensordaten, die außerhalb des Einsatzgebiets erfasst werden, darzustellen.

1.1.2 Virtualisierung und Visualisierung

1 Damit Sensorwerte mit Informationen aus dem Cyberspace kombiniert werden können, müssen physische Objekte *virtualisiert* werden. Damit ist die Erzeugung von virtuellen Objekten zu verstehen, die Abbilder der physischen Objekte darstellen. Virtualisierungen werden beispielsweise hergenommen, um Experimentierumgebungen darzustellen, in denen möglichst realistisch verschiedene Ereignisse simuliert und getestet werden können [Anderson et al., 2005]. Für die Erstellung von virtuellen Repäsentationen physischer Objekte können diese entweder in speziellen Modellierungsumgebungen, wie beispielsweise

[1] http://www.logitech.com/de-de/harmony-remotes, Zugriff November 2014

Cinema 4D[2], oder mittels Rekonstruktionsverfahren aus Bildern und Videos erstellt werden. Beispielsweise können aus Videosequenzen verschiedener Perspektiven der gleichen Szene Objekte und deren Bewegungsmuster erfasst, rekonstruiert und damit virtualisiert werden [Rander et al., 1997; Kanade et al., 1997]. Die physischen Objekte weisen dabei eine Verknüpfung mit ihren virtuellen Gegenstücken auf, indem möglichst alle Eigenschaften (z.B. Form und Größe) mit virtualisiert werden, so dass basierend auf dem virtualisierten Modell Rückschlüsse auf das physische Objekt hergestellt werden können. Die Virtualisierung aller in einer Umgebung befindlichen Objekte kann dazu verwendet werden, diese in Form eines 3D-Modells zu visualisieren, was ebenfalls ein Fokus im Bereich der CPE ist. Dabei ist eine einfache und verständliche Repräsentation wichtig, um ein Mapping zwischen Aktion und Reaktion zu ermöglichen.

Für die Darstellung von Sensordaten müssen geeignete und leicht verständliche Darstellungsformen gewählt werden, die die Masse an Daten entsprechend aufbereitet repräsentieren, um Managern von CPE eine Hilfestellung bei der Entscheidungsfindung zu bieten und diese nicht mit einer Vielfalt an Informationen zu überfordern. Norman ging bereits 2002 auf die Thematik der adäquaten Repräsentation von physischen Sensoren und deren Auswirkung auf die Aktuatoren ein. Am Beispiel von mehreren Lichtschaltern zur Steuerung von Lampen präsentierte er eine Methode zur adäquaten Repräsentation physischer Sensoren und deren Auswirkung auf Aktuatoren [Norman, 2002, Seite 95ff.]. Aufgrund der normierten Verkleidungen von Lichtschaltern werden diese in der Regel nebeneinander oder untereinander an der Wand angebracht, wie beispielsweise in Abbildung 1.3 (links) dargestellt ist. Für den Abgleich, welcher Schalter welches Licht steuert muss der Benutzer eine mentale Rotation erstellen, um die Deckenlichter mit den Wandschaltern assoziieren zu können. Aber selbst dies ist nicht immer eindeutig, da die zugehörigen Lampen beliebig im Raum verteilt sein können. Als Lösung für diese Problematik wurde prototypisch eine Verkleidung für die Schalter erstellt, auf die die Geometrie des Raums aufgezeichnet wurde. Die Lichtschalter wurden dabei an den Stellen angebracht, an denen sich die Lampen im Raum befinden, wodurch die Zuordnung zwischen Lichtschalter und Lampe vereinfacht wurde. Abbildung 1.3 (rechts) zeigt ein solches Lichtschalterlayout, welches an der Wand angebracht ist. Gemäß der Position der Lampen im Raum befinden sich die entsprechenden Lichtschalter an den Positionen auf dem abgebildeten Raummodell.

Im Umfeld von Fabriksystemen und zur Verkehrsflussüberwachung haben sich Konzepte und Umsetzungen zur Überwachung und Steuerung in Form von sogenannten Leitständen etabliert, wie beispielhaft in Abbildung 1.4 dargestellt. Angepasst auf das jeweilige Anwendungsfeld werden in solchen Leitständen Daten und Informationen mittels mehrerer Bildschirme dargestellt und aufbereitete Sensorinformationen inklusive potentieller Störungen visuell angezeigt, um Bedienern - den Operatoren - die Möglichkeit zu bieten, Abweichungen von vorgegebenen Prozessen schnell zu erfassen. Gleichzeitig kann eine sinnvolle Visualisierung als Entscheidungsunterstützung dienen, auf deren Basis die Operatoren aktiv in die zu kontrollierende Umgebung eingreifen können. Durch die Verwendung von domänen- und anwendungsspezifischen Softwarelösungen ist ein Einsatz solcher

[2]http://www.maxon.net/de/products/cinema-4d-studio.html, Zugriff November 14

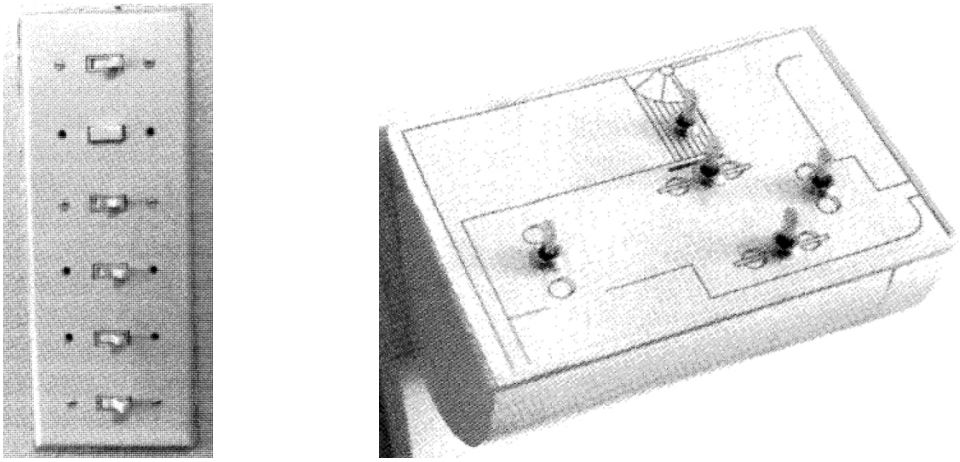


***Abbildung 1.3**: Lichtschalterlayout im klassischen Stil untereinander (links) und in Abhängigkeit vom Raummodell (rechts) (Quelle [Norman, 2002])*

Spezialsysteme in anderen Anwendungsfällen fast ausgeschlossen. Die enge Verzahnung solcher Leitstände mit den physischen Sensoren und Aktuatoren sowie die angepasste virtuelle Repräsentation der gesammelten Informationen erschwert zusätzlich den Transfer in andere Anwendungsszenarien.

***Abbildung 1.4**: Leitstand in einem Kraftwerk*[3]

Gerade im Zeitalter von Online-Welten und medialer Virtualität, wie sie beispielsweise in SecondLife[4] realisiert wird, können physische Objekte dreidimensional visualisiert werden und bieten daher eine geeignete Variante der Repräsentation von ubiquitären Umgebungen. Dabei besteht zudem die Möglichkeit, Personen trotz räumlicher Distanz mittels modellierter Welten miteinander zu verbinden, indem sie sich als virtuelle Charaktere in der künstlich

[3] Quelle: http://de.wikipedia.org/wiki/Leitstand, Zugriff: November 2014
[4] http://www.secondlife.com, Zugriff November 2014

geschaffenen Umgebung begegnen. Heutzutage werden diese Potentiale der digitalen Visualisierung und Virtualisierung vorwiegend in der Spielindustrie aufgegriffen. In Zukunft können existierende Konzepte dahingehend erweitert werden, dass die Virtualisierung von ubiquitären Umgebungen in Kombination mit Sensoren ebenfalls dazu verwendet werden kann, diese zu monitoren. Gleichzeitig können Benutzerinteraktionen mit der virtualisierten Repräsentation dazu verwendet werden, um Aktuatoren in der physischen Umgebung zu steuern und diese damit zu verändern.

Erste Ansätze für die dreidimensionale Repräsentation von physischen Räumen sind unter anderem in der Planungsphase zu finden, wie beispielsweise bei Architektur- und Einrichtungsprogrammen. Zum Beispiel existieren kommerzielle Applikationen, die eine dreidimensionale Planung von Umgebungen inklusive deren Ausstattung ermöglichen. Exemplarisch hierfür ist das Bauwesen zu nennen, bei dem vor dem Bau eines Gebäudes die Infrastruktur und Räume virtuell modelliert werden. Im speziellen Fall eines Supermarkts werden in diesem Modell neben den baulichen Gegebenheiten, die auf Basis eines Grundrissplans erfasst werden, auch die Regalierung bis hin zu der Bestückung aller Regale mit Produkten modelliert. Diese Applikationen werden als *Floor-* und *Spaceplanner* bezeichnet. Abbildung 1.5(a) zeigt eine solche virtuelle Verräumung im Spaceplanner *Spaceman*[5] der Firma Nielsen. Ein weiteres Beispiel für 3D-Raummodellierungstools sind Applikationen, die es auch Laien ermöglichen, sich ein Eigenheim virtuell zusammenzustellen, wie in Abbildung 1.5(b) zu sehen ist. Weiterhin existieren kostenlose Applikationen für die Modellierung von Räumen. Diese werden meist als Entscheidungsunterstützung für den Anwender entwickelt, wobei die Applikation selbst nicht im Vordergrund steht, sondern Objekte, die in eine virtuelle Repräsentation der eigenen Umgebung eingebettet werden können. Ein prominentes Beispiel hierfür ist der Küchenplaner des Einrichtungsunternehmens Ikea, der es ermöglicht, angebotene Möbelstücke in einem Umriss der Küche zu platzieren (siehe Abbildung 1.5(c)). Neben der 3D-Darstellung umfasst diese Applikation zusätzliche Funktionalitäten, um den Kunden bei der Platzierung zu unterstützen, wie z.B. Warnhinweise, wenn Elektrogeräte direkt neben Spülarmaturen positioniert werden und somit ein Sicherheitsrisiko darstellen. Während diese Modellierungssysteme bei der Planungsphase von Umgebungen eingesetzt werden, besteht aktuell noch keine Möglichkeit, diese auch während des Betriebs einzusetzen, um Benutzerinteraktionen zu repräsentieren und Sensorinformationen darzustellen.

1.1.3 Dual Reality Management Dashboard

Kombiniert man ein Netzwerk von Sensoren, Aktuatoren und Diensten mit einer Virtualisierungskomponente, so entsteht ein System, welches als Monitoring-Komponente verwendet werden kann, indem der aktuelle Zustand einer CPE virtuell repräsentiert und grafisch dargestellt wird. Parallel zur Überwachungsfunktionalität kann eine solche Virtualisierung basierend auf Interaktionen mit der virtuellen Umgebung auch zur Steuerungs- und Kontrollstation erweitert werden. Die gegenseitige Beeinflussbarkeit zwischen Objekten der realen

[5] http://spaceman-professional.software.informer.com, Zugriff: November 2014

(a) Virtuelle Regalbestückung in Spaceman

(b) 3D-Raumplanung für Heimbereiche[6]

(c) Bildschirmausschnitt aus dem Ikea Küchenplaner

***Abbildung 1.5**: 3D-Planungskomponenten für Raumausstattung*

Welt und einer Virtualisierung ist unter dem Ausdruck *Dual Reality* bekannt, welcher von Lifton folgendermaßen definiert wurde:

Definition 1.2 (Dual Reality) *Eine Verschmelzung resultierend aus dem Zusammenspiel zwischen der realen Welt und der virtuellen Welt, verbunden durch Netzwerke von Sensoren und Aktuatoren. Während beide Welten jeweils für sich alleine bereits komplett sind, werden sie erweitert durch die Fähigkeit sich gegenseitig widerzuspiegeln, zu beeinflussen und ineinander überzugehen [Lifton, 2007, Seite 16].*

Kombiniert man die Ansätze des Monitoring und der Steuerung einer physischen Umgebung basierend auf ihrer grafischen Repräsentation, erhält man ein Tool, welches Echtzeitinformationen aus CPE grafisch aufbereitet repräsentieren kann. Gleichzeitig bietet die Verzahnung der physischen Umgebung und ihrer virtuellen Repräsentation dem Manager einer solchen Umgebung die Möglichkeit, direkten Einfluss auf diese auszuüben (siehe Abbildung 1.6). Ein solches System wurde im Rahmen dieser Arbeit konzipiert und entwickelt, das *Dual Reality Management Dashboard (*DURMAD*)*.

Durch die Verbindung von realen Informationen, die mittels physischer Sensoren erfasst werden, und deren Integration in eine Virtualisierung der Umgebung in einem 3D-Modell, können die Vorteile beider Welten miteinander kombiniert werden. Beispielsweise können natürliche Benutzerinteraktionen in der CPE als Eingaben im virtuellen Modell gesehen werden, während geometrische Berechnungen, wie beispielsweise die Ermittlung der Abstände zwischen zwei Objekten, im Virtuellen einfach durchgeführt werden können. Neben der reinen Darstellung können, basierend auf diesen Berechnungen oder Benutzerinteraktionen eines Managers mit dem 3D-Modell, gleichzeitig Handlungsanweisungen in der physischen Umgebung angestoßen werden, wie zum Beispiel die Aufforderung zum Öffnen eines Tors. Dies setzt eine bi-direktionale Beeinflussbarkeit voraus. Um eine geeignete Entscheidungsunterstützung für solche Handlungsanweisungen zu bieten, können im Virtuellen Simulationskomponenten integriert werden, die basierend auf den erfassten Daten Vorhersagen gemäß spezifizierter Parameter treffen. Des Weiteren können mögliche eintreffende Ereignisse bereits im Vorfeld mittels solcher Simulationen getestet werden,

[6]Quelle: http://de.floorplanner.com, Zugriff: November 2014

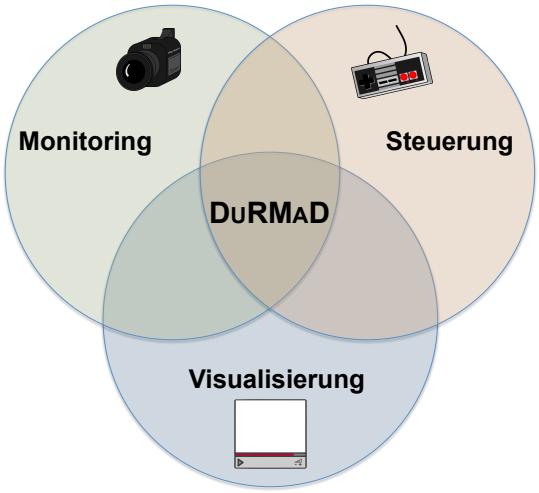


***Abbildung 1.6**: DuRMaD als Verbindung von Monitoring, Steuerung und Visualisierung*

um bereits pro-aktiv entsprechende Verbesserungsmaßnahmen in der realen Welt einleiten zu können. Beispielsweise kann ein möglicher Leerstand an einer Produktionsmaschine frühzeitig erkannt und automatisch entsprechend neues Material geordert werden.

1.2 Forschungsfragen

Im Rahmen der vorliegenden Dissertation werden mehrere Fragestellungen untersucht. Die wichtigsten wissenschaftlichen Fragen, die innerhalb der Dissertation beantwortet werden, sind folgende:

1. **Wie können Cyber-Physische Umgebungen semantisch repräsentiert und virtualisiert werden?**

   Cyber-Physische Umgebungen (CPE) sind komplexe Konstrukte aus Sensoren, Aktuatoren und Diensten, die durch entsprechende Netzwerke und Kommunikationsschnittstellen miteinander verbunden sind. Durch diese Komplexität ist eine Repräsentation einer solchen Umgebung relativ schwierig zu bewerkstelligen. Besonders bei agilen, sich ständig verändernden Umgebungen sind Änderungen nur schwer darzustellen. Mit Hilfe von Methoden semantischer Annotation wird eine Virtualisierung und darauf aufbauend eine interaktive Visualisierungsarchitektur realisiert.

2. **Wie können komplexe Cyber-Physische Umgebungen überwacht werden?**

   Durch die Integration von Sensorik in CPE können Veränderungen kontinuierlich erfasst werden. Diese Daten werden in der Regel primär dazu eingesetzt, um möglichst automatisiert auf Veränderungen reagieren zu können. Zudem können diese Daten miteinander kombiniert werden, um damit eine Überwachung der Umgebung zu realisieren und somit unübliche Vorkommnisse frühzeitig erkennen zu können. Neben der

Erfassung und Verarbeitung von Sensordaten ist eine Überwachung der ausführenden Dienste und weiterer Hardwarekomponenten sinnvoll. Dadurch kann eine korrekte Arbeitsweise der Umgebung sichergestellt werden. Im Rahmen dieser Arbeit wird ein Konzept sowie ein Softwarewerkzeug vorgestellt, welches Managern von CPE eine einfache Möglichkeit des Monitoring ermöglicht.

3. **Wie können Aktuatoren in Cyber-Physischen Umgebungen zentral koordiniert und angesteuert werden?**

   Neben dem Monitoring von CPE können diese auch gesteuert und verändert werden. Eine zentrale Ansteuerung von allen in der Umgebung befindlichen Komponenten, insbesondere der Aktuatoren, stellt dabei eine einfache Variante der Einflussnahme auf die Umgebung dar. Gerade die Möglichkeit der Koordination aus einem Programm heraus würde verantwortliche Personen bei schnell durchzuführenden Einflussnahmen unterstützen. In dieser Arbeit wird eine Methode präsentiert, welche durch Verknüpfung von Virtualisierungskomponenten und physischen Aktuatoren diese Fragestellung beantwortet.

4. **Wie können Visualisierungen und Simulationen auf Grundlage sich stetig verändernder Umgebungsdaten durchgeführt werden?**

   In der Arbeit wird ein Konzept vorgestellt, welches eine „diskrete“ Integration von Simulationen in bestehende Infrastrukturen ermöglicht, ohne dabei die Funktionsweise nachteilig zu beeinflussen. Dies bedeutet unter anderem, dass nach Beendigung der Simulation die Dienste ihren normalen Betrieb wiederaufnehmen, ohne Beeinträchtigung durch die zuvor durchgeführte Simulation. Die Simulationen können in diesem Fall sowohl auf die physische Umgebung als auch auf die Virtualisierung oder auf beide eine Auswirkung haben. Es wird erläutert, wie diese Integration es ermöglicht, eine Testumgebung zu schaffen, um bereits im Vorfeld neue Systeme testen zu können. Dadurch wird eine Testumgebung für komplexe Simulationen unter realitätsnahen Bedingungen geschaffen.

5. **Wie können heterogene Sensorinfrastrukturen und Datenströme visualisiert und Ereignisse protokolliert werden?**

   Gerade die Kommunikation zwischen verteilten Komponenten in CPE kann nur schwer nachverfolgt werden. Vor allem bei gewachsenen Umgebungen, zu denen kein protokollierter Informationsverlauf existiert. Im Rahmen dieser Arbeit werden Methoden vorgestellt, die eine Darstellung des aktuellen Datentransfers und eine Protokollierung der erfassten Veränderungen zwecks Analysen ermöglichen.

6. **Wie können Austausch- und Erweiterbarkeit von Sensoren, Aktuatoren und Diensten in Cyber-Physischen Umgebungen sichergestellt werden?**

   CPE sind meistens sehr heterogen und werden mit der Zeit erweitert. Dies kann durch neue Sensoren, Aktuatoren oder auch durch Anpassen der Dienste erfolgen. Richtlinien zum Aufbau von solchen Umgebungen im Zusammenspiel mit einer spezialisierten Kommunikationsinfrastruktur werden in dieser Arbeit vorgestellt. Die daraus

resultierende Infrastruktur ermöglicht eine einfache Erweiterung der Umgebung, ohne die alten Strukturen und Verbindungen zu beeinträchtigen.

7. **Wie können auftretende Fehlfunktionen in einer Cyber-Physischen Umgebung erkannt, protokolliert und zwecks Ursachenbehebung signalisiert werden?**

   Aus der Anwendungsentwicklung sind Fehleranalysen durch sogenanntes „Debugging“ bereits eine gängige Methode. Bei verteilten Systemen hingegen ist eine solche Analyse nur mit großem Aufwand zu bewerkstelligen. Gerade durch den enormen Datenfluss zwischen den Systemen wird das Nachvollziehen von Fehlfunktionen sehr komplex, da diese über mehrere Systeme propagiert werden können. Eine Untersuchung inwieweit eine Fehlerüberwachung und Notifikation auf Basis der Verbindung zwischen physischer Umgebung und deren Virtualisierung realisiert werden kann, liefert eine Antwort auf diese Fragestellung.

1.3 Aufbau der Promotionsarbeit

Die folgenden Kapitel beschreiben Grundlagen, Konzeptionen und Umsetzungen von Cyber-Physischen Umgebungen (CPE) mit besonderem Fokus auf die grafische Darstellung der Virtualisierung einer CPE und der Kommunikation zwischen ihren Komponenten. Dies wird untermauert mit einer Darstellung existierender Systeme inklusive einer detaillierten Abgrenzung zu den Konzepten, die im Rahmen dieser Arbeit entwickelt und prototypisch umgesetzt wurden. Die einzelnen Kapitel erläutern die spezifischen Themenaspekte der Arbeit detailliert und schließen jeweils mit einer Kurzzusammenfassung, in der die wichtigen Punkte nochmals kurz aufgeführt werden.

In Kapitel 2 wird auf Grundlagen eingegangen, die zum Verständnis und als Hintergrundwissen für die vorliegende Arbeit dienen. Darunter fallen Informationen zu instrumentierten Umgebungen, Arten von Datenermittlung, Ansätzen von Kommunikationsinfrastrukturen, Visualisierungs- und Virtualisierungsmethoden und bereits existierende Überwachungs- und Kontrollsysteme.

Im anschließenden Kapitel 3 werden verwandte Arbeiten detailliert beschrieben, die sowohl die Konzeption als auch die Umsetzung des im Rahmen dieser Arbeit entwickelten Frameworks beeinflusst haben. Darauf folgt ein umfassender Vergleich dieser Arbeiten mit dem Konzept, welches im Rahmen dieser Arbeit entwickelt wurde.

In Kapitel 4 werden Virtualisierungen und Visualisierungen von CPE analysiert. Dabei wird auf die Besonderheiten von CPE und die Unterschiede von aktuell existierenden Visualisierungskomponenten zu der in dieser Arbeit entwickelten Darstellungskomponente eingegangen. Des Weiteren wird in diesem Abschnitt die Definition von Dual Reality erstmalig formalisiert.

Kapitel 5 stellt die Konzeption und Umsetzung eines Tools zur interaktiven dreidimensionalen Visualisierung von CPE dar, das *Dual Reality Management Dashboard (*DuRMaD*)*.

Es wird präsentiert, wie dieses Tool zur Darstellung der Virtualisierung einer CPE verwendet werden kann und dabei die Möglichkeit bietet, Veränderungen in der physischen Umgebung automatisch im Virtuellen widerzuspiegeln. Gleichzeitig werden Mehrwertfunktionalitäten durch die in DURMAD implementierten Module und die Verknüpfung der realen Umgebung mit dem virtuellen Modell aufgezeigt.

In Kapitel 6 wird eine modulare Kommunikationsinfrastruktur präsentiert, die im Rahmen dieser Arbeit entwickelt wurde, der *Event Broadcasting Service (EBS)*. Insbesondere werden Filter- und Vorverarbeitungsmodule vorgestellt, welche in die Server- und in die Client-Struktur eingebunden werden können, um bereits in der Kommunikationsschnittstelle Vorverarbeitung realisieren zu können. Zudem wird präsentiert, wie DURMAD und EBS zusammenarbeiten und damit das *Dual Reality Framework* bilden.

Kapitel 7 zeigt die Vorteile des DR-Frameworks am Beispiel der Handelsdomäne auf. Durch die Erläuterung der Integration ins Innovative Retail Laboratory (IRL) werden die Potenziale präsentiert und spezifische Erweiterungen für die Handelsdomäne beschrieben. Hierbei werden die unterschiedlichen Sensor- und Aktuatortypen sowie die Dienste im IRL vorgestellt und erläutert, wie das DR-Framework darin eingebunden wurde.

Die in dieser Arbeit erstellten Konzepte und Entwicklungen werden in Kapitel 8 zusammengefasst. Gleichzeitig werden die im Rahmen dieser Arbeit geleisteten wissenschaftlichen Beiträge noch einmal explizit aufgeführt und ein Ausblick auf potentiell aufbauende Arbeiten gegeben.

2 Theoretische Grundlagen und Basiskonzepte

In diesem Kapitel werden Grundlagen vorgestellt, die dem Verständnis der weiteren Arbeit und der Eingruppierung der Ergebnisse in aktuelle Entwicklungen und Forschungsthemen dienen. Zuerst werden Konzepte der Datenverarbeitung und Interaktionsformen beschrieben. Anschließend werden Merkmale und Eigenschaften von Smarten Umgebungen bis hin zu Cyber-Physischen Umgebungen dargestellt. Im dritten Abschnitt wird speziell auf Kommunikationsinfrastrukturen und -architekturen eingegangen. Darauf folgend werden Visualisierungs- und Virtualisierungskonzepte aufgeführt. Abschließend wird auf Grundlagen der Darstellung von Sensorinformationen und die Möglichkeit auf diese zu reagieren eingegangen.

2.1 Datenakquise und -verarbeitung

Um das Potenzial von Sensor-Aktuator-Netzwerken auszunutzen, müssen die von den Sensoren erfassten Daten von Diensten aufbereitet und die Ergebnisse anschließend bereitgestellt werden. Dies kann durch eine Verarbeitungskette, wie sie in Abbildung 2.1 dargestellt ist, beschrieben werden. Für die Erfassung von Sensordaten spielt die Methode der Datenübertragung, die *Modalität* (siehe Abschnitt 2.1.1), eine wichtige Rolle, da Sensoren nur gewisse Methoden unterstützen. Beispielsweise können Mikrofone keine visuellen Daten erfassen. In Smarten Umgebungen müssen die durch Sensoren erfassten Rohdaten vor der Übertragung vorverarbeitet werden. Optional können bei der Vorverarbeitung die Daten gleichzeitig gefiltert werden, um nur relevante Daten weiter zu verteilen. Auf der Empfängerseite müssen die ankommenden Daten erneut aufbereitet werden. Beispielsweise müssen neben den Sensordaten zusätzliche Informationen aus externen Wissensquellen akquiriert werden, um eine Analyse durchführen zu können. Dies erfordert eine eindeutige Zuordnung von Objekten zu den entsprechenden Informationen, welches eine Zielstellung im Bereich des *Internet der Dinge* (siehe Abschnitt 2.1.2) ist. Eine spezialisierte Informationsquelle stellen hierbei die *digitalen Objektgedächtnisse* (siehe Abschnitt 2.1.3) dar, die alle objektspezifischen Informationen umfassen. Die gewonnenen Informationen stehen den Applikationen zur Verwendung bereit. Im einfachsten Fall werden die Rohdaten so aufbereitet, dass sie einem Endnutzer grafisch dargestellt werden können. Daneben können die Sensordaten auch hergenommen und verarbeitet werden, um weiterführende Dienstleistungen darauf aufzusetzen, wie beispielswei-

se eine Unterstützung von Personen, welche auf Basis der Resultate Entscheidungen treffen müssen. In diesem Fall muss der aktuelle Zustand der Umgebung erfasst und verarbeitet werden. Informationen zum aktuellen Zeitpunkt, zur Umgebung und zum Benutzer werden unter dem Begriff *Kontext* (siehe Abschnitt 2.1.4) zusammengefasst. Betrachtet man die Sensor-Aktuator-Netzwerke in Verbindung mit intelligenten Diensten, können Smarte Umgebungen als intelligente *Agentensysteme* (siehe Abschnitt 2.1.5) angesehen werden, die basierend auf erfassten Sensorinformationen automatisiert reagieren [Cook und Das, 2007].

***Abbildung 2.1**: Verarbeitungskette von Daten in Smarten Umgebungen*

2.1.1 Modalität

Bei der Kommunikation zwischen Menschen untereinander oder in der Mensch-Computer-Interaktion ist die geeignete Wahl und das Verständnis des Mediums und die Übertragungsart, die *Modalität*, wichtig. Einerseits können nicht alle Modalitäten von jedem System verstanden werden. Andererseits können Informationen in bestimmten Modalitäten nur schwer kommuniziert bzw. erfasst werden.

Definition 2.1 (Modalität) *Eine Modalität ist eine Methode oder ein Verfahren, um Informationen zwischen Menschen oder zwischen Menschen und Maschinen über irgendein Medium auszutauschen. Ein Medium ist dabei die physikalische Realisierung irgendeiner Informationspräsentation an der Schnittstelle zwischen Menschen bzw. Mensch und Maschine [Bernsen, 2002].*

Modalitäten sind zum Beispiel Text, Grafiken, Gesten oder Sprache. Eine elementare Modalität wird als *Unimodalität* bezeichnet, wie dies in der Taxonomie von Bernsen erläutert wird [Bernsen, 2002]. Viele Systeme arbeiten mit Kombinationen einzelner Unimodalitäten. Dies wird durch den Begriff *multimodal* ausgedrückt, der für die Kombination verschiedener Unimodalitäten steht. Dabei gibt es Unimodalitäten, die voneinander unabhängig sind und welche, die voneinander abhängen. Beispielsweise sind Text und Sprache unabhängig von anderen Modalitäten, da diese in der Regel ohne weitere Modalitäten verstanden werden. Grafiken hingegen benötigen daneben eine weitere Modalität, um verstanden zu werden, wie beispielsweise Text als Beschriftung. Ein virtueller Charakter vereint verschiedene Modalitäten, wie z.B. Sprache, Gestik und Mimik, und kann somit als multimodale Mensch-Maschine-Schnittstelle angesehen werden. Bei einer multimodalen Ausgabe ist oftmals die Synchronisation der einzelnen Modalitäten untereinander wichtig. Beispielsweise muss die Mundbewegung eines virtuellen Charakters mit der Sprachausgabe synchronisiert werden, damit eine natürliche und glaubwürdige Ausgabe zustande kommt.

Ein weiteres Beispiel für multimodale Ein- und Ausgabesysteme sind Smartphones, die mehrere Modalitäten verarbeiten können. Diese sind in der Lage zu vibrieren, Text

und Bilder auf dem Display auszugeben und über den Lautsprecher akustische Signale wiederzugeben. Neben den Ausgabemodalitäten bieten Smartphones auch multimodale Eingabemöglichkeit. Sie können beispielsweise über Sprache oder durch Gesten gesteuert werden. Gerade bei multimodalen Eingaben können mittels unterschiedlicher Modalitäten über die gleiche Eigenschaft einer Entität Aussagen getroffen werden. Zum Beispiel indem man über die Spracherkennung einen Anruf einleiten lassen will und gleichzeitig im Telefonbuch auf den Kontakt für einen Rufaufbau klickt. Daher müssen die Eingaben in Beziehung zueinander gesetzt, also fusioniert, werden [Wasinger et al., 2005]. Dabei können sich die multimodalen Eingaben entweder verstärken oder in einem Konflikt stehen, der aufgelöst werden muss. In dem zuvor erwähnten Beispiel würde eine Verstärkung eintreffen, wenn der mittels Gesteninteraktion gewählte Kontakt mit der verbalen Eingabe übereinstimmt. Andernfalls würde es zu einem Konflikt kommen, da nicht eindeutig entschieden werden kann, zu welchem der beiden Kontakte das Smartphone einen Rufaufbau einleiten soll. Stehen für Ausgabe die gleichen Modalitäten wie für die Eingaben zur Verfügung und umgekehrt, spricht man von *Symmetrischer Multimodalität* [Wahlster, 2003].

Der Einsatz symmetrischer Multimodalität in einem System ermöglicht es, auf eine intuitive und für den Benutzer natürlichen Art und Weise, Antworten auf Anfragen zu präsentieren. Dabei muss die gewählte Eingabemodalität berücksichtigt werden, um die entsprechende Ausgabemodalität wählen zu können [Wasinger und Wahlster, 2006]. Neben den verwendeten Modalitäten für die Eingabe spielen auch Aspekte der Privatsphäre eine wichtige Rolle bei der geeigneten Ausgabemodalitätenwahl. So wird in privaten Umgebungen (siehe Kapitel 2.2) oftmals Sprache als eine favorisierte Ausgabemodalität gewählt, wohingegen in einer öffentlichen Umgebung eher eine Ausgabe favorisiert wird, die von den umliegenden Personen nicht direkt wahrgenommen wird [Wasinger und Krüger, 2006]. Schließlich können noch weitere äußere Faktoren bei der Wahl der Ausgabemodalitäten berücksichtigt werden, wie beispielsweise die Lautstärke in der Umgebung, um welches Objekt oder um welche Eigenschaft es sich handelt. Beispielsweise sind Informationen zu Medikamenten eher als privat anzusehen, wohingegen Daten zu Digitalkameras keine intimen Informationen des Benutzers preisgeben [Kahl et al., 2008].

2.1.2 Internet der Dinge und Dienste

Für die Verknüpfung von Objekten mit Informationen aus dem Cyberspace wird eine eindeutige Identifikation der physischen Objekte benötigt. Die Grundbausteine hierzu werden im *Internet der Dinge (Internet of Things, IoT)* behandelt. Das Ziel des IoT liegt darin, dass man von überall zu jeder Zeit auf alle Daten aller Objekte zugreifen kann [Infso D.4 Networked Enterprise & RFID et al., 2008]. Dabei steht im Vordergrund, dass die Informationen der Objekte miteinander verknüpft werden können, um smartere Systeme zu erhalten, die darauf aufbauend autonomes und adaptives Verhalten aufweisen können. Das IoT ist in drei Ansätze unterteilt, welche sich orientieren nach: dem *Ding*, dem *Internet* und der *Semantik* [Atzori et al., 2010]. Die Dinge bedürfen hierbei einer eindeutigen Identifizierung mit Hilfe geeigneter Technologien, die entweder mit optischen Verfahren arbeiten (z.B. zweidimensionale Barcodes) oder mittels Funktechnologie (z.B. RFID). Um eine möglichst allgemeingültige Identifizierung zu erreichen, müssen entsprechende

Standards entwickelt werden, die die Adressierungsproblematik thematisieren. Beispiele für eindeutige Identifikationsmechanismen stellen unter anderem URN (Uniform Resource Name) oder IPv6 dar. Anhand einer solchen eindeutigen Identifizierung können sich Objekte gegenseitig referenzieren und untereinander kommunizieren [Gubbi et al., 2013].

In den letzten Jahren sind funkbasierte Erfassungstechnologien, wie beispielsweise Radiofrequenz Identifikation (RFID) oder Nahfeldkommunikation (NFC), in immer mehr Geschäftsbereichen eingeführt worden. Dies ermöglicht die Erfassung der Produkte auch auf eine größere Distanz ohne Sichtkontakt. Des Weiteren bieten solche Erfassungssysteme die Möglichkeit, jedes Objekt mit einem eindeutigen Identifikationsschlüssel auszustatten und zu referenzieren. Diese Eindeutigkeit muss gewährleistet werden, um das Objekt eineindeutig detektieren zu können. In Bezug auf das Internet beschäftigt sich das IoT mit den Übertragungs- und Zugriffsmechanismen durch die Entwicklung geeigneter Middleware-Architekturen. Durch die allgegenwärtige Verfügbarkeit der Daten spielen Mechanismen zum Schutz von Sicherheit und Privatsphäre eine wichtige Rolle. Darüber hinaus müssen die ermittelten Informationen miteinander verknüpft werden, weshalb die Semantik ebenfalls ein Hauptthemengebiet im Bereich des IoT ist. Ein Schwerpunkt beim IoT sind Standardisierungen, durch die es erst ermöglicht wird, die Eindeutigkeit und den allgemeinen Zugriff auf Daten zu gewährleisten. Neben der Zugänglichkeit zu Daten von Objekten gibt es auch die Bestrebung, Verarbeitungslogiken und Dienste zur Verfügung zu stellen. Aus diesem Grund gibt es neben dem IoT die Bestrebung des Internets der Dienste. Dabei stehen die gleichen Ziele im Vordergrund, dass die Dienste zu jeder Zeit von überall erreichbar sind und mittels Semantik - idealerweise automatisch - ihre Funktionsweise erfasst werden kann.

2.1.3 Digitale Objektgedächtnisse

Mit Hilfe einer eindeutigen Identifikation, wie sie mit dem Internet der Dinge angestrebt ist, können alle Informationen, die zu einem Objekt während seines Lebenszyklus erfasst werden, diesem zugeordnet und digital persistent gespeichert werden. Hierdurch entstehen sogenannte *digitale Objektgedächtnisse (Digital Object Memory, DOMe)*, die in Form eines Tagebuchs alle auftretenden Informationen zu einem Objekt speichern [Kröner et al., 2013]. Durch diese Zugriffsmöglichkeit wird gleichzeitig eine Transparenz geschaffen, die es ermöglicht, alle objektspezifischen Informationen zu jeder Zeit einzusehen, insofern ein Zugriff auf das digitale Objektgedächtnis gewährt ist. Dadurch entsteht eine Verknüpfung der physischen Instanz mit virtuellen Informationen [Kim et al., 2009].

Beispielsweise bietet das sogenannte *Object Memory Model (OMM)* eine Möglichkeit, solche digitalen Objektgedächtnisse anzulegen. Dieses wurde im Rahmen einer W3C[1] Incubator Group (XG) entwickelt [Kröner et al., 2011]. Basierend auf dem *Object Memory Format* können objektbezogene Parameter und Informationen in strukturierter Form in einem Speicher abgelegt werden. Das Object Memory Format ist dabei so aufgebaut, dass ein OMM mit einem Header beginnt, gefolgt von einem Inhaltsverzeichnis und den jeweiligen Informationsblöcken, wie in Abbildung 2.2 schematisch dargestellt. Im Header

[1] http://www.w3c.org, Zugriff: November 2014

stehen OMM spezifische Informationen, wie beispielsweise die Nummer der verwendeten Version. Das Inhaltsverzeichnis ermöglicht eine Übersicht über die Informationsblöcke, die im OMM enthalten sind und erlaubt durch eine Indizierung einen schnellen Zugriff auf die jeweiligen Informationsblöcke. Diese wiederum beginnen ebenfalls mit einem strukturiertem Block, welcher nähere Informationen, wie beispielsweise Ersteller und Format, enthält. Anschließend folgt der Nutzdatenblock, in dem die eigentlichen Daten hinterlegt werden.

***Abbildung 2.2**: Struktur eines Objektgedächtnisses basierend auf dem OMM-Format (Quelle [Haupert, 2013])*

Bei der objektbezogenen Speicherung der Daten können zwei Varianten voneinander unterschieden werden. Bei der ersten Variante werden die Informationen in einem Speicher hinterlegt, welcher sich direkt am jeweiligen Objekt befindet. Bei der zweiten Variante wird ein externes Speichermedium verwendet, in dem die Informationen hinterlegt werden. Bei der Speicherung am Objekt sind die Informationen zu jeder Zeit verfügbar und erreichbar, sofern man im Besitz des Objekts ist. Zudem bietet diese Methode einen Vorteil bezüglich der Sicherheit der Daten, da ein Zugriff auf diese nur mit Besitz des Objekts erfolgen kann. Ein Nachteil dieser Variante ist jedoch, dass ein Speicher an einem Objekt in der Regel stark größenbegrenzt ist. Zudem können zusätzliche externe Dienste nicht ohne Weiteres auf die Informationen des Objekts gemäß des IoT-Gedankens zugreifen. Werden die Daten in einem externen Speicher gehalten, so muss eine eindeutige Referenzierung vom Objekt zum Speicherort bestehen. Zudem muss dieser Speicherort von überall erreichbar sein, von wo die Informationen benötigt werden. Hinsichtlich Sicherheit müssen des Weiteren geeignete kryptografische Verfahren angewendet werden, da die Daten theoretisch von überall ausgelesen und eventuell verändert werden können. Bei beiden Varianten dient das Objekt als Schlüssel zu den entsprechenden Daten. Neben dem realen Produkt kann auch dessen Virtualisierung als ein solcher Schlüssel fungieren, wenn die entsprechenden Parameter bei der Virtualisierung mit berücksichtigt werden.

2.1.4 Kontext

Für Systeme, die auf die aktuellen Gegebenheiten angepasst sein sollen, beispielsweise um auf den Benutzer zugeschnittene Informationen aufzubereiten und darzustellen, ist der aktuelle *Kontext* wichtig. Der Kontext wird anhand der erfassten Sensorinformationen ermittelt, die den aktuellen Status sowie Veränderungen in der Umgebung erfassen. Der Begriff Kontext wurde erstmals von Schilit et al. in Bezug auf kontextbasierte Software aufgegriffen [Schilit et al., 1994]. Diese adaptiert sich an die Position, in der sie ausgeführt wird, an Personen, die sich in der Nähe befinden, an den Rechner, auf dem sie aufgespielt ist und an weitere verfügbare Geräte. Ebenso passt sie sich den zeitlichen Bedingungen an. Aus dieser

Aufzählung erfolgte eine formale Definition des Begriffs Kontext durch Dey wie folgt:

Definition 2.2 (Kontext) *Kontext sind Informationen, die zur Charakterisierung einer Situation einer Entität verwendet werden können. Eine Entität ist eine Person, ein Ort oder ein Objekt, welche als relevant für die Interaktion zwischen einem Benutzer und einer Applikation zu betrachten ist, inklusive des Benutzers und der Applikation [Dey, 2001].*

Bei der Einflussnahme mit Kontext kann zwischen aktiver und passiver Verwendung unterschieden werden [Chen et al., 2000]. Bei der aktiven passt sich die Anwendung automatisch dem erfassten Kontext an, indem das Applikationsverhalten verändert wird. Im passiven Fall stellt die Applikation den neuen oder veränderten Kontext einem interessierten Benutzer dar oder hinterlegt ihn, damit er später durch einen Benutzer abgerufen werden kann, ohne dass er explizit in der Anwendung verwendet wird. Eine Umsetzung von passiver Kontextverarbeitung erfolgte beispielsweise in Form sogenannter Kontext-Widgets, bei denen Kontextveränderungen analog zu Benutzereingaben verarbeitet werden [Dey et al., 2000]. Kontextbasierte Applikationen unterstützen drei Merkmale:

- **Präsentation** von Informationen und Diensten für einen Benutzer.
- Automatische **Ausführung** eines Dienstes für einen Benutzer.
- **Kennzeichnung** von Kontextinformationen zur späteren Identifikation.

Kontext kann man in fünf Kategorien unterteilen: *Individualität*, *Zeit*, *Lokation*, *Aktivität* und *Relation* [Zimmermann et al., 2007]. Dabei umfasst der Individualität-Kontext alle Informationen zur Entität, von der der Kontext abstammt. Er beschreibt somit alles, was über die Entität erfasst werden kann und damit in der Regel deren Status. Der Zeit-Kontext enthält alle Informationen in Bezug auf die Zeit, inklusive der Zeitzone sowie aktueller bzw. virtueller Zeit. Der Aktivität-Kontext beschreibt die Aktion, in welcher sich die Entität aktuell befindet, um Aussagen treffen zu können, was die Entität erreichen möchte und wie ihr Vorgehen diesbezüglich ist. Der Relation-Kontext subsumiert alle Daten über die Verbindung einer Entität zu weiteren Entitäten und kann dazu verwendet werden, um Informationen über Gruppen von Entitäten weiterzupropagieren. Der Lokation-Kontext umfasst alle Informationen zur Position, in der sich die Entität befindet. Besonders diese Art von Kontext wird in vielen Diensten zur Benutzerunterstützung eingesetzt. Solche Dienste sind beispielsweise Systeme, die abhängig von der aktuellen Position des Benutzers unterschiedliche Informationen liefern. Diese Systeme sind als *lokationsbasierte Informationssysteme [Location-Based Services]* bekannt. Eine Definition von lokationsbasierten Informationssystemen wurde von Virrantaus et al. folgendermaßen gegeben:

Definition 2.3 (Lokationsbasierte Informationssysteme) *Lokationsbasierte Informationssysteme sind Dienste, auf die über mobile Endgeräte durch das mobile Netzwerk zugegriffen werden kann und die die Möglichkeit bieten, von der aktuellen Position des Endgeräts Gebrauch zu machen [Virrantaus et al., 2001].*

Um eine automatische Reaktion auf Nutzerhandlungen basierend auf dem jeweiligen Kontext zu ermöglichen, müssen Regeln der Verarbeitungslogiken erstellt werden. Ein

Beispiel einer Implementierung, die es ermöglicht, Sensorinformationen aufzunehmen und anschließend zur späteren Anwendung semantisch zu annotieren ist aCAPpella [Dey et al., 2004]. Dies ist ein Tool, welches die Erkennung von Handlungen ermöglicht, um geeignete Aktionen in der Umgebung ausüben zu können. Dazu wird eine grafische Oberfläche bereitgestellt, die von Sensoren erfasste Informationen visuell darstellt, zum Beispiel die Geräuschpegel von Tonaufnahmen oder den Status, ob ein Sensor aktiv oder inaktiv war (z.B. ob eine Person im Raum detektiert wurde oder nicht). Diese Sensorinformationen werden aufgenommen, zwischengespeichert und automatisch in *Events* unterteilt, wie beispielsweise „Personen sprechen“ oder „Anzahl von Personen“. Diese Events werden in der grafischen Oberfläche aufbereitet visualisiert, so dass der Benutzer entsprechende Stellen aus den Sensordaten markieren und somit eigene Events definieren kann, auf die das System später reagieren soll. Mittels Methoden aus dem maschinellen Lernen werden anschließend die annotierten Events gelernt. Um eine zufriedenstellende automatische Erkennung von Events zu gewährleisten, müssen mehrere solcher Sensoraufnahmen annotiert und der Komponente des maschinellen Lernens zur Verfügung gestellt werden. Wurden genügend Aufzeichnungen des gleichen Typs annotiert und dem Lernalgorithmus übergeben, kann das System anschließend automatisch erkennen, wenn ein vordefiniertes Event eintrifft und eine vorgegebene Aktion ausführen. Ein Fokus bei dieser Arbeit liegt darauf, dass das Tool auch ohne Programmierkenntnisse verwendet werden kann, was durch die grafische Darstellung erreicht wurde, in der die erfasste Ausführung semantisch annotiert werden kann.

2.1.5 Agenten

Durch die intelligente Verarbeitung von Sensorinformationen können Smarte Umgebungen automatisch auf Veränderungen reagieren, wodurch sie als Agentensystem angesehen werden. Der Unterschied zwischen einem Dienstprogramm und einem Agentensystem besteht darin, dass Dienste passiv bleiben, solange sie nicht von einer externen Quelle aufgerufen werden. *Agenten* hingegen agieren autonom, indem sie pro-aktiv nach Lösungen suchen, selbstständig geeignete Dienste finden und diese aufrufen können, wenn die Notwendigkeit erkannt wurde [Klusch, 2008, Seite 41]. Dazu verwendet der Agent Sensoren, um den Zustand seiner Umgebung zu erfassen und Aktuatoren, um auf diese Einfluss auszuüben [Russell und Norvig, 1995, Seite 31]. Daneben sollte der Agent eigenes Wissen besitzen oder auf anderes zurückgreifen können, was ihm die Fähigkeit verleiht, autonom arbeiten zu können. Das Wissen und die Sensorinformationen werden dabei im Programm des Agenten verarbeitet und auf Reaktionen abgebildet. Neben physikalischen Agenten, z.B. Robotersysteme in Fabriken, können diese auch rein virtuell existieren und werden in diesem Fall als Softwareagenten (bzw. Softbots) bezeichnet.

Das Wichtige bei Agentensystemen ist, dass sie selbstständig in Echtzeit zwischen vielen verschiedenen Lösungsmöglichkeiten anhand ihres Wissens und der erfassten Sensorinformationen entscheiden müssen. Dabei werden drei Arten von Agenten voneinander unterschieden: *Reflex-Agenten*, *zielbasierte Agenten* und *nutzwertbasierte Agenten*. Reflex-Agenten agieren nach Bedingung-Aktions-Regeln, die auch mit den Begriffen Situation-Aktions-Regeln oder Produktionsregeln bezeichnet werden. Dies bedeutet, dass,

sobald eine Bedingung zutrifft, die entsprechende Regel feuert und damit die Aktion angewendet wird. Daneben gibt es die zielbasierten Agenten, die eine Menge von Regeln haben und diese versuchen so nacheinander auszuführen, dass ein gewünschter Zielzustand erreicht wird. Die Abfolge der zu feuernden Regeln wird auch als Kompositionsplan bezeichnet, da er sich aus einer definierten Abfolge von Handlungen zusammensetzt. Die letzte Gruppe der Agenten sind die nutzwertbasierten. Diese agieren so, dass das Ergebnis einer Handlung zu einem Weltzustand führt, der einen höheren Nutzwert für ihn besitzt, wobei der Nutzwert anhand einer vordefinierten Formel ermittelt wird. Beispielsweise stellt die prozentuale Gewinnchance einen möglichen Nutzwert bei einem Spiel dar.

Um entsprechend auf den Status der Umgebung wirken zu können, muss der Agent den gegebenen Umgebungsfaktoren angepasst werden. Dabei kann man unter den folgenden sechs Eigenschaften unterscheiden [Russell und Norvig, 1995, Seite 46]:

↪ **Zugänglich** vs. **unzugänglich**
Wenn der Agent zu dem kompletten Status der Umgebung Zugriff hat und diesen dort abrufen kann, wird die Umgebung als zugänglich für den Agenten bezeichnet. Dazu müssen die Sensoren alle für die Entscheidungsfindung hilfreichen Daten erfassen. Dies bietet den Vorteil, dass der Agent den aktuellen Zustand der Umgebung nicht selbstständig nachverfolgen und zwischenspeichern muss.

↪ **Deterministisch** vs. **nichtdeterministisch**
Kann der nächste Zustand einer Umgebung vollständig von dem aktuellen Status und der vom Agenten ausgeführten Aktion abgeleitet werden, bezeichnet man die Umgebung als deterministisch. In diesem Fall muss der Agent nicht mit Ungenauigkeiten und unerwarteten Phänomenen arbeiten.

↪ **Episodisch** vs. **nicht episodisch**
Bei episodischen Umgebungen wird die Erfahrung des Agenten in Episoden unterteilt, wobei jede Episode aus einer Abfolge von Erfassen des Umgebungszustands und entsprechender Handlung besteht. Dabei haben die Episoden keinen Einfluss aufeinander, so dass der Agent nicht vorausdenken muss.

↪ **Statisch** vs. **dynamisch**
Eine Umgebung ist dynamisch, wenn sie sich während der Entscheidungsfindung eines Agenten verändern kann und damit schwerer zu kontrollieren ist als statische, da die Zeitspanne zwischen Erfassung des Zustands und entsprechender Handlung so klein wie möglich sein sollte.

↪ **Diskret** vs. **kontinuierlich**
Gibt es ausschließlich eine begrenzte Anzahl von sich unterscheidenden, klar spezifizierten Erfassungen und Aktionen, spricht man von diskreten, andernfalls von kontinuierlichen Umgebungen.

2.2 Smarte Umgebungen

Smarte Umgebungen (siehe Definition 1.1, Seite 5) setzen sich aus physischen Objekten sowie Sensoren und Aktuatoren zusammen, die über verarbeitende Dienste miteinander verbunden sind. Während Sensoren Veränderungen in Umgebungen erfassen, führen Aktuatoren mittels mechanischer, elektromagnetischer oder pneumatischer Vorrichtungen Anpassungen in diesen aus (siehe Abschnitt 2.2.1). Um aus den erfassten Informationen einen Mehrwert für den Benutzer generieren zu können, müssen diese entsprechend übertragen und von Diensten verarbeitet werden (siehe Abschnitt 2.2.2). Sowohl die Datenübermittlung als auch die verarbeitenden Dienste müssen an den jeweiligen Verwendungszweck angepasst werden. Aus diesem Grund müssen bereits bei der Entwicklung von Smarten Umgebungen diese im Hinblick auf ihren Verwendungszweck unterschieden werden, was sich in diversen Unterscheidungskriterien widerspiegelt (siehe Abschnitt 2.2.3). Neben den physisch erfassten und verarbeiteten Informationen in der Smarten Umgebung werden durch die zunehmende Vernetzung mit Hilfe des Internets weitere Daten aus externen Quellen zur Verarbeitung zugezogen, was eine Verschmelzung der physischen und der virtuellen Welt mit sich bringt. Diese Ansätze werden im Themengebiet der Cyber-Physischen Umgebungen (CPE) aufgegriffen und weiterentwickelt (siehe Abschnitt 2.2.4).

2.2.1 Sensoren und Aktuatoren

Sensoren sind im Allgemeinen Geräte, welche auf einen physikalischen Stimulus reagieren, womit folgende Definition aufgestellt werden kann:

Definition 2.4 (Sensor) *Ein Gerät, welches auf einen physikalischen Stimulus, wie beispielsweise thermische Energie, elektromagnetische Energie, akustische Energie, Druck, Magnetismus oder Bewegung reagiert, indem ein Signal - normalerweise elektrisch - erzeugt wird [National Communications System Technology & Standards Division, 1996].*

Der physikalische Stimulus kann entweder direkt durch einen Benutzer erfolgen (z.B. Betätigung eines Schalters) oder automatisch anhand von Veränderungen erfasst werden (z.B. Temperaturveränderung). Während bei der ersten Variante der Benutzer bewusst und eigenständig bestimmt, wann der Sensor ausgelöst wird, funktioniert dies bei der zweiten Variante automatisch. Je nach Typ des Sensors sind beide oder nur eine der zwei Varianten zur Auslösung des Stimulus möglich. Beispielsweise ist eine Betätigung von Knöpfen und Schaltern nur mittels direkter Interaktion durch einen Benutzer möglich. Die Ausrichtung einer Kompassnadel hingegen erfolgt ohne menschlichen Einfluss, indem sie sich automatisch nach dem Erdmagnetfeld ausrichtet. Bewegungssensoren funktionieren in der Regel automatisch. Dies bedeutet, dass Personen ohne explizites Ziel, den Sensor auszulösen, sich durch dessen Erfassungsbereich bewegen, was durch den Sensor erfasst und kommuniziert wird. Jedoch können Benutzer auch willentlich diese Sensoren auslösen, indem sie sich absichtlich in den Erfassungsbereich des Sensors begeben. Die beiden Varianten haben zudem einen Einfluss auf die Wahrnehmung des Benutzers, wenn in der Umgebung Aktionen basierend auf den erfassten Sensorinformationen ausgeführt werden. Bei den direkten Interaktionen wird eine entsprechende Reaktion erwartet, wohingegen bei

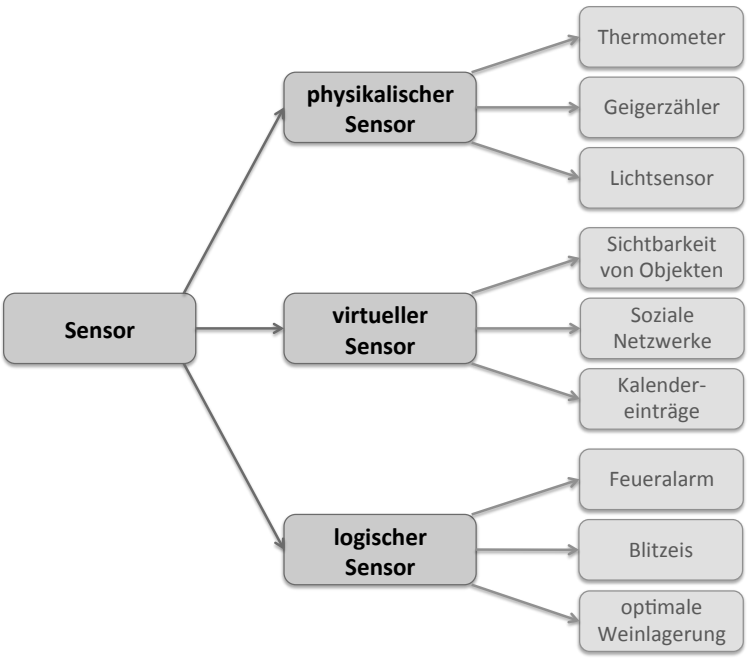


***Abbildung 2.3**: Unterteilung von Sensoren mit Beispielen in Anlehnung an Baldauf et al.*

indirekten Interaktionen die Umgebung aus Sicht des Benutzers autonom funktioniert und dieser daher nicht unbedingt mit einer Reaktion rechnet.

Des Weiteren kann zwischen *physikalischen*, *virtuellen* und *logischen Sensoren* unterschieden werden [Baldauf et al., 2007]. Die physikalischen Sensoren (auch als harte Sensoren bezeichnet) können Informationen in der Umgebung erfassen. Beispielsweise können mit Mikrofonen Audiosignale erfasst werden. Virtuelle Sensoren (bzw. softe Sensoren) bestehen nur aus Softwareapplikationen oder Diensten. Darunter fallen Programme, die das Netz durchsuchen und Informationen daraus zur Verfügung stellen. Logische Sensoren beziehen Daten von mehreren Informationsquellen. Sie kombinieren sowohl physikalische als auch virtuelle Sensoren mit weiteren Datenquellen, wie externen Datenbanken, um komplexere, höherwertige Informationen zu generieren. Beispielsweise können Regen- und Temperatursensoren verwendet werden, um Blitzeis zu prognostizieren. Ebenso können Temperaturwerte und Rauchmelder kombiniert werden, um Brände zu detektieren. Bei einer optimalen Weinlagerung müssen Luftfeuchtigkeit, Temperatur und Lichteinfall in einem vorgegebenen Rahmen liegen. Sobald einer dieser Werte die Richtwerte über- oder unterschreitet, kann eine nicht optimale Weinlagerung festgestellt werden. Abbildung 2.3 stellt die Einteilung der Sensoren inklusive einiger Beispiele grafisch dar.

Für die weite Verbreitung von Sensorik werden kostengünstige und stromsparende Sensorknoten benötigt, die mittels eines Sensornetzwerks miteinander verbunden werden. Dabei umfassen Sensorknoten, im Allgemeinen Sensoren genannt, neben den Sensoreinheiten Übertragungsmodule, Stromaggregate und Prozessoren, um die erfassten Rohdaten vor

der Bereitstellung aufzubereiten und somit nur die notwendigen Informationen weiterzuleiten [Akyildiz et al., 2002]. Alle in einem Sensornetzwerk erfassten Daten werden in sogenannten Datensenken gesammelt und darüber Endbenutzern oder darauf aufbauenden Diensten zur Verfügung gestellt. Die Größe der Sensorknoten kann dabei je nach Einsatzgebiet variieren und so klein werden, dass sie eine Größe von wenigen Millimetern besitzen. Solche miniaturisierten Sensoren sind auch unter dem Begriff „Smart Dust“ bekannt [Kahn et al., 1999]. Neben einer selbstständigen Stromversorgung mittels Batterien oder Solarzellen kann die Stromversorgung auch von externen Quellen erfolgen, so dass die Sensoren nur in der dafür vorgesehenen Umgebung eingesetzt werden können. Vorteil dabei ist, dass die Energiezufuhr einfacher überwacht und gesichert werden kann [Lifton et al., 2002].

Definition 2.5 (Aktuator) *Ein Aktuator ist ein auslösendes Gerät zur Betätigung einer mechanischen, elektromagnetischen oder pneumatischen Vorrichtung, die beispielsweise durch eine Sensorverknüpfung mit einem Computer verbunden ist.*

Aktuatoren stellen die Gegenstücke zu Sensoren dar und führen Veränderungen in der Umgebung aus. Dabei werden elektrische Signale, welche als Steuerungsbefehle versendet werden, in andere physikalisch messbare Aktionen umgewandelt. Die möglichen Effekte können dabei visuell, akustisch, haptisch oder olfaktorisch von Menschen erfasst werden. Beispiele von Aktuatoren sind Monitore, Lautsprecher, Vibrationsmotoren oder elektronische Lufterfrischer. Der Effekt der Veränderung kann entweder kurzfristig oder dauerhaft sein. Beispielsweise haben akustische Ausgaben nur zu dem Zeitpunkt der Ausgabe einen messbaren Einfluss, wohingegen das Öffnen eines Tors so lange bestehen bleibt, bis dieses wieder geschlossen wird. Im Gegensatz zu Sensoren müssen Aktuatoren explizit angesteuert werden und agieren nicht autonom. Die Ansteuerung erfolgt dabei basierend auf erfassten Sensorinformationen. Beispielsweise kann ein Licht eingeschaltet werden, sobald eine Bewegung beim zugehörigen Bewegungssensor erfasst wird. Neben Reaktionen auf Informationen, die von physikalischen Sensoren erfasst werden, können Aktuatoren auch basierend auf Daten von virtuellen Sensoren angesteuert werden, zum Beispiel, um alle fünf Sekunden ein neues Bild auf einem Monitor anzuzeigen. In diesem Fall erfasst der virtuelle Sensor die Zeitdifferenz, die für die Ansteuerungssignale herangezogen wird. Für die Analyse der erfassten Sensorinformationen und zur Ansteuerung der Aktuatoren werden in der Regel Verarbeitungslogiken eingesetzt, welche anhand vorgegebener Regeln agieren.

Neben Menschen können ebenfalls mittels Aktuatoren erfolgte Umgebungsveränderungen von Sensoren erfasst werden. Durch eine Verschmelzung zu Sensor-Aktuator-Netzwerken gepaart mit „intelligenten“ Verarbeitungslogiken entstehen dabei Umgebungen mit der Fähigkeit, autonom auf Veränderungen reagieren zu können. Abbildung 2.4 stellt den Verarbeitungskreislauf in solchen Smarten Umgebungen dar. Die Überwachung der Umgebung erfolgt durch Sensoren, deren Informationen an verarbeitende Dienste geschickt werden. Jede Sensorinformation stellt dabei eine Veränderung der Umgebung dar, die mittels der Dienste analysiert und verarbeitet wird. Entsprechend der Ergebnisse dieser Verarbeitung können Steuerungsmechanismen mittels der Aktuatoren in der Umgebung angeregt werden. Die daraus resultierende Reaktion stellt wiederum eine Veränderung der Umgebung dar, welche mittels der Sensoren erfasst und damit überwacht wird.

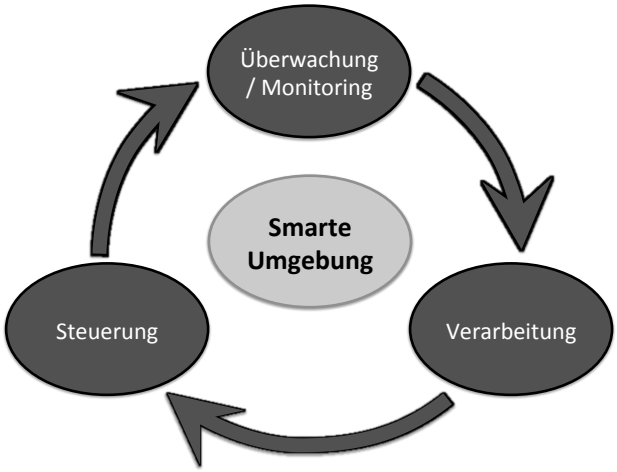


***Abbildung 2.4**: Verarbeitungskreislauf in Smarten Umgebungen*

2.2.2 Datenverarbeitung in Smarten Umgebungen

Daten können mittels fest installierter oder mobiler Sensorik erfasst werden, so dass sich die Thematik der Datenverarbeitung in Smarten Umgebungen aus den beiden Forschungsthemenfeldern der verteilten Systeme (*Distributed Systems*) und des mobilen Verarbeitens (*Mobile Computing*) zusammensetzt. Durch die verteilten Systeme können gewisse Aufgaben an spezialisierte Programme auf dedizierten Systemen ausgelagert werden. Mobile Verarbeitungsmöglichkeiten integrieren sich in diesen Ansatz, indem auch mobile Endgeräte als Verarbeitungssysteme hinzugezogen werden können. Durch eine entsprechende Verbindung zu Sensoren und Aktuatoren können Benutzerinteraktionen automatisch erfasst, verarbeitet und zur proaktiven Unterstützung verwendet werden, ohne dass der Benutzer selbst wissentlich aktiv agiert, sondern sich möglichst natürlich verhält. Dabei müssen die Systeme fehlertolerant und sicher im Hinblick auf Privatsphäre und sensible Daten sein [Satyanarayanan, 2001].

Bei Anwendungsprogrammen und Verarbeitungslogiken können zwei Typen unterschieden werden. Einerseits gibt es Dienste, die autonom ohne direkte Beeinflussung von Benutzern agieren. Sie arbeiten ausschließlich auf den erfassten Sensordaten sowie den ermittelten Ergebnissen anderer Programme. Andererseits gibt es Applikationen, mit denen der Benutzer direkt interagieren kann, zum Beispiel durch die Bedienung mit Hilfe einer Tastatur. Über eine entsprechende *Middleware* werden Dienste und Applikationen sowohl untereinander als auch mit den Sensor-Aktuator-Netzwerken der Umgebung verbunden. Aufgrund der Weiterentwicklung und der damit einhergehenden Verbreitung eingebetteter Sensoren und Aktuatoren entstand bereits frühzeitig die Bestrebung, basierend auf einer Vernetzung dieser Komponenten, Systeme zu erstellen, die autonom pro-aktiv agieren können [Tennenhouse, 2000]. Durch die Möglichkeit, mittels paralleler Verarbeitung

Informationen schnell auszuwerten, kann die Verarbeitungszeit stark reduziert werden, was eine zeitnahe Reaktion auf Sensorinformationen ermöglicht. Mittels der autonom agierenden Dienste, die in Echtzeit Reaktionen basierend auf Interaktionen und Veränderungen in der Umgebung durchführen können, kann man intelligentes Verhalten realisieren, weshalb man auch von *ambienter Intelligenz* spricht [Augusto et al., 2010]. Die Architektur im Sinne einer solchen ambienten Intelligenz ist in Abbildung 2.5 dargestellt.

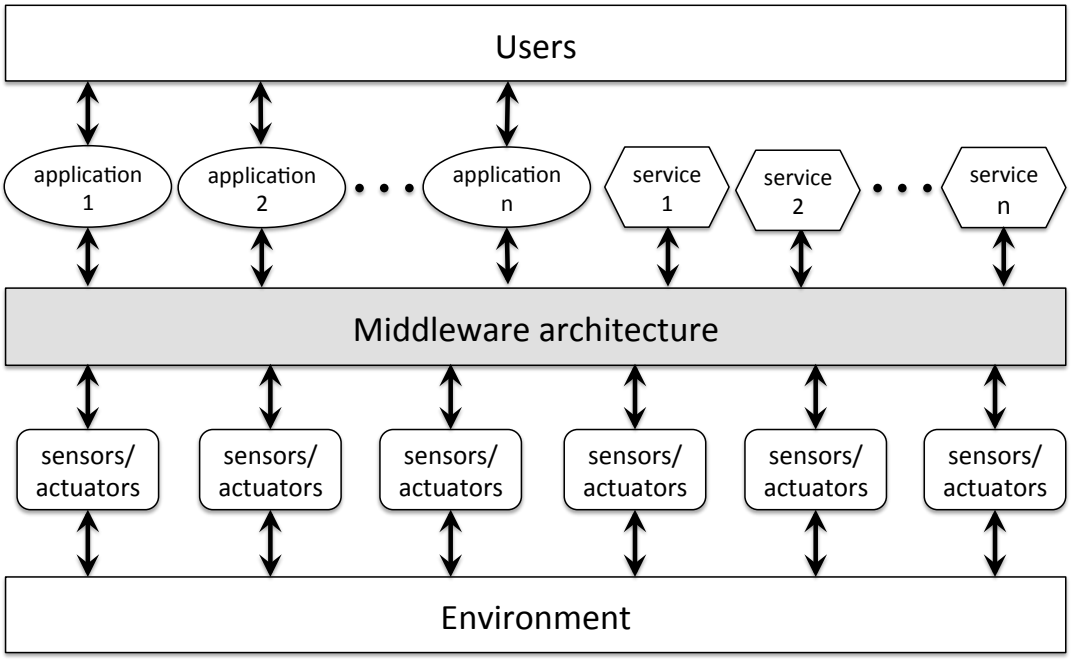


***Abbildung 2.5**: Architektur im Sinne der ambienten Intelligenz inklusive der Middleware (Quelle [Augusto et al., 2010])*

2.2.3 Unterscheidungskriterien von Smarten Umgebungen

Eine wichtige Eigenschaft von Smarten Umgebungen ist ihre Ausrichtung. Zum einen können sie vorwiegend dazu verwendet werden, um in ihr befindliche Personen zu unterstützen und sind daher *benutzerzentriert*. Zum anderen können solche Umgebungen *technologiezentriert* sein und sind in einem solchen Fall entsprechend konzipiert, um neue Hardware, Netzwerke und Middleware-Dienste zu testen [Abowd und Sterbenz, 2000]. Je nachdem, welche Ausrichtung die Smarte Umgebung hat, müssen geeignete Sensoren und Aktuatoren verwendet werden. Des Weiteren wirkt sich dies auch auf die verarbeitenden Dienste aus. Zum Beispiel müssen benutzerzentrierte Umgebungen Informationen über die jeweiligen Personen, beispielsweise aus einem Benutzermodell, abrufen können, um individuell und adaptiv auf deren Interaktionen und Präferenzen reagieren zu können. Da hierzu private Daten benötigt werden, müssen in solchen Fällen geeignete Maßnahmen getroffen werden, um die Privatsphäre der Personen nicht zu verletzen.

Neben der Ausrichtung kann bei Umgebungen auch zwischen den Domänen unterschieden werden, für die sie konzipiert sind. Im Laufe der vergangenen Jahre wurden bereits Umgebungen in mehreren Domänen instrumentiert, um Personen, die sich in diesen befinden, ziel- und zum Teil rollenbasiert zu unterstützen. Angepasst auf das jeweilige Einsatzgebiet weisen die Instrumentierungen und verfolgten Ziele unterschiedliche Richtungen auf. Prinzipiell sind drei differenzierbare Gruppen von Einsatzgebieten zu unterscheiden: *private*, *halböffentliche* und *öffentliche Umgebungen*. Abbildung 2.6 zeigt die im Rahmen dieser Arbeit erfolgte Einteilung von Smarten Umgebungen inklusive einiger Besipiele. Die zur Einteilung verwendeten Unterschiede zwischen den einzelnen Gruppierungen begründen sich unter anderem durch folgende Eigenschaften:

- Eigentümer der Umgebung
- Betreiber der Umgebung
- Nutzer der Umgebung
- Anzahl der Nutzer
- Wissen über die Nutzer
- Zugang zur Infrastruktur und Diensten
- Schutzmechanismen zur Wahrung der Privatsphäre

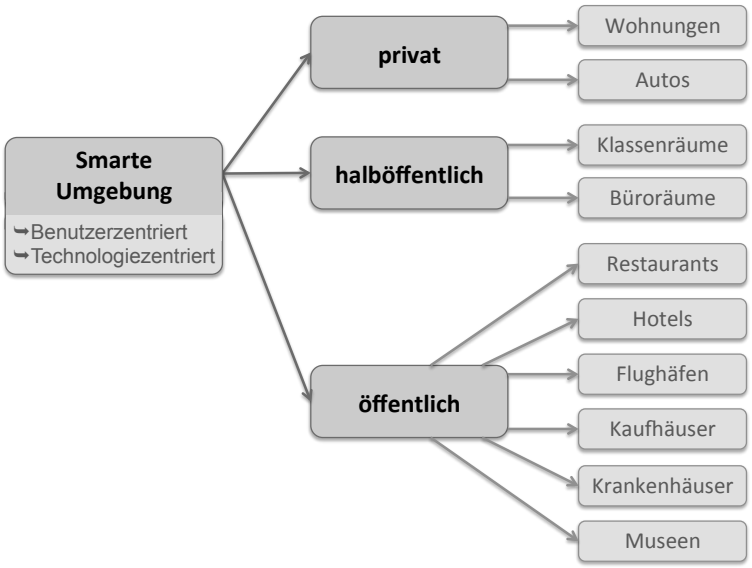


***Abbildung 2.6**: Unterteilung von Smarten Umgebungen inklusive Beispiele*

Private Umgebungen: In privaten Umgebungen sind in der Regel Eigentümer und Betreiber auch die Nutzer dieser Umgebungen. Es gibt einen kleinen geschlossenen Personenkreis, der Zugriff zur Infrastrukturumgebung hat, wobei sich dieser gut kennt. Aus diesem Grund können private Informationen bei der Assistenzfunktionalität meist bedenkenlos verwendet werden. Da der Personenkreis bekannt ist, kann einfach auf Informationen eines jeden Beteiligten zurückgegriffen werden. Fremde Personen benötigen keinen Zugriff auf die Infrastruktur, womit durch eine externe Sperre die Privatsphäre verhältnismäßig einfach geschützt werden kann.

Halböffentliche Umgebungen: In halböffentlichen Umgebungen sind die Benutzergruppen ebenfalls bekannt, wobei die Anzahl der Nutzer größer ist als bei privaten Umgebungen. Zudem ist es nicht zwingend der Fall, dass sich alle untereinander gut kennen. Daher müssen geeignete Schutzmechanismen implementiert sein, um die Privatsphäre jedes Einzelnen zumindest zeitweilig und situationsabhängig schützen zu können. Der Eigentümer der Umgebung ist dabei oftmals disjunkt vom Nutzerkreis.

Öffentliche Umgebungen: Bei öffentlichen Umgebungen ist weder der Nutzerkreis noch die Anzahl a priori bekannt. Zudem kann auch während des Betriebs nicht gewährleistet werden, dass Informationen über die aktuellen Nutzer akquiriert werden können. Zudem ist der heterogene Nutzerkreis noch größer als bei halb öffentlichen Umgebungen, so dass ein höherer Schutz der Privatsphäre eingehalten werden muss.

	Private Umgebungen	Halböffentliche Umgebungen	Öffentliche Umgebungen
Nutzer	bekannt	bekannt	unbekannt
Eigentümer, Betreiber, Nutzer	gleicher Personenkreis	disjunkter Personenkreis	disjunkter Personenkreis
Anzahl der Nutzer	wenige	viele	sehr viele
Wissen über die Nutzer	bekannt	teilweise bekannt	größtenteils unbekannt
Zugang zur Infrastruktur und Diensten	einfach	größtenteils einfach	schwierig
Schutzmechanismen zur Wahrung der Privatsphäre	einfach	komplex	sehr komplex

***Tabelle 2.1**: Unterscheidungsmerkmale der Gruppen von Smarten Umgebungen*

In Tabelle 2.1 werden die in dieser Arbeit spezifizierten Unterscheidungsmerkmale zwischen Gruppen von Smarten Umgebungen zusammengefasst und gegenübergestellt. Aufgrund dieser Unterschiede müssen je nach Einsatzgebiet sowohl geeignete Sensoren/Aktuatoren als auch Verarbeitungsalgorithmen entwickelt und eingesetzt werden. Entsprechend dazu muss die Middleware agil sein, um sich an diese Konstellation anpassen zu können.

2.2.4 Cyber-Physische Umgebungen

Betrachtet man neben der physischen Umgebung noch die virtuelle Welt, können neuartige Anwendungsfelder erschlossen werden, wie beispielsweise die Fernüberwachung autonom arbeitender Produktionssysteme. Dabei kommt es zu einer Verschmelzung der Smarten Umgebung und dem Cyberspace, was unter dem Begriff Cyber-Physische Umgebungen (Cyber-Physical Environments, CPE) bekannt ist. Eine CPE besteht hierbei aus einem Zusammenschluss aller in der Umgebung befindlichen Cyber-Physischen Systeme (Cyber-Physical Systems, CPS) (siehe Abbildung 2.7).

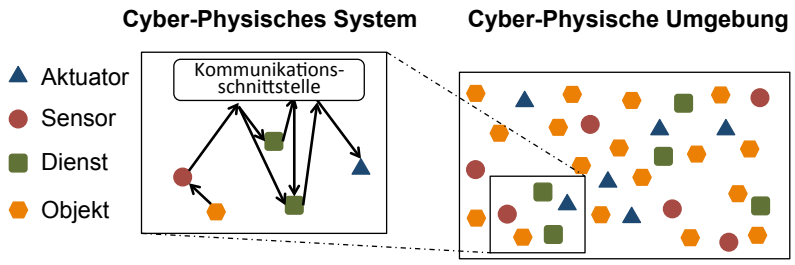


***Abbildung 2.7**: Cyber-Physische Umgebungen als Zusammenschluss mehrerer Cyber-Physischer Systeme*

CPS bestehen aus einer Vernetzung von Sensoren, Aktuatoren, Objekten und Diensten (siehe Abbildung 2.8). Objekte und deren Kontextparameter werden durch Sensoren erfasst und diese Daten über eine entsprechende Kommunikationsschnittstelle an Dienste geschickt. Nach der Auswertung der Sensorinformationen werden die Resultate zur Weiterverarbeitung an weitere Dienste kommuniziert, Anwendern grafisch repräsentiert, persistent zur späteren Verwendung gespeichert oder zur Ansteuerung von Aktuatoren verwendet. Zur Interaktion werden in CPS eine Reihe dedizierter, multimodaler Mensch-Maschine-Schnittstellen zur Verfügung gestellt. Für die Verarbeitung der Informationen durch Dienste werden zudem Daten aus dem Cyberspace hinzugezogen. Dies bedeutet, dass neben den Daten aus dem lokalen Netzwerk Informationen und Dienste aus dem World Wide Web (WWW) Berücksichtigung finden [acatech, 2011]. Zusätzlich existieren weitere Kommunikationsstrukturen innerhalb der CPS zu anderen CPS der Umgebung sowie globalen Netzen im WWW. Aus diesem Grund ist ein wichtiger Bestandteil neben der Sensorik und Aktuatorik in CPS die Kommunikations- und Informationstechnik, damit eine ad-hoc-Vernetzung aller Komponenten realisiert werden kann. „CPS erfordern unterschiedliche Applikationen schnell und unkompliziert miteinander zu vernetzen, sowohl statisch zur Entwicklungszeit als auch dynamisch im Betrieb.“ [acatech, 2011, S. 25].

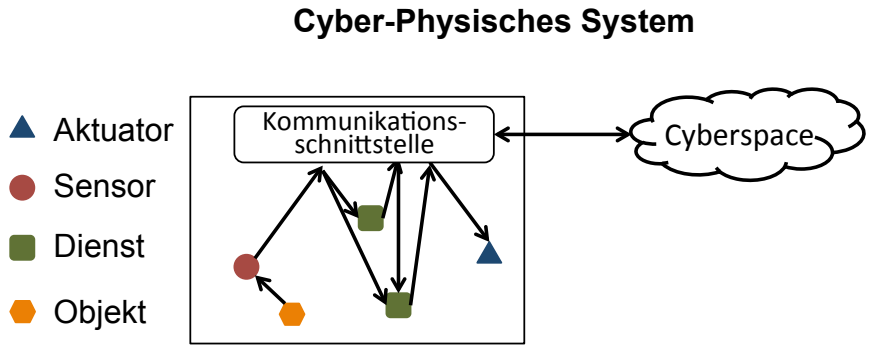


***Abbildung 2.8**: Cyber-Physisches System als Zusammenschluss von Sensoren, Aktuatoren, Objekten und Diensten*

Durch die Bereitstellung der Kommunikationsschnittstellen für externe Applikationen werden CPS als domänenübergreifende Vernetzungsmöglichkeit angesehen und bieten daher neuartige Anwendungsmöglichkeiten und somit auch neue Geschäftsmodelle. CPS ist eine Technologie, welche die virtuelle und die physische Welt zusammenbringt, um eine vernetzte Welt zu erreichen, in der intelligente Objekte mit allen anderen kommunizieren und interagieren können [MacDougall, 2013]. Dies ermöglicht entsprechend der Idee des Internets der Dinge, Daten und Dienste, da von überall auf die entsprechenden Ressourcen zugegriffen werden kann. Hierdurch wird der Markt für innovative Applikationen geöffnet, indem die Grenzen zwischen physischer und virtueller Welt verschwinden.

Damit die Komponenten in Cyber-Physischen Umgebungen (CPE), als Zusammenschluss aller in der Umgebung befindlichen CPS, interoperabel miteinander fungieren können, ist eine Trennung von Sensoren, Aktuatoren und Diensten notwendig. Ein Informationsaustausch muss in diesem Fall über die entsprechende Kommunikationsinfrastruktur erfolgen. Dies ermöglicht es, dass Sensorinformationen flexibel zu mehreren Diensten geschickt werden können, ohne die Kommunikationsverbindung zum Sensor aktiv verändern zu müssen. Anstelle dessen funktioniert die Informationsübertragung ad hoc automatisch. Ebenso können mehrere Dienste die Ansteuerung von Aktuatoren realisieren. Es kann jedoch zu Konflikten kommen, wenn zwei Dienste denselben Aktuator ansteuern möchten. Um solche parallelen Zugriffe adäquat zu verarbeiten, müssen geeignete Mechanismen der Zugriffsverwaltung verwendet werden, die den Konflikt auflösen, analog zur Konfliktauflösung bei multimodalen Applikationen (siehe Abschnitt 2.1.1).

Wie oben beschrieben bestehen CPE aus drei großen Komponenten, die miteinander in Verbindung stehen und sich gegenseitig beeinflussen. Auf der einen Seite existiert die physische instrumentierte Umgebung, in der Sensoren Interaktionen und Veränderungen erfassen und Aktuatoren eine Schnittstelle darstellen, um Änderungen in der Umgebung vorzunehmen. Auf der anderen Seite gibt es die virtuelle Komponente, die Informationen aus dem Cyberspace, wie beispielsweise dem WWW, entsprechend zu Objekten und

Diensten hinzuzieht. Mit Hilfe von geeigneten Benutzerschnittstellen und -oberflächen können die Informationen der Umgebung auch dazu verwendet werden, ein virtuelles Abbild der Umgebung zu erstellen, welches von einem Benutzer inspiziert werden kann. Um dies zu gewährleisten, müssen die beiden Komponenten durch eine entsprechende dritte Komponente, die Kommunikationsinfrastruktur, miteinander verbunden werden. Eine Architektur für CPE umfasst daher immer diese drei Komponenten.

Der Zusammenhang zwischen CPE und deren Komponenten wird in dieser Arbeit untersucht und in Abbildung 2.9 dargestellt. Dabei umfasst eine CPE alle darin befindlichen Benutzer, Objekte sowie die aufgespannten CPS. Zudem können der CPE Applikationen zugeordnet werden, die das Zusammenspiel der einzelnen Komponenten regelt und die auf den jeweiligen Kontext der CPE angepasst sind. Das gleiche gilt für die CPS, die ebenfalls spezifische Applikationen bereitstellen können. Daneben umfassen CPS Sensoren, Aktuatoren und Objekte. Objekte wiederum können eigene CPS aufspannen, wie beispielsweise Produktionsanlagen, die als ein Objekt angesehen werden können, aber gleichzeitig aus einem Zusammenschluss von Sensoren und Aktuatoren bestehen. Zusätzlich können objektspezifische Informationen in einem digitalen Objektgedächtnis (DOMe)

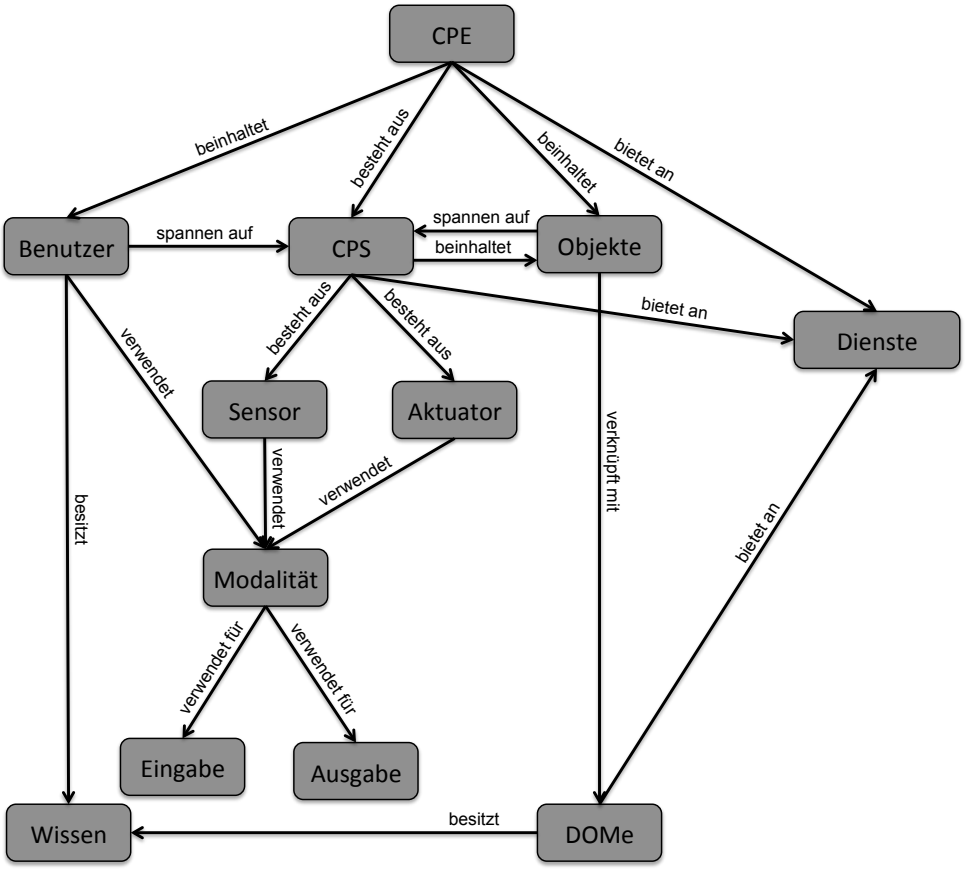


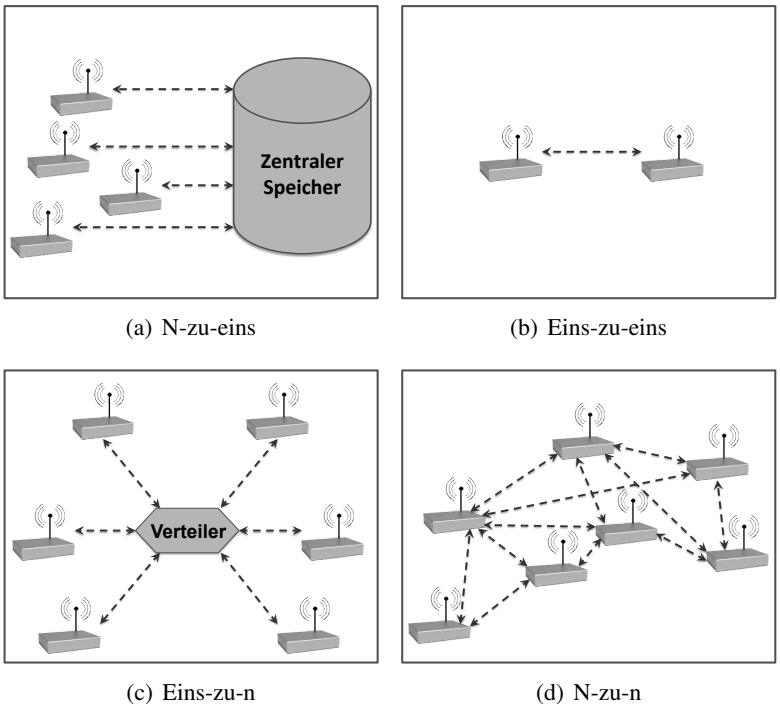
***Abbildung 2.9**: Virtualisierung im Sinne des Internet der Dinge und Dienste*

hinterlegt werden. Aufgrund aktiver Objektgedächtnisse, die pro-aktiv auf Veränderungen reagieren, können Objektgedächtnisse auch Applikationen zugeordnet werden. Durch die ständige Informationsspeicherung kann aus den Objektgedächtnissen auch spezifisches Wissen extrahiert werden. Benutzer haben ebenfalls Wissen, welches sie in der CPE und der Interaktion mit dieser einbringen. Des Weiteren können auch Benutzer CPS aufspannen, basierend auf einer digitalen Instrumentierung, z.B. durch Sensoren in der Kleidung, wie Handschuhe [Fishkin et al., 2005], oder in Form von Arm- und Brustbändern [Bonato, 2003; Seoane et al., 2013], Smartphone oder Smart Watch. Für die Kommunikation und Interaktion können sowohl von den Benutzern als auch von den Sensoren und Aktuatoren mehrere Modalitäten verwendet werden. Dabei kann zwischen Ein- und Ausgabemodalitäten unterschieden werden (siehe Abschnitt 2.1.1).

2.3 Kommunikation

Zur Übertragung von Daten zwischen Komponenten (Sensoren, Aktuatoren und verarbeitende Dienste) in CEP gibt es mehrere Kommunikationsmöglichkeiten, welche schematisch in Abbildung 2.10 aufgezeigt werden. Eine Methode ist die Verwendung eines *zentralen Speichers*, auf dem alle Informationen hinterlegt und von dort von allen Diensten abgerufen werden können. Da alle Dienste auf den passiven Speicher Zugriff haben, spricht man von einer *N-zu-eins*-Beziehung. Der Vorteil dabei besteht darin, dass es zwischen den einzelnen Komponenten untereinander keine eigenständige Verbindung gibt, die erzeugt und überwacht werden muss. Jede Komponente muss jedoch Zugriff auf den zentralen Speicher haben, um eine Verbindung mit diesem aufbauen zu können. Daneben gibt es die Möglichkeit der direkten Verbindung zwischen Komponenten, bei denen die einfachste und am längsten praktizierte Methode eine *Eins-zu-eins*-Beziehung ist. Es wird entweder eine direkte oder eine indirekte - über ein von beiden Komponenten bekanntes Medium - Verbindung aufgebaut, über die alle Informationen übertragen werden. Dabei muss der Sender den Empfänger kennen und ihn bei der Nachrichtenübermittlung explizit adressieren. Nachteilig bei dieser Methode hingegen ist, dass die Informationen nur einem Dienst, der zudem bekannt sein muss, zur Verfügung gestellt werden. Dies hat eine hohe Anzahl von Verbindungen in verteilten Systemen zur Folge, wenn Informationen zwischen vielen Komponenten untereinander ausgetauscht werden sollen. Eine Alternative dazu ist eine *Eins-zu-n*-Beziehung bei der Datenübermittlung. Dabei muss der Sender weder die möglichen Empfänger direkt adressieren noch muss er diese kennen. Die zu übermittelnde Nachricht wird in diesem Fall an eine zentrale Stelle geschickt und von dort an alle angeschlossenen Komponenten weiter verteilt. Dies ermöglicht eine Entkopplung von Sendern und Empfängern in Bezug auf Raum, Zeit und Synchronisation des Datenflusses.

Bei den zuvor beschriebenen Kommunikationsmöglichkeiten müssen die Adressaten a priori bereits bekannt sein, um die Verbindung aufbauen zu können. Eine Alternative der Informationsverteilung hierzu besteht aus dezentralisierten *Ad-hoc*-Netzwerken, in denen es kein zentrales System gibt, an welches die Informationen geschickt werden. Anstelle dessen bauen die Komponenten ein eigenes Netzwerk auf, über welches sie miteinander



(a) N-zu-eins (b) Eins-zu-eins

(c) Eins-zu-n (d) N-zu-n

***Abbildung 2.10**: Differenzierung von verteilten Systemen anhand ihrer Kommunikationsverbindungen*

kommunizieren. Aus diesem Grund spricht man hierbei auch von einer *N-zu-n*-Beziehung. Dadurch entstehen agile Beziehungen zwischen den Komponenten, wie sie beispielsweise bei Agentensystemen [Cook, 2009] oder bei Netzwerken kleiner Sensorknoten, z.B. Berkeley Motes [Mainwaring et al., 2002], zum Einsatz kommen. Dabei werden Informationen zu bekannten, wie beispielsweise benachbarten, Komponenten geschickt, die wiederum diese an ihre Bekannten weiterleiten, usw. Kommt eine neue Komponente in diese Infrastrukturumgebung hinzu, sucht diese selbstständig nach weiteren Komponenten in der Informationskette und macht sich autonom bei diesen bekannt. Dadurch entstehen sehr agile und möglicherweise nicht planbare Strukturen, die von den Komponenten selbstständig geregelt werden müssen. Dieses Verfahren resultiert in komplexen Systemen, die den Vorteil aufweisen, dass sie keine weitere Infrastruktur mit einer zentralen Komponente benötigen.

Neben der Art der Vernetzung kann bei Kommunikationsinfrastrukturen zwischen den Verbindungstypen unterschieden werden. Einerseits gibt es die Möglichkeit, eine Verbindung zwischen zwei Komponenten dauerhaft in Form einer Server-Client-Verbindung aufrecht zu erhalten. Dabei besteht eine kontinuierliche Datenverbindung zwischen beiden Komponenten, über die zu jeder Zeit bi-direktional Informationen übertragen werden können. Für den Aufbau einer solchen Standleitung müssen die Clients den Server a priori kennen, um sich bei diesem initial anmelden und damit die Verbindung aufbauen zu können. Andererseits können über entsprechende Schnittstellen die Verbindungen ad hoc aufgebaut

werden, um kurzfristig eine Datenübertragung durchzuführen. Dies wird oftmals in den Middleware-Architekturen aus dem Internet der Dienste verwendet, wie beispielsweise den dienstbasierten Architekturen (Service-Oriented Architecture, SOA) [Erl, 2004]. In diesem Fall können mittels bereitgestellter Schnittstelle Informationen aus den Diensten von externen Applikationen abgerufen werden. Hierbei müssen die Endpunkte, bei denen die Schnittstellen und anschließend die Daten abgerufen werden, im Vorhinein bekannt sein. Nachdem die Daten übertragen wurden, wird die Schnittstelle wieder geschlossen und muss für eine weitere Anfrage erneut aufgebaut werden.

Schließlich besteht noch die Möglichkeit in Form eines Verbindungsaufbaus nach dem Peer-to-Peer (P2P)-Ansatz. Dies findet vorwiegend bei den Ad-hoc-Netzwerken Anwendung, wobei neue Knoten automatisch nach benachbarten Knoten suchen, mit denen sie kommunizieren können und eine entsprechende Verbindung aufbauen. Entfernt sich ein Knoten aus dem Nachbarschaftsbereich, werden die zuvor initialisierten Schnittstellen automatisch wieder abgebaut. Eine spezialisierte Form sind azyklische P2P-Netzwerke. Dabei wird bei jeder Veränderung des Netzwerks, also wenn neue Komponenten hinzukommen oder welche wegfallen, sichergestellt, dass keine Zyklen unter den Verbindungen entstehen [Carzaniga et al., 2001]. Dies setzt zwar eine aufwändigere Verarbeitung bei Verbindungsauf- und -abbau voraus, hat anschließend aber den Vorteil, dass die zu übermittelten Nachrichten leichter an alle verbundenen Komponenten weiter verteilt werden können, ohne einem Knoten im Netzwerk die Nachricht mehrfach zukommen zu lassen. Aufgrund der ähnlichen Ansätze werden die P2P Infrastrukturen oftmals mittels eines Multi-Agentennetzwerks realisiert, welches die Verbindungen automatisch konsistent hält [Koubarakis, 2003; Qin et al., 2006].

Dazwischen gibt es auch vereinzelt hybride Lösungsansätze, die versuchen, die Vorteile der einzelnen Verbindungskonstellationen zu vereinen. Ein Beispiel hierfür ist der *DeviceManager* [Endres, 2003]. Bei diesem Ansatz können sich Dienste an einer zentralen Stelle, einem Dienstpool, anmelden und somit ihre Funktionalitäten für andere Dienste anbieten, wie beispielsweise die Fähigkeit, Sound über Lautsprecher auszugeben. Dienste, die aufgrund fehlender eigener Aktuatoren Aufgaben nicht erfüllen können, können beim Dienstpool Anfragen stellen. Dabei enthalten diese Anfragen spezifische Anforderungen, die vom geforderten Dienst benötigt werden. Anschließend werden die Anforderungen mit den Eigenschaften der registrierten Dienste abgeglichen. Im positiven Fall wird zwischen den beiden zusammenpassenden Diensten eine direkte Verbindung aufgebaut, bis die geforderte Aktion durchgeführt wurde. Abschließend wird die Verbindung wieder getrennt, so dass der erste Dienst erneut seine Funktionalitäten für weitere Anfragen zur Verfügung stellen kann.

Zusätzlich zu der Art und dem Aufbau von Verbindungen gibt es einen weiteren Unterschied in den diversen Systemen, welcher sich in der Art der Datenübertragung begründet. Dabei muss zwischen *synchroner* und *asynchroner* Übertragung unterschieden werden. Bei der synchronen Variante werden die Daten übertragen und auf eine Antwort des Empfängers gewartet. Beispielsweise können damit Funktionen auf entfernten Systemen aufgerufen und auf die Rückgabe der Ergebnisse gewartet werden, um diese später weiter zu verarbeiten.

Eine solche Funktionalität ist bei vielen Programmiersprachen bereits standardmäßig implementiert, wie beispielsweise *RMI* (Remote Message Invocation)[2] bei Java oder allgemeiner *RPC* (Remote Process Call) [Birrell und Nelson, 1984]. Diese Form der Datenübertragung hat den Vorteil, dass eine genaue Folge von Informationsflüssen eingehalten wird, da auf die Antworten gewartet wird und der eigentliche Prozess sich so lange in einer Art Schlafmodus befindet. Dies birgt jedoch den Nachteil, dass weitere Informationen, die zwischenzeitlich ankommen, nicht abgearbeitet werden. Um dies zu umgehen, können die neu eintreffenden Daten in eigene Arbeitsprozesse, sogenannte Threads, ausgelagert werden, was jedoch sowohl die Erstellung als auch die Wartung solcher Programme deutlich verkompliziert. Bei der asynchronen Übertragung hingegen wird nicht explizit auf das Resultat gewartet. Sobald das Ergebnis ermittelt wurde, wird der ursprüngliche Prozess notifiziert und kann diese Daten weiterverarbeiten. In der Zwischenzeit kann er jedoch voll funktionsfähig weiter agieren. Zudem muss nicht sichergestellt werden, dass der Empfänger zur Zeit der Übertragung die Daten direkt verarbeiten kann. Dazu werden die Daten in einem Puffer zwischengespeichert und dem Dienst übergeben, sobald dies möglich ist. Diese Variante vereinfacht die Unterstützung der Performanz von Systemen bei Datenübertragungen. Jedoch muss dabei berücksichtigt werden, dass keine fest definierte sequentielle Abfolge der Informationsübermittlung eingehalten werden kann.

Unterscheidungsparameter	**Ausprägungen**
Kommunikationsverbindung	N-zu-eins / Eins-zu-eins / Eins-zu-n / N-zu-n
Systemverteilung	zentral / dezentral
Verbindungsart	Standleitung / ad hoc
Übertragungsart	synchron / asynchron

***Tabelle 2.2**: Differenzierungsmerkmale der Kommunikation in Cyber-Physischen Umgebungen*

Die zuvor aufgeführten Differenzierungsmerkmale der Kommunikation in Cyber-Physischen Umgebungen sind in Tabelle 2.2 zusammengefasst. In den vergangenen Jahren wurden diverse Ausprägungen dieser Kommunikationsmöglichkeiten in mehreren Kommunikationsinfrastrukturen aufgegriffen. Der *Blackboard*-Ansatz (siehe Abschnitt 2.3.1) verwendet beispielsweise einen zentralen Speicher, der für die Kommunikation genutzt wird. Ähnlich dazu ist der *Tupelraum*-Ansatz (siehe Abschnitt 2.3.2), bei dem die im Speicher hinterlegten Informationen in klar definierten Paketen hinterlegt werden. *Publish/Subscribe*-Systeme (siehe Abschnitt 2.3.3) hingegen agieren nach der Eins-zu-n-Verbindungsmethode, um die Informationen untereinander zu verteilen. Daraus entwickelten sich die *komplexen Verarbeitungssysteme (Complex Event Processing, CEP)* (siehe Abschnitt 2.3.4) und die *Datenstrom Managementsysteme (DSMS)* (siehe Abschnitt 2.3.5), welche bereits in ihrer Kon-

[2] http://docs.oracle.com/javase/tutorial/rmi/, Zugriff: November 2014

zeption Verarbeitungslogiken integrieren. Letztere wurden zudem von *Datenbank Managementsystemen (DBMS)* beeinflusst, indem bereits etablierte Funktionalitäten aus den DBMS in DSMS zum Einsatz kommen. Im Folgenden werden die einzelnen Ansätze, welche Abbildung 2.11 nach deren Entwicklung grafisch darstellt, näher erläutert.

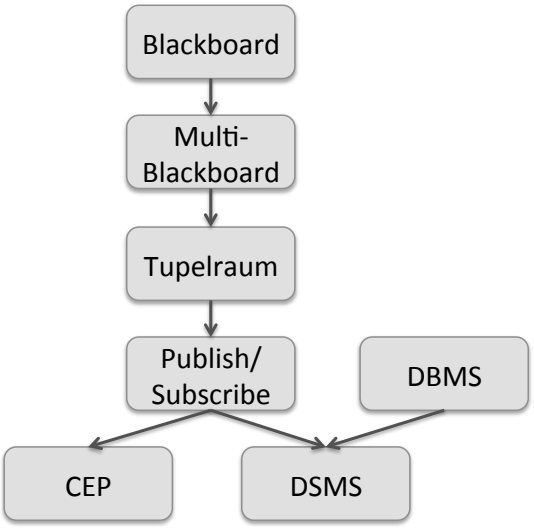


***Abbildung 2.11**: Entwicklung der Kommunikationsinfrastrukturen*

2.3.1 Blackboard und Multiblackboard Architekturen

Eine frühe Idee zur Übermittlung von Informationen war die Verwendung von gemeinsam nutzbarem Informationsspeicher, analog zur Abbildung 2.10(a). Erman et al. verwendeten dazu eine Datenbank, die sie als *Blackboard* bezeichneten [Erman et al., 1980]. Neben dem global verfügbaren Datenspeicher besteht eine Blackboard-Architektur aus voneinander unabhängigen Verarbeitungsmodulen, die als Wissensquellen bezeichnet werden [Pflegera und Hayes-Roth, 1997]. Entsprechend der zuvor beschriebenen Eigenschaften von Kommunikationsinfrastrukturen auf Basis eines zentrales Speichers haben alle vom System verwalteten Wissensquellen Zugriff auf das Blackboard und können dort sowohl Daten hinterlegen als auch von dort abrufen. Betrachtet man die Wissensquellen als autonome Agenten kann man folgende Definition festlegen:

Definition 2.6 (Blackboard) *Ein Blackboard ist eine globale Datenstruktur, die von Agenten als Kommunikationskanal verwendet wird. Dies bedeutet, dass ein Blackboard grundsätzlich eine gemeinsam genutzte Region ist, zu der alle Agenten Zugriff haben [Occello und Demazeau, 1994].*

Verwendet wurde diese Architektur unter anderem zum Sprachverstehen von einzelnen Befehlen und Anfragen [Erman et al., 1980]. Die Wissensquellen hatten dabei jeweils einen speziellen Anwendungseinsatz, so dass die Verarbeitung der komplexen Aufgabe nach dem „teile und herrsche“-Ansatz parallelisiert werden konnte. Die Zwischenergebnisse

wurden zurück auf das Blackboard geschrieben, um von anderen Wissensquellen abgerufen und weiterverarbeitet werden zu können. Daneben enthielt das Blackboard den aktuellen Verarbeitungsstatus, der von einem Blackboard-Monitor überwacht wurde. Dieser Monitor hatte neben der Überwachung die Funktion, Nachrichten an Wissensquellen zu versenden, um diese zu notifizieren und aktivieren, wenn neue Informationen im Blackboard zur Verfügung standen.

Die Anwendung von Blackboards als zentrale Kommunikationsschnittstelle wurde erweitert zu *Multi-Blackboards*, bei denen mehrere voneinander unabhängige Informationsspeicher eingesetzt werden. Durch die Entkopplung ist es möglich, diese räumlich voneinander zu trennen und mit unterschiedlichen Eigenschaften auszustatten. Beispielsweise kann mit diesem Ansatz die Rechtevergabe vereinfacht werden, indem für jede Gruppe ein eigenes Blackboard angelegt wird und nicht für jedes Blackboard ein dediziertes Rechtesystem integriert werden muss. Unter anderem kam dieser Ansatz bei Multi-Agentensystemen zum Einsatz, bei denen Agenten in mehreren Level zur gegenseitigen Kontrolle und Überwachung eingesetzt wurden [Tnazefti-Kerkeni et al., 2002]. Dabei wird davon ausgegangen, dass das zugrundeliegende Informationssystem persistent vorliegt. Bei der Verwendung wurde sich die Unabhängigkeit der Blackboards zu Nutze gemacht, indem jedes Level der Agenten mit einem eigenen Blackboard verbunden wurde. Jedes Blackboard eines bestimmten Levels beinhaltete nur die Informationen, welche für die zugehörigen Agenten notwendig waren. Zudem benötigen die diversen Level des Agentsystems unterschiedlich spezifische und detaillierte Informationen, was mittels der individuellen Zuweisung von Blackboards zu jeder Ebene einfach zu realisieren und kontrollieren war. Wurden Daten in mehreren Level benötigt, konnten diese auf mehreren Blackboards als Kopien angelegt werden. Die Synchronisation aller Kopien erfolgte durch einen entsprechenden Agenten. Dies bedeutet, dass Änderungen von Daten, die auf einem Blackboard angepasst wurden, anschließend auch auf den anderen Blackboards mit den entsprechenden Kopien nachgezogen wurden. Durch die Verteilung und geringere Zugriffsrechtevergabe wurde die Wartezeit der Agenten, um Informationen vom jeweiligen Blackboard zu lesen, verringert und gleichzeitig das mögliche Datenvolumen vergrößert.

2.3.2 Tupelräume

Bereits 1982 wurden erste Schritte hinsichtlich der Kommunikation von einzelnen dezentralen Knoten in einem Netzwerk mit anderen vollzogen. Ziel dabei war ein möglichst einfacher, generischer und erweiterbarer Datenaustausch zwischen allen Komponenten in einem solchen Netzwerk. Um eine einheitliche Struktur zu schaffen wurde der Ansatz verfolgt, alle Informationen in klar definierte Pakete, sogenannten *Tupel*, zu verpacken. Ähnlich wie bei dem Blackboard-Ansatz wurden diese Tupel über einen gemeinsamen strukturierten Speicher, den *Tupelraum*, untereinander ausgetauscht.

Definition 2.7 (Tupel) *Ein Tupel ist eine geordnete Sequenz von Feldern mit auswertbaren Formeln oder spezifischen Daten, die aus primitiven Datentypen bestehen [Gelernter und Bernstein, 1982].*

Definition 2.8 (Tupelraum) *Ein Tupelraum ist ein Raum, zu dem Tupel hinzugefügt und wieder entfernt, jedoch nicht verändert werden können. Der Tupelraum ist an einer zentralen Stelle in einem Netzwerk hinterlegt, so dass mehrere Komponenten darauf gleichzeitig zugreifen können [Gelernter und Bernstein, 1982].*

Erste Entwicklungen in diesem Bereich wurden in der Programmiersprache Linda umgesetzt [Gelernter und Bernstein, 1982]. Diese bietet die Funktionalität, mittels drei Operatoren Tupel in einem Tupelraum zu hinterlegen bzw. von dort abzurufen: *lesen*, *schreiben* und *verwenden*. Beim Schreiben werden die Tupel im Tupelraum hinterlegt und stehen fortan solange persistent für andere Dienste zur Verfügung, bis sie verwendet werden. Dies bedeutet, dass das Tupel vom Dienst ermittelt wird und der Eintrag aus dem Tupelraum entfernt wird. Im Gegensatz dazu erstellt der Lese-Operator eine Kopie des Tupels, mit dem der Dienst operieren kann, wobei das Original weiterhin im Speicher bleibt. Die zugrundeliegende Applikation, die einen Lese- oder Verwenden-Operator aufruft, erhält in beiden Fällen das gleiche Ergebnis. Der Unterschied zwischen beiden Operatoren liegt daher ausschließlich auf der Datenmenge im Tupelraum. Für die Ermittlung eines passenden Tupels muss ein Template erstellt werden, in welchem spezifiziert wird, welche Eigenschaften das gesuchte Tupel aufweisen muss. Dabei sind die Templates analog zu den Tupeln aus einer geordneten Sequenz von Feldern aufgebaut. Im Gegensatz zu den Tupeln müssen jedoch nicht alle Felder genau spezifiziert sein, sondern können durch einen Platzhalter als allgemeine Variable definiert sein. Zum Abgleich wird in erster Linie nach der Anzahl der im Template enthaltenen Felder geschaut und diesbezüglich der Tupelraum durchsucht und gefiltert. Das Set von Tupeln, welches eine übereinstimmende Anzahl von Feldern hat, wird mit den im Template bereits spezifizierten Werten von Feldern abgeglichen und somit erneut gefiltert. Schließlich wird das erste, zum Template passende Tupel als Antwort zurückgegeben.

Unter anderem wurde Linda verwendet, um Prozessausführungen zu parallelisieren [Carrier und Gelernter, 1998]. Über den gemeinsamen Speicher konnten Informationen zwischen mehreren Verarbeitungsprozessen geteilt und von diesen konsumiert werden. Somit wurden die Verarbeitungsschritte sowohl von den räumlichen als auch den zeitlichen Gegebenheiten getrennt. Sie konnten separat abgearbeitet und die Ergebnisse anschließend miteinander verschmolzen werden, was die positiven Eigenschaften von verteilten Systemen und parallelem Arbeiten vereint. Ein weiterer Vorteil dieser Vorgehensweise ist die Skalierbarkeit und Übertragbarkeit [Gelernter und Carriero, 1992]. Ebenso wie der Blackboard-Ansatz ist die Verteilung von Informationen empfängerunspezifisch und unabhängig von räumlichen und zeitlichen Gegebenheiten, da die Tupel ihre eigene Lebensdauer haben, die unabhängig von dem Prozess ist, der sie erzeugt hat. So können Tupel noch abrufbar sein, auch wenn der erzeugende Prozess schon nicht mehr aktiv ist.

2.3.3 Publish/Subscribe

Das Publish/Subscribe-Paradigma baut, wie der Name sagt, darauf auf, dass es Komponenten gibt, die Informationen publizieren und welche, die sich für den Erhalt solcher Informationen

registriert haben, indem sie Interesse dafür bekunden [Eugster et al., 2003]. Die zu übermittelnden Informationen werden hierbei in Form von *Events* übersendet.

Definition 2.9 (Event) *Ein Event bezeichnet ein beobachtbares Geschehen, beziehungsweise ein Geschehen, dessen Stattfinden angenommen wird. [Luckham, 2001].*

Sobald eine Aktivität festgestellt wird, wird ein entsprechendes Event dazu generiert und vom Erzeuger publiziert. Anschließend werden die Interessenten darüber informiert und erhalten automatisch die im Event enthaltenen Informationen. Dabei dient eine zentrale Komponente dazu, den Austausch der Daten zu realisieren, die als Broker bezeichnet wird. Wie bei den beiden zuvor präsentierten Ansätzen sind bei Publish/Subscribe Sender und Empfänger voneinander entkoppelt, sowohl in Bezug auf Zeit, Raum und Synchronisation. Zeitliche Entkopplung meint dabei, dass Events gesendet werden können, während der Empfänger nicht verbunden ist. Umgekehrt können Empfänger über Events notifiziert werden, wenn der Sender nicht mehr verbunden ist. Unter der räumlichen Entkopplung ist zu verstehen, dass die Parteien, die miteinander kommunizieren wollen, sich nicht unbedingt gegenseitig kennen. Die Sender publizieren ihre Events über den Broker und kennen weder Empfänger, die sich für dieses Event registriert haben, noch kennen sie deren Anzahl. Die Entkopplung in Bezug auf die Synchronisation meint, dass der Sender nicht blockiert wird, wenn er Nachrichten erstellt und versendet. Ebenso erhält der Empfänger die Nachrichten asynchron. Im Unterschied zum Blackboard- und Tupelraum-Ansatz wird hierbei kein zentraler Speicher verwendet, auf dem die Events abgelegt und abgerufen werden. Anstelle dessen gibt es einen zentralen Verteiler, zu dem die Events geschickt werden und der diese an die verbundenen Komponenten weiterleitet (siehe Abbildung 2.10(c)). Für die Verbindung zwischen den Komponenten und dem Broker werden meist Server-Client-Verbindungen eingesetzt, wobei der Broker als Server fungiert und sich die anderen Komponenten als Client an diesem registrieren.

Da die Events prinzipiell an alle registrierten Komponenten weiter verteilt werden, müssen diese vor der Verarbeitung zur Effizienzsteigerung und Schonung der Ressourcen gefiltert werden. Die Filterung kann dabei anhand des Typs der Events oder nach deren Inhalt gemäß vorgegebener Regeln erfolgen [Baldoni, 2005]. Neben der Möglichkeit, bereits vor der Versendung seitens des Brokers zu filtern, dem *globalen Filter*, können Filter auch lokal in jedem Client integriert werden. Die lokale Variante birgt den Nachteil, dass Informationen zu mehreren Empfängern geleitet und dort unverarbeitet ausgefiltert werden, was zu einer starken Belastung der Netzwerkinfrastruktur führen kann. Dies wirkt sich jedoch positiv auf die Ressourcen des Brokers aus, der die Events ausschließlich verteilen und nicht noch zuvor filtern und zuweisen muss. Bei der typspezifischen Filterung werden alle Events geblockt, für die ein Client sich nicht registriert hat. Wird nach dem Inhalt gefiltert, erfolgt dies anhand von Schlüsselwörtern innerhalb der Informationsblöcke des Events. Da die typbasierten Publish/Subscribe-Systeme eher statisch und primitiv sind, können diese relativ effizient implementiert werden. Inhaltsbasierte Systeme sind hingegen ausdrucksstärker, benötigen jedoch komplexere Protokolle, die den Verarbeitungsaufwand zur Laufzeit erhöhen.

Bei der Informationsübertragung wird zwischen zwei Varianten unterschieden, welche unter den Begriffen „*push*“ und „*pull*“ bekannt sind. Basierend auf der verwendeten Architek-

tur können Events automatisch durch den Broker an die registrierten Clients geschickt werden (push), ohne dass diese die Events explizit anfragen müssen. Einige Publish/Subscribe-Systeme bieten jedoch noch die Möglichkeit Events zwischenzuspeichern. Durch diese Zwischenspeicherung wird ermöglicht, dass die Clients Informationen explizit abfragen können und je nach Realisierung auch müssen (pull), wie beispielsweise bei den ersten beiden Kommunikationsinfrastrukturen. Demnach unterscheidet man die Systeme in der Art, inwieweit Events persistent hinterlegt werden. Bei Systemen, die Events nicht zwischenspeichern, werden die Daten, sobald sie am Broker eingetroffen sind, weiter verteilt, so dass nur die Clients diese erhalten, die sich zuvor am Server registriert haben. Im Gegensatz dazu werden bei Systemen mit persistenter Datenhaltung die Informationen auf der Seite des Brokers zwischengespeichert, so dass Clients zu dem Zeitpunkt, an dem sie sich registrieren, auch diese Daten verarbeiten können.

2.3.4 Komplexe Eventverarbeitungssysteme

Aus dem Publish/Subscribe-Ansatz hat sich unter anderem das Themengebiet der komplexen Eventverarbeitung (Complex Event Processing, CEP) entwickelt. Dabei geht es nicht nur darum, Änderungen zu erfassen und diese weiter zu verteilen, sondern auch darum, Unregelmäßigkeiten oder Verbindungen zwischen Events zu detektieren und diese zur weiteren Verarbeitung zu versenden. Dabei werden die Events anhand einer hierarchischen Ordnung unterteilt. Diejenigen, die aus den Sensoraufnahmen entstehen, sind Events niedriger Ordnung. Werden diese miteinander kombiniert entsprechend der Idee von logischen Sensoren (vgl. Abschnitt 2.1.4), entstehen Events höherer Ordnung. Neben der Erstellung von neuen Events steht bei diesem Ansatz auch die Filterung von Daten im Vordergrund, die anhand vordefinierter Regeln erfolgt. Die Ergebnisse von CEP-Systemen werden einerseits zur Überwachung und Erfassung von Störereignissen verwendet, andererseits aber auch um entsprechende Reaktionen auf diese auszuführen. Daher wird zwischen zwei Formen von Events unterschieden. Bei der ersten Form werden Nachrichten als Informationen zwischen den Komponenten verteilt. Dies dient ausschließlich der Notifikation und Visualisierung. Bei der zweiten Form steht, angelehnt an RPC (siehe Abschnitt 2.3), eine gezielte Ansteuerung einer Aktivität bei dem Empfänger im Vordergrund.

Im Allgemeinen bestehen CEP-Systeme aus einem Netzwerk von Filtern und Analyse-Tools mit der Fähigkeit, mehrere Events zu kombinieren und zusammen auszuwerten [Luckham und Frasca, 1998]. Bei der Verarbeitung können zusätzlich neue Events erzeugt und im Netzwerk weiter propagiert werden. Wichtig dabei ist, dass das gesamte Netzwerk agil ist und sich leicht adaptieren lässt, indem beispielsweise neue Filter oder erweiterte Analyse-Tools integriert werden können. Mittels einer kontinuierlichen Auswertung der erfassten Systemveränderungen können Visualisierungen angeschlossen werden, die eine Überwachung des Systems vereinfachen, indem die Rohdaten bereits vorverarbeitet werden. Dabei müssen aus den Events die enthaltenen Daten extrahiert, miteinander kombiniert und in zeitliche Zusammenhänge gebracht werden. Einsatzgebiete dafür sind unter anderem Prozess Management Systeme [Eckert und Bry, 2009].

2.3.5 Datenstrom Managementsysteme

Neben der Entwicklung von CEP-Systemen hat sich aus dem Publish/Subscribe-Ansatz eine weitere Kommunikationsinfrastruktur entwickelt, die auf bereits etablierte Verarbeitungslogiken aus dem Bereich der Datenbanken aufbaut, den *Datenstrom Managementsystemen (DSMS)*. Zugrundeliegend sind sogenannte Datenbank Managementsysteme (DBMS), welche auf Relationen operieren, die in einer Datenbank hinterlegt sind. Eine Relation besteht dabei aus Attributen und Tupeln, wobei ein Attribut den Typ eines möglichen Attributwertes beschreibt und ein Tupel eine konkrete Kombination von Attributwerten darstellt. Zum Bearbeiten und Abfragen von relationalen Daten gibt es bereits spezifische Sprachen. Bekannt sind beispielsweise relationale Abfragesprachen wie SQL, welche auf den in der Datenbank hinterlegten Relationen operieren. Dabei werden die Anfragen einmalig gestartet, ausgewertet und beantwortet. Agiert wird ausschließlich auf den Daten, die zuvor in der Datenbank hinterlegt wurden. Bei DSMS hingegen wird nicht auf bereits abgelegten Daten operiert, sondern auf Datenströmen, über die kontinuierlich neue Live-Daten zum System geschickt und ohne vorherige Speicherung ausgewertet werden.

Definition 2.10 (Datenstrom) *Ein Datenstrom S ist eine (möglicherweise unendliche) Sammlung (Multiset) von Elementen $\langle s, \tau \rangle$, in der s ein Tupel darstellt, welches zu dem Schema von S gehört und $\tau \in \mathcal{T}$ der Zeitstempel des Elements ist [Arasu et al., 2006].*

Da diese Datenströme möglicherweise unbegrenzt sind, können Operatoren nicht direkt auf diesen agieren. Im Speziellen können keine blockierenden Operatoren angewendet werden. Blockierend bedeutet in diesem Fall, dass kein (Teil-) Ergebnis erstellt werden kann, bevor nicht alle betroffenen Einträge angeschaut wurden. Beispiele hierfür sind die Ermittlung eines Minimalwerts oder einer Anzahl. Aus diesem Grund agieren die Operatoren bei DSMS auf Teilabschnitten, sogenannten *Fenstern*.

Definition 2.11 (Fenster) *Für jede Zeitinstanz definiert ein Fenster in einem Datenstrom ein Set aus Tupeln, über welches die Anfrage ausgeführt wird [Chandrasekaran et al., 2003a].*

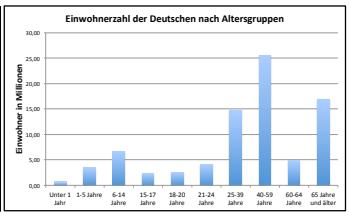
Typische Beispiele solcher Fenster sind das *Landmarkenfenster* und das *verschiebbare Fenster*. Bei dem ersten ist ein älterer Zeitpunkt im Datenstrom als Startpunkt festgelegt und wird nicht geändert. Der Endpunkt bewegt sich mit der Ankunft neuer Tupel mit, so dass das Fenster sich mit der Zeit vergrößert. In diesem Fall werden alle Tupel betrachtet, die vom spezifizierten Zeitpunkt an bis zum aktuellen Augenblick der Verarbeitung im System eingetroffen sind. Beim verschiebbaren Fenster bewegen sich beide Grenzen simultan mit der Ankunft neuer Tupel mit, so dass die Fenstergröße konstant bleibt. Mittels dieser Fenster können Teilabschnitte des Datenstroms in relationale Datenstrukturen überführt werden, auf denen die Operatoren arbeiten können. Aufgrund der Ableitung seitens DSMS von traditionellen DBMS werden auch meist erweiterte SQL-Anfragesprachen für die Verarbeitung der Datenströme verwendet. Eingesetzt werden DSMS beispielsweise zur Überwachung von Netzverkehr und bei Finanzapplikationen [Babcock et al., 2002].

2.4 Visualisierung, Virtualisierung und Simulation

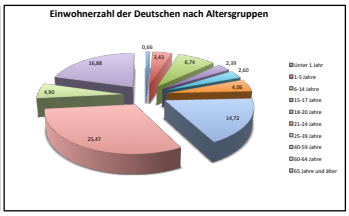
Bei der Visualisierung können mehrere Ansätze unterschieden werden. Beispielsweise kann für die Darstellung von Daten eine schematische, grafische Repräsentation gewählt werden oder eine Repräsentation in Form von Text und Zahlen basierend auf den Fakten. Bei der grafischen Darstellung kann wiederum die Anzahl der Dimensionen in Betracht gezogen werden, sprich ob die Grafik zweidimensional oder dreidimensional dargestellt wird. Während die textuelle Repräsentation oftmals eine schnellere Auffassung der einzelnen Fakten ermöglicht, können durch Grafiken die Beziehungen zwischen den Fakten schneller erfasst werden. Werden 3D-Darstellungen oftmals als schöner empfunden, hat sich in den letzten Jahren gezeigt, dass die Aufnahme der Fakten hierbei nicht besser ist als bei 2D-Darstellungen, sondern diese teilweise sogar verschlechtert. Abbildung 2.12 zeigt Beispiele für die unterschiedlichen Darstellungsvarianten. Als Grundlage hierfür dienen Informationen zur Altersstruktur in Deutschland aus dem Jahre 2011[3].

Einwohnerzahl der Deutschen nach Altersgruppen	
Unter 1 Jahr	0,66
1-5 Jahre	3,43
6-14 Jahre	6,74
15-17 Jahre	2,39
18-20 Jahre	2,60
21-24 Jahre	4,06
25-39 Jahre	14,72
40-59 Jahre	25,47
60-64 Jahre	4,90
65 Jahre und älter	16,88

(a) Textuelle Darstellung



(b) 2D Balkenchart



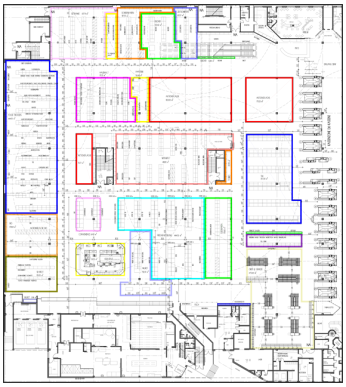
(c) 3D Tortendiagramm

***Abbildung 2.12**: Mögliche Darstellungsvarianten für Informationen*

Für gewisse Aufgaben hingegen bieten 3D-Darstellungen Vorteile, was in Studien belegt worden ist [Risden et al., 2000]. Unter anderem wurde gezeigt, dass die Suche in einer kombinierten Darstellung von 3D- und 2D-Listen das Auffinden von Suchbegriffen vereinfacht. Dabei bestanden die Suchbegriffe aus Kategorien, deren hierarchische Struktur durch eine dreidimensionale Darstellung repräsentiert wurde. Verglichen wurde diese Suchfunktion mit einer Dateiansicht in Form einer Ordnerstruktur und einer Internetseite, die alle Elemente der aktuell ausgewählten Kategorie sowie dem Pfad bis zu dieser anzeigte. Das Erstellen und Hinzufügen hingegen wurde von den Teilnehmern der Studie langsamer in der 3D-Darstellung als in 2D realisiert. Wie die Ergebnisse der Studie zeigen, muss je nach Aufgabe und Einsatz die entsprechende Repräsentationsart gewählt werden, um ein möglichst optimales Ergebnis zu erhalten.

Für die visuelle Darstellung von physischen Umgebungen müssen diese virtualisiert werden. Daraus entstehen Modelle, die als virtuelle Welten dargestellt werden können. Neben der rein reellen und der rein virtuellen Welt gibt es auch Mischformen, die unter dem Begriff „gemischte Realitäten“ zusammengefasst werden. Für eine dreidimensionale Darstellung von Modellen müssen die entsprechenden Szenen auf dem System generiert werden,

[3]Quelle: http://de.statista.com/statistik/daten/studie/1365/umfrage/bevoelkerung-deutschlands-nach-altersgruppen/, Zugriff: Oktober 2013



(a) Grundrissplan eines Supermarkts

(b) Dreidimensionale Darstellung einer Supermarktumgebung

***Abbildung 2.13**: 2D- und 3D-Darstellungen von Umgebungen*

was unter dem Begriff „Rendern“ bekannt ist. Für eine 3D-Darstellung unabhängig vom System, auf dem die Anzeige erfolgen soll, sind Internet-Browser prädestiniert, da diese bereits standardmäßig auf den Systemen installiert sind (siehe Abschnitt 2.4.2). Durch bereitgestellte Schnittstellen zur Hardware können die 3D-Inhalte direkt im Browser gerendert und angezeigt werden. Daneben gibt es auch spezialisierte Applikationen, die auf Modelle, die im Internet hinterlegt sind, zurückgreifen und diese visualisieren. Eine Virtualisierung von Cyber-Physischen Umgebungen hat neben der visuellen Darstellung zusätzlich den Vorteil, dass realitätsnahe Simulationen darin durchgeführt werden können (siehe Abschnitt 2.4.3). Solche Simulationen können herangezogen werden, um Prognosen und Vorhersagen zu erstellen.

2.4.1 Reale und virtuelle Welt

Bereits in Abschnitt 2.2 wurde auf die Thematik von instrumentierten Umgebungen eingegangen. Gegenüber der realen Welt gibt es die *Virtualität*, in der anstatt physischer Objekte virtuelle Objekte auf Displays dargestellt werden. Reale Objekte haben dabei eine aktuelle physische Existenz, während virtuelle Objekte nur als Wesen oder Effekt existieren, jedoch weder förmlich noch tatsächlich [Milgram und Kishino, 1994]. Wie auch Daten können Umgebungen in 2D oder in 3D dargestellt werden. Vor allem in der Spieleindustrie sind animierte 3D-Welten weit verbreitet, um den Benutzern ein möglichst realistisches Spieleempfinden zu bieten. Ebenso können mittels 3D-Darstellungen physische Gegebenheiten einfacher präsentiert werden, um dem Betrachter eine Impression der räumlichen Gegebenheiten zu vermitteln. Bei der Erstellung von Grundrissplänen wird hingegen aufgrund der besseren Übersichtlichkeit und Modellierung eine 2D-Ansicht verwendet (siehe Abbildung 2.13).

Erweiterte Realität

Neben der realen Welt und rein virtuellen Welt gibt es noch weitere Mischformen, die Teile aus beiden vereinen, wie in Abbildung 2.14 illustriert ist.

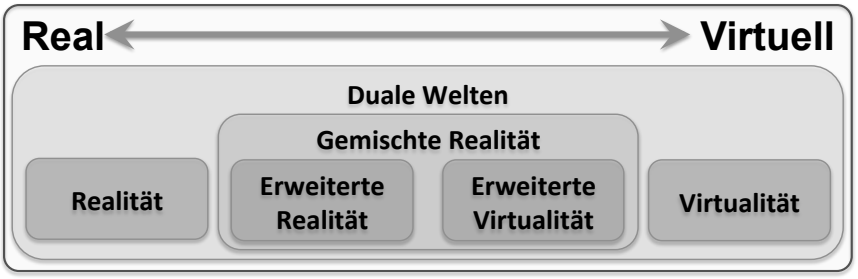


***Abbildung 2.14**: Einteilung der verschiedenen Welten und deren Verschmelzungen*

Erweiterte Realität beschreibt dabei eine physische Umgebung, welche durch virtuelle Objekte ergänzt wird [Milgram und Kishino, 1994]. Dies erfolgt beispielsweise durch tragbare Videobrillen, die als *Head Mounted Displays* (HMD) bezeichnet werden. Ein solches HMD ist beispielsweise die Google Glass[4]. Neben der realen Welt können durch die Brille vereinzelt virtuelle Objekte dargestellt werden. Ein anderes Beispiel sind visuelle Marker, die durch die Kamera eines Handys mit entsprechender Applikation betrachtet, ein 3D-Objekt auf dem Display darstellen. Unter anderem bietet das sogenannte ARToolkit eine solche Funktionalität an [Kato und Billinghurst, 1999].

Erweiterte Virtualität

Im Gegensatz zur erweiterten Realität wird bei der erweiterten Virtualität die im Mittelpunkt stehende virtuelle Darstellung um reale, physische Objekte erweitert. Dies kann beispielsweise ebenfalls mittels HMDs erfolgen, indem durch diese eine vollständig modellierte Welt zu sehen ist, aber die Hände des Benutzers aus dem Live-Kamerabild in diese Szene ergänzt werden. Eine weitere Alternative ist die Verwendung von sogenannten Cave-Umgebungen, bei denen sich der Benutzer inmitten mehrerer Displays befindet, die eine virtuelle Welt darstellen [Cruz-Neira et al., 1993]. Der Benutzer selbst und mögliche weitere Gegenstände liegen als physische Objekte vor, mit denen interagiert werden kann. Dies kann eingesetzt werden, um Tests in möglichst realistischen Umgebungen durchführen zu können, die in Realität nur sehr schwer und mit hohen Kosten und Aufwand erstellt werden könnten [Lok, 2004]. Ein Beispiel dieser Verwendung wird von Hühn et al. gegeben, die eine Cave-Umgebung verwenden, um einen virtuellen Supermarkt darzustellen. Ziel dabei ist es, lokationsbasierte Informationssysteme zu testen und zu evaluieren [Hühn et al., 2012]. Eine dritte Variante für den Einsatz von erweiterter Virtualität stellen Simulatoren dar, bei denen sich die Benutzer in einer virtuellen Umgebung bewegen, indem sie mit realen Objekten interagieren. Zum Beispiel wird dies bei der Ausbildung von Piloten eingesetzt, die in einem realen bzw. einem realgetreuen Abbild eines Flugzeug-Cockpits operieren. Dabei sind entweder die Scheiben durch Monitore ersetzt oder das Cockpit selbst ist in einer Cave-Umgebung positioniert. Somit befindet sich der auszubildende Pilot selber in der virtuellen Umgebung, die er durch die Interaktion mit den physischen Armaturen realistisch beeinflussen kann. Abbildung 2.15 zeigt ein Bild eines solchen Flugzeugsimulators. Die

[4] http://www.google.com/glass/start/, Zugriff: November 2014

Kombination von realer und virtueller Welt kann man auch unter dem Begriff *gemischte Realität* zusammenfassen. Dies umfasst jegliche Zusammensetzung aus realen und virtuellen Objekten.

***Abbildung 2.15**: Beispiel eines Flugzeugsimulators*[5]

Dual Reality

Wie bereits in Definition 1.2 beschrieben geht es bei dem Dual Reality (DR) Paradigma darum, dass Realität und Virtualität miteinander verschmelzen, indem durch eine geeignete Middleware Informationen zwischen beiden Welten ausgetauscht werden. Diese Informationen können einen Einfluss auf die jeweilige Welt haben und diese somit verändern. Ursprünglich wurde Dual Reality als Informationsmedium angesehen, indem natürliches Benutzerverhalten und -interaktionen in der physischen Umgebung durch Sensoren erfasst und automatisch als virtueller Kontext repräsentiert werden [Lifton, 2007]. Beispielsweise diente ein virtueller Avatar dazu, den aktuellen Aktivitätslevel in einem Gebäudekomplex zu repräsentieren. Je mehr Aktivität durch die Sensoren erfasst wurde, desto mehr wurde der Avatar basierend auf einem Morphismus in eine außerirdische Phantasiefigur verwandelt. Abbildung 2.16 zeigt einige Situationen dieser Verwandlung vom Avataren mit wenig Aktivität (links) bis hin zum Außerirdischen bei hohem Aktivitätslevel (rechts).

***Abbildung 2.16**: Darstellung des Aktivitätslevels mittels Morphismus eines Avataren (Quelle: [Lifton, 2007])*

[5] Quelle: http://www.radiohamburg.de:8080/Nachrichten/Hamburg-aktuell/Panorama/2011/Oktober/Abheben-wie-im-richtigen-Flugzeug-Interaktiver-Flugzeugsimulator-bald-in-Hamburg, Zugriff November 2014

Weiterhin wird DR als Möglichkeit verstanden, die Gleichstellung der physischen und der virtuellen Welt zu erreichen. In diesem Zusammenhang ist der Begriff Dual Reality auch unter *Cross Reality* (X-Reality) oder *gespiegelte Welten* (Mirrored Worlds) bekannt [Kimber et al., 2012]. Stahl et al. haben den Begriff DR in der Art erweitert, dass mehrere Welten untereinander synchronisiert werden und dies als *Synchronisierte Realitäten* definiert [Stahl et al., 2011]. In diesem Fall können beispielsweise mehrere virtuelle Welten miteinander synchronisiert werden, wie es bei Online Spielen der Fall ist. Ebenso besteht die Möglichkeit, reale mit virtueller Welt oder mehrere räumlich voneinander getrennte Umgebungen zu harmonisieren, so dass diese miteinander gepaart sind. So können TV-Geräte mehrerer Räume untereinander synchronisiert werden, um eine soziale Verbindung zwischen den Räumen und den darin befindlichen Personen Umgebungen und deren Einwohner herzustellen.

2.4.2 3D im Web

Für die computergestützte Darstellung von dreidimensionalen Inhalten ist eine Schnittstelle zur Grafikhardware notwendig, um die vordefinierten Szenen zu erzeugen, was als Rendern bezeichnet wird. Eine Software, die eine solche Schnittstelle aufbaut, ist die „Offene Grafikbibliothek“ OpenGL (Open Graphics Library) [Shreiner und The Khronos OpenGL ARB Working Group, 2009]. Dabei ist OpenGL unabhängig von der zugrundeliegenden Hardware und Softwareplattform und daher in den gängigen Betriebssystemen standardmäßig integriert. Typische Einsatzmöglichkeiten sind in 3D-Spielen, Darstellungen von virtuellen Welten oder Modellierungsprogrammen, wie CAD, zu sehen. Zusätzlich wird eine spezialisierte Version, OpenGL ES (Open Graphics Library for Embedded Systems), angeboten, die auch auf mobilen Endgeräten eingesetzt werden kann, um dort das Darstellen von 3D Szenen zu ermöglichen. Basierend auf dieser Spezialisierung wurde WebGL (Web Graphics Library) entwickelt, welches bereits ein fester Bestandteil einiger Webbrowser, wie Google Chrome und Mozilla Firefox, ist. Bei anderen Browsern, wie Safari, lässt sich WebGL nachträglich aktivieren. Durch die Spezialisierung für die Verwendung in Browsern bietet WebGL Schnittstellen für JavaScript an, wodurch standardmäßige Interaktionsschemata für Webseiten weiter verwendet werden können. Dies wurde unter anderem dazu verwendet, um die Integration und Erstellung von 3D-Szenen zu unterstützen [Leung und Salga, 2010; DeLillo, 2010].

Für das Rendern der 3D-Inhalte müssen die Szenen zur Darstellung in einem gewissen Datenformat beschrieben werden. Ein weit verbreitetes Format hierfür ist VRML (Virtual Reality Modeling Language), aus welcher die 3D Darstellung in Echtzeit generiert wird. Viele Modellierungswerkzeuge bieten die Möglichkeit, VRML-Dateien sowohl zu importieren als auch zu exportieren, wodurch sich dieses Format zum Austausch von 3D-Modellen etabliert hat. Für die Darstellung von VRML-Dateien im Browser werden jedoch spezielle Plugins benötigt, die zuvor installiert werden müssen. Der offizielle Nachfolger von VRML als Beschreibungssprache für 3D-Modelle ist X3D (Extensible 3D), welcher seit 2004 als offizieller ISO-Standard für 3D-Inhalte im Web spezifiziert ist. Jedoch benötigt man zur Darstellung wie auch beim Vorgänger spezielle Plugins bzw. browserunabhängige Programme.

XML3D

Parallel zu den Erweiterungen von X3D laufen Bestrebungen, 3D-Inhalten auf Internetseiten darzustellen, ohne zuvor ein Plugin installieren zu müssen. Ein Beispiel hierfür ist XML3D, welches eine Beschreibungssprache für 3D-Webinhalte ist [Sons et al., 2010]. Diese Sprache basiert auf der Kombination von JavaScript und WebGL, so dass die 3D Szenen in Browsern, welche WebGL unterstützen, gerendert werden können. Dabei wird die Beschreibung direkt in das Dokument der Webseite, das Document Object Model (DOM), integriert und kann daher mittels standardmäßigen JavaScript-Funktionen und CSS-Dokumenten (Cascading Style Sheets) manipuliert werden. Wie auch bei X3D wird für die Beschreibung das XML-Format verwendet, um konform mit dem Aufbau von Internetseiten zu bleiben. Somit ist es möglich, sowohl 2D- als auch 3D-Inhalte auf der gleichen Internetseite zu visualisieren und mit beiden Darstellungsformen in derselben Form zu interagieren.

Second Life

Neben der Darstellung von 3D-Webinhalten in Browsern besteht die Möglichkeit, diese in spezialisierten Programmen zu visualisieren. Dies wird unter anderem in der Spieleindustrie eingesetzt, bei der sich die Benutzer auf einem Server im Internet anmelden und sich anschließend durch die Synchronisation zwischen allen Spielern gemeinsam durch die 3D-Umgebung bewegen. Eine weitere Alternative für die Darstellung von interaktiven 3D-Szenen ist *Second Life*[6]. Neben der Visualisierung bereits modellierter Umgebungen bietet Second Life die Möglichkeit, neue Umgebungen zu erstellen. Dazu kann der Nutzer komplexe Modelle unter Verwendung von zur Verfügung gestellten Primitiven, wie Boxen, Kuglen und Kegeln, mittels eines Editors zusammenstellen. Der Benutzer selbst kann sich als Avatar durch die Umgebung, die in Form von Inseln aufgebaut ist, navigieren. Der Hauptfokus von Second Life liegt dabei mehr auf der sozialen Komponente, einen virtuellen Treffpunkt zu realisieren, als darauf ein Spiel darzustellen. Um sich oder seine Firma zu repräsentieren, kann man eine gewisse Fläche kaufen, die man dann selbstständig mit Hilfe des Editors modellieren kann. Zusätzlich beinhaltet Second Life auch eine Physik-Engine, mit der Objekte mit realistischem Verhalten animiert werden können.

Um von extern Einfluss auf diese Welt zu haben, gibt es eine Schnittstelle, welche über die Linden Scripting Language (LSL) angesprochen werden kann. Damit können Objekte referenziert und interaktives bzw. intelligentes Verhalten angehängt werden. Ebenso besteht damit die Möglichkeit, Informationen aus Second Life, wie beispielsweise Nutzerinteraktionen, externen Applikationen zur Verfügung zu stellen. Für die Verwendung von Second Life ist eine Internetverbindung notwendig. Um auch Offline die gleiche Funktionalität nutzen zu können und ohne zur Generierung einer eigenen Umgebung Land kaufen zu müssen, wurde die Alternative *OpenSimulator*[7] entwickelt. Diese hat sich dem Funktionsumfang von Second Life angepasst und unterstützt ebenfalls LSL als Kommunikationsprotokoll. Der Server, zu

[6] www.secondlife.com, Zugriff: November 2014
[7] www.opensimulator.org, Zugriff: November 2014

dem man sich verbindet, kann eigenständig verwaltet werden, womit man unabhängig von einer Internetverbindung ist.

2.4.3 Simulation und Vorhersagen

In der Regel sind Installationen von neuen Sensoren und Aktuatoren in der realen Welt mit viel Aufwand und oftmals erheblichen Kosten verbunden. Aus diesem Grund werden Simulationen verwendet, um die Auswirkung dieser Komponenten bereits im Vorfeld virtuell zu testen und Antworten auf die Frage „was wäre wenn...?“ zu geben. Zudem können sie dazu herangezogen werden, potentielle Ergebnisse zu präsentieren. So können Hardware-Komponenten simuliert und Sensordatenerfassung und -verarbeitung emuliert werden [Bruneau und Consel, 2012]. Bei einer vollständigen Simulation wird jeder mögliche erreichbare Zustand getestet. Hierdurch kann die uneingeschränkte Funktionalität des zu untersuchenden Sachverhalts unter den modellierten Bedingungen festgestellt werden. Dies ist in den meisten Fällen nicht möglich, da der zu untersuchende Zustandsraum beliebig groß und teilweise nicht endlich ist. In diesem Fall können mittels einer partiellen Simulation möglichst viele Zustände, die bestenfalls nicht triviale Sachverhalte integrieren, geprüft werden. Neben der Vielfalt der möglichen Zustände spielt auch die Genauigkeit der Modellierung und Datenqualität bei Simulationen eine wichtige Rolle. Je nachdem, was simuliert werden soll, ist eine exakte Modellierung nicht möglich. Nimmt man die Erfassung eines Bluetooth-Signals als Beispiel, benötigt man ein sehr komplexes Modell des Senders, Empfängers und der gesamten Umgebung. Durch äußere Einflüsse, wie Luftfeuchtigkeit oder Metall, werden die Bluetooth-Signale beeinflusst, was für die Modellierung einen zu komplexen Sachverhalt darstellt. Aus diesem Grund werden in solchen Fällen einfachere Modelle angenommen, z.B. eine eindeutige Signalerfassung in einem gewissen Umkreis um den Sender. Hierdurch werden die Daten ungenauer und führen zu einer nicht einhundertprozentig realistischen Simulation.

Simulationen können in vier Komponenten unterteilt werden: *Interaktion*, *Verarbeitung*, *geometrisches Modell* und *Präsentation* [Shaw et al., 1992]. Abhängig davon, was genau getestet werden soll und welche Daten zur Verarbeitung herangezogen werden sollen, kann jede dieser Komponenten simuliert bzw. emuliert werden. Beispielsweise können virtuelle Charaktere dazu verwendet werden, ein Positionierungssystem zu testen. Dabei bewegen sich die Avatare im virtuellen Raum und simulieren somit die Interaktion mit dem Positionierungssystem [Brandherm et al., 2008a]. Meist steht die Verarbeitungslogik, wie im vorherigen Beispiel die Positionsermittlung, selbst im Vordergrund und soll mit Hilfe der Simulation getestet werden. Diese hängt aber oftmals von weiteren Faktoren ab, wie beispielsweise der Signalerkennung. Diese kann wie bereits oben beschrieben durch einfache Modelle simuliert werden. Auch geometrische Modelle können vereinfacht werden. Beispielsweise reicht es oftmals aus, dass Objekte über ihr Begrenzungsrechteck beschrieben werden. Dies vereinfacht die Erstellung von Simulationen, indem zuvor keine komplexen Modelle erstellt werden müssen. Schließlich ist ein wichtiger Bestandteil von Simulationen die Darstellung. Hierbei kann man wiederum zwischen einer rein virtuellen und einer physischen Darstellung unterscheiden. Bei der virtuellen Variante werden die Simulationsergebnisse im virtuellen Modell dargestellt, wie beispielsweise in Second

Life [Ullrich et al., 2008]. Daneben können die Simulationsergebnisse in realen Modellen gezeigt werden, wie beispielsweise in Nachbildungen von Umgebungen aus Holz oder Lego [Armac und Retkowitz, 2007]. Wie bereits zuvor beschrieben werden Simulationen für die Ausbildung von Flugzeugpiloten eingesetzt. Auch im Automobilbereich gibt es ähnliche Ansätze, um Fahrverhalten und neue Systeme zu testen [Medenica et al., 2011; Kim und Dey, 2009].

Neben dem Testen der richtigen Funktionsweise werden Simulationen eingesetzt, um Vorhersagen treffen zu können. Diese werden zur Unterstützung bei anstehenden Entscheidungen herangezogen. Dabei werden mögliche zukünftige Veränderungen basierend auf aktuellen und historischen Informationen ermittelt. Ein prominentes Beispiel hierfür ist die Wettervorhersage, die basierend auf komplexen Wettermodellen, aktuellen Messdaten und Erfahrungswerten mögliche zukünftige Ereignisse ermittelt und diese als Wetterprognosen bekannt gibt [Zhu et al., 2002]. Dabei werden komplexe mathematische Modelle und Berechnungen herangezogen, um Simulationen durchzuführen, bei denen mehrere Alternativen mit leicht unterschiedlichen Konditionen getestet werden. Da die Simulation aufgrund der Komplexität nicht vollständig und exakt modelliert werden kann, sind die Ergebnisse mit gewissen Fehlern und Unsicherheiten behaftet. Diese werden mittels prozentualer Angaben, z.B. bei der Regenwahrscheinlichkeit, angegeben. Aufgrund besserer und genauerer Modelle sowie gesteigerter Performance bei der Simulation durch schnellere Hardware, sind die Simulationen und damit auch Prognosen in den vergangenen Jahren stetig verbessert worden. Dies stellt einerseits die Wichtigkeit der Genauigkeit bei der Modellierung und Datenverwendung dar und zeigt andererseits den Einfluss der potentiellen Rechenleistung auf. Steht mehr Rechenleistung zur Verfügung, können mehr potentielle Zustände simuliert, analysiert und in die Bewertung mit einbezogen werden.

2.5 Monitoring und Steuerung von Smarten Umgebungen

Zusätzlich zu Simulationen zum Testen von bestimmten Ereignissen ist bei Smarten Umgebungen wichtig, deren Funktionsfähigkeit zu überwachen und den Ablauf zu monitoren (siehe Abschnitt 2.5.1). Dazu können die in der Umgebung verbauten Sensorsysteme verwendet werden. Die erzielten Ergebnisse müssen entsprechend aufbereitet und einem Handlungsweisenden zur Verfügung gestellt werden, damit dieser sie als Handlungs- und Entscheidungsunterstützung heranziehen kann. Beispielsweise können über Sensornetzwerke große Menschenansammlungen erfasst und kommuniziert werden (siehe Abschnitt 2.5.2). Durch eine intelligente Verarbeitung und a-priori-Wissen können hierbei Ungenauigkeiten von Sensoren und Messungen kompensiert werden, wodurch ein Einsatz kostengünstiger Sensoren ermöglicht wird. Aufgrund der großen Anzahl von Sensoren und Messungen müssen die erfassten Daten aufbereitet und intuitiv verständlich repräsentiert werden. Hierzu werden oftmals sogenannte *Business Intelligence* (siehe Abschnitt 2.5.3) Systeme und *Dashboards* (siehe Abschnitt 2.5.4) verwendet. Weitere Monitoring- und Steuerungsumgebungen stellen für den Anwendungszweck spezialisierte *Leitstände* dar (siehe Abschnitt 2.5.5). Diese werden beispielsweise eingesetzt, um aus einer zentralen Stelle den Verkehrsfluss von Zügen zu kontrollieren und bei unerwarteten Störungen aktiv eingreifen zu können.

Zudem existieren bereits Planungstools, die den Benutzer bei der Konzeption von Smarten Umgebungen durch Hinweise unterstützen und dadurch dessen Arbeit erleichtern (siehe Abschnitt 2.5.6). Durch eine Verbindung dieser Planungstools mit dem Sensor-Aktuator-Netzwerk der Umgebung könnten diese zu Monitoring- und Steuerungssystemen erweitert werden.

2.5.1 Monitoring von Smarten Umgebungen

Neben dem Analysieren von Daten ist das Überwachen von Smarten Umgebungen eine wichtige Aufgabe, um die korrekte Funktionsweise gewährleisten zu können, was unter Monitoring bekannt ist. Dies ist einerseits für die Mensch-Umgebungs-Interaktion relevant, aber auch für die Maschine- bzw. Roboter-Umgebungs-Interaktion. So können beispielsweise virtuelle „Gucklöcher“ verwendet werden, um einen Einblick in die Umgebung zu ermöglichen [Butz und Krüger, 2006]. Diese sogenannte „Peephole Metapher“ kann mit Hilfe von Kamerasystemen realisiert werden, welche die Informationen in der Umgebung in Form von Bildern aufnehmen, die auf einem beliebigen Bildschirm angezeigt werden können. Beim Monitoring wird auf die verbaute Sensor-Infrastruktur zurückgegriffen, die eine Repräsentation der Umgebung ermöglicht. Unter anderem können solche Informationen für kollaborative Wartungsarbeiten an Maschinen eingesetzt werden. In diesem Fall werden die erfassten Sensordaten, wie beispielsweise ein Videobild, einem Experten zur Verfügung gestellt, welcher sich nicht vor Ort befindet. Dieser kann die Daten analysieren und mögliche Fehlerquellen ermitteln, die er einem Arbeiter vor Ort mitteilen kann. Neben dem Videobild können weitere Informationen visualisiert werden, wie beispielsweise ein 3D-Modell der zu reparierenden Anlage [Grün et al., 2012]. Diese Repräsentation kann auch als Kommunikationsunterstützung zwischen dem Experten und dem Arbeiter vor Ort verwendet werden, indem Geräteteile in diesem selektiert und somit dem anderen mittels Synchronisation der Darstellungen präsentiert werden.

Bei einer Roboter-Umgebungs-Interaktion können die Sensorinformationen der Umgebung dazu verwendet werden, das Weltmodell des Roboters den aktuellen Umständen, dem Kontext, anzupassen. Dabei werden die Sensorinformationen vom Roboter ausgewertet und das Modell seiner Umgebung angepasst [Fichtner et al., 2003]. Beispielsweise können Sensoren an Türen Aufschluss darüber geben, ob diese geöffnet oder geschlossen sind, so dass der Roboter dies bei seiner Wegplanung mit berücksichtigen kann. Je genauer die Sensorinformationen sind, desto exakter kann die Planung der Aufgabe erfolgen. Werden nicht alle Änderungen in der Umgebung erfasst oder sind die Erfassungen mit gewissen Fehleranfälligkeiten behaftet, kann es bei der Durchführung des Plans zu Komplikationen kommen, die in einer kostenintensiven Umplanung resultieren können. Neben der gesamten Umgebung können auch gewisse Parameter oder auch einzelne Komponenten kontrolliert werden, wie beispielsweise Performanz, Temperaturentwicklung und Geschwindigkeit. Dadurch können außergewöhnliche Ereignisse erfasst und proaktiv entsprechenden Entscheidungsträgern präsentiert werden [Chan et al., 2004], damit diese gegebenenfalls geeignete Gegenmaßnahmen einleiten können.

Für das Monitoring von Umgebungen werden die Änderungen als Events erfasst und kommuniziert. Somit umschreibt Monitoring die Erkennung, Generierung, Verarbeitung, Verteilung, Filterung und Visualisierung von Events, welche Statusveränderungen der Umgebung in möglichst Echtzeit ermitteln [Mansouri-Samani, 1995]. Gerade in verteilten Systemen bieten Monitoring Systeme die Möglichkeit, fehlerhafte Prozesse oder Verarbeitungsschritte möglichst frühzeitig zu erkennen, um gegen diese entsprechende Reaktionen durchführen und auf mögliche Alternativen zurückgreifen zu können. In diesem Fall verfügen die übermittelten Events über einen Status Report Charakter. Wichtig dabei ist jedoch, dass die Sensoren zur Überwachung den eigentlichen operationalen Fluss nicht negativ beeinflussen.

2.5.2 Erfassung von Menschenansammlungen

In Großstädten werden zunehmend Systeme eingesetzt, welche Probleme frühzeitig erkennen und entsprechend darauf reagieren können. Ein Schwerpunkt bei der Überwachung von Großstädten und Festivitäten ist die Erfassung von Ansammlungen großer Menschenmengen. Dies wird zum einen dazu verwendet, um Menschen darüber informieren zu können, diese Bereiche zu umgehen und zum anderen, um gezielt Einsatzpersonal (z.B. von Polizei und Feuerwehr) einteilen zu können. Aufgrund der Gegebenheit, dass solche Menschenansammlungen nur bei gewissen Ereignissen eintreten, ist eine ständige Überwachung der Gebiete nicht notwendig und würde nur unnötige Kosten verursachen. Eine kurzfristige Instrumentierung hingegen, zum Beispiel mittels Kamerasystemen, kann sehr kostenintensiv sein oder auch die Privatsphäre der Menschen in dieser Umgebung verletzen. Aus diesem Grund gibt es bereits Ansätze, welche kostengünstige Sensoren zur Datenerfassung verwenden, die jedoch keine exakte Erfassung gewährleisten. Die erfassten Daten können anhand statistischer Methoden hochgerechnet werden und somit ein annähernd reales Ergebnis abbilden. Ein Beispiel hierfür ist die Erfassung von Smartphones mit aktiviertem Bluetooth Modul [Weppner und Lukowicz, 2013]. Anhand dieser Informationen und statistischer Daten, wie beispielsweise, dass nur circa jede zehnte Person Bluetooth aktiviert hat, kann eine Hochrechnung ermittelt werden. Zusätzlich fließen Daten über die erwartete Altersklasse mit ein, da junge Leute eher ein Smartphone besitzen als ältere. Des Weiteren werden die geometrischen Gegebenheiten berücksichtigt, was die Reichweite der Sensoren mit einschließt, die in Abhängigkeit von der erfassten Personenzahl aufgrund von Interferenzen variiert. Alternativ zu fest installierten Sensoren können auch mobile Sensoren verwendet werden. Hierbei wird deren Position bei jedem Scan der Analyse mit berücksichtigt. Anhand dieser Daten können anschließend die Personenzahlen im erfassten Bereich hochgerechnet werden, wobei eine Genauigkeit von bis zu 75 Prozent erzielt werden kann, was Ergebnisse mehrerer Tests belegen [Weppner und Lukowicz, 2013].

2.5.3 Business Intelligence

Neben den aus der Umgebung erfassten Sensordaten müssen weitere Informationen aus externen Quellen zur Auswertung mit berücksichtigt werden. Speziell um wichtige Entscheidungen in Unternehmen treffen zu können, müssen Daten aus verschiedenen Quellen herangezogen und analysiert werden. Um diese Tätigkeit zu unterstützen, werden *Business Intelligence (BI)* Systeme eingesetzt, welche eine spezialisierte Form von Monitoring-

Komponenten darstellen [Negash und Gray, 2008]. Der Begriff BI wurde im Jahre 1989 von Gartner kreiert und beschreibt datengetriebene *Entscheidungs-Unterstützungs-Systeme* (Decision Supporting System , DSS). Diese verwenden sowohl geschäftsinterne als auch externe Informationen, um hieraus Schlüsse zu ziehen und diese Entscheidungsträgern im Unternehmen als Unterstützung zur Verfügung zu stellen. Dabei werden neben strukturierten Daten, wie beispielsweise aus einer Datenbank, unstrukturierte Informationen, wie beispielsweise E-Mails, herangezogen, um Anomalien und Ausnahmen zu erkennen.

Definition 2.12 (Business Intelligence Systeme) *Business Intelligence Systeme vereinen die Sammlung von Daten, Datenspeicherung und Wissensmanagement mit analytischen Tools, um komplexe betriebsinterne und konkurrierende Informationen Planern und Entscheidungsträgern zu präsentieren [Negash, 2004].*

Eine wichtige Aufgabe von BI Systemen ist es, die unterschiedlichen Datenquellen möglichst in Echtzeit zu erfassen, zu kombinieren und zu vergleichen. Die Resultate können anschließend gepaart mit menschlicher Analyse zu Wissen transferiert werden. Um auch Nichtspezialisten einen Mehrwert zu bringen, müssen die Systeme benutzerfreundliche Oberflächen haben, so dass Berichte leicht erstellt und von den Benutzern verstanden werden können. Eingesetzt werden sie unter anderem dazu, Vorhersagen basierend auf historischen Daten und aktueller Arbeitsleistung zu erstellen, sowie ad hoc Antworten auf spezifische Anfragen und strategische Einblicke zu gewähren [Negash, 2004].

2.5.4 Dashboard

Für die Darstellung der Resultate von BI Systemen werden unter anderem sogenannte *Dashboards* verwendet. Diese stellen ein Medium zur Kommunikation und Darstellung von Informationen dar. Im Business-Sektor werden sie bereits als ein wirksames Hilfsmittel angesehen, um Informationen verständlich auf einem Bildschirm darzustellen und daher zur Repräsentation von Berichterstattung und Analysen eingesetzt [Few, 2006]. Dabei muss die Darstellung so gewählt werden, dass sie leicht verständlich und der Inhalt schnell erfassbar ist. Somit spielt das Design und die Adaption auf die jeweilige Anforderung eine wichtige Rolle. Nach Fews Meinung zielen viele Firmen eher auf die äußere Wirkung der Dashboards ab, um zu erreichen, mit viel Farben und Animationen diese interessanter aussehen zu lassen. Darunter wiederum leidet meistens die Verständlichkeit und somit auch der unterstützende Nutzen. Ein Einsatzgebiet ist die Darstellung der Erreichung vordefinierter Ziele gemäß vorgegebener Kennzahlen, den sogenannten *KPIs* (Key Performance Indicators). Neben der geeigneten Wahl der Darstellung müssen auch entsprechende Updateraten spezifiziert werden, in denen die Dashboards sich aktualisieren, um einerseits nicht zu selten wichtige neue Informationen zu präsentieren und andererseits die Rechenleistung nicht unnötig auszulasten, da oftmals große Datenmengen analysiert und aggregiert werden müssen.

Definition 2.13 (Dashboard) *Ein Dashboard ist eine visuelle Darstellung der wichtigsten Informationen, die zur Erreichung von einem oder mehrerer Ziele benötigt werden; konsolidiert und angeordnet auf einem einzigen Bildschirm, so dass die Informationen im Überblick überwacht werden können [Few, 2007].*

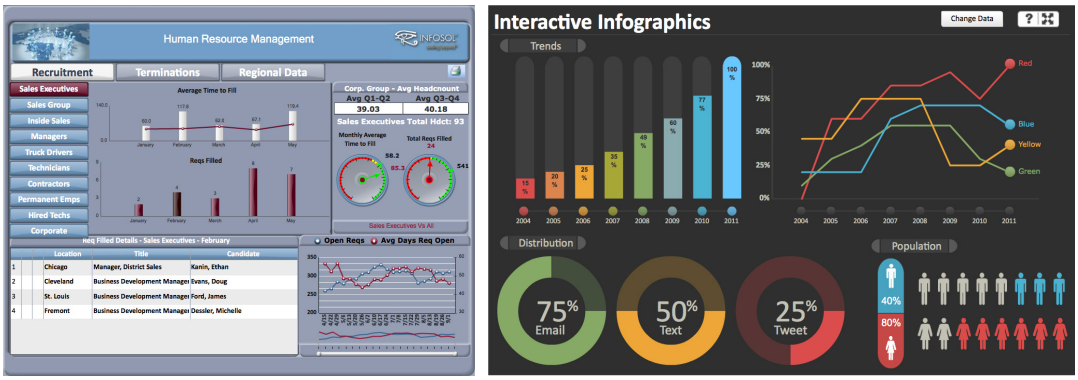


***Abbildung 2.17**: Darstellung zweier Dashboards mit mehreren Darstellungskomponenten*[8]

Beispieldarstellungen für primitive Dashboards sind unter anderem Ampeln, Tachometer, Geschwindigkeitsmesser und Diagramme. Abbildung 2.17 zeigt zwei Dashboards mit mehreren der zuvor erwähnten Darstellungskomponenten. Im Allgemeinen sammeln Dashboards auf intelligente Weise Daten, um diese in einer Art anzuzeigen, dass Verantwortliche durch eine einfache Repräsentation Anomalien, mögliche Missstände und potentielle Trends erkennen und proaktiv gegen diese vorgehen zu können. Im optimalen Fall kann dies mit Hilfe interaktiver Dashboards unterstützt werden. Neben der Darstellungsmöglichkeit von quantifizierbaren Daten können Dashboards auch verwendet werden, um nicht zählbare Informationen zu präsentieren, wie beispielsweise Listen der bestverdienensten Personen, der größten Städte oder der ertragreichsten Filialen.

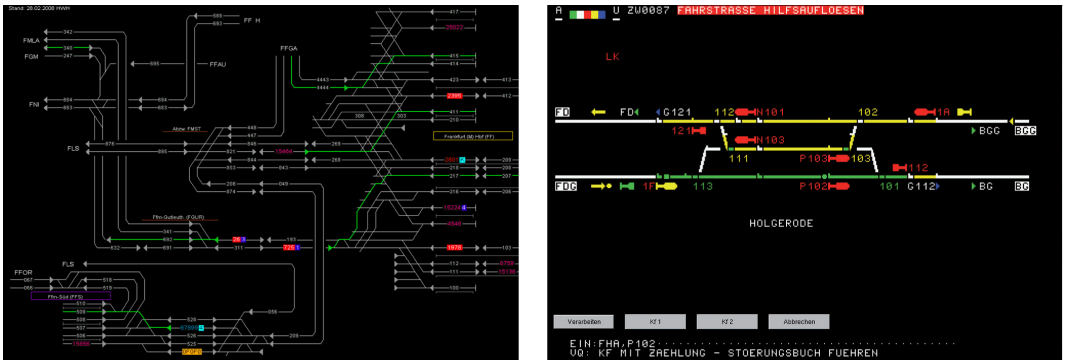
2.5.5 Leitstände zur Verkehrsflussüberwachung

Ein weiteres prominentes Beispiel zur Kontrolle komplexer Systeme mit mehreren unabhängig voneinander agierenden Komponenten sind *Leitstände*. Diese sind beispielsweise auf Flughäfen zur Flugüberwachung oder bei der Verkehrsflussüberwachung im Einsatz. Bei der Bahn werden alle Informationen, die ein Fahrdienstleiter zur Kontrolle und Steuerung benötigt, in einem sogenannten „Elektronischen Stellwerk“ Bedienraum (ESTW-Bedienraum) zur Verfügung gestellt (siehe auch Abbildung 1.4 in Kapitel 1). In Form eines Dashboards können auf mehreren Monitoren die aktuellen Einstellungen der Weichen abgefragt, angezeigt und geändert werden. Damit verfügen die Eisenbahnunternehmen über eine grafische und tabellarische Darstellung des aktuellen Betriebsstatus ihrer Züge auf dem Schienennetz. Bei unplanmäßigen Änderungen, wie beispielsweise Störungen, können aus einem solchen ESTW-Bedienraum Steuerungen vorgenommen werden, um möglichst schnell auf die Änderungen reagieren zu können.

Neben der Visualisierung des Verkehrsflusses in den Bedienräumen werden die gesammelten Informationen zunehmend der breiten Bevölkerung zur Verfügung gestellt, da die ständige Verfügbarkeit von Informationen im Fern- und Nahverkehr in den letzten Jahrzehnten entscheidend an Bedeutung gewonnen hat [Funkwerk Information Technologies, 2009].

[8] Quelle: http://www.sapdashboardgallery.com/, Zugriff: Oktober 2013

Durch gezielte Aufbereitung und ein intelligentes Informationsmanagement können Verkehrsunternehmen und Transporteure sowie Fahrgäste von den gesammelten Informationen profitieren. Dazu werden alle erfassten Daten in einem zentralen Betriebsinformationssystem zusammengeführt, gespeichert und entsprechend aufbereitet den jeweiligen Zielgruppen bereitgestellt.



(a) Streckenspiegel im ESTW-Bedienraum[9] (b) Detailansicht[10]

***Abbildung 2.18**: Darstellung in den Betriebszentralen von Verkehrsbetrieben*

2.5.6 Planungstools

Neben der Überwachung von Smarten Umgebungen gibt es auch Systeme, die zu deren Erstellung eingesetzt werden. Diese Planungstools unterstützen Architekten und Entscheidungsträger in einer Art und Weise, dass sie Vorschläge für das Layout und die Einrichtung der Umgebung aufzeigen. Beispiele solcher Planungstools sind unter anderem im Handelsbereich zu finden. Dort werden sogenannte Floor- und Spaceplanner eingesetzt, um bereits im Vorfeld sowohl die Regalpositionen als auch deren Bestückung zu spezifizieren. Basierend auf einem Grundrissplan können in einer Visualisierung virtuell Regale positioniert werden. Dabei können optional Unterstützungsfunktionalitäten eingeschaltet werden, wie beispielsweise eine Warnung bei Verletzung von Mindestabständen zwischen einzelnen Regalen.

Ist die Positionierung der Regale erfolgt, können diese im nächsten Schritt virtuell mit Produkten befüllt werden, deren Informationen aus dem eigenen Warenwirtschaftssystem entnommen werden. Dazu haben diese Systeme Schnittstellen, die an die Warenwirtschaftssysteme der Einzelhändler angeschlossen sind. Des Weiteren bieten diese Systeme Schnittstellen zu bereits bestehenden BI Systemen des Händlers, um beispielsweise einen Abgleich von prognostizierten Abverkaufszahlen mit der anvisierten Verräumung zu ermöglichen und potentiell anstehende Leerverkäufe anzuzeigen. Auch im privaten Umfeld werden solche Planungstools eingesetzt, um schon im Vorfeld Wohnungseinrichtungen zu planen, wie bereits in

[9] Quelle: http://www.db-netz.de/file/2360520/data/produktpraesentation_dispositionsarbeitsplatz.pdf, Zugriff: November 2014

[10] Quelle: http://www.stellwerke.de/formen/lupe.html, Zugriff: November 2014

Kapitel 1.1.2 präsentiert. Die beschriebenen Systeme zielen jedoch nicht auf das Monitoring instrumentierter Umgebungen ab, obwohl sie durch die bereits geschaffenen Schnittstellen und Möglichkeiten der Darstellung großes Potenzial hierzu hätten.

2.6 Zusammenfassung

In diesem Kapitel wurde auf Grundlagen eingegangen sowie Definitionen aufgeführt, die beim Verständnis der in dieser Arbeit behandelten Themen behilflich sind. Begonnen wurde mit grundlegenden Konzepten zur Datengenerierung und -verarbeitung in Bezug auf die Verarbeitungskette in Smarten Umgebungen. Ein besonderer Fokus lag auf der Definition von Modalität, welche eine wichtige Eigenschaft bei der Datenerfassung und -analyse dargestellt. Eine eindeutige Referenzierung von Objekten wurde im Themengebiet des Internet der Dinge und Dienste erläutert. Diese eindeutige Identifikationsmöglichkeit kann zum Aufbau von digitalen Objektgedächtnissen verwendet werden, wie am Beispiel von OMM beschrieben. Bei der situationsbezogenen Anwendung von erfassten Sensorinformationen müssen zusätzlich weitere Informationen berücksichtigt werden. In diesem Zusammenhang wurde der Begriff Kontext eingeführt und definiert. Des Weiteren wurde der Begriff Agent erläutert, welcher mittels Sensoren Informationen in der Umgebung erfasst, diese auf „intelligente“ Art miteinander fusioniert und verarbeitet, um selbstständig mittels Aktuatoren hierauf reagieren zu können.

Anschließend wurde auf die Beschreibung von Smarten Umgebungen inklusive der detaillierten Erläuterungen von Sensoren und Aktuatoren, mit Hilfe derer Umgebungen die Möglichkeit erhalten, Informationen zu erfassen und entsprechende Handlungen auszuführen, eingegangen. Dabei wurde zwischen verschiedenen Arten von Smarten Umgebungen unterschieden, für deren Planung unterschiedliche Parameter berücksichtigt werden müssen. Des Weiteren wurde auf Verarbeitungsmethoden für Smarte Umgebungen eingegangen. Im Speziellen wurde die Kombination physischer Umgebungen mit dem Cyberspace, was mit dem Begriff Cyber-Physische Umgebungen (CPE) bezeichnet wird, betrachtet.

Weiterhin wurden diverse Kommunikationsmöglichkeiten beschrieben und deren Differenzierungsmerkmale erläutert. Die vorgestellten Kommunikationstechnologien umfassen Blackboard-Ansätze, Tupelräume, Publish/Subscribe-Systeme, komplexe Eventverarbeitungssysteme (CEP) sowie Datenstrom Managementsysteme (DSMS). Im Speziellen wurden die letzten beiden Systeme miteinander in Verbindung gebracht, da sie zwei konkurrierende Ansätze darstellen, die aus unterschiedlichen wissenschaftlichen Gebieten stammen [Cugola und Margara, 2012b]. Sie unterscheiden sich vorwiegend in der Art, wie die Datenübertragung stattfindet. Beim ersten Ansatz geht man von festen Ereignissen aus, die als Event publiziert werden. Der zweite baut auf dem Konzept von kontinuierlichen Datenströmen auf, über die die Informationen verbreitet werden.

Im Anschluss daran wurden Systeme und Methoden erläutert, die eine Virtualisierung und Visualisierung von CPE ermöglichen. Es wurden unterschiedliche Darstellungsformen präsentiert und diskutiert. Durch die Virtualisierung entstehen virtuelle Welten und

Mischformen zwischen der realen und der virtuellen Welt. Diese wurden in das Real-Virtuell-Kontinuum eingeordnet und gegeneinander differenziert. Bei der Visualisierung von 3D-Modellen lag ein besonderer Fokus auf der Darstellung im Internet. Neben der reinen Visualisierung wurde auf die Möglichkeit der Einbindung von Simulationen, die einerseits zum virtuellen Testen von Systemen und andererseits für Vorhersagen verwendet werden können, eingegangen.

Das Kapitel endet mit der Vorstellung von existierenden Monitoring-Systemen, die bereits heutzutage angewendet werden. Die Beschreibung umfasst die Definition und Erläuterung von Business Intelligence Systemen zur Auswertung von Informationen sowie Dashboards als Darstellungsvariante. Neben Monitoring- und Steuerungsmechanismen, den Leitständen, für öffentliche Plätze und den öffentlichen Transport wurden Methoden präsentiert, die basierend auf erfassten Informationen, diese entsprechend auswerten und visuell Entscheidungsträgern darstellen. Neben Systemen, die während des laufenden Betriebs eine Kontrolleinrichtung für die Umgebung zur Verfügung stellen, wurden exemplarisch für die Handelsdomäne Tools vorgestellt, die eine Planung der Umgebung im Vorfeld ermöglichen.

Im folgenden Kapitel werden verwandte Arbeiten aufgeführt, welche mögliche Implementierungen der in diesem Kapitel vorgestellten Grundlagen aufzeigen. Im Speziellen werden mehrere Kommunikationstechniken und Systeme zur visuellen Repräsentation von virtualisierten Umgebungen vorgestellt.

Teil II

Stand der Wissenschaft und Technik

3 VERWANDTE ARBEITEN

In der vorliegenden Arbeit wird die Thematik der Virtualisierung von smarten Umgebungen inklusive ihrer interaktiven Visualisierung behandelt. Dies umfasst einerseits die visuelle 3D-Darstellung der Umgebung und andererseits die Kommunikation zu und zwischen den darin befindlichen Sensoren, Aktuatoren und Diensten. In diesem Zusammenhang wurden sowohl Kommunikationsinfrastrukturen als auch Visualisierungskonzeptionen, die nach dem Dual Reality Paradigma aufgebaut sind, hinsichtlich ihrer Generalisierbarkeit untersucht. Im folgenden Kapitel werden relevante Arbeiten aus diesen beiden Themengebieten aufgeführt und detailliert beschrieben. Das Kapitel endet mit einem Vergleich dieser Arbeiten und dem Ansatz der vorliegenden Arbeit, indem Gemeinsamkeiten und Unterschiede erörtert werden.

3.1 Kommunikationsinfrastrukturen

Seit der Entstehung von verteilten und vernetzten Systemen wird an geeigneten Kommunikationsinfrastrukturen geforscht. Wie bereits in Abschnitt 2.3 beschrieben, können die Infrastrukturen in verschiedene Klassen kategorisiert werden. In den kommenden Abschnitten werden mehrere Beispiele von Systemen vorgestellt, die in diese Klassen eingeordnet werden können. Begonnen wird mit einem System, welches sich an der Idee des Tupelraums orientiert. Im Anschluss daran werden drei Systeme vorgestellt, die nach dem Publish/Subscribe-Prinzip agieren. Darauf folgen vier Systeme, die aus den Bereichen von aktiven Datenbanken und Datenstrom Managementsystemen (DSMS) stammen. Abschließend werden weitere vier Arbeiten aufgeführt, die unter die komplexen Eventverarbeitungssysteme (CEP) fallen.

3.1.1 T Spaces

T Spaces ist eine von IBM in Java implementierte Version des Tupelraum-Ansatzes mit dem Ziel, mehrere Betriebssystemplattformen unterstützen zu können [Wyckoff et al., 1998]. Wie bei dem Tupelraum-Ansatz in Linda (siehe Abschnitt 2.3.2) bestehen die Tupel für die Übertragung von Informationen aus einer geordneten Sequenz von Objekten. Als Erweiterung zu der Realisierung in Linda wird bei T Spaces der Vorteil der objektorientierten Programmierung berücksichtigt. So können bei den Templates Objektklassen sowie abgeleitete Klassen angegeben werden, um entsprechende Tupel zu erhalten. Für die Abfrage, ob ein Tupel zu einem Template passt, wird zuerst geprüft, ob die Anzahl der Felder

übereinstimmt. Anschließend wird für jedes Feld des Tupels geprüft, ob es eine Instanz der Klassen des zugehörigen Felds im Template ist. So können auch abgeleitete Klassen und damit Spezialisierungen in Tupel ermittelt werden. Zuletzt werden die im Template explizit spezifizierten Werte mit denen des Tupels verglichen.

Des Weiteren wurden neben den Standardoperatoren für das Schreiben, Lesen und Entnehmen zwei weitere implementiert, die es ermöglichen, alle Tupel, die dem Template entsprechen, zu ermitteln und gegebenenfalls dem Tupelraum zu entnehmen. Im Gegensatz zum Lese-Operator liefert der *Scan*-Operator alle Tupel gemäß des spezifizierten Templates zurück. Analog ist der *konsumierende Scan*-Operator das Pendant zum Entnahme-Operator. In Linda wurden die Operatoren so implementiert, dass sie blockieren. Dies bedeutet, dass sie wie bei der synchronen Datenübertragung nach einer Anfrage so lange warten, bis die geforderte Antwort kommt, während der zugehörige Prozess sich in einem Schlafmodus befindet. Ergänzend zu den blockierenden Funkionen wurde bei T Spaces zusätzlich eine nicht-blockierende Variante implementiert [Lehman et al., 1999]. Ähnlich des Ansatzes von Multi-Blackboards ist es in T Spaces möglich, mehrere Tupelräume zu spezifizieren, zu denen sich die Quellen verbinden können. Bei der Anmeldung zum Tupelraum wird eine zugehörige Gruppenspezifikation angegeben, über die ein entsprechendes Rechtesystem verknüpft ist. Somit können sich die Quellen nur zu gewissen Tupelräumen verbinden, für die sie autorisiert sind. Jeder Raum beinhaltet dafür eine Liste von Gruppen, die die Berechtigung haben, auf ihn zugreifen zu dürfen.

Beispielhaft wurde T Spaces zum iterativen Durchsuchen von Internetseiten verwendet, was als *Crawling* bezeichnet wird. Es wurden mehrere Crawler parallel verwendet, um effizient einen Internetbereich zu durchsuchen. Der Bereich wurde dabei in kleinere Subbereiche unterteilt und für jeden Subbereich ein separater Crawler instantiiert, welcher eine Zusammenfassung der Internetseiten für den jeweiligen Subbereich erstellt. Wurden auf den Seiten neue Verlinkungen zu Adressen gefunden, wurden diese im Tupelraum hinterlegt, damit der zugehörige Crawler diese Internetseite bearbeiten kann.

3.1.2 Event Heap / iROS

Als Erweiterung von T Spaces wurde von Johanson und Fox der *Event Heap* als Publish/Subscribe-System entwickelt [Johanson und Fox, 2002]. Ziel der Implementierung war es, eine Middleware zu kreieren, die in ubiquitären Umgebungen mit mobilen und eingebetteten Geräten agieren kann, um kollaborativ an Lösungen von Aufgaben zu arbeiten. Die Middleware wurde so entwickelt, dass sie in unterschiedlichen Arten von ubiquitären Applikationen eingesetzt werden kann, um die Aufgaben der Übertragung von Informationen zwischen diesen zu übernehmen. Des Weiteren war eine Anforderung, dass das System einfach in existierende Sprachen und Umgebungen eingebunden werden kann und möglichst robust gegenüber Flüchtigkeitsfehlern ist, um beim Experimentieren mit neuen Geräten das existierende System nicht zu destabilisieren. Um diese Zielsetzung zu erreichen, haben Johanson und Fox als Grundlage den Tupelraum-Ansatz gewählt, da er durch den einfachen Aufbau auf mehreren Betriebssystemen ausgeführt werden kann. Die Basisimplementierung von Server und Client ist in Java realisiert. Daneben werden auch eine Webseite zum

Versenden und Empfangen von Events sowie Wrapper für C++ und Python bereitgestellt, damit die übertragenen Daten auch auf anderen Betriebssystemen und in Applikationen auf PDAs verwendet werden können.

Die einzelnen Tupel werden in Form von Events gebündelt. Dabei werden die Events anhand ihrer Typen unterschieden. Jeder Eventtyp umfasst ein Set aus vordefinierten Tupeln und somit spezifische Informationen. Für die Übertragung der Events werden die Informationen in die jeweiligen Tupel integriert. Anschließend wird das gesamte Event serialisiert und zum Server geschickt, der dieses mittels push an die registrierten Clients verteilt. Diese werden mit Hilfe von speziellen Funktionen über neu eintreffende Events notifiziert. Diese Funktionen werden Callbacks genannt, da sie automatisch aufgerufen werden, wenn die jeweiligen Events am System eintreffen. Damit die Clients bei der Verarbeitung wissen, um welchen Eventtyp es sich handelt, wird neben den zuvor erwähnten Tupeln zusätzlich ein weiteres Tupel dem Event hinzugefügt, welches dessen Typ spezifiziert. Anhand dieses Typs wissen die Clients bei der Verarbeitung, welche Tupel und somit welche Informationen im Event hinterlegt sind. Des Weiteren haben die Events einen Zeitstempel, der deren Gültigkeit spezifiziert. Ist dieser gesetzt, sind die Events so lange gültig, bis der Zeitstempel abgelaufen ist und werden in diesem Zeitraum persistent hinterlegt. Dank der automatischen Notifikation anhand der Callbacks wird die Gefahr eliminiert, dass Events aufgrund kurzer Lebensdauer und zu großer Abfragefrequenz durch die Clients verloren gehen, da die Daten nach dem Ablauf nicht mehr persistent vom Server gehalten werden. Wird der Server neu gestartet, melden sich die Clients automatisch wieder am System an.

Prototypisch wurde der Event Heap als essentielle Komponente in eine Infrastrukturkomponente integriert, das sogenannte *iROS*-System (Interactive Room Operating System), welches in einem interaktiven Konferenzraum in Stanford eingesetzt wurde, dem *iRoom* (Interactive Room) [Johanson et al., 2002]. Mittels einer internetbasierten Kontrollseite konnten die Aktuatoren, wie beispielsweise das Licht und mehrere Bildschirme, in der Umgebung über das iROS-Systems gesteuert werden. Dies ermöglichte es, dass die Steuerung auch über mobile Endgeräte funktionierte. Unter anderem lief in der Umgebung ein Dienst, welcher mehrere Bildschirme und Präsentationsobjekte ansteuern konnte, der *SmartPresenter*. Dieser Dienst wurde über Events gesteuert und konnte Bildschirminhalte, wie beispielsweise eine PowerPoint-Präsentation oder 3D-Modelle, auf einem der Displays anzeigen.

3.1.3 MundoCore

MundoCore ist eine Kommunikationsmiddleware, welche spezialisiert auf die Informationsübertragung in verteilten Systemen mit mobilen Endgeräten und kleinen Sensorknoten entwickelt wurde [Aitenbichler et al., 2007]. Die Nachrichten werden als einfache oder hierarchisch aufgebaute Events übermittelt. Ein einfaches Event setzt sich aus dem Namen und einer Liste von Attributen zusammen. Jedes Attribut ist hierbei ein Tupel aus Namen, Typ und Wert. Hierarchisch strukturierte Events bestehen aus einer Verschachtelung von Attributen, wie es beispielsweise bei XML der Fall ist. Neben einem Publish/Subscribe-Ansatz wird auch eine verteilte objektorientierte Programmierung, analog zu den Implementierungen von RMI, und Streaming unterstützt. Somit kann die Middleware auf die benötigten

Eigenschaften der Umgebung angepasst werden. Für Publish/Subscribe werden neben den typbasierten auch inhaltsbasierte Subskriptionen unterstützt. Im zweiten Fall wird bei der Anmeldung ein entsprechender Filter angegeben, welcher den Inhalt prüft und im positiven Fall die Information weiterleitet. Einfache Filter enthalten hierbei den Namen, des zu überprüfenden Tupels, dessen Typ, einen Operator und eine Konstante mit der abgeglichen werden soll. Einfache Filter können auch zu zusammengesetzten Filtern kombiniert werden, indem beispielsweise zwei einfache Filter als Konjunktion zusammengeschlossen werden und somit beide Bedingungen erfüllt sein müssen. Entgegen dem verteilten Funktionsaufruf und Datenstreaming wird bei Publish/Subscribe die Entkopplung der Sender und Empfänger hinsichtlich Zeit, Raum und Datenfluss realisiert.

Von MundoCore existieren Implementierungen in C++, Java und Python. Für die Erstellung von Stub-Dateien kann eine Schnittstellenbeschreibungssprache (Interface Definition Language, IDL) verwendet werden, aus der die entsprechenden Stub-Dateien für die jeweiligen Programmiersprachen erstellt werden [Aitenbichler, 2006]. Zur Datenübertragung werden die Events binär oder in ein XML-Format serialisiert. Zur sicheren Übertragung können die Events auch verschlüsselt werden. Neben der Möglichkeit der Vernetzung über einen zentralen Server unterstützt MundoCore eine ad-hoc-Vernetzung der Clients. In diesem Fall werden erreichbare Clients gesucht und sich mit diesen automatisch verbunden. Basierend auf dem zugrundeliegenden Übertragungsprotokoll können neben TCP und UDP auch serielle Verbindungen oder Infrarotverbindungen zur Kommunikation genutzt werden.

Exemplarisch wurde MundoCore für eine Raumsteuerung eingesetzt [Aitenbichler et al., 2005]. Der Raum war hierbei mit mehreren Displays und Monitoren instrumentiert, wobei diverse Dienste Zugriff auf die Displays benötigten. Durch einfache Interaktion sollten die unterschiedlichen Dienste gestartet und gestoppt werden können. Dies erfolgte durch die Kommunikation als Events, deren Erhalt durch eine Bestätigungsnachricht beantwortet wurde, gefolgt von der Durchführung des entsprechenden Diensts. Nach Vollendung der Handlung wurde erneut eine Nachricht als Event versendet. Mittels smarter Netzstecker konnten Geräte zudem über die Eventkommunikation ein- und ausgeschaltet werden, wobei eine erfolgreiche Durchführung quittiert wurde.

3.1.4 MQTT / MQTT-SN

MQTT (Message Query Telemetry Transport) ist ein Protokoll zur Datenübertragung, welches von IBM entwickelt wurde und noch weiterentwickelt wird[1]. Ziel dieses Protokolls ist es, einen Standard für die Informationsübertragung im industriellen Umfeld zu definieren, um im Speziellen bei Systemen mit eingeschränkten Verarbeitungsressourcen eingesetzt werden zu können [Hunkeler et al., 2008]. Das Protokoll ist als Publish/Subscribe-System konzipiert worden, um die Vorteile der Austauschbarkeit von Komponenten und die lose Kopplung der Datenübertragung zu unterstützen. Dabei werden drei Qualitätsmerkmale (Quality of Service, QoS) unterstützt, welche sich in der Zuverlässigkeit der Informationsübertragung und im Verarbeitungsaufwand sowie der Komplexität unterscheiden. Zum

[1] http://mqtt.org, Zugriff Januar 2014

einen können Informationen maximal einmal an jeden Client geschickt werden. Dies impliziert, dass auch Nachrichten möglicherweise verloren gehen. Zum anderen kann angegeben werden, dass Nachrichten mindestens einmal an jeden Client geschickt werden, was möglicherweise zu mehrfachem Eintreffen des gleichen Events bei den Empfängern führen kann. Schließlich gibt es noch die Möglichkeit, jede Nachricht genau einmal an jeden Client zu schicken. Je nach Anwendung kann die gewünschte QoS-Einstellung für jedes Event einzeln eingestellt werden, um die Verarbeitung auf dem Broker so gering wie möglich zu halten.

Ähnlich wie beim Event Heap unterstützt das Protokoll von MQTT die Interessensbekundung und damit eine Registrierung für Eventtypen. Des Weiteren können auch Events bezüglich ihres Themas angefragt werden, wobei die potentiellen Themen a priori allen Clients bekannt sind. Neben dem Eventtyp und Thema beinhalten Events auch die Information zur Identifikation des Senders. Alle Informationen werden gemäß des Protokolls im Binärformat kodiert, welches auch bei der Übertragung verwendet wird. Die standardmäßige Registrierung am Broker enthält einen Benutzernamen und ein Passwort, über die eine spätere SSL-Verschlüsselung bei der Kommunikation realisiert werden kann. Daneben kann ein sogenanntes „Will“-Event registriert werden. Dieses wird vom Broker versendet, sobald der Client unerwartet seine Verbindung verliert. Dies kann unter anderem zur Erkennung von Systemfehlern verwendet werden. Zudem senden die Clients in regelmäßigen Abständen eine Information, dass sie noch ordnungsgemäß funktionieren, wodurch mögliche Fehlerquellen detektiert werden können, wenn diese Events nicht mehr versendet werden.

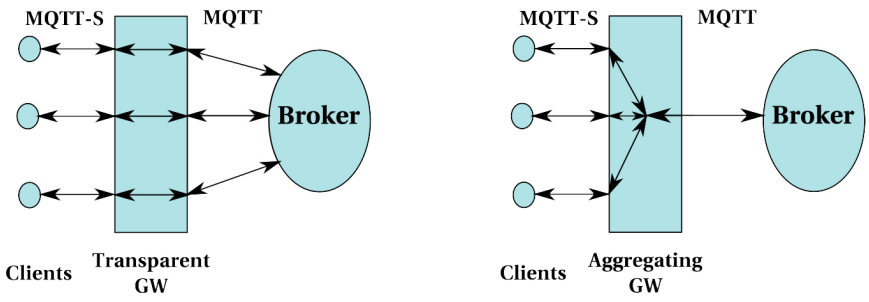


***Abbildung 3.1**: Zwei Alternativen für Gateways (GW) in MQTT-SN (Quelle: [Hunkeler et al., 2008])*

Um auch mobile Netzwerke zu unterstützen gibt es die Weiterentwicklung: *MQTT-SN* [MQTT for Sensor Networks]. Publiziert wurde die Erweiterung vorerst unter dem Akronym MQTT-S, was aber zu Verwechselungen und Unklarheiten führte, da auch Sicherheit [Security] unter „S“ verstanden werden konnte. Die einzelnen drahtlosen Netzwerke kommunizieren dabei über Gateways mit dem Broker, wobei zwei Typen unterschieden werden [Hunkeler et al., 2008]. Das transparente Gateway dient als direkte Zwischenschicht zwischen den Clients und dem Broker. Dies bedeutet, dass zwischen dem transparenten Gateway und dem Broker ebenso viele Verbindungen aufrecht gehalten werden, wie Clients am Gateway verbunden sind (siehe Abbildung 3.1, links). Beim aggregierenden Gateway werden die Events

gebündelt und nur über eine Verbindung zum Broker geleitet (siehe Abbildung 3.1, rechts). Nachteil dabei ist jedoch, dass das Gateway die Informationsweiterleitung gemäß den angegebenen QoS-Parametern selber gestalten muss. Beispielhafte Umsetzungen von MQTT-SN gibt es für ZigBee und TinyOS, welche Übertragungsprotokolle für verteilte Sensornetzwerke sind.

3.1.5 Snoop / Sentinel

Snoop ist eine Spezifikationssprache für Events im Kontext von aktiven Datenbank Managementsystemen (DBMS) [Chakravarthy und Mishra, 1994]. Die Detektion von Events erfolgt in Form von Regeln. Diese sind so aufgebaut, dass sie abhängig von Events vorgegebene Bedingungen prüfen und gegebenenfalls anschließend Aktionen ausführen. Aus diesem Grund sind sie unter dem Begriff „Event-Bedingung-Aktion"-Regeln bekannt. In Snoop wird zwischen primitiven und zusammengesetzten bzw. komplexen Events unterschieden. Die primitiven Events sind als atomar anzusehen und treten zu einem bestimmten Zeitpunkt auf, wobei die potentiellen Typen a priori bekannt sind. Beispielsweise fallen Funktionsaufrufe oder Datenbankoperationen in diese Kategorie. Zusammengesetzte Events hingegen umfassen eine Zeitspanne und beschreiben mitunter zeitliche Beziehungen zwischen den Events. Ein solches Event setzt sich dabei aus mehreren primären oder komplexen Events zusammen. Da primitive Events dem System bereits im Vorfeld bekannt sind, können diese problemlos detektiert werden. Für die Erfassung von zusammengesetzten Events wurden mehrere Operatoren definiert. Der *Sequenz-Operator* hat als Bedingung, dass ein vorgegebenes Event detektiert wird, nachdem bereits ein anderes spezifiziertes Event erfasst wurde. Bei der *Disjunktion* muss eines von zwei Events detektiert werden und bei der *Konjunktion* beide. Daneben gibt es den *A-Operator*, der jedes Auftreten eines spezifizierten Events zwischen zwei vorgegebenen Events, die als Schranken gelten, detektiert und die Aktion auslöst. Um nur eine Aktion auf mehrere Auftreten des gesuchten Events in der vorgegebenen Schranke zu ermöglichen gibt es den *A*-Operator*. Die periodische Alternative, der *P-Operator*, wird ab der Erkennung eines vorgegebenen Startevents in periodischen Zeitabständen ausgelöst, bis ein definiertes Terminalevent detektiert wird. Mit dem *P*-Operator* werden die periodischen Auftreten der Aktionsauslösung nicht sofort weitergegeben, sondern nach der Detektion des Terminalevents akkumuliert versendet.

Neben dem Eventtyp wird auch nach der Art unterschieden, wie die Erkennung stattfindet. Bei der *kürzlichen* Erfassung wird das letzte Auftreten der Events hergenommen und mit diesem weitergearbeitet. Bei der *aufzeichnenden* Variante werden die zwischenzeitlich erfassten zusammengesetzten Events gepuffert und anschließend in chronologischer Reihenfolge abgearbeitet, wobei bei jeder Verarbeitung der Events die letzte Version aus dem Speicher „verbraucht" wird. Bei der *kontinuierlichen* Verarbeitung werden die Startpunkte der erfassten Events mit den weiteren eintreffenden Events gepuffert und bei vollständiger Spezifikation gefeuert. Damit realisiert Snoop den Ansatz von verschiebbaren Fenstern. Die letzte Variante ist die *kumulative*, bei der alle möglichen Kombinationen der eingetroffenen Events, die die Spezifikation erfüllen, ein Auslösen der geforderten Reaktion verursachen. Umgesetzt wurde die Spezifikationssprache beispielsweise in Sentinel, welches ein objektorientiertes DBMS zugrunde legt [Chakravarthy, 1997]. In diesem Rahmen wurde

Snoop um die Operatoren *Plus*, der eine Mindestzeitspanne zwischen dem Auftreten von zwei Events umfasst, und *Nicht* ergänzt. Für die Erfassung und Detektion von Events werden Graphen-basierte Ansätze zu Hilfe genommen.

Komplexe Events können in zwei Arten unterteilt werden [Galton und Augusto, 2002]. Zum einen kann die Detektion zum Zeitpunkt des Auftretens des letzten Events durchgeführt werden und zum anderen kann die gesamte Zeitspanne bei der Detektion berücksichtigt werden, was zu unterschiedlichen Ergebnissen führen kann. Je nach System und Anwendung muss die geeignete Variante gewählt werden. Um die intervallbasierte Erkennung von komplexen Events zu ermöglichen wurde Snoop erweitert. Diese Änderungen wurden in *SnoopIB* formalisiert und implementiert [Adaikkalavan und Chakravarthy, 2006].

3.1.6 Aurora / SQuAl

Aurora ist ein DSMS, welches als Monitoring System für Applikationen ausgelegt ist [Abadi et al., 2003]. Es basiert auf dem Grundgedanken, dass das Datenbank-System aktiv agiert und der Mensch dabei passiv bleiben kann. Bei normalen DBMS muss der Mensch hingegen aktiv über Anfragen Informationen akquirieren und das System wird ausschließlich passiv zum Speichern der ankommenden Daten verwendet. Als Voraussetzung für das angestrebte Monitoring System sollen die Daten von externen Quellen stammen und mit historischen Informationen verknüpft werden können. Es soll Trigger-basiert sein und in Realzeit agieren können, wobei nur relevante Daten bei der Bearbeitung betrachtet werden. Aus diesem Grund ist der Aufbau von Aurora so, dass mehrere Datenströme parallel eintreffen können. Diese werden im System gemäß der auf Anfragen basierenden Vorgaben verarbeitet und die Ergebnisse an externe Applikationen weitergeleitet.

Die externen Applikationen können sich für die gewünschten Verarbeitungen beim System registrieren und Anfragen stellen. Dabei sind drei Arten von Anfragen zu unterscheiden. Erstens gibt es die *kontinuierlichen Anfragen*, die alle eintreffenden Informationen verarbeiten und an die angeschlossenen Applikationen weiterleiten. Zweitens gibt es *ad-hoc-Anfragen*, die ähnlich wie kontinuierliche Anfragen funktionieren, jedoch zum Zeitpunkt der Erstellung zusätzlich noch ältere zwischengespeicherte Informationen berücksichtigen. Im Gegensatz zu den kontinuierlichen Anfragen können diese zu einem späteren Zeitpunkt wieder von der Verarbeitung getrennt werden. Schließlich gibt es die sogenannten *Views*, an die keine Applikationen angeschlossen sind. Die ermittelten Informationen können zwischengespeichert und später bei Bedarf anderen Programmen zur Verfügung gestellt werden. Dabei werden bei der Berechnung Qualitätsmerkmale ermittelt, welche das System versucht zu maximieren. Diese Merkmale stellen dabei mehrdimensionale Funktionen verschiedener Attribute dar. Darunter fallen Informationen zur Antwortzeit und damit verbundene Verzögerung, wie viele Tupel zur Performanzoptimierung nicht betrachtet werden und wie viele „wichtige“ Daten erstellt wurden.

Die Anfragen werden mittels *SQuAl*, die Anfrage-Algebra von Aurora, erstellt. Diese unterstützt insgesamt sieben Operatoren. Der *Filter-Operator* ermöglicht anhand gewisser Merkmale einen Datenstrom zu unterteilen, indem er eine Fallunterscheidung durchführt. Der

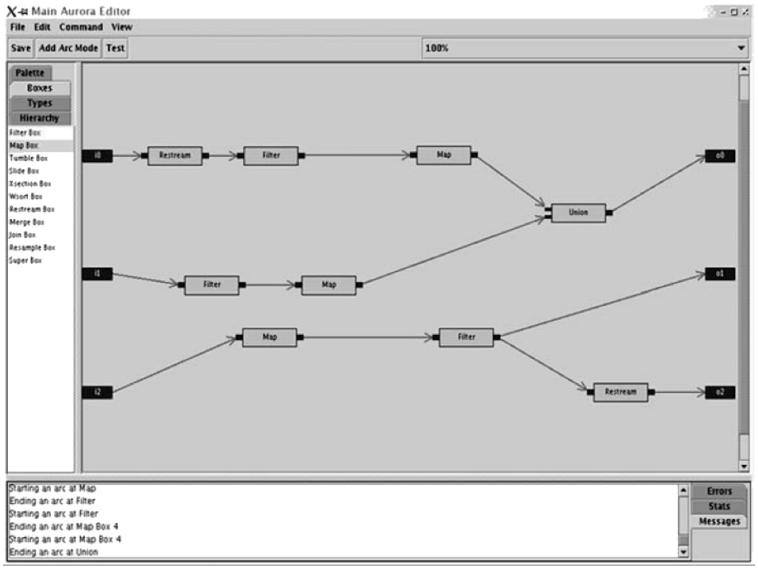


***Abbildung 3.2**: Grafische Oberfläche zur Erstellung und Bearbeitung von Verarbeitungsgraphen in Aurora (Quelle: [Abadi et al., 2003])*

Abbildungs-Operator kann mittels einer Funktion die Informationen eines Datenstroms abändern und diese weiterleiten, ohne dabei die Struktur zu verändern. Ein *Sortierungs-Operator* kann verwendet werden, um aus n Tupeln eines nach definierten Merkmalen herauszufiltern und weiterzuleiten, wie beispielsweise das Minimum oder Maximum. Trifft anschließend ein neues Event ein, wird das zuvor herausgefilterte durch dieses ersetzt und die Filterung erneut durchgeführt. Der *Aggregations-Operator* kann eingesetzt werden, um Fenster in den Datenströmen zu definieren. Der *Verbindungs-Operator* ermöglicht den Zusammenschluss mehrerer Datenströme zu einem einzigen, der anschließend weiterverarbeitet werden kann. Schließlich gibt es den *Resample-Operator*, der eine teilweise Vermischung von Attributen zweier Events aus zwei Datenströmen in einem vorgegebenen Fenster realisiert. Diese Operatoren können auch in Form einer Ausführungskette miteinander kombiniert werden. Um die Anforderungen an Echtzeitverarbeitung so gut wie möglich zu realisieren, wurden diverse Optimierungsalgorithmen berücksichtigt, wie beispielsweise die automatische Sortierung der Reihenfolge der Operatoren oder der Verschmelzung von Teilpfaden in diesen Ausführungsketten. Zur Erstellung solcher Verarbeitungsgraphen wurde eine grafische Oberfläche entwickelt, die es ermöglicht, die Operatoren hintereinander zu schalten. Dafür wurde das „Boxen und Pfeile“-Layout gewählt, bei dem die Operatoren als Boxen dargestellt werden und die Informationkanäle als Pfeile zwischen diesen (siehe Abbildung 3.2). Mittels Drag’n’Drop kann über diese GUI der Verarbeitungsgraph mit den diversen Ketten erstellt, angeschaut und bearbeitet werden.

3.1.7 TelegraphCQ

Wie auch Aurora ist *TelegraphCQ* eine Erweiterung von Datenbank Managementsystemen. Während die traditionellen DBMS ausschließlich auf Relationen aus einer Datenbank

operieren, verwendet das in Berkeley entwickelte System zusätzlich Informationen aus den Live-Datenströmen ohne diese vorher zwischenzuspeichern [Chandrasekaran et al., 2003a,b]. Zusätzlich wird jedoch noch eine Datenbank verwendet, um historische Informationen zwischenzuspeichern, damit diese für spätere Operationen zur Verfügung stehen. Neben der Möglichkeit, sowohl bei neuen Anfragen alte Daten zu verwenden, als auch neue Daten in älteren kontinuierlichen Anfragen zu berücksichtigen, besteht das Ziel in TelegraphCQ darin, die Verarbeitungslogik so weit wie möglich zu bündeln, um ressourceneffizient zu arbeiten. Dazu agiert es als ein deterministischer Zustandsautomat mit nicht-blockierenden Operatoren und baut auf PostgreSQL auf. Zudem verwendet es einen geteilten Speicher, auf dem die vom Benutzer spezifizierten Anfragen ausgeführt werden. In diesen Speicher werden einerseits Informationen aus den Datenströmen und andererseits Informationen, die aus der Datenbank stammen, geschrieben. Die ermittelten Informationen können entweder automatisch an verbundene Clients geschickt (push) oder von diesen abgerufen werden (pull). Verwendet wird dazu eine an SQL angelehnte Abfragesprache, wobei sich Clients über eine TCP/IP-Schnittstelle oder eine Kommandozeilenschnittstelle zu dem verarbeitenden Server verbinden können. Dieser schließt mehrere Module ein, die das Zwischenspeichern von weiteren Daten aus externen Datenquellen, standardmäßige relationale Operatoren und dynamische Optimierungsverfahren umfassen. Einzelne Anfragen von Clients werden dabei in einer „Super“-Anfrage zusammengefasst, die sich entsprechend anpasst, sobald neue Anfragen hinzukommen.

Um ressourceneffizient arbeiten zu können, ist es in der Regel wichtig, nicht die gesamte Historie zu verwenden, sondern die Daten auf einen geringeren Prozentsatz herunterzubrechen. Da dabei Informationen verloren gehen, sollte es aus der Anwendung heraus möglich sein, den Faktor der Reduktion zu bestimmen. Neben der Verarbeitungszeit kann auch die Speicherauslastung ein Problem darstellen. Basierend auf der Erkennung einer Überlastung des Speichers müssen ebenso geeignete Verfahren verwendet werden, um auch in solchen Fällen fehlerfrei arbeiten zu können. Aus diesem Grund wurde TelegraphCQ um eine Methode namens *OSCAR* (Overload-sensitive Stream Capture and Archive Reduction) erweitert, die diese Problematik aufgreift [Chandrasekaran und Franklin, 2004]. Diese Methode wurde in drei verschiedenen Arten implementiert und in TelegraphCQ integriert. Bei der ersten „*eifrigen*“ Version werden die ankommenden Daten aus dem Datenstrom komplett auf der Festplatte abgespeichert. Gleichzeitig werden Kopien von reduzierten Fassungen dieser Daten hinterlegt. Die entsprechenden Reduktionsstufen müssen dabei a priori festgelegt werden. Wird anschließend zur Anfragezeit eine Reduktionsstufe gefordert, die noch nicht als Kopie hinterlegt ist, werden die nächst höhere Stufe gewählt und mittels einer Entnahmefunktion so viele Einträge ausgewählt, bis der Reduktionsgrad erreicht wurde. Vorteil dabei ist, dass die Entnahmefunktion nur auf der kleineren Datenmenge operieren muss. Jedoch erhöhen sich bei diesem Verfahren die Zeit beim Schreiben sowie der Speicherplatzbedarf. Bei der zweiten „*faulen*“ Version wird beim Schreiben der Informationen aus dem Datenfluss nur eine Kopie auf der Festplatte hinterlegt. Sobald die erste Anfrage kommt, werden gemäß des Reduktionsfaktors entsprechend viele Einträge durch die Entnahmefunktion ermittelt und der Rest als „tot“ spezifiziert. In einem weiteren Schritt wird eine Kopie erstellt, die nur die Tupel enthält, die nicht als tot markiert wurden. Wird erneut eine Anfrage gestellt, bei

der der Reduktionsgrad größer oder gleich ist, werden entsprechend viele Einträge aus der Kopie durch die Entnahmefunktion spezifiziert, die verbleibenden als tot erklärt, woraufhin diese aus der Kopie gelöscht werden. Wird ein geringerer Reduktionsgrad gefordert, wird die komplette Kopie gelöscht und die Prozedur beginnt von vorne. Vorteil hierbei ist, dass beim Schreiben nur eine Version angelegt wird. Jedoch kann abhängig von den Reduktionsgraden bei den Anfragen der Algorithmus länger benötigen als bei der eifrigen Version, da die Kopie im schlimmsten Fall bei jeder zweiten Anfrage erneut aufgebaut werden muss. Bei der dritten Variante handelt es sich um eine „hybride“ Version. Diese agiert nach dem „teile und herrsche“-Prinzip. Bei Ankunft der Daten werden diese uniform jedoch zufällig in einen von n unterschiedlichen Blöcken geschrieben, wobei n vorab spezifiziert sein muss. Anschließend wird die Entnahmefunktion gemäß des Reduktionsfaktors auf jedem Block ausgeführt und die Resultate zusammengefügt. Diese Version hat den Vorteil, dass sie nur eine Kopie auf der Festplatte benötigt und die Entnahmefunktion auf kleineren Eingabewerten parallel operieren kann.

Des Weiteren wurde TelegraphCQ dahingehend erweitert, dass es trotz hardwareseitiger Probleme und hoher Informationsdurchsätze stabil arbeiten kann [Shah et al., 2004]. Dazu werden parallele Datenströme als redundantes Übertragungsmedium verwendet, wobei beide Datenströme auf unterschiedlichen Systemen laufen. Durch ein hinterlegtes Protokoll, in dem der Empfang von Daten bestätigt wird, können Fehler bei der Übertragung erfasst werden. Dazu wurde der sogenannte *Flux*-Operator integriert, der es ermöglicht, solche Fehler zu erkennen und automatisch darauf zu reagieren, so dass auch bei der Umlagerung der Informationen auf andere Systeme oder Kommunikationskanäle keine Daten verloren gehen.

3.1.8 CQL / STREAM

CQL [Continous Query Language] ist eine generelle, abstrakte Sprache inklusive einer Semantik für Verarbeitungssysteme von Datenströmen, welche in Stanford entwickelt wurde [Arasu et al., 2006]. Neben der Integration und Verarbeitung von Datenströmen können bei CQL ebenfalls Relationen bei den Berechnungen verwendet werden. Beispielsweise werden Relationen aus einer Verbindung zu einer Datenbank extrahiert und mit Datenströmen kombiniert. Hierzu werden drei Operationen angeboten: Datenstrom-zu-Relation (D2R), Relation-zu-Relation (R2R) und Relation-zu-Datenstrom (R2D). Die eigentliche Verarbeitung besteht immer aus einer Abfolge eines D2R, gefolgt von einem oder mehreren R2R und schließlich einem R2D Operator. Insgesamt inkludiert CQL drei D2R Operatoren. Bei dem ersten, dem *zeitbasierten Gleitfenster* (time-based sliding window), können Informationen abhängig von einem Zeitfenster abgerufen werden, wie beispielsweise alle Tupel, die im letzten Zeitfenster τ eingetroffen sind oder alle, die seit Beginn als Datenstrom an das System geschickt wurden. Der zweite ist der *tupelbasierte Fenster* (tuple-based window) Operator. Dieser ermöglicht es, die letzten n Tupel des Datenstroms zu referenzieren. Der letzte ist der *geteilte Fenster* (partitioned windows) Operator und bietet die Möglichkeit, Informationen aus den Datenströmen abhängig von gewissen Attributen $A_1, ..., A_k$ zu gruppieren. Jede dieser k Gruppierungen ist dabei von den anderen unabhängig. Von jeder werden anschließend die letzten n Tupel ermittelt und schließlich wieder zusammengefügt. Ähnlich zu TelegraphCQ können Anfragen im System über eine erweiterte Form der

SQL-Syntax gestellt werden, wobei davon ausgegangen wird, dass die Tupel in zeitlich geordneter Reihenfolge eintreffen. Dies wird bei den R2R Operatoren eingesetzt, die von SQL abgeleitet sind. Bei den R2D Operatoren werden wiederum drei Versionen angeboten. Der erste ist der *Istream* Operator, was für Einfügestrom [insert stream] steht. Er beinhaltet alle Relationen, die im letzten Zeitfenster neu hinzugekommen sind. Analog dazu gibt es den *Dstream* Operator, abgeleitet vom englischen „delete stream“, welcher die Relationen enthält, die im letzten Zeitfenster gelöscht wurden. Schließlich gibt es noch den *Rstream* Operator [relation stream], welcher alle Relationen aus dem letzten Zeitfenster liefert.

Die CQL Syntax wurde unter anderem in STREAM umgesetzt [Arasu et al., 2003]. STREAM steht für *STanford stREam datA Manager* und ist ein universelles, relationelles Datenstrom Managementsystem. Es wurde so entwickelt, dass eine große Anzahl von kontinuierlichen Anfragen gestellt werden kann. Für jede Anfrage wird ein separater Plan erstellt und anschließend im System mit anderen Plänen, die entweder übereinstimmen oder gleiche Teilpläne beinhalten, zusammengeschlossen. Somit können Ressourcen gespart werden. Ein Anfrageplan besteht dabei aus den Anfrageoperatoren, Verbindungen innerhalb der Operatoren und sogenannten Synopsen [The STREAM Group, 2003]. Die Anfrageoperatoren stehen dabei zu den drei Operatorenklassen der abstrakten Semantic von CQL in Beziehung. Die Verbindungen innerhalb der Operatoren dienen als Puffer und die Synopsen bewahren den Laufzeitstatus der zugehörigen Operatoren. Beispielsweise können sie aus Listen bestehen, welche die Informationen des letzten Zeitstempels beinhalten. Zur effizienten Ressourcenschonung können sich auch mehrere Operatoren die gleichen Synopsen teilen. Da CQL bei der Bearbeitung der Anfragen von einer geordneten Ankunft von Tupeln der Datenströme ausgeht, muss bei der Umsetzung sichergestellt werden, dass bis zu einem Zeitpunkt alle Informationen von den Quellen an den Server geschickt wurden. Dies wurde in STREAM so umgesetzt, dass die Quellen in regelmäßigen Zeitabständen einen sogenannten *Herzschlag* (Heartbeat) zum Server schicken, welcher ausschließlich den Zeitstempel τ beinhaltet. Dies wird auf dem Server so interpretiert, dass alle anschließend eintreffenden Informationen über diesen Datenstrom mit einem Zeitstempel versehen sind, der größer als τ ist. Aus diesem Grund können die Berechnungen für den Zeitstempel τ durchgeführt werden, ohne die Gefahr, dass nachträglich Events eintreffen, die noch berücksichtigt hätten werden müssen. Dabei wird der kleinste Zeitstempel verwendet, der von allen Quellen geschickt wurde, wobei soweit bekannt die Verzögerung der Übertragung von den Zeitstempeln abgezogen wird. Durch die Transformation in ein relationales Schema gehen jedoch die zeitlichen Abfolgen verloren und müssen bei Bedarf künstlich ergänzt werden.

Der resultierende Ausführungsplan kann anschießend grafisch aufbereitet werden, wie beispielhaft in Abbildung 3.3 dargestellt ist [Arasu et al., 2004]. Das grafische Interface bietet neben der Darstellung des Plans auch die Möglichkeit, die Ausführung zu überwachen und Anfragepläne anzupassen. Dazu bietet STREAM ein Webinterface über HTTP, welches es ermöglicht, einerseits neue Anfragen zu erstellen und sich andererseits Ergebnisse präsentieren zu lassen.

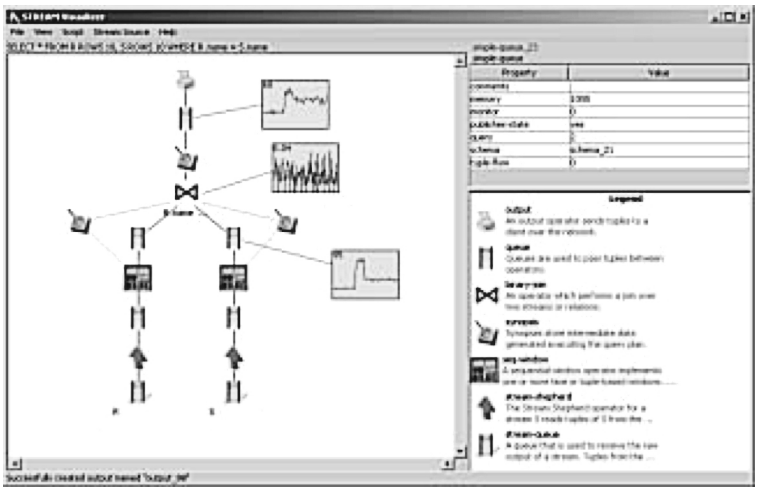


Abbildung 3.3: *Grafische Oberfläche von STREAM (Quelle: [Arasu et al., 2004])*

3.1.9 GEM

GEM (Generalized Event Monitoring Language) ist eine Beschreibungssprache zur Erkennung von komplexen Events und wurde zum Monitoren von Systemen konzipiert, bei denen Aktivitäten und Statusänderungen als Events registriert werden. Ein besonderer Fokus liegt hierbei auf der Erfassung und Berücksichtigung der zeitlichen Abfolge, in der Events auftreten [Mansouri-Samani, 1995]. Ein Event ist dabei spezifiziert durch einen Typ, einen Zeitstempel, wann dieses stattgefunden hat, eine Identifikation, an welcher Quelle das Event aufgetreten ist und eine Liste von spezifischen Parametern. Anhand des Eventtyps ist dabei definiert, welche Parameter enthalten sind. Grundsätzlich wird zwischen primitiven Events und komplexen unterschieden. Die primitiven werden von den Quellen erfasst und an das Verarbeitungssystem übergeben, welches anhand von Regeln diese in Form von komplexen Events kombiniert. Diese Events beinhalten neben den zuvor aufgelisteten Eigenschaften Informationen über den Zeitstempel des ersten und des letzten primitiven Events, aus dem es zusammengesetzt ist. Darüber kann die Zeitspanne ermittelt werden, welche das komplexe Event umfasst. Als Operatoren werden *Selektion* und *Komposition* angeboten. Im Speziellen werden der *Und-Operator*, bei dem beide Events auftreten müssen, der *Sequenz-Operator*, welcher zudem noch eine Abfolge der beiden Events bestimmt, der *Oder-Operator*, bei dem eines der beiden Events auftreten muss, und der *Verzögerungs-Operator*, der die Detektion nach einem definierten Zeitintervall weiterleitet, angeboten. Zudem gibt es eine erweiterte Form des Sequenz-Operators, mit Hilfe dessen noch spezifiziert werden kann, dass zwischen den beiden Events ein vordefiniertes Event nicht eintreffen darf [Mansouri-Samani und Sloman, 1997].

Für die Erfassung von Events werden Spezifikationen in Form von Regeln erstellt, welche analog zu den Events durch eine Identifikation eindeutig spezifiziert werden. Diese Regeln bestehen aus einem Zeitfenster, in dem sie aktiv sind, der Beschreibung des komplexen Events, auf welches sie reagieren sollen und einer Aktionssequenz, die bei der

Erkennung des komplexen Events ausgelöst werden soll. Mit Hilfe der eindeutigen Identifikation können sowohl Regeln als auch Events aktiviert, deaktiviert oder gelöscht werden. Neben der Weiterleitung der Aktionen an externe Programme können die Erfassungen von komplexen Events auch als interne Trigger innerhalb des Systems verwendet werden. Für die Verarbeitung der Regeln werden diese ähnlich dem Ansatz bei mathematischen Funktionen in Form von Bäumen ausgewertet.

Wie bereits zuvor erwähnt, stand bei der Entwicklung von GEM die zeitliche Erfassung von Events im Vordergrund. Dabei wurde ein besonderer Fokus auf die Verzögerung bei der Übertragung gelegt, weshalb es trotz synchronisierter Uhren bei allen Systemen zu einer ungeordneten Ankunft der Events am Hauptsystem kommen kann. Um dieses Problem zu umgehen werden die Events vor der eigentlichen Weiterleitung im System zurückgehalten. Dabei wird vorausgesetzt, dass die maximale Verzögerung bei der Übertragung bekannt ist und dies als Pufferzeit im Verzögerungsoperator gewählt. Damit wird sichergestellt, dass die weitere Verarbeitung alle Events in geordneter Reihenfolge erhält.

3.1.10 Cayuga

Cayuga ist ein CEP-System, welches mit dem Hintergrund entwickelt wurde, einerseits auf einer wohlspezifizierten Algebra aufzubauen und andererseits eine optimierte Variante einer Implementierung zur Verfügung zu stellen, so dass eine sehr große Anzahl von Events trotz mehrerer registrierter Anfragen mit möglichst kurzer Verzögerung bearbeitet werden kann [Brenna et al., 2007]. Dabei orientierte sich die Entwicklung an den Verarbeitungsmethoden zum Filtern von XML-Dateien. Insgesamt umfasst die Algebra von Cayuga vier unäre und drei binäre Operatoren. Der *Projektions-Operator* ermöglicht es Werte innerhalb von Events zu verändern. Der *Auswahl-Operator* kann verwendet werden, um Events nach spezifischem Inhalt zu filtern. Mit dem *Umbenennungs-Operator* kann das Schema der Events verändert werden. Der *Aggregations-Operator* operiert über eine Sequenz von Events. Der *Vereinigungs-Operator* ermöglicht es zwei Datenströme mit dem gleichen Schema zu fusionieren. Mit dem *konditionalen Sequenz-Operator* können Elemente von zwei Datenströmen miteinander fusioniert werden, die gleiche Werte eines spezifizierten Attributs aufweisen. Schließlich wird der *Iterations-Operator* angeboten, der eine Aufeinanderfolge von konditionalen Sequenz-Operatoren darstellt.

Für die Optimierung der Implementierung wurde die Verarbeitung der einzelnen Schritte in Form eines endlichen Automaten realisiert, dessen Knoten die einzelnen Anfrageoperatoren darstellen [Demers et al., 2006]. Anfragen können über eine Webschnittstelle mittels der vorgegebenen Algebra erstellt werden. Diese werden anschließend in den Automaten integriert, wobei mögliche Teilpfade gefunden und miteinander kombiniert werden. Die Anfragen sind dabei ähnlich zu denen in SQL aufgebaut. Des Weiteren beinhaltet der Automat eine Indizierung der eintreffenden Events, um diese in den Operatoren bei späteren Übereinstimmungen mit den Anfragen schneller zuweisen und verarbeiten zu können.

Das eigentliche System wurde in C++ entwickelt. Daneben wurde mittels Python die Webschnittstelle zum Erstellen der Anfragen generiert. Sowohl der resultierende Automat als

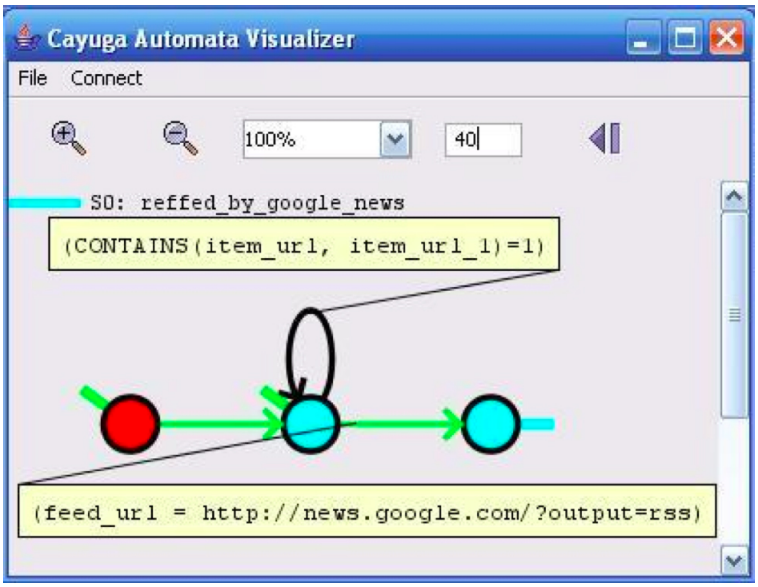


***Abbildung 3.4**: Grafische Oberfläche von Cayuga (Quelle: [Brenna et al., 2007])*

auch die einzelnen Verarbeitungsschritte bei eintreffenden Events können in externen Dateien protokolliert werden. Mittels einer Java-Anwendung ist es damit möglich, den endlichen Automaten inklusive einer Darstellung von Durchläufen einzelner Events zu visualisieren (siehe Abbildung 3.4).

3.1.11 RACED

RACED (Rate-Adaptive Complex Event Detection) ist ein CEP-System, welches basierend auf inhaltsbasierten Publish/Subscribe-Systemen eine Erfassung von komplexen Events ermöglicht [Cugola und Margara, 2009]. Die Architektur umfasst einerseits die Quellen, die die Informationen in Form von Events bereitstellen und Abonnenten, die sich für verarbeitete Daten registrieren. Die eigentliche Verarbeitung läuft dezentral auf mehreren Systemen, die über Vermittler miteinander verbunden sind. Durch „Systemadministratoren“ können die Ausführungen überwacht und angepasst werden. Dabei können die Definitionen von komplexen Events mittels einer API spezifiziert werden.

Als verarbeitende Funktionalitäten werden der Filter-, der Kompositions- und der Parameter-Operator angeboten. Der erste ermöglicht es, Events nach deren Inhalt herauszufiltern. Über den Kompositions-Operator können mehrere Eventfilter miteinander kombiniert werden. Mittels der Parameter-Operatoren können Informationen in mehreren Events miteinander in Verbindung gebracht und diese dadurch verglichen werden. Mit Hilfe eines Fenster-Operators können zeitliche Aspekte festgelegt werden, die bei der Auswertung von komplexen Events berücksichtigt werden sollen. Schließlich können neue Eventtypen spezifiziert und erzeugt werden.

Zur Datenübermittlung zwischen den Vermittlern können einerseits alle Informationen direkt, d.h sobald sie erstellt wurden, weitergeleitet werden (push). Und andererseits können die Daten auch nur durch explizite Abrufe erfragt werden (pull), womit die Netzauslastung geschont werden soll. Hierzu wird eine Historie angelegt, die fortlaufend erweitert wird. Je nach Auslastung des Netzwerks können die Broker sich selbstständig konfigurieren und zwischen push und pull wechseln.

3.1.12 T-REX / TESLA

TESLA [Trio-based Event Specification LAnguage] zielt auf die zeitnahe Verarbeitung einer großen Anzahl an Informationen ab mit dem Hintergrund, ein hohes Maß an Ausdrucksfähigkeit bei einer einfachen Anfragesprache zu bieten [Cugola und Margara, 2010]. Sie baut auf Prädikatenlogik erster Stufe auf, welche in TRIO [Ghezzi et al., 1990] spezifiziert wurde. Dadurch wurde eine Spezifikation erreicht, um im Speziellen Sequenzen von Events zu detektieren. Als Operatoren werden Selektion, Parametrisierung, Negation, Aggregation, Sequenzen und Iteration angeboten, wobei die Selektion basierend auf vorgegebenen Werten erfolgt und die Parametrisierung einen dynamischen Abgleich zwischen zwei Werten von unterschiedlichen Events ermöglicht. Des Weiteren ist es möglich, einzelne Regeln für Eventselektion und -konsum zu erstellen. Dies bedeutet, dass explizit ausgewählt werden kann, welche Events bei mehreren möglichen Eventpaaren berücksichtigt werden - alle zutreffenden, die n ersten oder die n letzten. Des Weiteren kann angegeben werden, ob dieses Event für eine spätere Verarbeitung noch zur Verfügung stehen soll oder nicht.

Umgesetzt wurde Tesla im System *T-Rex*, welches als CEP System entwickelt wurde [Cugola und Margara, 2012a]. An dieses können sich Empfänger per Anfragen registrieren. Ähnlich dem Publish/Subscribe-Ansatz bekommen sie anschließend Informationen gemäß der Anfragen zugesendet, sobald diese vorhanden sind. Neben dem Typ, welcher Anzahl und Art der beinhalteten Parameter definiert, enthält jedes Event einen Zeitstempel, an dem es detektiert wurde. Das eigentliche System ist als Automat aufgebaut, welcher die eintreffenden primitiven Events gemäß der spezifizierten Regeln zu komplexen Events verarbeitet und diese weiter verteilt. Jede Regel wird dabei in ein Automatenmodell transferiert, welches aus einem oder mehreren Sequenzmodellen besteht. Sobald ein neues Event eintrifft, wird eine entsprechende Sequenz generiert, die die Abfolge des Automaten beschreibt, welche das Event durchlaufen muss. Um mit einer großen Anzahl von Events agieren zu können werden diese beim Eintreffen nach der Transformation in einen Puffer geschrieben. Ist dieser voll gehen die weiteren Events verloren. Zur Optimierung kann die Größe des Puffers a priori festgelegt werden. Ist er zu klein, gehen mehr Events verloren. Ist er zu groß, kann die Bearbeitung zu viel Zeit benötigen und hierdurch eine lange Verzögerung entstehen.

Tesla ist in C++ entwickelt und bietet Schnittstellen für Java und C++ an, welche anhand von Adaptern mit dem Verarbeitungssystem verbunden sind. Die eigentliche Verarbeitung läuft auf einem zentralen System. Ähnlich zu Cayuga wurden Indexierungsmechanismen verwendet, um die Verarbeitungsgeschwindigkeit zu optimieren. Des Weiteren wird über mehrere Threads die Bearbeitung von Anfragen parallelisiert. Die Optimierungen wurden

spezifisch hinsichtlich des Ziels entwickelt, Eventsequenzen bei einer großen Anzahl von eintreffenden Events zu erfassen, zu verarbeiten und an die Empfänger zu verteilen.

3.2 Dual Reality Visualisierungskomponenten & Managementtools

Die Kombination von Informationen über und aus der realen Umgebung und einem virtuellen Modell, inklusive einer bi-direktionalen Beeinflussung, wurde bereits in mehreren Systemen integriert. Im kommenden Abschnitt werden mehrere Beispiele hierzu vorgestellt und detailliert beschrieben. Die Verbindung zwischen der physischen Umgebung und ihrer Virtualisierung wird dabei einerseits zur Darstellung von Kontextfaktoren, wie beispielsweise Lichtintensität und Lautstärke, verwendet und andererseits als realitätsnahe Testumgebung für Verarbeitungssysteme sowie als Monitoring- und Steuerungstool. Zudem wird ein System beschrieben, welches Benutzerverhalten aus der realen Umgebung aufnimmt, um dieses als realistisches Verhaltensmuster für virtuelle Charaktere zu verwenden. Zuletzt wird ein Tool vorgestellt, welches eine einfache Modellierung von Smarten Umgebungen ermöglicht und durch entsprechende Schnittstellen Informationen bi-direktional mit den Systemen in der Smarten Umgebung austauschen kann.

3.2.1 Repräsentation von Kontextfaktoren

Lifton, der den Begriff Dual Reality definiert hat, präsentierte sein Paradigma mit Hilfe einer prototypischen Installation. Als physische Komponente in der realen Welt entwickelte er eine instrumentierte Steckerleiste (Plug), welche mit diversen Sensoren und Aktuatoren ausgestattet ist, die über einen verbauten Mikrocontroller gesteuert werden können [Lifton, 2007]. Die Steckerleiste umfasst insgesamt vier standardmäßige Stromsteckdosen mit

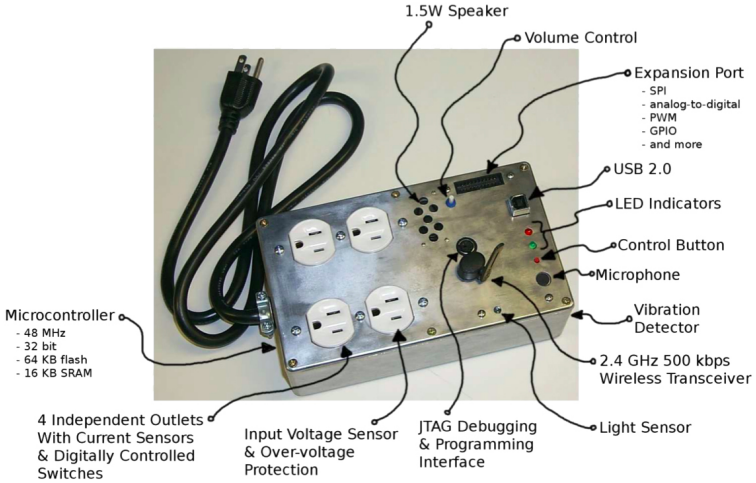


***Abbildung 3.5**: Darstellung der instrumentierten Steckerleiste von Lifton (Quelle: [Lifton, 2007])*

digitalen Schaltern zur Regulierung der Stromzufuhr, Lautsprecher und LED-Leuchten, die als Aktuatoren dienen. Zudem wurden neben einem Knopf weitere Sensoren verbaut, die den durchfließenden Strom messen, den Lichteinfall erfassen, Vibration detektieren, Geräusche wahrnehmen und die Temperatur messen. Die Programmierung der instrumentierten Steckerleiste erfolgt über eine Programmierschnittstelle, welche gleichzeitig zur Fehlersuche bei der Ausführung verwendet werden kann, indem ein Rechner an diese Schnittstelle angeschlossen wird. Abbildung 3.5 zeigt den Aufbau des Sensor-Aktuator-Verbunds.

Zur Repräsentation der erfassten Sensorinformationen dient ein sogenannter „Datenteich“ (Data Pond), welcher in Second Life (siehe Abschnitt 2.4.2) visualisiert wird. Abbildung 3.6 (links) zeigt die Darstellung dieses Datenteichs. Dieser umfasst einen Tümpel mit einem gewissen Durchmesser (in der Visualisierung blau dargestellt), mehrere grüne bewegliche Halme und ein Feuer in dessen Mitte. Veränderungen in der realen Welt werden mittels der Sensoren erfasst und in die virtuelle Umgebung transferiert, wo sie die Darstellung der einzelnen Teile des Datenteichs beeinflussen. Der erfasste Lichteinfall beeinflusst hierbei die Länge der Halme und die gemessene Temperatur deren Farbe. Die Bewegung der Halme in der virtuellen Repräsentation passt sich der erfassten Bewegung in der physischen Umgebung an. Die Geräuschlautstärke spiegelt sich in der Größe des Tümpels wider und der aktuell verwendete elektrische Strom beeinflusst die Intensität des Feuers. Anhand dieses Morphismus kann der aktuelle Kontext und damit das Aktivitätslevel im Bereich der Steckerleiste durch die Betrachtung des virtuellen Datenteichs erfasst werden.

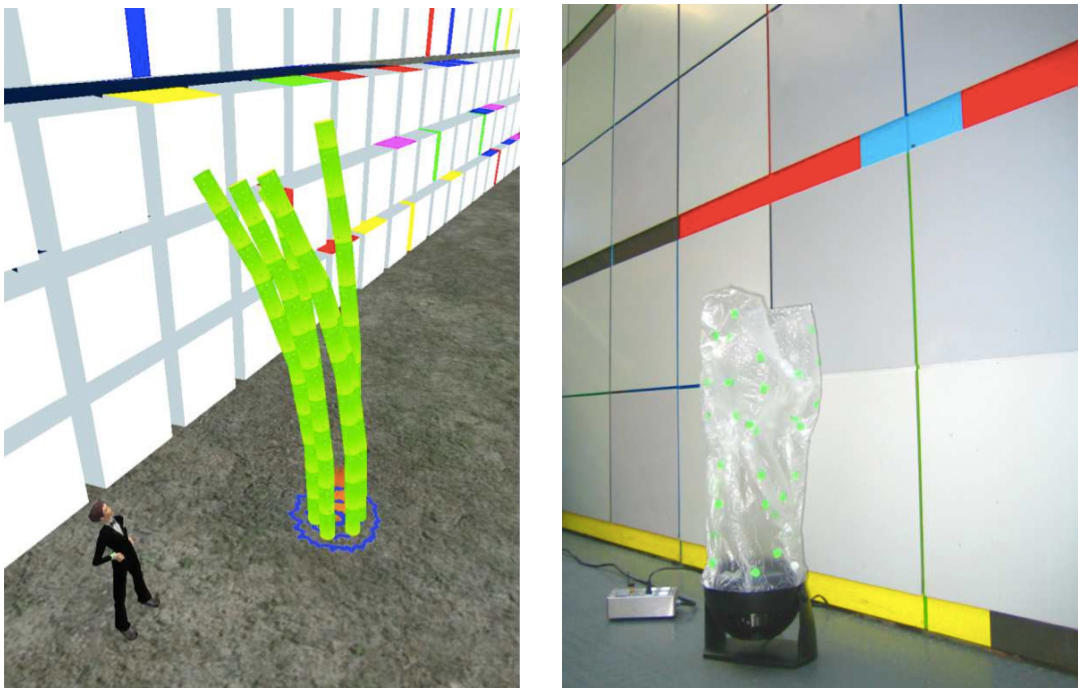


***Abbildung 3.6**: Virtueller Datenteich mit Avatar (links) zur Repräsentation von erfassten Kontextfaktoren und reales Gegenstück (rechts), Quelle: [Lifton und Paradiso, 2009]*

Um die Beeinflussung der realen Welt durch Interaktionen mit dem virtuellen Modell zu präsentieren, wurde ein Ventilator an die Steckerleiste angeschlossen, an den ein gepunktetes, sonst durchsichtiges Plastiklaken angebracht ist (siehe Abbildung 3.6, rechts) [Lifton et al., 2009]. Durch eine Pulssteuerung des Stroms kann die Intensität des Ventilators gesteuert werden und damit die Bewegung des Lakens, welches mittels der Punkte besser zu erkennen ist. Neben der Bewegung wird implizit die Lautstärke in der Umgebung durch die Aktivierung des Ventilators verändert. Die Intensität des Ventilators spiegelt hierbei die Distanz eines Avataren zum virtuellen Datenteich in Second Life wider. Somit kann die Aktivität im Virtuellen durch den physischen Ventilator repräsentiert werden.

3.2.2 Twin-World Mediator

Brandherm et al. entwickelten einen Ansatz, mit dessen Hilfe es ermöglicht wird, existierende sensorbasierte Systeme im Virtuellen zu simulieren und damit zu evaluieren [Brandherm et al., 2008a]. Bei diesem Ansatz wird - wie bereits bei Lifton - Second Life (SL) zur Darstellung der virtuellen Umgebung eingesetzt. Die grundsätzliche Idee bei diesem Ansatz ist, dass Sensoren mittels einer Simulation in SL nachgebildet werden können. Die daraus generierten Sensorinformationen werden anschließend an das zu testende System übergeben, dessen Resultat wiederum zurück in die virtuelle Welt gespiegelt und dort visuell repräsentiert wird. Für die Kommunikation mit SL wird die Linden Scripting Language (LSL) verwendet. Dabei werden benötigte Skripte erstellt, die im sogenannten *Twin-World Mediator* integriert werden [Ullrich et al., 2008]. Dieser dient als Schnittstelle zwischen der realen Welt und SL. Neben den Skripten umfasst der Twin-World Mediator eine Datenbank, welche notwendige Informationen aus der virtuellen Welt zwischenspeichert. Beispielsweise werden alle statischen Objektinformationen einmalig bei der Initialisierung der Szene kommuniziert und in dieser Datenbank zwischengespeichert, um Ressourcen zu sparen. Bei beweglichen Objekten hingegen werden deren Position und Orientierung in vorher festgelegten zyklischen Zeitabständen aus SL an den Twin-World Mediator mittels einfacher SLS Skripte geschickt [Prendinger et al., 2009]. Neben der Möglichkeit, Informationen aus SL in die reale Welt zu übermitteln, werden Schnittstellen nach SL angeboten, so dass ein bi-direktionaler Informationsaustausch gewährleistet wird.

Demonstriert wird der Twin-World Mediator an einem Positionierungssystem [Brandherm et al., 2008b]. Dieses System verwendet Sensorinformationen, die einerseits von RFID-Tags und andererseits von Infrarotbaken ausgestrahlt werden [Brandherm und Schwartz, 2005]. Basierend auf den erfassten Sensorinformationen, welche Position, Ausrichtung und Signalstärke der empfangenen Sensorsignale enthalten wird durch einen entsprechenden Positionierungsalgorithmus die Position des Empfangsmoduls ermittelt. Bei der Simulation in SL dient ein virtueller PDA als Empfänger, welcher an einem Avatar angebracht ist. Daneben werden virtuelle RFID-Tags in SL positioniert. Ein Simulationsalgorithmus berechnet anhand der Position und Orientierung des Avataren und der Sensoren die am PDA eintreffenden Sensorinformationen, welche an den Positionierungsalgorithmus weitergegeben werden. Dieser ermittelt daraus die Position des PDA und damit die des Avataren und sendet dieses Ergebnis an SL, wo es in Form einer transparenten Kugel dargestellt wird. Abbildung 3.7 zeigt einen Screenshot einer solchen Darstellung in SL.

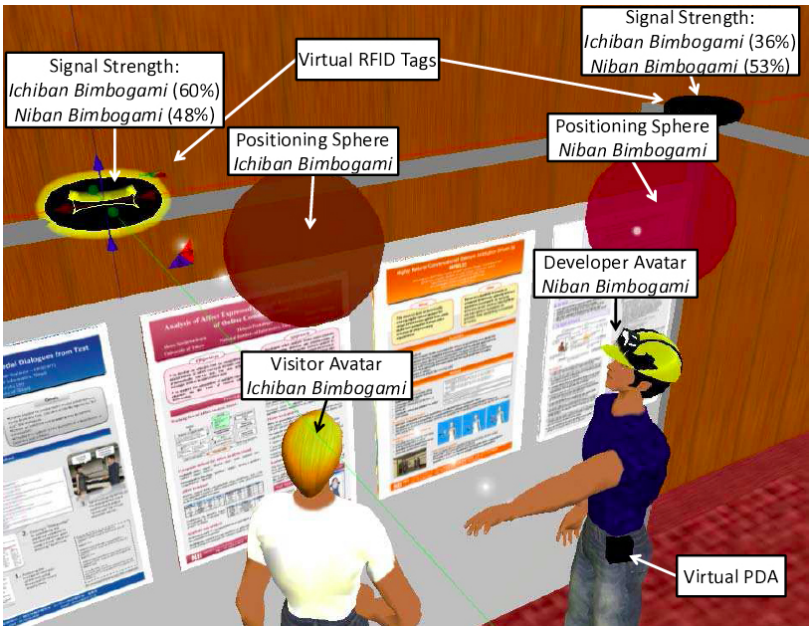


***Abbildung 3.7**: Second Life als virtuelle Testumgebung für Systeme zur Positionserkennung (Quelle: [Ullrich et al., 2008])*

Durch Hinzufügen weiterer virtueller Sensoren bzw. Umpositionieren der existierenden können verschiedene Installationen simuliert und der Positionierungsalgorithmus für diese Varianten evaluiert werden. Das Evaluationsergebnis ist dabei abhängig von der Genauigkeit der Simulation der Sensorerfassung, wie beispielsweise der Berücksichtigung von Objekten, die das Signal stören.

3.2.3 Eolus One

Eolus One ist eine in der Schweiz entwickelte X-Reality Plattform, welche eine sichere Übertragung und Darstellung von Gebäude-Sensordaten nach Second Life (SL) gewährleisten soll. Anstelle von SL kann auch die kostenlose Alternative OpenSimulator als eigene Applikation auf einem Server im Intranet verwendet werden. Die erfassten Daten der Gebäudeüberwachungssensorik werden bei diesem Ansatz als Events erfasst und mittels der 3D-Darstellung visualisiert, um den aktuellen Status des Gebäudes repräsentieren zu können. Unter anderem umfasst das Konzept Sensorinformationen wie Temperatur, Luftfeuchtigkeit und Öffnungsstatus von Türen. Im Speziellen sollen damit die Informationen in virtuelle Kommandozentren transferiert werden, welche es Experten ermöglichen, auch aus großer Entfernung mögliche Fehlerquellen zu identifizieren und entsprechend geeignetes Personal zur Problembeseitigung anweisen zu können. Neben der reinen Überwachung sind auch Schnittstellen zur Gebäudesteuerung gegeben, um beispielsweise die Heizung, die Klimaanlage oder das Licht über die interaktive 3D-Darstellung zu verändern, was eine Möglichkeit der Energieersparnis darstellen kann [Coleman, 2009]. Eolus One wurde entwickelt,

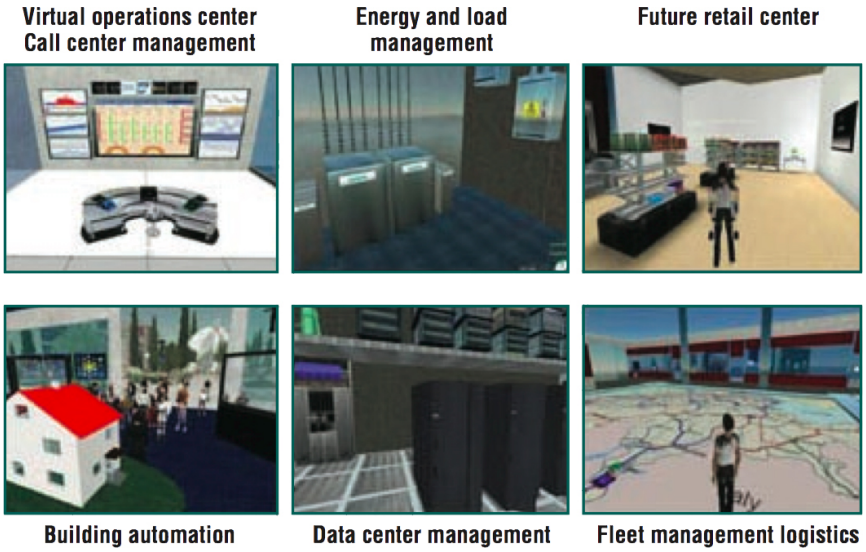


***Abbildung 3.8**: Darstellung des Eolus One Park in Second Life (Quelle: [Coleman, 2009])*

um in mehreren Anwendungsszenarien eingesetzt zu werden, beispielsweise in zukünftigen Einkaufszentren, virtuellen Callcentern oder Datenzentren, wie in Abbildung 3.8 gezeigt.

Eine erste Realisierung zur Präsentation des Potenzials der X-Reality Plattform wurde in Form eines Puppenhauses realisiert. Dieses wurde mit diversen Sensoren und Aktuatoren ausgestattet und eine entsprechende Abbildung in Second Life erstellt[2]. Unter anderem wird der Öffnungsstatus der Türen, ob das Licht ein- oder ausgeschaltet ist und die Temperatur im Puppenhaus erfasst und in SL virtuell repräsentiert. Dort kann man wiederum das Pendant zum Heizungsthermostat durch einen Avataren verändern, was gleichzeitig in das reale Modell zurück gespiegelt wird. Neben dem Puppenhaus wurde auch ein Gebäudeteil der Entwicklungsfirma mit dem System ausgestattet, so dass Zugriffskontrollen, Lichter, Fahrstühle, Strom, Klimaanlage und Brandschutzsystem remote aus dem sicheren virtuellen Kontrollraum in SL überwacht und gesteuert werden können [Coleman, 2009].

3.2.4 Virtuelle Schokoladenfabrik

Eine weitere Anwendung der Verbindung einer Smarten Umgebung und ihrer virtuellen Abbildung wurde anhand einer Schokoladenfabrik gezeigt [Back et al., 2010b]. Diese wurde virtuell nachmodelliert und in Multiverse, einer 3D-Plattform für Onlinespiele, dargestellt. Abbildung 3.9 stellt ein Foto der realen Fabrikumgebung (links) ihrer Virtualisierung in Multiverse (rechts) gegenüber. Die Schokoladenfabrik wurde mit dem Hintergrund errichtet, hoch technologisch Schokolade zu produzieren, weshalb bereits bei Konzeption und Bau mehrere Sensoren und Steuerungsmöglichkeiten integriert wurden. Hierzu zählen Temperatur- und Luftfeuchtigkeitsmesser, mehrere Überwachungskameras in der Fabrikhalle sowie in Maschinen und weitere Maschinensensoren. Die erfassten Sensorwerte werden

[2]http://www.theguardian.com/technology/2008/feb/26/internet.buildings, Zugriff: November 2014

über eine entsprechende Schnittstelle in die virtuelle Welt transferiert und dort grafisch repräsentiert, so dass eine Überwachung der Umgebung und der Maschinen realisiert wurde. Beispielsweise werden Sensordaten einer Maschine dargestellt, wenn man sich mit einem Avataren in deren Umgebung begibt. Geht man bis auf einen kurzen Abstand an die Maschine heran, wird das Live-Kamerabild präsentiert.

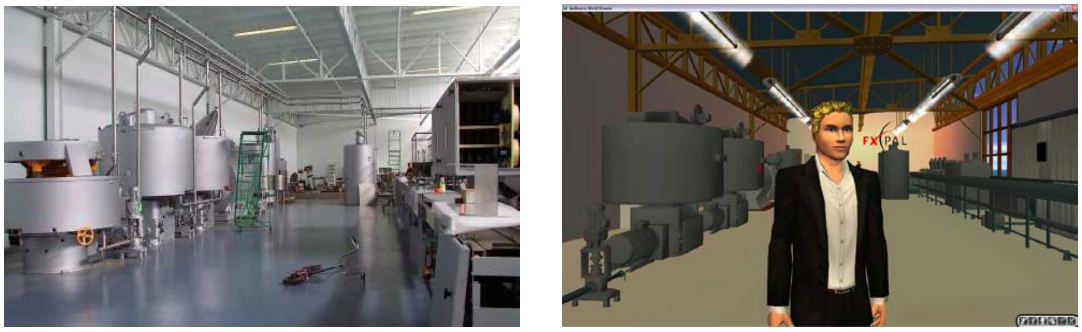


***Abbildung 3.9**: Darstellung der Schokoladenfabrik als reales Bild (links) und virtuelles Modell in Multiverse (rechts), Quelle: [Back et al., 2010a]*

Eingesetzt wird das System, um einerseits Kunden die Fabrik in Form eines virtuellen Rundgangs zu präsentieren und andererseits als Monitoring- und Steuerungsumgebung für Mitarbeiter. Aus diesem Grund wurde ein rollenbasierter Zugriff realisiert, um nur autorisierten Personen die Möglichkeit der Steuerung zu bieten [Back et al., 2010a]. Neben der Integration in die 3D-Darstellung werden Sensorinformationen auch in einer mobilen Applikation verwendet. In dieser ist es möglich, den Zustand aller Geräte sowie Live-Kamerabilder zu inspizieren. Neben der Darstellung und Überwachung der Fabrikanlage können auch einzelne Maschinen aus dieser Applikation heraus gesteuert werden. Beispielsweise können Mischmaschinen eingeschaltet, ausgeschaltet und deren Temperatur geregelt werden, ohne physisch vor Ort sein zu müssen. Zudem können Alarmmechanismen aktiviert werden, die als Push-Notifikationen zu der Applikation geschickt werden, damit entsprechende Handlungen initiiert und somit gewisse Automatismen pro-aktiv geschaltet werden.

3.2.5 iWorlds

Davies und Callaghan untersuchten die Möglichkeit, menschliches Verhalten durch Sensoren zu erfassen und zu erlernen, um dieses für die Ansteuerung von virtuellen, autonom agierenden Avataren zu verwenden [Davies und Callaghan, 2012]. Das Hauptziel dabei ist, dass die Avatare menschenähnliches Verhalten aufweisen, so dass diese in Spielen eingesetzt werden können, ohne dass reale Spieler sie als simulierte Agenten wahrnehmen. Da die Entwicklung und Implementierung von menschlichen Verhaltensmustern komplex, zeitaufwändig und kostenintesiv ist, wurde die Idee verfolgt, dass Interaktionen von Benutzern mit einer virtuellen oder realen Umgebung erfasst und gespeichert werden, so dass diese von virtuellen Avataren nachgeahmt werden können. Zur Erfassung des Nutzerverhaltens werden einerseits Interaktionen von Benutzern mit einem virtuellen Modell aufgezeichnet und andererseits

eine instrumentierte Umgebung, der sogenannte *iSpace*[3], als Interaktionsmedium verwendet. Der iSpace beinhaltet insgesamt vier Räume (Wohnzimmer, Arbeitszimmer, Schlafzimmer und Bad), welche mit einer Vielzahl an Sensoren und Aktuatoren ausgestattet sind, die über ein UPnP-Framework (Universal Plug and Play) gesteuert werden können. Abbildung 3.10 (links) zeigt einen Ausschnitt des Wohnzimmers, in dem auch eine Küchenzeile integriert ist.

***Abbildung 3.10**: Virtuelle Darstellung (rechts) einer instrumentierten Wohnung (links) bei iWorlds, Quelle: [Davies und Callaghan, 2012]*

Als Visualisierungsumgebung für das virtuelle Modell des iSpace wurde RealXtend[4] verwendet, welches ein Ableger von OpenSimulator ist, jedoch die Möglichkeit bietet, extern erstellte 3D-Objekte in das Modell zu laden. Für die Repräsentation der Möbelstücke dienten 3D-Modelle, die kostenlos im Google 3D Warehouse, einer Online-Sammlung von Google SketchUp Modellen[5], zur Verfügung standen. Die virtuellen Objekte wurden mittels der Sensoren und Aktuatoren mit den realen Gegenstücken verknüpft. Beispielsweise konnte durch Interaktion im Modell das Licht im iSpace gesteuert werden. Für die Synchronisation zwischen den beiden Welten wurde eine Schnittstelle erstellt, die Informationen zwischen dem UPnP Framework und RealXtend austauschte. Durch dieses Vorgehen können Benutzer sich natürlich in der Umgebung verhalten während deren Interaktionen erfasst, gespeichert und im virtuellen Modell dargestellt werden. Ebenso werden Benutzerinteraktionen mit dem Modell protokolliert, indem sie einen Avatar darin steuern. Beide Ansätze - Interaktion mit der physischen oder der virtuellen Umgebung - dienten dem gleichen Ziel, dass das Verhalten protokolliert wird und daraus ein „nachgeahmtes“ Verhalten für virtuelle Avatare erstellt werden kann.

3.2.6 YAMAMOTO

Für eine Fußgänger-Navigation in Gebäuden müssen deren Räume, Flure, Treppen und Fahrstühle entsprechend modelliert werden. Aufgrund der verschiedenen Ebenen ist dies

[3] http://dces.essex.ac.uk/Research/iieg/idorm2/, Zugriff: November 2014
[4] http://realxtend.org, Zugriff: November 2014
[5] http://sketchup.google.com/3dwarehouse/?hl=de&hl=de, Zugriff: November 2014

mit standardisierten Tools oftmals nur schwierig realisierbar. Aus diesem Grund wurde ein Framework entwickelt, welches eine einfache Möglichkeit der Modellierung von mehrstöckigen Gebäuden unterstützt [Stahl und Haupert, 2006; Stahl, 2009]. Diese Modellierungsumgebung ist unter dem Begriff YAMAMOTO (Yet Another Modeling Toolkit) bekannt und ist in der Programmierungssprache C# implementiert. Basierend auf einem Grundrissplan, welcher als Bild in das System geladen wird, kann die Struktur des Gebäudes in Form von Polygonen modelliert werden. Die Polygone enthalten dabei spezifische Eigenschaften, wie beispielsweise, ob diese von Personen passiert werden können oder nicht. Angrenzende Polygone teilen sich dabei die entsprechenden Punkte und Kanten. Wie die Polygone enthalten auch die Kanten spezifische Eigenschaften, die unter anderem spezifizieren, um welche Art von Verbindung es sich zwischen den beiden Polygonen handelt (z.B. Tür, Fenster, offen, Wand) und in welcher Richtung diese passierbar ist. Mehrere Ebenen von Gebäuden können über zusätzliche Polygone miteinander verbunden werden, die als Treppe, Schräge oder Aufzug modelliert werden können. Basierend auf der Modellierung und den Passierungseigenschaften der Kanten und Polygone kann ein Verbindungsgraph für begehbare Pfade erstellt werden. Dieser wird in Kombination mit einem A*-Algorithmus herangezogen, um eine Route zwischen zwei Punkten im Modell zu ermitteln, die als Navigationspfad verwendet werden kann.

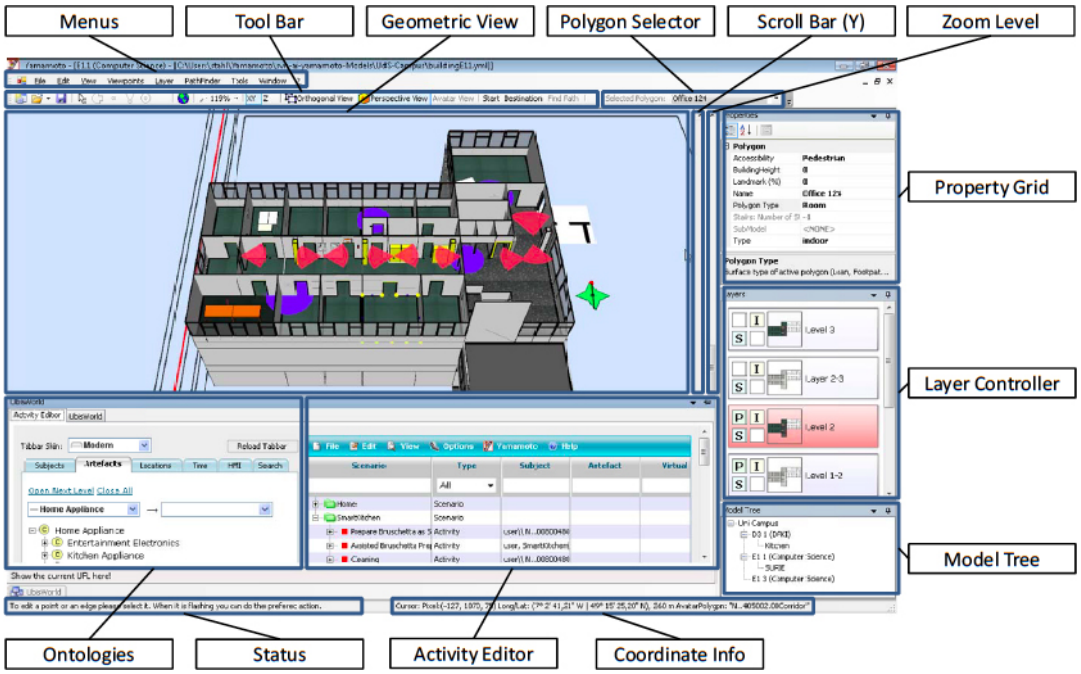


***Abbildung 3.11**: Ausschnitt aus dem Modellierungstool* YAMAMOTO *(Quelle: [Stahl, 2009])*

Zudem können Sensoren in das Modell integriert werden. Um den Wirkungsbereich dieser Sensoren zu repräsentieren, können sie als Punktquellen (z.B. für RFID Tags), Segmente (z.B. für Infrarotbaken) oder Kreise (z.B. für Bluetooth) integriert werden. Die Modellierung kann dabei in 2D erfolgen. Durch Höhenangaben bei den Kanten und den

Sensoren wird daraus eine 3D-Darstellung gerendert, die ebenfalls editiert werden kann. Weitere Objekte, wie beispielsweise Möbelstücke, können in Form von Boxen erstellt werden, die ineinander geschachtelt werden können. Die Abspeicherung erfolgt im eigens hierfür definierten yml-Format, welches sich an XML orientiert, so dass die Modellierung auch manuell in einem Texteditor inspiziert und angepasst werden kann. Abbildung 3.11 zeigt die grafische Oberfläche von YAMAMOTO [Stahl, 2009]. Neben der Darstellung in YAMAMOTO können die Modelle in die beiden Formate VRML (Virtual Reality Modeling Language) (siehe Abschnitt 2.4.2) und RoSiML (Robot Simulation Modeling Language)[6] exportiert werden, um beispielsweise auch auf mobilen Endgeräten dargestellt werden zu können.

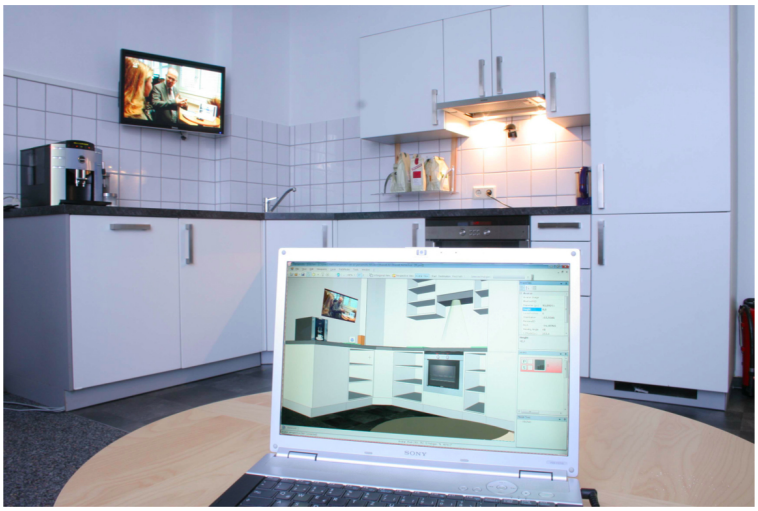


***Abbildung 3.12**: Vergleich real virtuell bei* YAMAMOTO *(Quelle: [Stahl et al., 2011])*

Neben der reinen Modellierung umfasst YAMAMOTO Schnittstellen zu Sensoren und Aktuatoren in der physischen Umgebung. Das Ziel dieser Integration besteht darin, die physische Umgebung mit dem virtuellen Gegenstück zu synchronisieren. Beispielsweise wird der Zustand von Lichtern und Bildschirminformationen unter Verwendung des VNC-Protokolls in YAMAMOTO repräsentiert. Interaktionen mit diesen Repräsentationen haben gleichzeitig einen Einfluss auf die physischen Aktuatoren. Beispielsweise kann dadurch das Licht an- und ausgestellt werden. Abbildung 3.12 zeigt eine instrumentierte Küche und die virtuelle Repräsentation in YAMAMOTO. Für die Ansteuerung der Aktuatoren verwendet YAMAMOTO eine Anbindung an einen sogenannten *Universal Control Hub* (UCH) [Zimmermann und Vanderheiden, 2007; Frey et al., 2011]. Der UCH baut auf dem *Universal Remote Console* (URC) Standard [ISO, 2008] auf, welcher einen Fokus auf der Erreichbarkeit von Controllern beinhaltet inklusive geeigneter Benutzeroberflächen und -schnittstellen für die jeweils verwendeten Systeme. Neben der Synchronisation zwischen physischer Umgebung und virtuellem Modell sind auch Verbindungen zwischen mehreren physischen Umgebungen angedacht. Unter anderem besteht eine Idee der Synchronisation des angeschalte-

[6] http://www.informatik.uni-bremen.de/spprobocup/RoSiML.html, Zugriff: November 2014

ten TV Senders oder Lichtern zwischen den Umgebungen, um trotz der Distanz eine gewisse soziale Verbindung der Benutzer darzustellen. Dieses Konzept ist unter dem Begriff *Synchronisierte Realitäten* bekannt [Stahl et al., 2011].

3.3 Zusammenfassung

In diesem Kapitel wurden mehrere Systeme zur Kommunikation und Verarbeitung von Informationen sowie Visualisierungssysteme, welche nach dem Dual Reality Paradigma aufgebaut sind, beschrieben. Aufgrund von Spezialisierung hinsichtlich der Anwendungsdomäne hat jedes der vorgestellten Systeme seine Vorzüge entsprechend der jeweiligen Bedingungen. Der Event Broadcasting Service (EBS) und das Dual Reality Management Dashboard (DuRMaD), welche im Rahmen dieser Arbeit entwickelt wurden und in den folgenden Kapiteln detailliert beschrieben werden, wurden ebenso gemäß spezialisierten Anforderungen konzipiert. Unter anderem ist eine Anforderung, dass diese Systeme möglichst einfach einzurichten und zu verwenden sind. Gleichzeitig sind sie jedoch generisch und modular aufgebaut, so dass sie in mehreren Anwendungsdomänen eingesetzt werden können. Im Folgenden werden die Gemeinsamkeiten und Unterschiede der zuvor aufgeführten Systeme mit dem EBS und dem DuRMaD erläutert.

3.3.1 Kommunikationsinfrastrukturen im Vergleich

Im Hinblick auf eine einfache Anwendbarkeit des Systems sowie eine Anpassung auf geforderte Einsatzdomänen inklusive derer Anforderungen wurden die vorgestellten Systeme auf folgende Faktoren und Eigenschaften hin untersucht:

- *Datenübertragung*. Die Art der Informationsdarstellung beeinflusst sowohl die Übertragung als auch die Verarbeitung. Während Tupel generische Konstrukte sind, haben Events vorgegebene Strukturen, die in der Regel a priori bekannt sind. Bei der Übermittlung in Form von Datenströmen müssen diese zuerst in ein relationales Schema transferiert werden, wobei die zeitliche Reihenfolge des Eintreffens der Informationen nur durch eine „künstliche“ Erweiterung verwendet werden kann, da Relationen standardmäßig keiner Ordnung unterliegen.

- *Informationsverarbeitung*. Bei der Informationsverarbeitung kann zwischen zentralen und verteilten Systemen unterschieden werden. Während bei den zentral agierenden Systemen die gesamte Verarbeitung auf einem System erfolgt, wird bei der verteilten Informationsverarbeitung ein Netzwerk aus Systemen zur Bearbeitung der Informationen verwendet.

- *Fehlererkennung*. In verteilten Umgebungen können sich Systemfehler über mehrere Systeme erstrecken und somit eine Ursachenidentifikation erschweren. Aus diesem Grund ist eine Fehlererkennung, beispielsweise dass Systeme nicht mehr ordnungsgemäß funktionieren, bereits als integraler Bestandteil der Kommunikationsinfrastruktur hilfreich.

- *Sicherheit / Verschlüsselung*. Für den Einsatz in öffentlichen und halböffentlichen Umgebungen müssen gewisse Sicherheitsmechanismen zum Schutz von vertraulichen und privaten Informationen bereits bei der Kommunikation von Informationen berücksichtigt werden (siehe Abschnitt 2.2.3).

- *Datenverteilung*. Die generierten und zu übermittelnden Informationen können entweder von Interessenten aktiv angefordert werden (pull) oder automatisch an diese weiter verteilt werden (push).

- *Sequenzverarbeitung*. Je nach Anwendung kann eine vorgegebene Reihenfolge von nacheinander auftretenden Events von Interesse sein und muss in einem solchen Fall detektiert werden können.

- *Eventtransformation*. Eine Vorverarbeitung durch eine Kombination zwischen mehreren Events und die Anpassung gewisser Parameter eines Events kann eine weiterführende Verarbeitung vereinfachen und Redundanzen minimieren.

- *Zeitoperatoren*. Analog zur Reihenfolge von Events kann die Erfassung von bestimmten Events in einem definierten Zeitfenster erforderlich sein. In diesem Fall müssen entsprechende Zeitoperatoren für die Eventverarbeitung zur Verfügung stehen.

- *Aggregation*. Eine Analyse von mehreren Events, wie beispielsweise die Ermittlung des Minimums, Maximums oder Durchschnittwerts eines Parameters, unterstützt eine Detektion von Anomalien und kann als Vorverarbeitungskomponente eingesetzt werden.

- *GUI*. Eine Erstellung von Verarbeitungsschritten anhand einer grafischen Oberfläche stellt eine Vereinfachung dar, vor allem für Benutzer, die keine Experten auf diesem Gebiet sind.

- *Informationsflussvisualisierung*. Neben der Fehlererkennung von Systemen bietet eine Darstellung des Informationsverlaufs von Sender zu Empfängern der Nachricht eine Möglichkeit, Probleme in der Kommunikationsarchitektur visuell zu identifizieren. Beispielsweise kann somit festgestellt werden, wenn relevante Informationen nicht wie vorhergesehen übermittelt werden.

Die in diesem Kapitel vorgestellten Kommunikationsinfrastrukturen sowie der in dieser Arbeit entwickelte Event Broadcasting Service (EBS) wurden gemäß der beschriebenen Faktoren und Eigenschaften miteinander verglichen. Das Ergebnis ist in Tabelle 3.1 dargestellt. Der EBS basiert auf Events. Beispielsweise werden Veränderungen in der Umgebung durch Sensoren erfasst und in Form von Events weitergegeben. Durch die Einbindung von Verarbeitungsfunktionalität auf der Server- sowie auf der Client-Seite erfolgt die Verarbeitung in einer dezentralen verteilten Form. Wie auch TelegraphCQ und MQTT werden beim EBS Systemfehler erkannt und notifiziert, um entsprechend darauf reagieren zu können. Für die Übertragung der Events können diese verschlüsselt und beim Empfänger entsprechend entschlüsselt werden. In der Regel werden die Events direkt zu den entsprechenden Clients weitergeleitet. Parallel dazu können sie auch in einer Datenbank hinterlegt werden, deren Funktion analog zu einem Tupelraum konzipiert ist. Wie alle vorgestellten Systeme unterstützt der EBS

Typfilter, die beispielsweise verwendet werden, wenn der EBS als Publish/Subscribe-System agieren soll. Zudem sind Mechanismen zur Sequenzverarbeitung, Eventtransformation sowie Zeitoperatoren und Aggregation in die Verarbeitungsmöglichkeiten der Kommunikationsinfrastruktur integriert. Mit Hilfe einer grafischen Oberfläche können die Verarbeitungsregeln visuell erstellt und angepasst werden. Schließlich werden Informationen bezüglich Sender und Empfänger der Events dazu verwendet, um eine visuelle Darstellung des Informationsflusses zu realisieren. Im Allgemeinen ist das System so aufgebaut, dass es modular angepasst werden kann, um die Anforderungen der Einsatzdomäne zu erfüllen.

	Datenübertragung	Informationsverarbeitung	Fehlererkennung	Sicherheit / Verschlüsselung	Datenverteilung	Sequenzverarbeitung	Eventtransformation	Zeitoperatoren	Aggregation	GUI	Informationsflussvisualisierung
T Spaces	Tupel	verteilt	-	-	pull	-	-	-	-	-	-
Event Heap / iROS	Tupel/Event	verteilt	-	-	push	-	-	-	-	-	-
MundoCore	Tupel/Event	verteilt	-	√	push	-	-	-	-	-	-
MQTT / MQTT-SN	Event/Relation	verteilt	√	√	push	-	-	-	-	-	-
Snoop / Sentinel	Datenstrom/Relation	zentral	-	-	push	√	-	-	-	-	-
Aurora / SQuAl	Datenstrom/Relation	verteilt	-	-	push	-	√	√	√	√	-
TelegraphCQ	Datenstrom/Relation	verteilt	√	-	push/pull	-	√	√	√	-	-
CQL / STREAM	Datenstrom/Relation	zentral	-	-	push	-	√	√	√	√	-
GEM	Event	verteilt	-	-	push	√	-	√	-	-	-
Cayuga	Event	zentral	-	-	push	√	√	-	√	√	-
RACED	Event	verteilt	-	-	push/pull	√	-	√	-	-	-
T-Rex / TESLA	Event	zentral	-	-	push	√	-	√	√	-	-
EBS	Event	verteilt	√	√	push/pull	√	√	√	√	√	√

***Tabelle 3.1**: Vergleich der Kommunikationsinfrastrukturen mit dem Event Broadcasting Service* (EBS)

3.3.2 Visualisierungskonzepte im Vergleich

Analog zu den Kommunikationsinfrastrukturen wurden die Visualisierungskonzepte hinsichtlich spezifischer Parameter untersucht. Diese umfassen:

- *Anwendung*. Ein Motivationsgrund für die Entwicklung der Systeme liegt im jeweiligen Anwendungsfeld. Einerseits kann die visuelle Repräsentation als Morphismus von Sensorinformationen der einen Welt in die andere verwendet werden. Andererseits kann der Fokus darin bestehen, eine möglichst realgetreue virtuelle Testumgebung zu erhalten. Des Weiteren kann die bi-direktionale Verbindung zwischen physischer Umgebung und virtuellem Modell dazu verwendet werden, um natürliche Benutzerinteraktionen zur späteren Verwendung aufzuzeichnen. Schließlich können Dual Reality Systeme dazu eingesetzt werden, um eine Umgebung zu monitoren und in Form eines virtuellen Leitstands zu steuern.

- *Visualisierung*. Für die dreidimensionale Darstellung des virtuellen Modells können diverse Visualisierungskomponenten eingesetzt werden, wie beispielsweise Second Life (SL), OpenSimulator (OpenSim), Multiverse, RealXtend oder spezialisierte Editoren.

- *Kommunikation*. Entsprechend der Systeme, die zur Visualisierung verwendet werden, werden unterschiedliche Kommunikationsschnittstellen für den Datenaustausch zwischen den beiden Welten verwendet, wie beispielsweise die Linden Scripting Language (LSL).

- *Modell*. Neben den Kommunikationsschnittstellen zeigen sich Unterschiede der Visualisierungskomponente in der Speicherung des Modells. In der Regel wird das gesamte Modell einmalig erstellt und als eine Szene abgespeichert.

- *Monitoring*. Für das Monitoring einer Umgebung muss diese möglichst realgetreu im Virtuellen abgebildet werden. Zudem müssen Informationen von Sensoren mittels der entsprechenden Kommunikationsschnittstellen in das virtuelle Modell übertragen und dort repräsentiert werden.

- *Steuerung*. Analog zum Monitoren müssen für das Steuern von Umgebungen deren Aktuatoren aus dem interaktiven 3D-Modell ansprechbar sein.

- *Simulation*. Zusätzlich zur Erfassung von Benutzerinteraktionen und der Darstellung von Sensorinformationen weisen virtuelle Umgebungen Potenzial auf, unter Verwendung von Simulationen gewisse Aspekte zu testen, zu evaluieren und damit Prognosen zu erstellen.

Gemäß der oben aufgeführten Eigenschaften wurden die vorgestellten Systeme eingeordnet (siehe Tabelle 3.2). Das Dual Reality Management Dashboard (DURMAD) ist vorwiegend für den Einsatz als Leitstand von Cyber-Physischen Umgebungen mit der Möglichkeit der Einbindung von Simulationen entwickelt. Aufgrund der Verknüpfung mit dem EBS werden alle Benutzerinteraktionen sowohl im Virtuellen als auch im Realen protokolliert und können im Nachhinein weiterverwendet werden. Neben der Darstellung in Java3D ist eine Schnittstelle zu weiteren Visualisierungskomponenten enthalten. Exemplarisch ist

die Darstellung in XML3D (siehe Abschnitt 2.4.2) im DURMAD angebunden. Neben der Anbindung des EBS sind weitere Schnittstellen in Form von VNC zum Darstellen von Bildschirminhalten und REST, um Daten an externe Applikationen bereitzustellen, integriert. Im Gegensatz zu den anderen Systemen wird die Szene nicht a priori modelliert, sondern sie wird basierend auf einem Grundrissplan dynamisch generiert. Aufgrund der Funktionalität als Leitstand und Testumgebung unterstützt DURMAD Simulationen sowie Monitoring und Steuerung von Umgebungen.

	Anwendung	Visualisierung	Kommunikation	Modell	Monitoring	Steuerung	Simulation
Steckerleiste & Datenteich	Morphismus	SL	LSL	Szene	√	-	-
Positionierungssimulation	Testumgebung	SL/OpenSim	Twin-World Mediator/ LSL	Szene	-	-	√
Eolus One	Leitstand	SL/OpenSim	LSL	Szene	√	√	-
Virtuelle Schokoladenfabrik	Leitstand	Multiverse	Python Skripte	Szene	√	√	-
iWorlds	Verhaltens-aufzeichnung	RealXtend	UPnP Framework	Szene	*	-	-
YAMAMOTO	Leitstand/ Testumgebung	YAMAMOTO(C#)/ VRML-/RoSiML-Viewer	Rest/VNC/UCH	Szene (yml)	√	√	+
DURMAD	Leitstand/ Testumgebung/ Aufzeichnung	Java3D/XML3D	REST/VNC/EBS	dynamisch (yml/gml)	√	√	√

***Tabelle 3.2**: Vergleich der Dual Reality Systeme mit dem Dual Reality Management Dashboard (*DURMAD*)*

*: Daten werden aufgezeichnet, dienen aber nicht dem Monitoring der Umgebung
+: Enthält eine Abspielfunktion von virtuellen Avatarrundgängen, sonst keine Simulationen

Teil III

Dual Reality Framework für Cyber-Physische Umgebungen

4 Virtualisierung und Visualisierung von Cyber-Physischen Umgebungen

Um eine digitale Repräsentation einer Cyber-Physischen Umgebung (CPE) zu ermöglichen, muss ein entsprechendes Modell dieser Umgebung existieren. Bei der Modellierung von Objekten und von gesamten Umgebungen müssen dabei gewisse Eigenschaften berücksichtigt werden. Zur Erstellung solcher Modelle gibt es spezialisierte Tools, die den Anwender durch entsprechende Funktionen unterstützen. Mittels der hiermit generierten Modelle können CPE digital repräsentiert werden. Sollen auf diesen Repräsentationen Simulationen durchgeführt werden, oder ist eine Verschmelzung von physischer und virtueller Umgebung gewünscht, müssen den einzelnen physischen Objekten eindeutige Modelle zugeordnet werden, was unter dem Begriff Virtualisierung verstanden wird. Die Virtualisierung einer CPE stellt dabei die Grundlage von Dual Reality (DR) dar, bei welcher eine gegenseitige Beeinflussung zwischen der physischen und der virtuellen Umgebung erfolgt. Für die Erweiterung von DR hinsichtlich der Integration von Simulationen wird eine Formalisierung benötigt, die in dieser Arbeit aufgestellt wird und eine klare Abgrenzung und Einteilung von DR-Applikationen ermöglicht. Um Ergebnisse von Simulationen im Virtuellen präsentieren zu können, muss eine Visualisierung der Virtualisierung gegeben sein. Dazu können mehrere Unterscheidungskriterien für die Darstellung und diverse Visualisierungskomponenten voneinander abgegrenzt werden.

4.1 Modellierung

Wie bereits erwähnt, werden für eine digitale Visualisierung von Objekten virtuelle Repräsentationen dieser Objekte benötigt. Je nachdem welche Visualisierungssoftware zum Einsatz kommt, muss die virtuelle Repräsentation in einem gewissen Dateiformat vorliegen. Dieses Dateiformat wird von der Software gelesen, analysiert und entsprechend der Spezifikationen visuell dargestellt. Für die Erstellung solcher Repräsentationen müssen die Objekte zuerst modelliert werden. In dieser Arbeit wird folgende allgemeingültige Definition von Modellierung aufgestellt und verwendet:

Definition 4.1 (Modellierung) *Modellierung beschreibt die Erstellung einer physischen oder einer virtuellen Repräsentation eines existierenden oder eines fiktiven Objekts. Vorwiegend wird die äußere Form des Objekts modelliert, welche die geometrische Figur und*

Textur umfasst. Ebenso können Eigenschaften, wie beispielsweise Materialien des Objekts, modelliert werden.

Eine Modellierung umfasst nach dieser Definition unter anderem die Erstellung von digitalen Objekten, wie beispielhaft in Abbildung 4.1 dargestellt. Vorwiegend werden solche Modelle dazu verwendet, um gewisse Aspekte grafisch darstellen zu können. Zusätzlich zu existierenden Objekten können auch fiktive Objekte modelliert werden, beispielsweise für die Erstellung von virtuellen Prototypen. Mittels geeigneter Rendering-Methoden können durch solche Ansätze neue Objektformen realistisch im Virtuellen abgebildet und somit Personen präsentiert werden. Neben der Erstellung von virtuellen Modellen können auch miniaturisierte Abbilder von großen Objekten im Reellen erstellt werden. Unter anderem wird dies in der Architektur eingesetzt, um zukünftige Gebäude und Landschaften mittels eines miniaturisierten Modells bereits während der Planung darzustellen und Entscheidungsträgern zu präsentieren. Zusätzlich zur visuellen Modellierung können auch gewisse Parameter und Eigenschaften durch eine Modellierung digitalisiert werden. Dies findet beispielsweise bei der Benutzermodellierung Verwendung, bei der gewisse personen- oder gruppenspezifische Eigenschaften in digitaler Form abgebildet werden. Diese Daten können anschließend zur Personalisierung von bestimmten Diensten verwendet werden. Ein Beispiel für die Erstellung und Speicherung solcher Benutzermodelle stellt UbisWorld dar [Heckmann, 2006].

***Abbildung 4.1**: Darstellung einer modellierten Umgebung*

Wird eine Modellierung zur visuellen Darstellung verwendet, sollten bei der digitalen Version die prägnanten Eigenschaften des physischen Objekts leicht erkennbar sein, um eine Zuordnung herstellen zu können. Oftmals werden Modellierungen verwendet, um komplexere Strukturen, wie beispielsweise Grundrisse von Gebäuden, in Form eines virtuellen Modells darzustellen. Dabei kann hinsichtlich der Realitätsgenauigkeit unterschieden werden. Oftmals reicht eine relativ grobe Darstellung aus, um eine Zuordnung zu ermöglichen. In diesen Fällen werden neben dem grob modellierten Objekt noch zusätzliche Objekte benötigt, damit die Zuordnung für die Benutzer eindeutig erkennbar ist. Beispielsweise kann eine grobe Darstellung einer Umgebung, bei welcher die Objekte im Modell an

den entsprechenden Stellen, an denen sie in der physischen Umgebung platziert sind, die Zuordnung von einem Würfel zu einem Tisch ermöglichen. Dazu wird das Wissen des Benutzers um die genaue Einrichtung der realen Umgebung vorausgesetzt. Ohne dieses Wissen wäre eine eindeutige Zuordnung nicht möglich. In einem solchen Fall muss die Modellierung des Objekts möglichst exakt das physische Objekt in Form, Größe und Darstellung widerspiegeln, um eine Identifikation zu gewährleisten.

Neben der Darstellung können Modellierungen dazu eingesetzt werden, um mit Objekten Simulationen durchzuführen. Gerade bei komplexen Konstrukten kommt es bei der Entwicklung neuer Objekte zu einer engen Verzahnung zwischen Modellierung und physischen Prototypen. Während die neuen komplexen Konstrukte vorerst im Virtuellen so lange getestet und modifiziert werden, bis eine gewünschte Eigenschaft erreicht ist, dienen Prototypen dazu, diese Eigenschaften auch unter realen Bedingungen zu evaluieren. Entsprechend der daraus resultierenden Erkenntnisse können die Modellierungen angepasst und erneut mittels der Simulationen getestet werden, bis wiederum ein zufriedenstellendes Ergebnis erreicht wird. Diese Vorgehensweise ist z.B. beim Design neuer Fahrzeugmodelle üblich, die zunächst im Virtuellen modelliert und anhand von Simulationen getestet werden. Anschließend werden die Ergebnisse anhand eines physischen Prototyps evaluiert und die Modellierung gegebenenfalls angepasst. So entsteht ein Kreislauf zwischen einem virtuellen Modell und der entsprechenden physischen Umsetzung in Form eines Prototypen, wie in Abbildung 4.2 dargestellt. Der Unterschied zwischen dem Verhalten in der virtuellen und in der realen Umgebung ist einerseits dem Detailgrad der Modellierung geschuldet und andererseits der Genauigkeit der Simulation. Je genauer die Modellierung dem realen Gegenstück entspricht, desto kleiner ist der Unterschied zwischen dem Virtuellen und dem Reellen. Entsprechendes gilt auch für die Realitätsgenauigkeit der zugrunde liegenden Simulation.

Für die vorliegende Arbeit sind vorwiegend Modellierungen einzelner Objekte (siehe Abschnitt 4.1.1) und gesamter Umgebungen (siehe Abschnitt 4.1.2) relevant. Deren Besonderheiten sowie potentielle Modellierungstools inklusive der zugrunde liegenden Dateiformate, welche auch im Rahmen dieser Arbeit zum Einsatz kamen, werden im folgenden Abschnitt beschrieben.

4.1.1 Objektmodellierung

Die Modellierung eines Objekts kann unter Zuhilfenahme entsprechender Tools erfolgen. Beispiele für solche 3D-Grafiktools sind Maya[1], Cinema 4D[2] und Blender[3]. Sie ermöglichen eine dreidimensionale Modellierung von Objekten inklusive Texturen und Farbgebungen. Aufgrund ihres großen Funktionsumfangs sind sie oftmals nur von „Experten“ effizient bedienbar. Mit Experten sind hierbei Personen gemeint, die sich mit den jeweiligen 3D-Modellierungstools auskennen und den gebotenen Funktionsumfang gezielt verwenden

[1] http://www.autodesk.de/products/autodesk-maya/overview, Zugriff: November 2014
[2] http://www.maxon.net/de/products/cinema-4d-studio.html, Zugriff: November 2014
[3] http://www.blender.org, Zugriff: April 2014

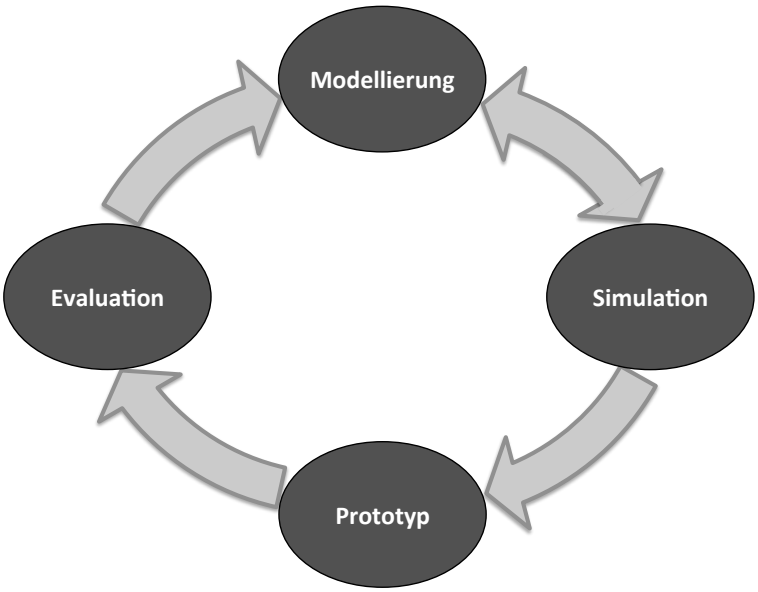


***Abbildung 4.2**: Kreislauf der Modellierung mit virtuellen Simulationen und Validierung reeller Prototypen*

können. Zur besseren Übersicht können komplexe Modelle auch aus mehreren Submodellen in Form einer hierarchischen Modellierung erstellt werden. Bei der Darstellung können diese Submodelle dann jeweils einzeln angezeigt oder ausgeblendet werden. Zusätzlich zu den Modellen können auch Bewegungsmuster in Form von Animationspfaden modelliert werden. Diese können bei einer späteren Wiedergabe abgespielt werden. Abbildung 4.3 zeigt einen Bildschirmauszug des Modellierungstools Blender. Die dargestellten Fenster innerhalb der Applikation sind modular aufgebaut und können über das Menü an die Präferenzen des Benutzers angepasst werden. Die Objektbefehle stellen kurze Funktionen dar, die über Knopfdruck aktiviert werden können. Detailliertere Anpassungen mit spezifischer Eingabe von Werten können über die Objekteigenschaften bzw. die Modellierungsparameter ausgeführt werden. Die Objekthierarchie bietet eine Übersicht der einzelnen Submodelle. Die Animationszeitleiste zeigt den zeitlichen Verlauf bei der Modellierung von Animationen an. In der 3D-Modellierungsumgebung kann das Modell basierend auf der Visualisierung angepasst werden. Neben der Speicherung in einem für Blender spezifischen Format werden auch diverse Exporter für weitere Dateiformate unterstützt.

Zusätzlich zur Möglichkeit, Modelle händisch zu erstellen, können die geometrischen Eigenschaften physisch vorhandener Objekte mit Hilfe von optischen Verfahren automatisch erfasst werden, beispielsweise durch 3D-Scanner und 3D-Kameras, wie die Microsoft Kinect [Smisek et al., 2013]. Ein Ansatz für die dreidimensionale Erfassung von Objekten mittels einer Kinect-Kamera ist *KinectFusion* [Izadi et al., 2011]. Dabei steht entweder das Objekt statisch im Raum und die Kinect-Kamera wird um dieses Objekt geführt oder die Kinect-Kamera wird statisch fixiert und das Objekt muss vor der Kamera bewegt werden, um eine dreidimensionale Abbildung zu erhalten. Zusätzlich zu den räumlichen Informa-

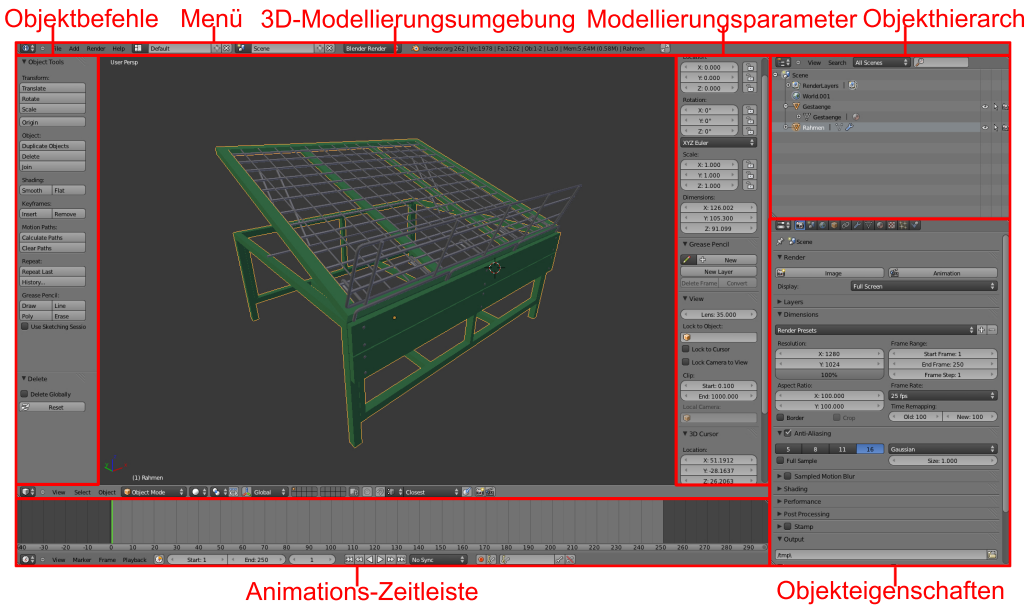


***Abbildung 4.3**: Bearbeitungsoberfläche von Blender*

tionen wird auch das RGB-Kamerabild verwendet, um Texturen der gescannten Objekte zu erhalten. Beispielanwendungen für solche dreidimensionalen Erfassungen sind neben der Simulation, welche durch die Physik-Engine unterstützt wird, auch das Replizieren von Objekten. Hierzu können 3D-Drucker verwendet werden, die basierend auf dem Modell ein Replikat des Objekts erstellen.

Eine weitere Alternative zur Modellierung existierender Objekte ist die Verwendung mehrerer Kameras, welche das Objekt gleichzeitig aus unterschiedlichen Blickwinkeln aufnehmen. Die entsprechenden Kamerabilder werden anschließend durch geeignete Verfahren miteinander kombiniert mit dem Resultat einer dreidimensionalen Rekonstruktion des Objekts [Bleser et al., 2007]. Dabei können sowohl die Kameras als auch das Objekt statisch bleiben. Beispielsweise kann eine Cave-Umgebung mit diversen Kameras ausgestattet werden, welche die Objekte im Inneren dieser Umgebung aus verschiedenen Perspektiven aufzeichnen und zur Verarbeitung an ein zentrales System übertragen [Kanade et al., 1997]. Diese Methode hat zudem den Vorteil, dass auch Bewegungen aufgenommen und somit Bewegungsmuster erfasst werden können, indem die zeitliche Abfolge mit protokolliert und bei der Rekonstruktion berücksichtigt wird. Dadurch kann beispielsweise die Ausführung des Schwungs eines Baseballschlags aufgezeichnet und später als dreidimensionales Video abgespielt werden [Rander et al., 1997].

4.1.2 Grundrissmodellierung

Neben der Erstellung virtueller Repräsentationen einzelner Objekte werden Modellierungstools auch zur Erstellung von Repräsentationen gesamter Umgebungen verwendet.

Eine solche Umgebung kann als modulares Konstrukt erstellt werden, indem mehrere Objektmodelle zusammengefügt werden. Dazu muss eine computerverständliche Beschreibung der Umgebung vorliegen, welche geografische Informationen enthält. Die Arten der Beschreibungen variieren ja nach Einsatzgebiet. Beispielsweise werden Grundrisse von Gebäuden oftmals in sogenannten CAD-Dateien gespeichert, die in der Regel nur eine zweidimensionale Repräsentation der Gebäudegrundfläche enthalten. Für eine dreidimensionale Ansicht müssen daher noch weitere Informationen hinterlegt werden, welche unter anderem Höhenangaben beinhalten. Zur Erfassung, Bearbeitung und Präsentation räumlicher Daten haben sich *Geoinformationssysteme (GIS)* etabliert, welche beispielsweise für Stadtplanungen eingesetzt werden. Ein GIS kann im Allgemeinen als ein System definiert werden, welches geografische Daten verarbeitet [Virrantaus et al., 2001]. GIS stellen dabei eine Erweiterung von Kartierungsapplikationen dar, indem sie Zusatzfunktionen als semantische Informationen beinhalten. Diese ermöglichen beispielsweise, Distanzen zu messen oder Informationen mittels Überblendungen darzustellen. Des Weiteren ermöglichen GIS topologische Analysen, um beispielsweise Nachbargrundstücke von Flurstücken automatisiert ermitteln zu können. Neben der Analyse und Verarbeitung geographischer Informationen ist eine Hauptaufgabe von GIS die grafische Darstellung dieser Daten. Zusätzlich zu kommerziellen GIS, wie beispielsweise ArcGIS[4], gibt es auch zahlreiche freie Varianten, wie beispielsweise QGIS[5]. Ergänzend zu lokal verwendbaren Softwareapplikationen gibt es auch webbasierte GIS, wie beispielsweise Diercke WebGIS [6].

Die unterschiedlichen Programme nutzen spezielle Datenformate zur Speicherung der geometrischen Modelle. Die zugrunde liegenden Datenmodelle umfassen dabei Informationen zur Art der Daten, der Struktur, in der sie hinterlegt sind und weitere Attribute. Im Speziellen umfassen GIS-Modelle Geometriedaten wie Lage, Form, Orientierung und Größe von Objekten. Als Quasi-Standard hat sich dabei das *Shapefile*-Format etabliert. Dieses Datenformat wurde von der Firma ESRI definiert und wird von den meisten GIS unterstützt. Ein weiteres, weit verbreitetes Datenformat zur Speicherung von raumbezogenen Objektinformationen ist *GML (Geography Markup Language)*. Dies ist eine Erweiterung von XML und orientiert sich an internationalen Normen und Standards. Dank der generischen Struktur von XML können standardisierte Objekttypen wie Punkte, Linien und Polygone mit zusätzlichen eigenen Attributen versehen werden. Das zugehörige Schema wird in einer Schemadatei (XSD) gespeichert, damit die GIS diese zusätzlichen Parameter ebenfalls verarbeiten und anzeigen können. Da beide Dateiformate von den meisten Systemen unterstützt werden, können sie als Austauschformat fungieren. Zudem gibt es mehrere Konverter, die Dateien zwischen den beiden Formaten übersetzen können. Während Shapefiles aufgrund ihrer binären Speicherung eine geringere Dateigröße aufweisen, sind GML-Dateien basierend auf dem XML-Format auch für Menschen lesbar und können manuell verändert werden.

Das in Abschnitt 3.2.6 und in Abbildung 3.11 vorgestellte System, YAMAMOTO, zur Modellierung mehrstöckiger Gebäude ist ebenfalls ein GIS. Die Datenspeicherung erfolgt

[4]http://www.esri.com/software/arcgis, Zugriff: November 2014
[5]http://www.qgis.org/, Zugriff: November 2014
[6]http://www.diercke.de/webgis, Zugriff: November 2014

bei diesem Programm im spezialisierten yml-Format (YAMAMOTO XML). In diesem wird die Grundfläche von Räumen in Form von Punkten, Kanten und Polygonen repräsentiert [Stahl, 2009]. Die Punkte werden dabei durch ihre dreidimensionale Koordinate im Raum spezifiziert, in Bezug zu einem selbst definierten Ursprung. Eine Kante wird mittels zwei Punkten als Verbindung zwischen diesen beschrieben. Ein Polygon besteht aus einer geordneten Folge von Punkten. Zusätzlich zu den geometrischen Darstellungen können vordefinierte Parameter angegeben werden, wie beispielsweise der Typ einer Kante (z.B. Wand, Tür oder Fenster) und ihre Höhe. Anhand dieser Daten kann eine dreidimensionale Darstellung des modellierten Raums erstellt werden. Zusätzlich besteht die Möglichkeit, Objekte wie Tische oder Schränke, anhand von verschachtelten geometrischen Grundformen, wie beispielsweise Boxen, im yml-Format zu repräsentieren. Sensoren können entsprechend ihrer Art als Punkte, Kegel oder Kreise im yml-Format modelliert werden. Basierend auf dieser Beschreibung der Umgebung kann eine dreidimensionale Darstellung des Modells generiert werden. Da die eigentliche Modellierung in 2D erfolgt und die Visualisierung in 3D, wird YAMAMOTO als 2,5D Modellierungstool bezeichnet.

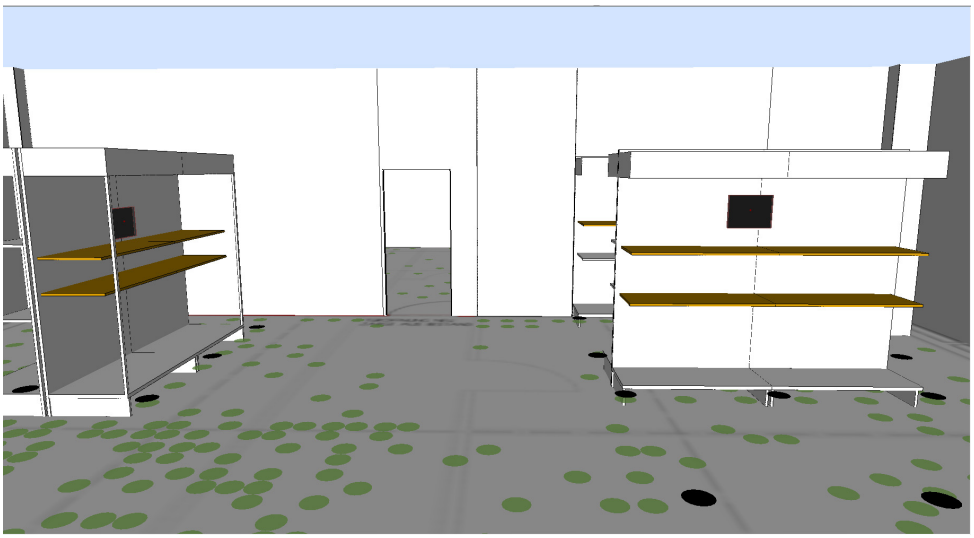


***Abbildung 4.4**: Darstellung eines Modells in* YAMAMOTO *mit Regalen und RFID-Tags in Form von Punktobjekten*

Abbildung 4.4 zeigt die 3D-Darstellung eines Modells in YAMAMOTO. Der Ausschnitt zeigt eine kleine Umgebung mit mehreren Regalen und RFID-Tags auf dem Boden, welche durch Punktobjekte repräsentiert werden. Die Regale sind dabei durch Verschachtelungen von Boxen mit unterschiedlichen Texturen modelliert. Für die Erstellung einer solchen yml-Datei kann man einerseits den in Abschnitt 3.2.6 beschriebenen visuellen YAMAMOTO-Editor verwenden. Andererseits kann man die auf XML basierte yml-Datei auch in einem Texteditor manuell erstellen und editieren, wobei dies eine genaue Kenntnis über die spezifizierte Struktur des yml-Formats voraussetzt und fehleranfällig ist. Um die Modellierung in YAMAMOTO zu vereinfachen kann man den Grundrissplan der zu modellierenden Umgebung

als Bild laden, welches im Hintergrund dargestellt werden kann. Anschließend können Polygone gezeichnet werden, aus welchen automatisch sowohl die entsprechenden Punkte als auch die Kanten erzeugt werden. Analog können Repräsentanten für Sensoren sowie Boxen ergänzt werden.

4.2 Virtualisierung

Zum virtuellen Testen und Evaluieren neuer Systeme und Softwareapplikationen müssen physisch existierende Objekte möglichst detailgetreu nachgebildet werden. Dieser Prozess wird als Virtualisierung bezeichnet. Vorwiegend wird Virtualisierung mit Betriebssystemen in Verbindung gebracht, wobei eine Hardware emuliert wird, so dass das gewünschte Betriebssystem auch auf einem Rechner läuft, dessen Hardwarevoraussetzungen eigentlich nicht zum Betriebssystem passen. Im Rahmen der Arbeit wurde dieses Konzept verallgemeinert und in folgender Definition zusammengefasst.

Definition 4.2 (Virtualisierung) *Eine Virtualisierung ist ein virtuelles Modell eines physischen Objekts, bei dem eine Verknüpfung zwischen dem realen Objekt und seiner Modellierung vorliegt. Bei einer Virtualisierung müssen die Eigenschaften des physischen Objekts der entsprechenden virtuellen Repräsentation zugeordnet werden.*

Im Allgemeinen beschreibt Virtualisierung die Erzeugung virtueller Modelle existenter Objekte. Die Modellierung bietet dabei eine Grundlage zur Erstellung von virtuellen Repräsentationen. Im Gegensatz zur Modellierung müssen Virtualisierungen einem spezifischen physisch existierenden Objekt entsprechen und diesem auch zugeordnet werden. Typischerweise wird dieses Verfahren eingesetzt, um Software unter realistischen Gegebenheiten zu testen, ohne auf eine physische Variante der Hardware angewiesen zu sein, die gegebenenfalls später zum Einsatz kommt. Beispielsweise kann mittels einer Virtualisierung Hardware emuliert werden, ohne dass diese physisch verwendet wird. Neben der virtuellen Repräsentation als Testumgebung können Virtualisierungen auch dazu verwendet werden, um Objekte originalgetreu digital zu visualisieren. Dazu werden neben den physikalischen Eigenschaften noch ästhetische Merkmale bei der Virtualisierung benötigt. Wird die Virtualisierung eines Objekts zum Testen von Softwareapplikationen verwendet, wird im Grunde keine Farbe und Textur des Objekts benötigt. Soll die Virtualisierung jedoch auch visuell dargestellt werden, sollten solche Parameter berücksichtigt werden. Somit muss bei einer Virtualisierung auch der jeweilige Anwendungszweck berücksichtigt werden, um benötigte Parameter im Vorfeld eingrenzen zu können, da die Virtualisierung ansonsten beliebig komplex werden kann. Eine vollständig virtualisierte Repräsentation eines Objekts kann oftmals - wenn überhaupt - nur mit sehr großem Aufwand realisiert werden. Ein Beispiel hierfür ist der Wirkungsradius bei funkbasierten Sensoren. Dieser wird neben Witterungsbedingungen auch durch Rückkopplungen anderer Funkwellen sowie weiterer Objekte in seiner Umgebung beeinflusst. Ein realistisches Modell ist daher nicht realisierbar. Ein vereinfachtes Modell, bei dem der Wirkungsradius mittels eines radialen Umkreises ohne Berücksichtigung weiterer Störquellen spezifiziert wird, ist für die meisten Simulationen ausreichend.

Somit kann man bei einer Virtualisierung zwischen zwei grundlegenden Eigenschaften unterscheiden. Einerseits spielt die Verwendung von wiedererkennbaren Merkmalen bei der Visualisierung eines Objekts eine wichtige Rolle. Andererseits werden physikalische Eigenschaften des Objekts benötigt, um beispielsweise eine Simulation damit durchführen zu können. Oftmals werden auch beide Eigenschaften miteinander kombiniert, so dass eine Simulation durchgeführt und gleichzeitig auch visualisiert werden kann. Somit können Virtualisierungen von größeren Objektgruppen, wie beispielsweise von Umgebungen, dazu verwendet werden, eine Testumgebung für neue Applikationen und Interaktionsformen zu schaffen. Unter anderem weisen solche virtualisierten Testumgebungen den Vorteil auf, dass die in ihnen herrschenden Bedingungen sehr genau kontrolliert werden können [Anderson et al., 2005; Leichtenstern et al., 2010]. Ebenso ist es darüber möglich, kostengünstig Sensoren und Applikationen zu testen [Oh et al., 2009]. Des Weiteren können mehrere Instanzen solcher Virtualisierungen parallel betrachtet und mit unterschiedlich konfigurierten Simulationen versehen werden. Durch die kontrollierte Umgebung können die Applikationen anschließend abhängig von den jeweiligen Eingaben und Bedingungen getestet werden, um eine optimale Einstellung zu ermitteln.

Um eine Testumgebung zu erhalten, muss im Vorfeld eine CPE samt aller enthaltener Objekte virtualisiert werden (siehe Abschnitt 4.2.1). Da der Detaillierungsgrad einer Virtualisierung unterschiedlich und je nach Anwendungszweck weit ausgeprägt sein kann, ist eine formalisierte Definition von Virtualisierungen hilfreich (siehe Abschnitt 4.2.2). Dies ermöglicht es, eine klare Eingruppierung zu ermöglichen. Um eine Zuordnung zwischen dem physischen Objekt und dessen Virtualisierung herstellen zu können, müssen eindeutige Identifikationsmechanismen verwendet werden. Durch diese Verknüpfung entsteht zudem eine Verschmelzung von CPE und deren Virtualisierung gemäß dem Dual Reality-Gedanken (siehe Abschnitt 4.2.3). Dies ermöglicht es, dass Ergebnisse aus einer virtuellen Simulation zeitnah in der realen Umgebung umgesetzt werden können. Basierend auf der Formalisierung von Virtualisierung kann auch eine formale Definition von Dual Reality aufgestellt werden (siehe Abschnitt 4.2.4).

4.2.1 Virtualisierung von Cyber-Physischen Umgebungen

Modelle, welche Ergebnisse der Modellierung und Virtualisierung von CPE darstellen, dienen neben der Möglichkeit als Simulationsumgebung auch dazu, Veränderungen in räumlichem Zusammenhang darzustellen, indem eine grafische Visualisierung in der 3D-Darstellung erfolgt. In der Regel kann ein komplexes Objekt als komplettes Konstrukt modelliert werden, was oftmals einfacher ist, da dabei nur ein Modell erstellt werden muss, in welchem alle Teilobjekte und Informationen enthalten sind. Spätere Anpassungen und Veränderungen können bei dieser Methode jedoch sehr zeitaufwändig sein. Eine alternative Variante besteht darin, jedes einzelne Objekt zu modellieren und anschließend alle Objekte zu dem benötigten Gesamtmodell zusammenzufügen, wie bei der Erstellung von Grundrissmodellen beschrieben (siehe Abschnitt 4.1.2). Für das Zusammenfügen wird dabei eine entsprechende Grundlage benötigt, auf der dies erfolgen kann, wie beispielsweise ein Bauplan. Dieser Ansatz hat den Vorteil, dass die einzelnen Bausteine modularisiert sind und somit Änderungen relativ schnell im Modell angepasst werden können. Ebenso können mittels der

Modularität einzelne Objekte verhältnismäßig einfach durch andere ersetzt werden, was eine gewisse Agilität mit sich bringt. Die Virtualisierung einer CPE kann durch gewisse Verfahren und Tools automatisiert werden. Falls ein Objekt mit einem digitalen Objektgedächtnis (siehe Kapitel 2.1.3) ausgestattet ist, so kann das jeweilige virtuelle Modell des Objekts dort hinterlegt und bei Bedarf automatisch in ein CPE-Modell eingefügt werden. Sensoren und Aktuatoren stellen dabei besondere Objekte dar, da sie weitere Eigenschaften aufweisen, welche mitberücksichtigt werden müssen.

Sensoren: Eine detailgetreue Virtualisierung von Sensoren ist vor allem für besonders realistische Simulationen notwendig. Dabei müssen mehrere Eigenschaften des Sensors, wie z.B. Empfangsbereich und potentielle Störquellen, Verwendung finden, um anschließend eine möglichst genaue virtuelle Abbildung zu erhalten und dadurch realistische Simulationsergebnisse zu erzielen. Beispielsweise sind der Erfassungsbereich und die Erfassungsfrequenz wichtige Parameter, die berücksichtigt werden müssen. Diese Daten können z.B. dazu dienen, um eine geeignete Positionierung der Sensoren in der realen Umgebung festzulegen. Ebenso können weitere Einflussfaktoren, wie beispielsweise die Genauigkeit der Erfassung sowie das Verhalten bei externen Störquellen, wichtige Parameter sein, welche modelliert werden müssen. Schließlich gibt es noch weitere Eigenschaften, wie beispielsweise die Anzahl der gleichzeitig erfassbaren Objekte, die in einer möglichst realitätsnahen Virtualisierung berücksichtigt werden müssen.

Aktuatoren: Ebenso wie bei den Sensoren dient eine detaillierte Virtualisierung von Aktuatoren vorwiegend dem Ziel, Simulationen möglichst realitätsnah durchführen zu können. Des Weiteren bietet dies die Möglichkeit, in einer Visualisierung die Auswirkung der Aktivität von Aktuatoren zu präsentieren. Insbesondere können damit bereits vor einer physischen Installation Aktionsradien und gewisse Gegebenheiten getestet werden, wie beispielsweise zur Erkennung von möglichen Kollisionen bei der automatisierten Bewegung von Objekten. Neben dem Aktionsradius spielen bei Aktuatoren die Verzögerung bis zum Starten sowie die Geschwindigkeit der Aktion eine wichtige Rolle. Analog zu den Sensoren können auch bei den Aktuatoren Störanfälligkeiten und Beeinflussungen durch externe Faktoren bei der Virtualisierung berücksichtigt werden.

4.2.2 Formalisierte Definition von Virtualisierung

Gemäß der zuvor beschriebenen Eigenschaften kann man die Virtualisierung von Objekten sowie von physischen Umgebungen wie folgt formal definieren. Sei o ein physisches Objekt, dann ist o_v dessen Virtualisierung, wenn eine eindeutige Verbindung zwischen beiden Instanzen vorliegt (4.1). Dies bedeutet, dass man ohne weiteres Wissen erfassen kann, dass o_v dem realen Objekt o entspricht, indem es sowohl die äußere Form als auch die Textur und die physikalischen Eigenschaften des physischen Objekts möglichst genau abbildet.

Virtualisierung
Die Menge aller Objekte o wird in O zusammengefasst. Entsprechendes gilt für die virtualisierten Objekte, die in der Menge O_v zusammengefasst werden. O_v entspricht daher einer

Virtualisierung von O, wenn es für jedes Objekt $o \in O$ eine Virtualisierung $o_v \in O_v$ gibt und umgekehrt.

$$o \overset{\text{V}}{\simeq} o_v :\Leftrightarrow o_v \text{ ist Virtualisierung von } o \tag{4.1}$$

$$O \overset{\text{V}}{\simeq} O_v :\Leftrightarrow \forall o \in O, \exists o_v \in O_v : o \overset{\text{V}}{\simeq} o_v \wedge |O| = |O_v| \tag{4.2}$$

Eindeutigkeit
Um eine eindeutige Zuordnung zu erreichen, wird bei der Virtualisierung eine Eindeutigkeit vorausgesetzt (4.3). Dies bedeutet, dass jedes physische Objekt $o \in O$ maximal eine virtuelle Entsprechung in der entsprechenden Virtualisierung $o_v \in O_v$ hat. Analog kann ein virtualisiertes Objekt nur eine physische Entsprechung haben. Aufgrund dieser Eindeutigkeit kann die Definition der Virtualisierung der Menge O auch wie in Gleichung (4.2) aufgestellt werden.

$$\begin{aligned} \forall o \in O, \forall o_v, o'_v \in O_v : o \overset{\text{V}}{\simeq} o_v \wedge o \overset{\text{V}}{\simeq} o'_v \overset{\text{def}}{\Rightarrow} o_v = o'_v \\ \forall o, o' \in O, \forall o_v \in O_v : o \overset{\text{V}}{\simeq} o_v \wedge o' \overset{\text{V}}{\simeq} o_v \overset{\text{def}}{\Rightarrow} o = o' \end{aligned} \tag{4.3}$$

Nimmt man eine Cyber-Physische Umgebung (CPE) als Grundlage für eine Virtualisierung, so umfasst die Menge von Objekten $O(CPE)$ alle Objekte dieser Umgebung. Eine Virtualisierung der Objekte wird in der Menge $O_v(CPE)$ zusammengefasst. Bei der Virtualisierung von CPE können insgesamt vier Varianten unterschieden werden.

Vollständige Virtualisierung
Bei einer *vollständigen Virtualisierung* gibt es zu jedem Objekt der physischen Umgebung eine virtualisierte Entsprechung (4.4). Ebenso gibt es für jedes Objekt in der Virtualisierung ein physisches Gegenstück.

$$O(CPE) \overset{\text{V}}{\simeq} O_v(CPE) \tag{4.4}$$

Partielle Virtualisierung
Bei der *partiellen Virtualisierung* hat nicht jedes physische Objekt der CPE eine Virtualisierung in der Menge $O_v(CPE)$ (4.5).

$$\exists O'(CPE) \subseteq O(CPE) : O'(CPE) \overset{\text{V}}{\simeq} O_v(CPE) \tag{4.5}$$

Erweiterte Virtualisierung
Im Gegensatz dazu umfasst bei der *erweiterten Virtualisierung* die Menge der virtuellen Objekte O_v nicht nur die virtualisierten Objekte der physischen Umgebung, sondern auch weitere virtuelle Objekte, deren physische Entsprechungen sich nicht in der CPE befinden und daher nicht in der Menge $O(CPE)$ enthalten sind (4.6).

$$\exists O'_v(CPE) \subseteq O_v(CPE) : O(CPE) \overset{\text{V}}{\simeq} O'_v(CPE) \tag{4.6}$$

Partiell erweiterte Virtualisierung
Schließlich kann eine Virtualisierung nur einen Teil der virtualisierten Objekte der CPE und zusätzliche virtuelle Objekte ohne physisches Gegenstück in der CPE beinhalten. Dies wird als *partiell erweiterte Virtualisierung* bezeichnet (4.7).

$$\exists O'_v(CPE) \subseteq O_v(CPE), \exists O'(CPE) \subseteq O(CPE) : O'(CPE) \overset{\text{v}}{\simeq} O'_v(CPE) \quad (4.7)$$

Zusammengefasst kann man feststellen, dass die partielle und die erweiterte Virtualisierung Spezialfälle der partiell erweiterten Virtualisierung sind. Ist die Teilmenge der physischen Objekte gleich der Menge an Objekten der CPE ($O'(CPE) = O(CPE)$), handelt es sich um eine erweiterte Virtualisierung. Entspricht die Teilmenge der virtuellen Objekte der virtuell modellierten Objekte ($O'_v(CPE) = O_v(CPE)$), handelt es sich um eine partielle Virtualisierung. Die vollständige Virtualisierung ist wiederum ein Spezialfall sowohl der partiellen als auch der erweiterten Virtualisierung und ist in der Regel nicht oder nur mit einem sehr großen Aufwand zu erreichen. Das gleiche gilt für eine erweiterte Virtualisierung, die ebenfalls eine Virtualisierung aller Objekte der CPE voraussetzt. In der Praxis reicht meist eine partielle Virtualisierung aus, um gewisse Bereiche der CPE zu monitoren. Möchte man Simulationen durchführen, müssen oftmals weitere virtuelle Objekte modelliert werden, die keine physische Entsprechung in der CPE haben, wodurch eine partiell erweiterte Virtualisierung entsteht.

Alternativ zur aufgeführten Formalisierung von Virtualisierung mittels Prädikatenlogik kann auch eine äquivalente Formalisierung auf Basis von Abbildungen aufgestellt werden. Eine Virtualisierung (4.2) entspricht hierbei einer bijektiven Abbildung eines physischen Objekts auf die virtuelle Repräsentation. Eine vollständige Virtualisierung beschreibt ein Isomorphismus zwischen der CPE und der virtuellen Umgebung. Eine partielle Virtualisierung entspräche einem injektiven Homomorphismus von der virtuellen in die reale Umgebung, wobei in diesem Fall zwischen der virtuellen Umgebung und einer Teilmenge der Objekte der realen Umgebung wiederum ein Isomorphismus vorliegt. Bei der erweiterten Virtualisierung liegt ein injektiver Homomorphismus von der realen in die virtuelle Umgebung vor. In diesem Fall besteht zwischen einer Teilmenge der Objekte der virtuellen und der realen Umgebung ein Isomorphismus. Aufgrund der Erweiterung der realen als auch der virtuellen Umgebung um Objekte, die keinen Gegenpart in der jeweils anderen Umgebung haben, stellt die partiell erweiterte Virtualisierung keine eigenständige Abbildung dar. Ausschließlich Teilmengen können als Homomorphismen bzw. Isomorphismen dargestellt werden.

4.2.3 Dual Reality in Cyber-Physischen Umgebungen

Die Virtualisierung stellt die Grundlage für Dual Reality (DR) dar (siehe Kapitel 2.4.1). Durch eine entsprechende Modellierung bekommt man eine virtuelle Repräsentation, welche durch geeignete Identifikationsmechanismen mit dem physischen Objekt verknüpft ist. Diese eindeutige Identifikation kann analog zu den Ansätzen des Internet der Dinge erfolgen (siehe Kapitel 2.1.2). Bei einer Virtualisierung einer CPE existiert im optimalen Fall zu jedem Sensor (S^i) und zu jedem Aktuator (A^i) eine entsprechende Virtualisierung (S^i_v bzw. A^i_v), wie in Abbildung 4.5 dargestellt. Diese virtuellen Repräsentationen weisen

dabei die gleichen Eigenschaften wie ihre physischen Entsprechungen auf und agieren daher in gleicher Art und Weise. Mit Hilfe einer entsprechenden Kommunikationsinfrastruktur können Informationen zwischen der realen und der virtuellen Welt ausgetauscht werden. Somit besteht eine gegenseitige Beeinflussung gemäß der Definition von Dual Reality (siehe Definition 1.2).

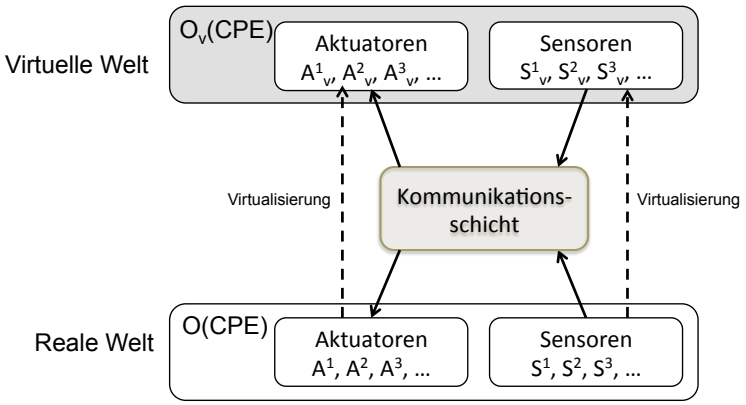


***Abbildung 4.5**: Interaktion zwischen Sensoren und Aktuatoren in Dual Reality Systemen*

Durch die Virtualisierung einer gesamten CPE werden mittels einer solchen Kommunikationsschicht auch die Verknüpfungen zwischen den Sensoren und Aktuatoren miteinander modelliert. Basierend auf dieser Kommunikation können alle Sensordaten, die in der CPE erfasst werden, in deren Virtualisierung $O'(CPE)$ übermittelt werden. Ebenso besteht die Möglichkeit, in der Virtualisierung initiierte Aktuatoransteuerungen in die physische Umgebung zurückzuspiegeln und dort auszuführen. Durch diesen Ansatz wird eine Synchronisation der beiden Umgebungen angestrebt, mit dem Ziel einer Verschmelzung zwischen der CPE und deren Virtualisierung. Hierdurch entsteht eine DR-Umgebung, die aus einer physischen und einer virtuellen Komponente besteht. Die Kommunikation zwischen diesen beiden erfolgt dabei mittels einer entsprechenden Sensor-Aktuator-Middleware und ist ebenfalls Teil dieser DR-Umgebung, wie in Abbildung 4.6 dargestellt. Mittels der Verschmelzung können die Vorteile beider Komponenten und damit der realen und der virtuellen Welten miteinander verbunden werden. Beispielsweise können Benutzerinteraktionen in der realen Umgebung erfasst und durch Auswertungen in der virtuellen Welt ergänzt werden. Die daraus resultierenden Aktionen werden anschließend sowohl in der virtuellen als auch in der realen Komponente ausgeführt. Somit kann die Virtualisierung mit der entsprechenden CPE synchron gehalten werden.

4.2.4 Formalisierte Definition von Dual Reality

Basierend auf der Formalisierung der Virtualisierung von CPE kann eine formalisierte Definition von Dual Reality (DR) aufgestellt werden. Für eine virtuelle Darstellung von Objekten müssen neben dem äußeren Erscheinungsbild auch weitere Eigenschaften berücksichtigt

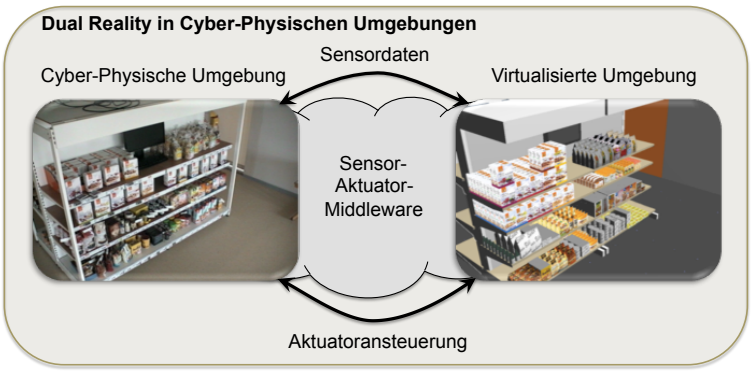


***Abbildung 4.6**: Dual Reality in Cyber-Physischen Umgebungen*

werden, wie beispielsweise die Position des Objekts in der Umgebung. Diese Eigenschaften inklusive aller Kontextparameter des Objekts o werden durch dessen Zustand $z(o)$ definiert. $z(O)$ beschreibt die Menge der Zustände aller Objekte $o \in O$.

Modellierung von DR über die Objektmenge O
Ist o_v eine Virtualisierung von o, so beschreibt der Zustand des virtuellen Objekts $z(o_v)$ eine DR-Virtualisierung des Zustands des physischen Objekts $z(o)$, wenn beider Eigenschaften einander entsprechen (4.8). Dies ist beispielsweise der Fall, wenn Form, Farbe und Position beider Objekte miteinander übereinstimmen. Nimmt beispielsweise ein Thermometer die Farbe Rot an, wenn die gemessene Temperatur über einem Schwellenwert liegt und hat sonst die Farbe Blau, muss die Virtualisierung dieses Verhalten ebenfalls annehmen. Analog dazu ist die Zustandsmenge von virtualisierten Objekten $z(O_v)$ eine DR-Virtualisierung der Zustandsmenge ihrer physischen Gegenstücke $z(O)$, wenn der Zustand eines jeden virtualisierten Objekts eine DR-Virtualisierung seiner physischen Entsprechung ist (4.9).

$$z(o) \stackrel{\wedge}{=}_{\text{DR}} z(o_v) :\Leftrightarrow o \overset{\text{V}}{\simeq} o_v \wedge z(o_v) \text{ ist DR-Virtualisierung von } z(o) \tag{4.8}$$

$$z(O) \stackrel{\wedge}{=}_{\text{DR}} z(O_v) :\Leftrightarrow O \overset{\text{V}}{\simeq} O_v \wedge \forall o \in O, \forall o_v \in O_v : o \overset{\text{V}}{\simeq} o_v \Rightarrow z(o) \stackrel{\wedge}{=}_{\text{DR}} z(o_v) \tag{4.9}$$

$z(O(CPE))$ umfasst alle Zustände und Informationen der in der CPE befindlichen Objekte. In diesem Fall kann man auch vom Zustand der gesamten Umgebung sprechen. $Z(O(CPE))$ umfasst alle möglichen Zustände, die die Objektmenge annehmen kann. Wie bereits in Kapitel 2.1.5 beschrieben, können Umgebungen hinsichtlich ihrer möglichen Zustände unterschieden werden. Im Fall von diskreten Umgebungen ist der Zustandsraum endlich, kann jedoch beliebig groß werden. Im kontinuierlichen Fall hingegen ist der Zustandsraum unendlich groß. Allein die Tatsache, dass Objekte beliebige Positionen annehmen können, führt zu einer kontinuierlichen Umgebung. Analog zu $Z(O(CPE))$ beschreibt $Z(O_v(CPE))$ alle potentiellen Zustände der virtuellen Objekte. Neben der Modellierung der Zustände müssen bei DR auch die Übergänge von einem Zustand in einen anderen betrachtet werden.

Zustandsänderungen durch Aktionen Die Übergänge erfolgen in Form von Aktionen $a(O)$, wobei zwischen atomaren und komplexen Aktionen unterschieden wird (4.10). Eine atomare Aktion beschreibt hierbei eine Handlung, die nur von einer Person oder einem Objekt als ein zusammenhängendes Ereignis durchgeführt werden kann. Eine komplexe Aktion besteht aus einer Abfolge mehrerer atomarer Aktionen. Die Menge aller möglichen Aktionen wird in $A(O)$ zusammengefasst.

$$z(O) \xrightarrow{a(O)} z'(O) :\Leftrightarrow z(O) \text{ geht aufgrund der Aktion } a(O) \text{ in Zustand } z'(O) \text{ über} \quad (4.10)$$

Vollständige DR In einer *vollständigen DR* wird jede Zustandsänderung der realen Umgebung in deren Virtualisierung widergespiegelt werden und umgekehrt (4.11). Hierbei liegen sowohl vor Durchführung der Aktion als auch im Anschluss daran Virtualisierungen der Zustände aller physischen und virtuellen Objekte vor.

$$\begin{aligned}
&\forall a(O(CPE)) \in A(O(CPE))\exists! a(O_v(CPE)) \in A(O_v(CPE)) : \\
&\qquad |A(O(CPE))| = |A(O_v(CPE))| \wedge \\
&\Big(z(O(CPE)) \xrightarrow{a(O(CPE))} z'(O(CPE)) \wedge z(O(CPE)) \stackrel{\wedge}{=}_{\text{DR}} z(O_v(CPE)) \wedge \\
&z'(O(CPE)) \stackrel{\wedge}{=}_{\text{DR}} z'(O_v(CPE)\Big) \Rightarrow z(O_v(CPE)) \xrightarrow{a(O_v(CPE))} z'(O_v(CPE)))
\end{aligned} \quad (4.11)$$

Partielle DR Hingegen existieren in einer *partiellen DR* lediglich für eine Submenge der Aktionen, die in der realen Welt ausgeführt werden können, entsprechende Aktionen in der virtuellen Umgebung, die dazu führen, dass nach der Ausführung die virtuelle Umgebung eine DR-Virtualisierung ihrer physischen Entsprechung ist (4.12).

$$\begin{aligned}
&\exists A'(O(CPE)) \subseteq A(O(CPE)) : \\
&\qquad \forall a'(O(CPE)) \in A'(O(CPE))\exists! a(O_v(CPE)) \in A(O_v(CPE)) : \\
&\qquad\qquad |A'(O(CPE))| = |A(O_v(CPE))| \wedge \\
&\Big(z(O(CPE)) \xrightarrow{a'(O(CPE))} z'(O(CPE)) \wedge z(O(CPE)) \stackrel{\wedge}{=}_{\text{DR}} z(O_v(CPE)) \wedge \\
&z'(O(CPE)) \stackrel{\wedge}{=}_{\text{DR}} z'(O_v(CPE)\Big) \Rightarrow z(O_v(CPE)) \xrightarrow{a(O_v(CPE))} z'(O_v(CPE)))
\end{aligned} \quad (4.12)$$

Erweiterte DR Analog können im Virtuellen zusätzliche Aktionen integriert sein, welche keine Entsprechung in der physischen Umgebung haben. Dies wird mit dem Begriff *erweiterte DR* bezeichnet (4.13).

$$\begin{aligned}
&\exists A'(O_v(CPE)) \subseteq A(O_v(CPE)) : \\
&\qquad \forall a(O(CPE)) \in A(O(CPE))\exists! a'(O_v(CPE)) \in A'(O_v(CPE)) : \\
&\qquad\qquad |A(O(CPE))| = |A'(O_v(CPE))| \wedge \\
&\Big(z(O(CPE)) \xrightarrow{a(O(CPE))} z'(O(CPE)) \wedge z(O(CPE)) \stackrel{\wedge}{=}_{\text{DR}} z(O_v(CPE)) \wedge \\
&z'(O(CPE)) \stackrel{\wedge}{=}_{\text{DR}} z'(O_v(CPE)\Big) \Rightarrow z(O_v(CPE)) \xrightarrow{a'(O_v(CPE))} z'(O_v(CPE)))
\end{aligned} \quad (4.13)$$

Partiell erweiterte DR Schließlich gibt es noch die *partiell erweiterte DR*, in der es sowohl eine Teilmenge von Aktionen in der realen als auch in der virtuellen Umgebung geben kann, die in der jeweils anderen Umgebung keine Entsprechung haben (4.14).

$$\begin{aligned}
&\exists A'(O_v(CPE)) \subseteq A(O_v(CPE)), \exists A'(O(CPE)) \subseteq A(O(CPE)) : \\
&\quad \forall a'(O(CPE)) \in A'(O(CPE)) \exists! a'(O_v(CPE)) \in A'(O_v(CPE)) : \\
&\quad\quad |A'(O(CPE))| = |A'(O_v(CPE))| \wedge \\
&\quad \Big(z(O(CPE)) \xrightarrow{a'(O(CPE))} z'(O(CPE)) \wedge z(O(CPE)) \stackrel{\wedge}{=}_{\text{DR}} z(O_v(CPE)) \wedge \\
&z'(O(CPE)) \stackrel{\wedge}{=}_{\text{DR}} z'(O_v(CPE)) \Big) \Rightarrow z(O_v(CPE)) \xrightarrow{a'(O_v(CPE))} z'(O_v(CPE)))
\end{aligned} \tag{4.14}$$

Aufgrund des sehr großen bis zu unendlich großen Zustandsraums und der entsprechend großen Menge an möglichen Aktionen ist eine vollständige DR in der Regel nicht realisierbar. Daher wird im allgemeinen Gebrauch unter DR eine partielle DR verstanden. Um Virtualisierungen auch als Testumgebungen einsetzen zu können, müssen oftmals zusätzliche Objekte im Virtuellen integriert und weitere virtuelle Zustandsänderungen ermöglicht werden, um die gewünschten Resultate zu erzielen. In diesem Fall ist es notwendig, eine partiell erweiterte DR zur Verfügung zu stellen. Analog zur Formalisierung von Virtualisierung kann auch die Formalisierung von DR in Form von Abbildungen erstellt werden.

Mittels Simulationen können darauf aufbauend die Objekte sowohl in der physischen als auch in der virtuellen Umgebung beeinflusst werden. Simulationen dienen der Erprobung von möglichen Veränderungen, welche in der Realität nicht stattgefunden haben. Diese sollen jedoch keinen folgenden Einfluss auf die Realität aufweisen. Aus diesem Grund muss nach Abschluss der Simulationen die Möglichkeit bestehen, die physische Umgebung auf den Zustand zu bringen, der ohne diese Simulationen vorliegen würde. Um diesen Sachverhalt formal zu beschreiben, wurde im Rahmen dieser Arbeit der Begriff *Erweiterte Dual Reality (DR++)* entwickelt und folgendermaßen definiert.

Definition 4.3 (Erweiterte Dual Reality) *Erweiterte Dual Reality beschreibt die Verschmelzung von einer realen Umgebung und deren Virtualisierung in Form einer virtuellen Umgebung, welche durch Netzwerke von Sensoren und Aktuatoren miteinander verbunden sind. Dadurch weisen beide Umgebungen die Fähigkeit auf, sich gegenseitig widerzuspiegeln. Neben real erfassten Sensorinformationen können während Simulationen auch simulierte Daten entweder die reale Umgebung, deren Virtualisierung oder beide beeinflussen, wodurch möglicherweise die Synchronisation zwischen beiden Umgebungen nicht mehr gewährleistet ist. Nach Abschluss der Simulation wird in der realen Umgebung der Zustand hergestellt, welcher ohne die Simulation existiert hätte. Zusätzlich ist die Möglichkeit geboten, beide Repräsentationen wieder zu synchronisieren, so dass die virtuelle Umgebung wieder einer Virtualisierung der physischen Umgebung entspricht.*

4.3 Visualisierung

Neben der Möglichkeit Simulationen durchführen zu können ist eine visuelle Darstellung der Umgebung der Hauptgrund für deren Virtualisierung. Visualisierungen können typischerweise zwei Verwendungszwecke haben: einerseits um ein Verständnis der Phänomene zugrunde liegender Daten zu erhalten und andererseits um dieses Verständnis einer Zuhörerschaft mitzuteilen [Kreylos et al., 2003]. Die Tools zur 3D-Modellierung von Objekten können diese zwar darstellen und in hochauflösender Form rendern, jedoch sind sie nicht für eine interaktive Handhabung gedacht. Hierzu werden spezielle 3D-Visualisierungskomponenten benötigt, die basierend auf der Modellierung die entsprechenden Objekte darstellen und eine Möglichkeit zur Interaktion mit der entstandenen Szene bieten. Neben der reinen Darstellung können bei solchen Visualisierungskomponenten weitere Informationen, welche von externen Applikationen stammen, der Szene hinzugefügt werden. Beispielsweise können zusätzliche Daten aus einer Datenbank mit dem Objekt verknüpft und dadurch dem Modell hinzugefügt werden. Dazu werden von diesen Komponenten entsprechende Schnittstellen bereitgestellt. Im Folgenden werden einige solcher Visualisierungskomponenten vorgestellt. In Tabelle 4.1 werden die Differnzierungsmerkmale der Komponenten zusammengefasst und einander gegenübergestellt.

YAMAMOTO: Das aus einem Forschungskontext entstandene Tool YAMAMOTO ist sowohl für die Modellierung von Umgebungen als auch für eine anschließende Interaktion mit diesen gedacht. Aufgrund der Implementierung in C# ist die Ausführung jedoch vorwiegend auf Windows-Systeme ausgelegt. Neben der Integration eines virtuellen Avatars können auch externe Einflüsse von Anwendungen, die außerhalb von YAMAMOTO laufen, in der Darstellung berücksichtigt werden. Beispielsweise wurde das VNC-Protokoll integriert, so dass Bildschirminhalte aus der realen Umgebung auch in der virtuellen Umgebung dargestellt werden können. Zudem können externe Sensoren und Aktuatoren eingebunden werden, so dass eine Lampe mit dessen Virtualisierung gemäß des Dual Reality Paradigmas synchronisiert werden kann [Stahl, 2009]. Die Darstellung, ob die Lampe ein- oder ausgeschaltet ist, wird durch einen hellen bzw. dunklen Kegel ausgehend von deren Virtualisierung repräsentiert. Das Anklicken dieses Kegels führt zu dem gleichen Ergebnis, als wenn in der realen Welt auf den zugehörigen Lichtschalter gedrückt wird. Für die Synchronisation werden in der Regel spezielle Schnittstellen benötigt. Um dies zu verallgemeinern wurde in YAMAMOTO eine Schnittstelle zum Universal Control Hub (siehe Abschnitt 3.2.6) eingebunden, welche einen standardisierten Zugriff auf externe Komponenten ermöglicht [Frey et al., 2011]. Ein Import extern modellierter Objekte ist nicht möglich, was sich insbesondere bei komplexen Objekten negativ bemerkbar macht, da diese basierend auf einer Verschachtelung von Boxen in YAMAMOTO nachmodelliert werden müssen. Neben der Möglichkeit der Darstellung im Tool selbst bietet YAMAMOTO eine Exportfunktionalität in VRML (siehe Abschnitt 2.4.2), wodurch eine mobile Darstellung der Szene ermöglicht wird. Die Darstellung mittels des exportierten Formats unterstützt jedoch keine Synchronisation mit der realen Umgebung, sondern dient einzig der Visualisierung des Umgebungsmodells.

SecondLife: Eine weitere 3D-Visualisierungskomponente stellt SecondLife dar (siehe Kapitel 2.4.2). Diese ist als Webapplikation realisiert, wobei das Modell auf einem zentralen Server im Internet hinterlegt wird, die Clients sich mit diesem verbinden und auf das Modell zugreifen. Um eine öffentlich zugängliche Darstellung einer modellierten Umgebung realisieren zu können, muss jedoch ein entsprechendes Stück Land in Form einer „Insel“ gekauft werden, die die jeweilige Darstellungsumgebung repräsentiert. Zur Darstellung muss der Client sich mit dem Server verbinden, was eine Internetverbindung zwingend erforderlich macht. Eine mobile Darstellung von SecondLife ist in einer eingeschränkten Fassung ebenfalls möglich, z.B. mit einem Viewer namens Lumiya[7], welcher für mobile Endgeräte auf Basis von Android zur Verfügung steht. Für die Kommunikation zwischen Second Life und externen Applikationen gibt es eine spezielle Skriptsprache, die Linden Scripting Language (LSL). Mit dieser können Informationen in das virtuelle Modell importiert oder von dort aus an externe Dienste exportiert werden. Dabei hat die LSL eine limitierte Skriptgröße für die Übertragung, was bei komplexen Kommunikationsanweisungen problematisch sein kann [Prendinger et al., 2009]. Um extern modellierte Objekte importieren zu können, müssen diese zunächst in ein spezielles Second Life-Format umgewandelt werden.

OpenSimulator: OpenSimulator ist eine Alternative zu Second Life, welche komplett kostenlos erhältlich ist. Dabei kann der Server auf einem lokalen Computer eingerichtet werden, mit dem sich die Clients direkt im Intranet verbinden können. Somit ist keine Internetverbindung notwendig. Für die Darstellung der Modelle werden mehrere externe, lokal installierte Viewer unterstützt. Für die Kommunikation wird ebenfalls LSL verwendet, wodurch eine Kompatibilität mit Second Life gewährleistet wird. Damit ist man bei der Kommunikation auch in diesem Fall auf die reduzierte Skriptgröße beschränkt. Des Weiteren werden jedoch zusätzliche Schnittstellen basierend auf REST und JSON angeboten, um eine flexiblere Kommunikation zu ermöglichen. Analog zu Second Life ist die Größe der zu konstruierenden Welt auf maximal 256×256 m beschränkt.

Java3D: Als plattformunabhängige Programmiersprache unterstützt Java auch eine 3D-Darstellung mittels der Erweiterung Java3D. Die Darstellung erfolgt hierbei über die OpenGL-Schnittstelle. Dabei werden mehrere Funktionalitäten und Parametereinstellungen durch die Implementierung von Java3D gekapselt, was den Entwickler in der Anwendung unterstützt. Durch die vollständige Integration der 3D-Visualisierung in Java können externe Java-Applikationen mit wenig Aufwand integriert und weiterverwendet werden. Besonders der große Funktionsumfang, der durch Oracle und mehrere Entwicklungsgruppen bereitgestellt wird, kann bei Auswertungen wie beispielsweise Distanz- und Schnittberechnungen und der Verarbeitung der 3D-Szenen eingesetzt werden. Die Darstellung der 3D-Szene erfolgt in der Regel auf dem gleichen Rechner, welcher auch das Modell lädt. Eine Mehrbenutzer-Unterstützung ist in Java3D nicht gegeben.

XML3D: Neben den oben genannten Anwendungen, die die Installation einer speziellen Software zur Visualisierung der 3D-Szene voraussetzen, existieren auch webbasierte

[7] http://www.lumiyaviewer.com, Zugriff: November 2014

	YAMAMOTO	Second Life	Open Simulator	Java3D	XML3D
Mehrbenutzer-Unterstützung	nein	ja	ja	nein	in Entwicklung
Server- / Webbasiert	lokal	Internet	Intranet	lokal	Intranet
Client / Applikation / Viewer	spezielle Applikation (Windows)	spezielle Applikation (Cross-Plattform)	spezielle Applikation (Cross-Plattform)	Java Applikation	Browser
Mobile Version	VRML Viewer	spezieller Viewer	spezieller Viewer	nein	*
Schnittstelle / Implementierung	C# / UCH	LSL	LSL	Java	JavaScript
Eigenschaften / Einschränkungen	kein Import von extern modellierten 3D-Modellen	Größenbeschränkung der Modellierungsfläche	Größenbeschränkung der Modellierungsfläche	keine Weiterentwicklung, große Funktionsvielfalt	in Weiterentwicklung

* Neuere Smartphones unterstützen bereits WebGL/OpenGL, zudem laufen aktuelle Entwicklungen auf das Rendering auf einem Server mit anschließender Übertragung auf ein (mobiles) Endgerät hinaus.

***Tabelle 4.1**: Differenzierungsmerkmale zwischen den Visualisierungskomponenten*

Darstellungsarten (siehe Kapitel 2.4.2). Diese benötigen keine spezielle Visualisierungssoftware, sondern verwenden Internetbrowser mit deren Schnittstellen über WebGL zur Grafikkarte für die Visualisierung. Ein Beispiel hierzu stellt die deklarative Beschreibungssprache XML3D dar, welche eine Darstellung von 3D-Inhalten in Webbrowsern ohne eine Installation eines externen Plugins mittels einer OpenGL-Unterstützung ermöglicht [Sons et al., 2010]. Dadurch können 3D-Inhalte neben standardmäßigen Webinhalten auf der gleichen Internetseite integriert werden. Durch die Unterstützung von JavaScript können auch Benutzerinteraktionen mit der Szene erkannt und darauf reagiert werden. Mobile Endgeräte, welche OpenGL unterstützen, können die 3D-Szenen ebenfalls darstellen. Zudem wird seitens XML3D die Bibliothek *XFlow* bereitgestellt, welche die Verwendung von Animationen unterstützt [Klein et al., 2012]. Des Weiteren umfasst XFlow Funktionalitäten aus den Bereichen der Bildverarbeitung und der Erweiterten Realität (siehe Kapitel 2.4.1), basierend auf der Integration von Markererkennung [Klein et al., 2013]. Neben der Erweiterung von XML3D durch XFlow sind weitere Ergänzungen geplant, wie beispielsweise ein serverbasiertes Rendering. Dies hat den Vorteil, dass leistungsstarke Server die 3D-Szene rendern und als Videostreams mobilen Endgeräten zur Verfügung stellen können. Somit wird es auch möglich sein, komplexere Szenen auf mobilen Endgeräten darzustellen, was aktuell aufgrund der Ressourcenbeschränkung noch nicht möglich ist.

Neben den Visualisierungskomponenten können auch die Darstellungsvarianten gemäß gewisser Parameter voneinander abgegrenzt werden. Zum Beispiel können Bewegungen in

einer Szene, wie das Wehen einer Fahne, zu einer dynamischen und lebendigen Darstellung führen. Für eine reine Informationsdarbietung können jedoch statische Visualisierungen zielführender sein, da diese den Benutzer nicht von wichtigen Informationen ablenken, um die kognitive Belastung des Betrachters möglichst gering halten. Des Weiteren kann eine Szene als einfache Darstellung visualisiert werden, ohne die Möglichkeit für den Benutzer, in die Szene einzugreifen, analog zu Fotos oder Videos. Demgegenüber steht eine interaktive Repräsentation, bei der ein Benutzer die Darstellung mittels Interaktion selbstständig verändern kann, wie es unter anderem in 3D-Spielen der Fall ist. Für eine Informationsrepräsentation, wie beispielsweise des aktuellen Stromverbrauchs von elektronischen Geräten, können einerseits textuelle Darstellungen gewählt werden, welche eine exakte Wertangabe ermöglichen. Andererseits können diese Informationen auch in eine grafische Darstellung kodiert werden, indem beispielsweise das Objekt über eine farbliche Überblendung hervorgehoben wird, wobei unterschiedliche Farben unterschiedliche Informationen repräsentieren. Schließlich können sich visuelle Darstellungen im Detailgrad der Repräsentation unterscheiden. Einerseits kann die Visualisierung eines Objekts gemäß einer exakten Virtualisierung besonders realitätsgetreu sein. Andererseits kann für bestimmte Einsatzgebiete eine abstrakte Darstellung vorteilhaft sein, z.B. in Form eines Comic-Stils oder mittels einer Repräsentation basierend auf einfachen geometrischen Objekten. Damit einhergehend ist auch der Realitätsgrad der Modellierung der Objekte in einer Visualisierung zu unterscheiden. Zum einen kann das Modell eine exakte Abbildung existierender Objekte sein und zum anderen kann es eine erweiterte Form mittels einer Simulation darstellen. Die Unterscheidungskriterien der aufgeführten Darstellungsvarianten sind in Tabelle 4.2 zusammengefasst.

Unterscheidungsparameter	**Ausprägungen**		
Bewegung	statisch	vs.	dynamisch
Interaktionsmöglichkeit	interaktiv	vs.	darstellend
Informationsdarstellung	grafisch	vs.	textuell
Darstellung	von realitätsgetreu	bis	abstrakt

***Tabelle 4.2**: Unterscheidungskriterien bei Darstellungsvarianten*

4.4 Zusammenfassung

In diesem Kapitel wurde die Modellierung von einzelnen Objekten und von gesamten Umgebungen thematisiert. Anhand existierender Tools wurde dargelegt, wie eine Modellierung erfolgen kann und welche Unterschiede bei den entsprechenden Datenformaten existieren. Anschließend wurde auf eine spezialisierte Form der Modellierung, die Virtualisierung, eingegangen, bei der das virtuelle Modell eine physische Entsprechung aufweisen muss. In diesem Zusammenhang wurde der Begriff Virtualisierung definiert und Besonderheiten bei der Virtualisierung von CPE beschrieben, insbesondere im Hinblick auf Sensoren und

Aktuatoren. Anschließend wurde die Verschmelzung von physischer Umgebung und ihrer Virtualisierung beschrieben, was unter dem Begriff Dual Reality (DR) bekannt ist. Basierend auf einer Formalisierung von Virtualisierung wurde auch das Dual Reality-Konzept formalisiert, um eine Möglichkeit zu schaffen, DR-Applikationen voneinander abzugrenzen und klassifizieren zu können. Hierbei wurde dahingehend unterschieden, welche Aktionen in den beiden Welten durchgeführt werden können. Anhand dieser Formalisierung wurde der Begriff Erweiterte Dual Reality (DR++) definiert, welcher DR um die Möglichkeit der Integration von Simulationen erweitert. Schließlich wurde auf die Visualisierung von virtualisierten CPE eingegangen. Dabei wurden mehrere 3D-Visualisierungskomponenten vorgestellt und ihre Vor- und Nachteile erörtert. Des Weiteren wurden Unterscheidungskriterien bei der Visualisierung erarbeitet und erläutert, die je nach Applikation und Verwendungszweck unterschiedlich ausgeprägt sein sollten.

Im nachfolgenden Kapitel wird das Konzept und die Implementierung des Dual Reality Management Dashboards vorgestellt - ein 3D-Visualisierungstool, welches die interaktive Darstellung einer CPE unterstützt. Durch eine Verknüpfung mit den Sensoren und Aktuatoren der entsprechenden physischen Umgebung erfüllt dieses Tool die Definition von DR++, indem es die Möglichkeit der Integration von Simulatoren explizit unterstützt.

5 DUAL REALITY MANAGEMENT DASHBOARD DURMAD

Um eine interaktive 3D-Darstellung einer Cyber-Physischen Umgebung (CPE) zu ermöglichen, wurde ein Tool entwickelt, welches basierend auf einem Grundrissplan und dreidimensional modellierten Objekten automatisiert das entsprechende Umgebungsmodell generiert und visualisiert. Basierend auf der interaktiven 3D-Visualisierung kann dieses Tool auch als Monitoring- und Steuerungskomponente für CPE verwendet werden, da eine gegenseitige Beeinflussung zwischen der physischen Umgebung und ihrer Virtualisierung vorliegt. Es trägt den Namen *Dual Reality Management Dashboard (*DURMAD*)* [Kahl et al., 2011b; Kahl, 2013b], da es Entscheidungsträgern, wie Führungskräften, alle relevanten Informationen zum aktuellen Zustand der Umgebung aufbereitet präsentieren kann, gemäß der Definition eines Dashboards (siehe Definition 2.13 in Kapitel 2.5.4). Der Begriff Management wird in dieser Arbeit in Anlehnung an die Erläuterung aus dem Gabler Wirtschaftslexikon[1] folgendermaßen definiert.

Definition 5.1 (Management) *Unter Management ist die Vertretung und Überwachung der Einhaltung von Unternehmensinteressen durch Führungskräfte zu verstehen. Neben der Aufgabendefinition ist die Kontrolle basierend auf einem Soll-/Istvergleich eine wichtige Aufgabe, um in einem nächsten Schritt die weiteren Planungen zu steuern.*

DURMAD besteht aus fünf Bereichen, die miteinander kombiniert werden. Einerseits gibt es die *Datenquellen*, welche Informationen zu der Umgebung sowie zu den einzelnen Objekten beinhalten. Andererseits bietet DURMAD spezifische *Schnittstellen*, um Informationen aus weiteren Datenquellen in DURMAD verwenden zu können und um Daten externen Applikationen bereitzustellen. Des Weiteren gibt es die *Kernmodule*, welche die grafische Repräsentation und Interaktionsfunktionalität umfassen. Diese sind in Java mit der Erweiterung Java3D implementiert und sind daher plattformunabhängig einsetzbar. Zudem gibt es *Kontroll- und Steuerungssysteme*, welche eine Verbindung zwischen den physischen Objekten und deren Virtualisierungen ermöglichen. Schließlich können externe *Verarbeitungsautomatismen* an DURMAD angeschlossen werden, so dass ein direkter Zugriff auf die enthaltenen Daten zur Verarbeitung, wie beispielsweise bei Simulationen, ermöglicht wird. Die aus diesen Komponenten resultierende Architektur ist in Abbildung 5.1

[1] http://wirtschaftslexikon.gabler.de/Archiv/55279/management-v9.html, Zugriff: November 2014

dargestellt. Während in der Architekturskizze nur die bereichsübergreifenden Kommunikationsschnittstellen dargestellt sind, werden im Folgenden die einzelnen Bereiche inklusive des internen Informationsausstauschs detailliert beschrieben.

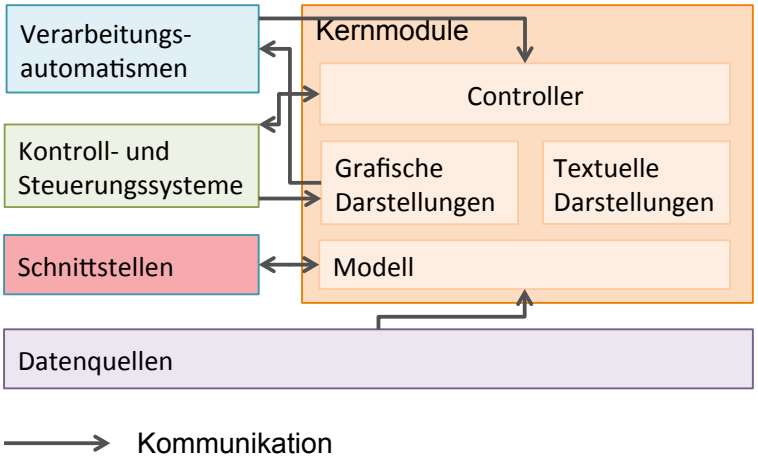


***Abbildung 5.1**: Architekturskizze des Dual Reality Management Dashboards*

5.1 Datenquellen

Für eine möglichst realistische Darstellung einer CPE müssen die entsprechenden Modelle der einzelnen Objekte in Form von Virtualisierungen vorliegen, damit diese entsprechend geladen und dargestellt werden können (siehe Abschnitt 5.1.1). Neben diesen Modellen ist ein Grundrissplan erforderlich, mit Hilfe dessen die Szene automatisch aufgebaut werden kann (siehe Abschnitt 5.1.2). Schließlich können auch semantische Beschreibungen von Objekten verwendet werden, um eine automatisierte Unterstützung des Benutzers zu ermöglichen (siehe Abschnitt 5.1.3). Eine Darstellung der Datenquellen in DURMAD inklusive möglicher Ausprägungen ist in Abbildung 5.2 aufgezeigt.

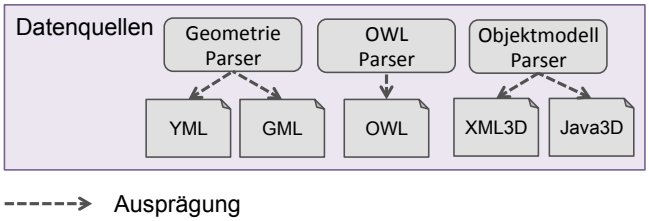


***Abbildung 5.2**: Datenquellen des Dual Reality Management Dashboards*

5.1.1 Objektmodelle

Die Objektmodellierung erfolgt, indem zunächst mittels eines Modellierungstools, wie beispielsweise Blender (siehe Abschnitt 4.1.1), das entsprechende Objekt virtualisiert wird. Neben den äußeren Geometriefaktoren werden dabei bereits Farbgebung und Texturen mit berücksichtigt. Anschließend kann das erstellte Modell exportiert werden. Blender bietet einen Exporter, der es ermöglicht, Modelle in einem von Java3D nativ unterstützten Format zu speichern, so dass DuRMaD diese importieren und darstellen kann. Dieses Format verwendet die XML-Syntax und spiegelt die interne Datenstruktur der resultierenden Java-Objekte wider. Ein Beispiel eines solchen Formats befindet sich im Anhang in Programmauszug A.1. Darin enthalten sind Punkt- und Texturkoordinaten, Informationen zur Textur und zum Material in Form von Farbwerten. Die Schnittstelle zwischen Java3D und OpenGL wurde seit 2004 nicht erweitert, so dass aktuelle Operationen in OpenGL, welche eine realistischere Darstellung ermöglichen würden, nicht unterstützt werden. Beispielsweise können in Java3D nicht alle Farbeigenschaften und Shader spezifiziert werden, welche von OpenGL zum Rendern realistischer Szenen benötigt werden. Aus diesem Grund ist keine realistische Darstellung mit Java3D möglich.

Zur Verwendung von anderen, aktuelleren Dateiformaten müssen entsprechende Parser entwickelt werden, welche die Dateien einlesen und in ein Java3D-kompatibles Format konvertieren. Exemplarisch hierfür wurde im Rahmen dieser Arbeit ein XML3D-Parser entwickelt. In Programmauszug A.2 im Anhang wird dasselbe Objekt in XML3D-Syntax dargestellt, welches in Programmauszug A.1 in Java3D-Syntax veranschaulicht ist. Wie der Name des Formats bereits spezifiziert, basiert XML3D ebenfalls auf XML. Zusätzlich zu den Punkt- und Texturkoordinaten umfasst das Modell Koordinaten für die Normalen, welche für das spätere Rendering wichtig sind. Des Weiteren sind spezifische Informationen zu den Materialeigenschaften im Modell hinterlegt. XML3D-Repräsentationen können ebenfalls aus diversen Modellierungstools, wie beispielsweise Blender oder Cinema4D, mittels entsprechender Skripte exportiert werden. Alternativ zu einer komplett eigenständigen Modellierung können auch bereits vorhandene Modelle in eine Modellierungsumgebung geladen, darin modifiziert und in eines der beiden Formate (Java3D oder XML3D) exportiert werden. Beispielsweise stellt Google Sketchup ein großes Spektrum von bereits modellierten und frei erhältlichen Modellen zur Verfügung.

5.1.2 Grundrissplan

Für die Repräsentation einer CPE wird neben den Objektmodellen auch ein Grundrissmodell benötigt, welches die Positionen der Objekte umfasst. Als Basis für solch einen Grundrissplan werden zwei Dateiformate unterstützt, für die jeweils ein entsprechender Parser implementiert wurde, nämlich yml und gml. Da das Tool YAMAMOTO [Stahl und Haupert, 2006] speziell für die einfache Erstellung von Modellen instrumentierter Umgebungen konzipiert wurde, besteht durch die Unterstützung des yml-Formats eine Möglichkeit, verhältnismäßig schnell und ohne explizites Modellierungswissen räumliche Modelle zu erstellen und in DuRMaD darzustellen. Basierend auf den vom Benutzer definierten Kanteneigenschaften werden automatisiert die entsprechenden 3D-Konstrukte von Wänden,

Türen sowie Fenstern erstellt und dem Modell hinzugefügt. Analog dazu werden modellierte Sensoren ebenfalls in die 3D-Szene integriert. Gleiches gilt für weitere Objekte, die in der yml-Datei hinterlegt sind, wie beispielsweise Displays oder verschachtelte Boxen, um komplexere Strukturen darstellen zu können. Dabei wandelt der Parser die in der yml-Datei vorhandenen Objekte in eine intern spezifizierte Struktur um, welche Typen, Namen, Dimensionen, Positionen und Orientierungen der Objekte enthält. Eine Zuordnung der in yml spezifizierten Objekte zu den entsprechenden vormodellierten Objekten für die Darstellung und zu den physischen Repräsentationen erfolgt mittels ihrer Typen und Namen. Hierzu fungieren die Namen als eindeutige Identifikationsschlüssel. Die resultierende Java3D-Struktur wird den Kernkomponenten von DURMAD zur Verfügung gestellt und ermöglicht somit, die entsprechenden Objekte zu erfassen, automatisch zu laden und an die entsprechende Position in der Szene zu setzen. Ein Auszug aus einer beispielhaften yml-Datei einer instrumentierten Umgebung ist im Anhang in Programmauszug A.3 zu finden.

Für die Verwendung von gml-Dateien wurde ebenfalls ein Parser entwickelt, der die entsprechenden Konstrukte erfasst. Der Vorteil des implementierten Parsers ist, dass er die Semantik, um welchen Objekttyp es sich handelt und wie sich dieser zusammensetzt, bereits beim Laden berücksichtigen kann. Dies ermöglicht die automatische Zuordnung der entsprechenden 3D-Modelle zur späteren Darstellung. Dabei müssen, wie bei yml, die vorkommenden Objekttypen bereits a priori bekannt und im Parser in einer Bibliothek definiert sein. Analog zum Vorgehen beim yml-Parser werden die geladenen Daten in eine entsprechende Struktur gepackt, welche anschließend von DURMAD verarbeitet und angezeigt werden kann. Ähnlich wie beim yml-Modell werden auch bei den gml-Dateien eindeutige Zuordnungen zur physischen Instanz mit Hilfe eines Identifikationsparameters geschaffen. Beispiele für zwei verwendete gml-Dateien sind im Anhang in Programmauszug A.4 und A.5 aufgeführt. In diesem Fall besteht zwischen den beiden Modellen eine Korrelation, welche über die „ShelfId“ beschrieben und entsprechend vom Parser berücksichtigt wird. In diesem Fall wird eine Unterteilung des Regals in kleinere Abschnitte, den Regalmetern, dem eigentlichen Regal zugeordnet. Nachdem der Grundrissplan durch den jeweiligen Parser geladen wurde, wird anhand der Informationen zu jedem einzelnen Objekt das entsprechende Objektmodell geladen und anschließend visualisiert.

5.1.3 Semantik

Zusätzlich zu visuellen Merkmalen können Modelle auch weitere Eigenschaften umfassen. Diese sind nicht in den zuvor beschriebenen Objektmodellen inkludiert, sondern werden separat spezifiziert. Im Speziellen bieten Sensoren und Aktuatoren gemäß ihrer Typen diverse Funktionalitäten. Beispielsweise kann ein Temperatursensor die Temperatur messen oder ein Tor sich öffnen und schließen. Damit auch Computer solche Funktionalitäten verstehen und damit operieren können, müssen diese Eigenschaften in einer entsprechend verständlichen Form hinterlegt werden. Dies erfolgt in der Regel durch eine entsprechende Ontologie, welche eine semantische Beschreibung darstellt. Eine Ontologie wird dabei als Menge von formal definierten Konzepten und Relationen, die für eine Wissensdomäne relevant sind, verstanden [Russomanno et al., 2005; Gruber, 1993]. Die Verknüpfung

eines solchen Konzepts mit einem physischen Objekt erfolgt mittels einer entsprechenden Instanziierung. Dies bedeutet, dass für jeden Sensor- bzw. Aktuatortyp ein entsprechendes Konzept in der Ontologie modelliert wird. Für jedes Objekt eines solchen Sensor- oder Aktuatortyps wird eine entsprechende Instanz des zugehörigen Konzepts angelegt und mit dem physischen Objekt verknüpft. Eine solche semantische Annotation kann anschließend hergenommen werden, um diverse Automatismen über die Objekte durchzuführen, was in der Regel in Agentensystemen verwendet wird (siehe Kapitel 2.1.5). In DURMAD werden die Eigenschaften der Sensoren und Aktuatoren semantisch beschrieben, um darüber Zusatzinformationen anzeigen und mögliche Interaktionen in der 3D-Visualisierung ermöglichen zu können.

Ein weit verbreitetes Format zur Beschreibung von Ontologien ist *OWL (Web Ontology Language)*, welches eine Spezifikation des World Wide Web Consortium (W3C) ist. Zur Erstellung solcher OWL-Dateien gibt es Programme, die den Benutzer bei der Modellierung unterstützen, analog zu den grafischen Modellierungstools. Ein solches Programm ist z.B. Protégé[2]. Dieses Programm ermöglicht die einfache Erstellung von Klassen, Instanzen sowie Verbindungen zwischen diesen. Abbildung 5.3 zeigt eine Visualisierung einer in dieser Arbeit entwickelten Ontologie für CPE, welche Klassen für Sensoren, Aktuatoren und Objekte umfasst. Die semantische Modellierung von Personen wurde bei der Grundfunktionalität der erstellten Ontologie ausgeklammert, da diese hauptsächlich zur Personalisierung der jeweiligen Systeme herangezogen wird, für die Verbindung der Sensoren und Aktuatoren in CPE jedoch keine wichtige Rolle einnimmt. Dennoch können auch semantische Beschreibungen von Personen für DURMAD relevant sein. In diesem Fall kann auf bereits existierende Benutzerontologien, wie beispielsweise *GUMO* [Heckmann et al., 2005], zurückgegriffen werden.

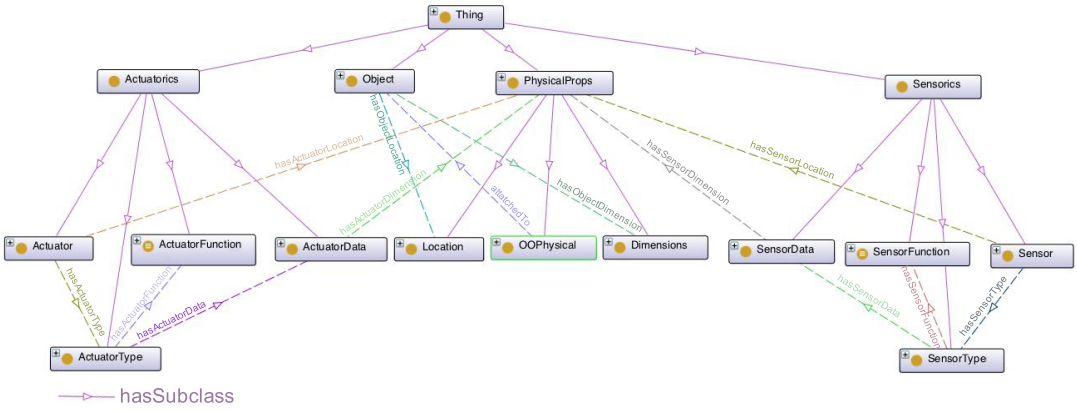


***Abbildung 5.3**: Darstellung der OWL-Semantik für Sensoren und Aktuatoren in* DURMAD

Wie in einer Ontologie typisch, werden die jeweiligen Klassen in der DURMAD-Ontologie von der allgemeinen Oberklasse „Thing“ abgeleitet. Zudem sind Informationen

[2]http://protege.stanford.edu, Zugriff: November 2014

bezüglich des Aktuatortyps (*ActuatorType*), der zugehörigen Daten (*ActuatorData*), Funktionen (*ActuatorFunction*) sowie der physikalischen Eigenschaften (*Actuator*) in einer Oberklasse (*Actuatorics*) zusammengefasst. Entsprechendes gilt auch für die Sensorinformationen (*Sensorics*). Des Weiteren werden physikalische Eigenschaften, welche Position und Größen umfassen, in der Klasse *PhysicalProps* zusammengefasst. Neben den Spezifikationen der Subklassen zeigt die in Abbildung 5.3 dargestellte Ontologie die Verbindung zwischen diesen auf. Zum Beispiel beschreibt eine physikalische Eigenschaft, dass eine physische Instanz eines Sensors oder Aktuators an einem Objekt angebracht sein kann (*attachedTo*). Zudem werden Sensoren und Aktuatoren sowohl Positionen als auch Größeninformationen zugeordnet (*hasSensorLocation*, *hasSensorDimension*, *hasActuatorLocation*, *hasActuatorDimenision*). Des Weiteren haben Sensoren und Aktuatoren einen spezifischen Typ (*hasSensorType*, *hasActuatorType*), umfassen spezifische Daten (*hasSensorData*, *hasActuatorData*) und bieten entsprechende Funktionalitäten an (*hasSensorFunction*, *hasActuatorFunction*). Ebenso haben Objekte eine Position und Größeninformationen (*hasObjectLocation*, *hasObjectDimension*).

Gemäß der entsprechenden CPE und den verwendeten Sensoren und Aktuatoren können in dieser Ontologie Instanzen angelegt werden, die diese beschreiben. Um diese Informationen auch bei der Darstellung in DURMAD zu verwenden, wurde ein entsprechender Parser dieser Ontologie erstellt, welcher die Instanzinformationen ausliest und mit den entsprechenden Virtualisierungen verknüpft. Dazu wird der eindeutige Identifikationsschlüssel verwendet, durch den die Verbindung zwischen physischem Objekt und dessen Virtualisierung initiiert wird. Dieser Schlüssel muss der entsprechenden Instanz in der Ontologie hinzugefügt werden. Anschließend stehen alle semantischen Informationen in DURMAD zur Verfügung.

5.2 Schnittstellen

Um weitere Informationen von externen Datenquellen zu einzelnen Objekten zu erhalten, wurde in DURMAD die Möglichkeit zur Verbindung mit einer Datenbank (siehe Abschnitt 5.2.1) sowie mit digitalen Objektgedächtnissen (siehe Abschnitt 5.2.2) realisiert. Zudem umfasst DURMAD eine SOAP-Schnittstelle, mit Hilfe derer externe Programme auf die erfassten Informationen zugreifen und diese in ihrer eigenen Verarbeitung verwenden können. Beispielsweise ist es darüber möglich, alle Objekte auf einem Tisch abzufragen oder eine Liste von Produkten abzurufen, die sich in der Umgebung befinden und innerhalb einer gewissen Zeitspanne ihr Mindesthaltbarkeitsdatum erreicht haben werden. Abbildung 5.4 stellt die in DURMAD implementierten Schnittstellen dar.

5.2.1 Datenbankschnittstelle

Eine feste Zuordnung der Position von dynamisch veränderlichen Objekten im Grundrissmodell ist nicht sinnvoll, da diese bei jeder Veränderung angepasst werden müsste. Daher sollten solche Positionsdaten an einem separaten Speicherort, wie beispielsweise einer Datenbank, hinterlegt werden. Zudem können weitere dynamisch veränderbare Informationen zu den einzelnen Objekten in einer solchen Datenquelle hinterlegt werden.

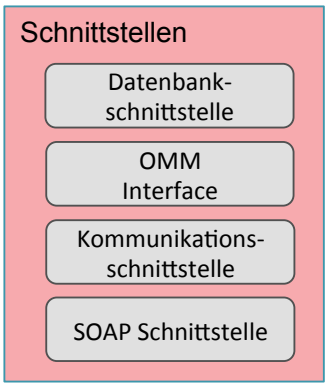


***Abbildung 5.4**: Schnittstellen des Dual Reality Management Dashboards*

Um dies zu ermöglichen, wurde ein möglichst generisches Datenbankschema entwickelt, welches in Abbildung 5.5 dargestellt ist. Als Grundkonstrukte sind Personen und Objekte in jeweiligen Tabellen der Datenbank spezifiziert, welche mittels einer eindeutigen ID identifiziert werden. Daneben hat die Personentabelle Felder, in denen Benutzername, ein spezifiziertes Passwort im MD5-Hash, sowie Vor- und Nachname hinterlegt sind. Neben der ID muss hierbei auch der Benutzername eindeutig sein. Personen können wiederum in Gruppen eingeteilt werden. Da ein Benutzer mehreren Gruppen zugeordnet sein kann und eine Gruppe aus mehreren Personen bestehen kann, liegt zwischen beiden Tabellen eine n:m-Beziehung vor. Um spätere Funktionen zu vereinfachen wird davon ausgegangen, dass jede Person in mindestens einer Gruppe ist. Diese hat den gleichen Namen wie die Person und umfasst ausschließlich die Person selbst als Mitglied. Analog zur Einteilung in Gruppen können Personen auch mehrere Rollen zugewiesen werden. Die jeweiligen Rollen werden dabei über eindeutige Bezeichnungen spezifiziert, zu denen Rechte hinterlegt werden können.

Für eine eindeutige Identifikation von Personen ist eine weitere Tabelle spezifiziert, die Identifikationsschlüssel von beispielsweise RFID- oder NFC-Tags umfasst. Somit können z.B. über die eindeutige Nummer einer Schlüsselkarte die zugehörige Person und damit auch deren Rechte ermittelt werden. Objekten können ebenfalls solche Identifikationsschlüssel zugewiesen werden. Dabei wird bei der Datenbank sichergestellt, dass ein Eintrag in der RFID-Tabelle entweder einer Person oder einem Objekt zugewiesen ist. Neben der Verbindung zu den Identifikationsmöglichkeiten umfasst die Objekt-Tabelle Daten über Namen und Typ des Objekts. Der Typ kann dabei als Filtermechanismus verwendet werden. Um zusätzlich zu den Grunddaten beliebige weitere Informationen zu einzelnen Personen oder Objekten hinterlegen zu können, gibt es eine allgemeine Informationstabelle. Diese umfasst alle Daten in Form von Schlüssel-Wert-Paaren. Jedes Objekt sowie jede Person kann beliebig viele Zusatzinformationen in dieser Tabelle speichern. Dies ermöglicht es, beispielsweise neue Daten zu einzelnen Objekten zu hinterlegen, die zur Konzeption der Datenbank nicht angedacht waren und daher im Datenbankschema nicht enthalten sind. Anstatt das Schema anzupassen, was womöglich einen negativen Einfluss auf andere Systeme hat, kann durch diese Herangehensweise eine agile und beliebig erweiterbare Datenbankstruktur geschaffen

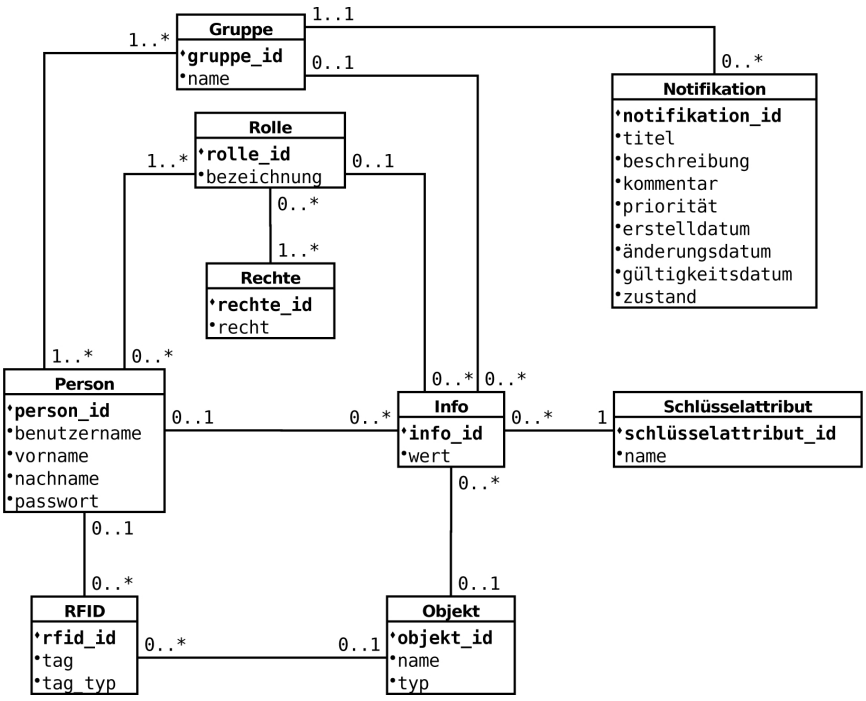


***Abbildung 5.5**: Ausschnitt aus dem Datenbankschema von* DURMAD

werden, die gerade bei sich ständig verändernden CPE einen Vorteil an Flexibilität bietet. Schließlich können Benachrichtigungen und Aufgaben mittels sogenannter Notifikationen in der Datenbank hinterlegt werden. Diese umfassen einen Titel, eine Beschreibung und einen Kommentar. Während Titel und Beschreibung nach der Erstellung festgelegt sind, kann der Kommentar verändert und ergänzt werden. Sowohl das Erstelldatum als auch die letzte Änderung werden als Zeitstempel in der Notifikation hinterlegt. Ebenso kann eine Gültigkeit angegeben werden, bis zu welchem Zeitpunkt die Information relevant ist. Die Wichtigkeit einer Notifikation kann mittels einer Priorität angegeben werden, welche Werte von „sehr hoch“, „hoch“, „normal“, „gering“ bis „sehr gering“ annehmen kann. Bei der Erstellung wird zudem eine Gruppe zugewiesen, für welche die Notifikation relevant ist. Schließlich kann eine Notifikation zwei Zustände einnehmen, einerseits kann sie noch offen und relevant sein und andererseits erledigt und abgeschlossen.

Das beschriebene Datenbankschema stellt eine Basisfunktionalität zur Verfügung, die jederzeit erweitert werden kann. Während die Objekttabelle für allgemeine Konstrukte verwendet werden kann, können spezifischere Daten in gesonderten Tabellen hinterlegt werden, welche mehrere Felder mit spezifischen Informationen umfassen. Durch eine Anbindung an die Info-Tabelle können ebenso beliebige Daten ergänzt werden, ohne das einmal spezifizierte Datenbankschema zu verletzen. Durch die hinterlegten Verknüpfungen können Daten auf unterschiedliche Weise abgerufen werden. Beispielsweise kann über den Namen (optional in Kombination mit dem zugehörigen Passwort) eine Person und damit ihre Gruppen, Rollen und Rechte ermittelt werden. Ebenso kann dies auch durch einen anderen

Identifikationsmechanismus erfolgen, wie z.B. im zuvor erwähnten Beispiel mittels einer Schlüsselkarte. Des Weiteren können Personen und Objekte auch anhand von Daten, welche in der Informationstabelle hinterlegt sind, ermittelt werden.

Um einerseits die Konsistenz der Datenbank zu gewährleisten und andererseits eine einfache Abfrage der Daten zu ermöglichen, wurde für das skizzierte Schema eine entsprechende Schnittstelle entwickelt. Dadurch können vordefinierte Funktionen über die Objekte aufgerufen werden, welche im Hintergrund in entsprechende SQL-Befehle umgewandelt werden. Gleichzeitig wurden diverse Caching-Mechanismen in diese Schnittstelle integriert, die es ermöglichen, Informationen lokal vorzuhalten, um nicht bei jeder Suche eine Datenbankanfrage zu schalten. Zum einen gibt es einen Modus, welcher bei jeder Anfrage auf die Datenbank zugreift, ohne die ermittelten Informationen anschließend zwischenzuspeichern. Dies ist sinnvoll, wenn die Daten sehr agil sind und sich zu jeder Zeit ändern können, wie beispielsweise bei Positionsangaben. Zum anderen gibt es eine Möglichkeit, dass alle Informationen, die man von der Datenbank abgefragt hat, anschließend zwischengespeichert werden. Wird diese Information erneut benötigt, wird sie aus dem Cache geladen, anstatt dass eine erneute Datenbankabfrage erfolgt. Dies erspart eine unnötige Belastung der Datenbank durch vielfältige Anfragen. Beispielsweise kann ein solcher Modus bei der Abfrage von Bildinformationen genutzt werden, welche sich in der Regel nicht verändern. Der Cache kann auch jederzeit gelöscht werden, so dass anschließend erneut die aktuellen Informationen aus der Datenbank geladen werden. Um dynamische Daten, welche sich nur in langen Zeiträumen verändern, zu laden, könnte der dauerhafte Caching-Modus verwendet und der Cache in regelmäßigen Abständen manuell geleert werden. Um dies jedoch zu vereinfachen, gibt es einen zeitgesteuerten Caching-Modus. Hierbei wird die Information nur in einem festlegbaren Zeitraum im Cache zwischengespeichert. Schließlich gibt es noch eine weitere Anfragevariante, welche es ermöglicht, Informationen in der Datenbank zu verändern. In den drei zuvor erwähnten Modi würde eine mögliche Informationsänderung nur lokal auf der jeweiligen Maschine erfolgen, ohne dass diese an die Datenbank zurückgespiegelt wird. Zur Änderung in der Datenbank müsste anschließend ein erneuter Befehl aufgerufen werden. Durch den letzten Modus werden die Informationen jedoch nicht nur lokal, sondern auch gleichzeitig in der Datenbank angepasst. Wird beispielsweise der Wert eines Info-Objekts verändert, so wird dies in diesem Modus automatisch auch in der Datenbank entsprechend angepasst.

5.2.2 OMM

Neben den Informationen aus einer zentralen Datenbank können auch objektspezische Daten aus digitalen Objektgedächtnissen (Object Memory Model, OMM) entnommen werden (siehe Kapitel 2.1.3). Hierzu wurde eine Schnittstelle von DURMAD zum OMM-Server implementiert, die sowohl einen lesenden als auch einen schreibenden Zugriff ermöglicht. Basierend auf dem jeweiligen Identifikationsschlüssel des Objekts kann eine Verbindung zu den im entsprechenden Objektgedächtnis hinterlegten Informationen erfolgen. Dieser Schlüssel kann entweder direkt am Produkt gespeichert sein oder über einen Zugriff auf die Datenbank ermittelt werden. In der Regel ist der Schlüssel in Form einer URL hinterlegt, so dass das

zugehörige Objektgedächtnis direkt abgerufen werden kann. Bei der Hinterlegung am Produkt kann eine solche URL beispielsweise im Speicher eines Funketiketts hinterlegt oder in einem zweidimensionalen Barcode kodiert werden. Für einen Datenbankzugriff reicht eine eindeutige Identifikation des Objekts aus, welche unabhängig vom jeweiligen Speicherort der Informationen ist. Ein Beispiel hierzu stellt die eindeutige ID eines RFID-Etiketts dar. In Verbindung mit dem zuvor beschriebenen Datenbankschema kann anhand dieser ID - sofern diese in der Datenbank hinterlegt ist - das zugehörige Objekt und damit auch die Verknüpfung zum Objektgedächtnis ermittelt werden. Sobald die Verbindung zum OMM aufgebaut und das Objektgedächtnis abgerufen wurde, stehen sowohl die Header-Informationen samt des Inhaltsverzeichnisses als auch die jeweiligen Informationsblöcke zur Verfügung und können anschließend bei der Verarbeitung berücksichtigt werden. Des Weiteren können über diese Schnittstelle neue Informationsblöcke erstellt und dem Objektgedächtnis hinzugefügt werden. Somit kann das digitale Tagebuch des Objekts (siehe Kapitel 2.1.3), in dem Objektinformationen gespeichert werden, mit den aus DURMAD ermittelten Daten weiter fortgeschrieben werden.

5.3 Kernmodule

Die Kernmodule umfassen die Teile des Dual Reality Management Dashboards, welche sich mit der Verarbeitung und Darstellung der ermittelten Informationen befassen (siehe Abbildung 5.6). DURMAD ist nach dem Model-View-Controller-Prinzip aufgebaut, so dass drei große Bereiche voneinander unterschieden werden. Das Modell umfasst dabei alle Informationen zum aktuellen Zustand der Szene (siehe Abschnitt 5.3.1), die einerseits für interne Berechnungen und andererseits für die Visualisierung bereitstehen. Die 3D-Darstellung erfolgt in einem Teil des Hauptfensters (siehe Abschnitt 5.3.2), welches in mehrere Teile untergliedert ist. Neben textuellen Darstellungen (siehe Abschnitt 5.3.3) umfasst die Visualisierung auch weitere grafische Darstellungen (siehe Abschnitt 5.3.4). Gerade die Präsentation von Informationen ist eine Hauptaufgabe des DURMAD, so dass auch mehrere Darstellungsmöglichkeiten implementiert wurden, um aufgabenspezifisch (siehe Kapitel 2.4) die geeignete Darstellungsform zu ermöglichen (siehe Abschnitt 5.3.5). Schließlich müssen erfasste Änderungen verarbeitet und das Modell entsprechend angepasst werden. Dies erfolgt durch den Controller, welcher gleichzeitig auch einfache Verarbeitungsmöglichkeiten für die erfassten Daten beinhaltet (siehe Abschnitt 5.3.6).

5.3.1 Modell

Das Grundrissmodell der CPE inklusive der darin verknüpften Objektmodelle bildet die Datengrundlage des Modells in DURMAD. Hierzu wird im ersten Schritt der Grundrissplan mit Hilfe des entsprechenden Parsers eingelesen und in eine interne Datenstruktur überführt. Anhand der Annotationen in diesem Modell werden im nächsten Schritt die zugehörigen 3D-Modelle geladen und entsprechend ihrer Position im Modell platziert. Neben den objektinternen Beziehungen schließt das Modell Schnittstellen zu externen Informationsquellen ein. So können optional unter Verwendung der Verbindung zum OMM oder einer Datenbank

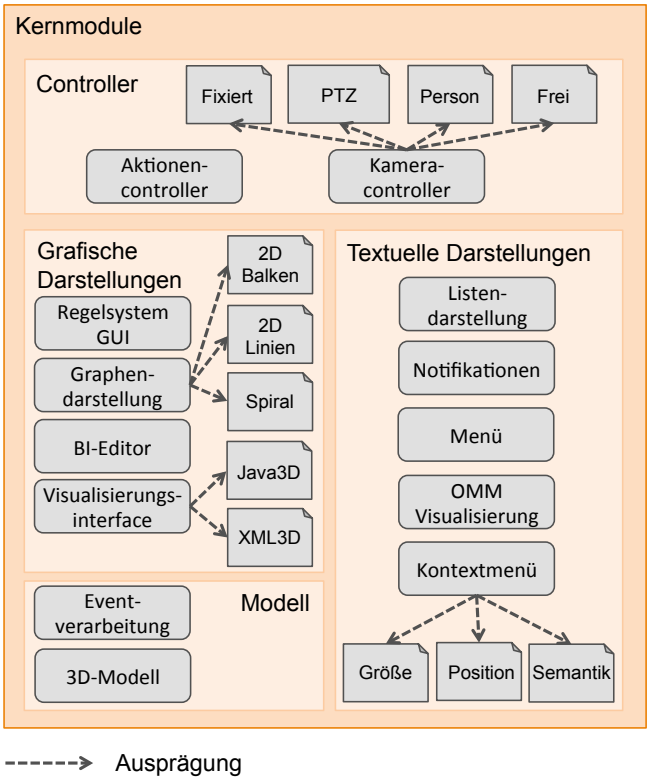


***Abbildung 5.6**: Kernmodule des Dual Reality Management Dashboards*

weitere Modelle hinzugefügt werden, die mobil und daher im Grundrissplan nicht enthalten sind. Mit Hilfe der Ontologie (siehe Abschnitt 5.1.3) werden zusätzliche Informationen zu den Objekten geladen und in die interne Datenstruktur aufgenommen. Dadurch entsteht eine Virtualisierung der gesamten CPE und das Modell umfasst alle Informationen der Objekte inklusive aller zugehörigen Informationen. Basierend auf diesem Modell können anschließend diverse Berechnungen erfolgen, wie beispielsweise die Ermittlung des Abstands zwischen zwei Objekten. Aufgrund der Implementierung in Java3D steht DURMAD ein großes Spektrum an vorgefertigten 3D-Strukturen und -Funktionen zur Verfügung. In der Regel erfolgen die Berechnungen jedoch nicht direkt im Modell, sondern in ausgelagerten Klassen. Für diese sind entsprechende Schnittstellen implementiert, so dass auf alle Objekte der virtualisierten CPE zugegriffen werden kann und diese zur Verarbeitung verwendet werden können.

Da DURMAD nach dem DR++-Prinzip entwickelt wurde, können sowohl Sensorinformationen aus der realen Umgebung als auch aus Simulationen das Modell verändern. Um potentielle Operationen zu vereinfachen, wurden einige Eigenschaften einer realen Umgebung im Modell nachgebildet. Beispielsweise können Objekte anstelle von absoluten Positionskoordinaten auch relative Positionen erhalten, indem sie in eine Gruppenstruktur des Referenzobjekts eingebunden werden. So können Objekte, welche sich auf einem Tisch be-

finden, mit diesem eine Gruppe bilden und Positionen auf der Tischplatte einnehmen. Diese Gruppierung erfolgt automatisch abhängig von den Sensorinformationen. Wird der Tisch verschoben, wird die entsprechende Gruppe mit transliert. Dabei wurde bei der Implementierung auf eine Trennung zwischen Translationsgruppen und Skalierungen geachtet. Bei einer Skalierung des Tischs aufgrund einer Simulation, werden die Objekte davon nicht beeinflusst und behalten ihre ursprüngliche Dimension. Basierend auf diesen Gruppierungen können zudem weitere Funktionen bereitgestellt werden, wie beispielsweise die Ermittlung der Anzahl an Objekten auf einem Tisch.

5.3.2 Darstellung des Hauptfensters

DURMAD bietet die Möglichkeit einer rollen- und rechtebasierten Darstellung und Verwendung des Hauptfensters, indem abhängig von den Benutzerrechten unterschiedliche Darstellungen und Informationen angezeigt werden können. Beispielsweise kann mittels dieser Rechte spezifiziert werden, ob der aktuelle Benutzer Einträge in der Datenbank verändern darf. Für eine Authentifizierung muss sich der Benutzer daher erstmals mittels seines Benutzernamens und Passworts anmelden. Nach einem Abgleich dieser Daten mit den Einträgen in der Datenbank werden die Rollen- und Rechteinformationen geladen und für die spätere Verwendung zwischengespeichert. Anschließend wird das Hauptfenster dargestellt, welches in drei Bereiche unterteilt ist (siehe Abbildung 5.7).

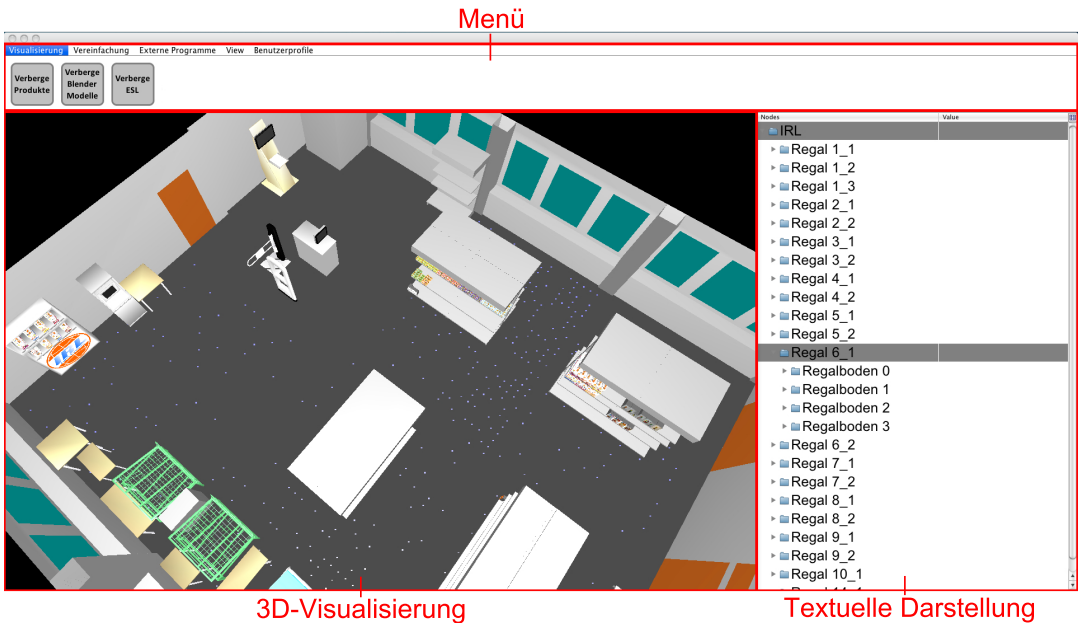


***Abbildung 5.7**: Grafische Oberfläche von* DURMAD *in der Java3D-Version*

Der erste Bereich stellt die Visualisierung einer CPE in einer 3D-Darstellung zur Verfügung. Neben der reinen Darstellung kann über diesen Bereich auch mit dem Modell interagiert werden, indem beispielsweise Objekte selektiert oder die virtuelle Kameraposition

verändert werden. Durch Interaktion mit Maus oder Tastatur kann hierbei die Kamera im Modell bewegt werden, wodurch ein virtueller Rundgang durch die virtualisierte CPE ermöglicht wird. Darüber hinaus können mehrere Kamerapositionen gespeichert werden. Diese können über ein Menü, welches den zweiten Bereich darstellt, ausgewählt werden. Zudem können über dieses Menü gewisse Objekttypen aus- und eingeblendet werden, um dem Benutzer eine übersichtliche Darstellung in der 3D-Visualisierung zu ermöglichen, wenn dieser auf spezielle Objekte und Ausprägungen achten möchte. Ebenfalls bietet dieses Menü den Einstieg für weitere Funktionalitäten, welche DURMAD zur Verfügung stellt. Darunter fällt beispielsweise das Öffnen weiterer grafischer Oberflächen oder das Starten von externen Applikationen. Schließlich gibt es eine weitere Darstellungskomponente, die je nach aktueller Anwendung variieren kann. Im Beispiel der Abbildung 5.7 enthält dieser Bereich eine textuelle Darstellung, welche Informationen aus einer zugehörigen Datenbank darstellt.

5.3.3 Textuelle Darstellungen

Neben der Textdarstellung, die Informationen aus der Datenbank präsentiert, kann in diesem Bereich der Benutzeroberfläche auch eine textuelle Listendarstellung von Objektgedächtnissen erfolgen. Mittels des Identifikationsschlüssels eines Objekts werden über die Schnittstelle zum OMM alle zugehörigen Informationen abgerufen und dargestellt. Die Auswahl des Objekts und damit des entsprechenden Identifikationsschlüssels kann beispielsweise durch eine Selektion in der 3D-Darstellung erfolgen. Des Weiteren wird durch Rechtsklick auf ein Objekt in der Visualisierung neben der aktuellen Cursorposition ein zugehöriges Kontextmenü als Überblendung angezeigt, welches abhängig vom jeweiligen Objekt und dessen Eigenschaften automatisch generiert wird. Beispielsweise können über dieses Kontextmenü gewisse Objekte hinsichtlich ihrer Position, Orientierung oder Dimension verändert werden. Weitere Funktionen können der dem Objekt zugehörigen Semantik entnommen werden, wie z.B. ein Eintrag zum Öffnen oder Schließen eines Tors.

Eine weitere Möglichkeit, textuelle Informationen darzustellen, erfolgt in Form von Notifikationen. Diese können in der Datenbank hinterlegt und für gewisse Gruppen zugänglich gemacht werden. Hierzu werden zwei Möglichkeiten angeboten, welche sich hinsichtlich Art und Ort der Darstellung unterscheiden. Zum einen können Notifikationen über das Kontextmenü angezeigt und abgerufen werden und zum anderen können sie in der 3D-Visualisierung dargestellt werden. Im ersten Fall wird bereits im Menü angezeigt, wie viele offene Notifikationen für den aktuell registrierten Benutzer vorliegen. In DURMAD werden Notifikationen in Aufgaben und Benachrichtigungen (Report) unterteilt, die separat angezeigt werden können (siehe Abbildung 5.8). Durch einen Klick auf den entsprechenden Button werden die Aufgaben mit den wichtigsten Informationen, wie Titel und Beschreibung, aufgelistet. Werden detailliertere Informationen, wie beispielsweise das Erstelldatum, benötigt, so kann durch einen Klick auf den Titel ein weiteres Fenster geöffnet werden. Dieses umfasst alle zur Notifikation gehörenden Daten und ermöglicht es, den hinterlegten Kommentar anzupassen und die Änderungen in die Datenbank zu schreiben. Damit wird die Möglichkeit eines einfachen Ticket-Systems geschaffen, über welches eine Kommunikation und Aufgabenverteilung zwischen Personen und Gruppen erfolgen kann.

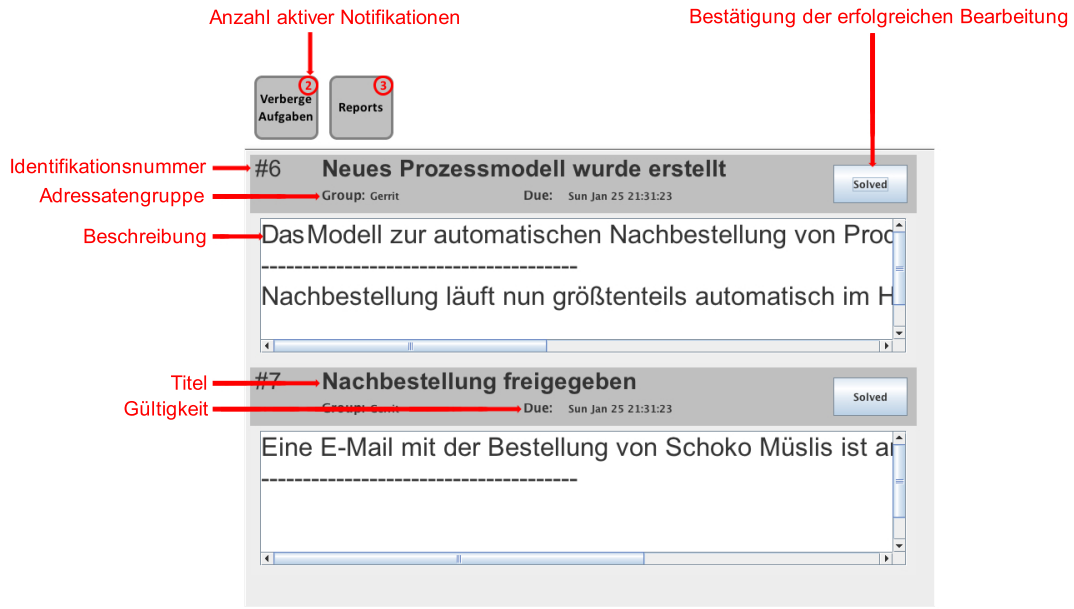


***Abbildung 5.8**: Visualisierung von Notifikationen mittels des Kontextmenüs von* DURMAD

Für die Darstellung von Informationen innerhalb der 3D-Visualisierung können diese den jeweiligen Objekten räumlich zugeordnet werden. Das Vorhandensein solcher Informationen wird dann durch eine Box symbolisiert, welche sich über dem jeweiligen Objekt in der 3D-Szene befindet, wobei durch die räumliche Korrelation die Zugehörigkeit leicht erschlossen werden kann (siehe Abbildung 5.9). Zudem ist die Box aufgrund ihrer Integration in die Szene von verschiedenen Perspektiven aus sichtbar. Klickt man auf diese Box, öffnet sich ein weiteres Fenster, welches die zugehörigen Informationen enthält. Diese können sowohl textuell als auch in Form von Bildern vorliegen. Neben der Darstellung des OMM im Kontextmenü stellt die Integration dieser Informationen in die 3D-Visualisierung eine zweite Variante der Informationsdarbietung für digitale Objektgedächtnisse dar.

5.3.4 Grafische Darstellungen

Eine grafische Darstellung in DURMAD wurde bereits am Beispiel des Hauptfensters präsentiert, welches die 3D-Visualisierung der virtualisierten CPE in Java3D zeigt. Aufgrund des Model-View-Controller-Ansatzes ist es jedoch möglich, bei Bedarf weitere Visualisierungskomponenten für die Darstellung der 3D-Szene zu verwenden (siehe Kapitel 2.4). Dabei werden alle Befehle, die die Visualisierung betreffen, an alle Visualisierungskomponenten übermittelt, welche diese entsprechend verarbeiten müssen und hierdurch synchron gehalten werden können. Als eine weitere Visualisierungskomponente wurde exemplarisch XML3D in DURMAD integriert. Abbildung 5.10 zeigt die Visualisierung derselben 3D-Umgebung in Java3D (a) und in XML3D (b). Durch die Unterstützung von XML3D wird es ermöglicht, dass aufgrund des browserbasierten Ansatzes die 3D-Visualisierung auf

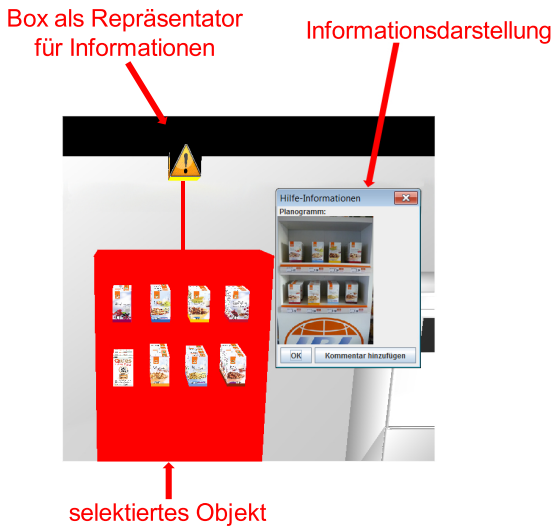


***Abbildung 5.9**: Objektbezogene Informationsdarstellung im 3D-Raum*

entfernten Rechnern erfolgen kann. Des Weiteren können so auch mobile Endgeräte zur Darstellung verwendet werden. Werden Veränderungen in der CPE detektiert, werden diese an DURMAD gesendet, woraufhin das Modell angepasst und die Visualisierungskomponenten benachrichtigt werden. Die Kommunikation zwischen dem 3D-Modell von DURMAD und der XML3D-Visualisierung erfolgt mittels JSON. Über spezielle Befehle können beispielsweise Objekte an einer bestimmten Position mit einer vorgegebenen Dimension der Szene hinzugefügt werden. Zudem können Kameraposition und -orientierung über diese Schnittstelle übermittelt werden, so dass die Darstellung der Bildausschnitte synchron gehalten werden kann. Da DURMAD ebenfalls XML3D-Repräsentationen für Modell und Darstellung unterstützt, kann hierbei bei beiden Visualisierungskomponenten auf die gleiche Datenbasis (Einzelmodelle) zurückgegriffen werden, was eine fehlerhafte Darstellung aufgrund unterschiedlicher Modelle vermeidet. Neben der reinen Visualisierung kann mit der XML3D-Darstellung auch interagiert werden. Informationen über die Selektion von Objekten oder die Veränderung der Kamera werden über dieselbe JSON-Schnittstelle an das Modell kommuniziert.

Auch für abstrakte Informationen bietet DURMAD eine grafische Darstellungsform in Form einer Graphendarstellung. Hierzu werden Darstellungsoptionen in Form von 2D-Linien, 2D-Balken, Kreisdiagrammen und Spiralgraphen bereitgestellt. Während Linien-, Balkendarstellung und Kreisdiagramme bereits aus Tabellenkalkulationsprogrammen bekannt sind, sind Spiralgraphen eher unüblich. Diese weisen jedoch den Vorteil auf, dass periodische Ereignisse durch diese Visualisierung schnell ersichtlich sind. Dazu besteht die Möglichkeit, die Zeitintervalle gemäß des zu betrachtenden Turnus anzupassen. Dazu ist eine stunden-, tage-, wochen- und jahresweise Darstellung möglich. Der entsprechende Zeitverlauf wird dabei als eine gesamte Runde dargestellt, wobei mehrere Runden in Form einer Spirale visualisiert werden. Eine Runde ist wiederum in diskrete Abschnitte unterteilt. Die

(a) Darstellung in Java3D (b) Darstellung in XML3D

***Abbildung 5.10**: Darstellung derselben Szene in zwei unterschiedlichen Visualisierungskomponenten*

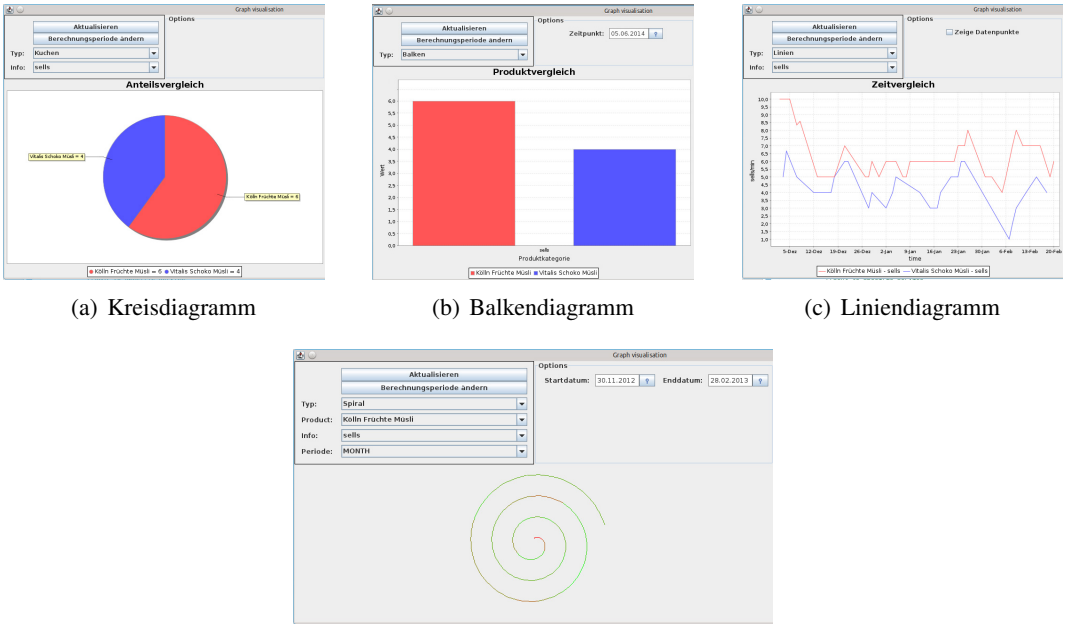
Anzahl der resultierenden Segmente ist hierbei abhängig vom jeweiligen Intervall. Beispielsweise ist die wochenweise Darstellung in sieben Segmente unterteilt, welche jeweils einen Tag repräsentieren. Entsprechend eines Farbschemas, welches sich an definierte Schwellenwerte hinsichtlich eines speziellen Parameters (z.B. Temperatur, Abverkaufszahlen, etc.) richtet, werden diese Segmente eingefärbt. Abbildung 5.11 zeigt Beispiele der verschiedenen Graphendarstellungen am Beispiel von Abverkaufszahlen für ein spezifisches Produkt. Bei der Linien- und Balkendarstellung können zusätzlich auch Informationen mehrerer Objekte angezeigt und damit miteinander verglichen werden, wobei die Graphen mittels einer Farbkodierung den jeweiligen Informationen zugeordnet werden können. Dies ermöglicht einen Vergleich eines spezifischen Parameters bei mehreren Objekten.

5.3.5 Informationsrepräsentation im 3D-Modell

In den vorangegangenen Abschnitten wurde die Darstellung von Informationen in textueller und in graphenbasierter Form beschrieben. Zusätzlich wurde in DURMAD die Möglichkeit geschaffen, eine Darstellung von quantifizierbaren Parametern im 3D-Modell zu verwenden. Beispielsweise kann dadurch der Stromverbrauch von Geräten mittels einer optischen Hervorhebung dargestellt werden. Während ein niedriger Stromverbrauch durch eine grüne Überlagerung repräsentiert werden kann, könnte diese immer weiter in ein Rot übergehen, je höher der Stromverbrauch des Objekts ist. Eine spezielle Form von Parametern stellt die Datenqualität dar. Um eine möglichst genaue Informationswiedergabe zu ermöglichen, ist es wichtig, dass bei der Darstellung auch die unterschiedliche Qualität der Daten berücksichtigt wird und diese dem Benutzer entsprechend dargestellt werden kann. Grund für die variierende Datenqualität sind folgende Faktoren [Gershon, 1999]:

↪ **Fehlerhafte Daten:** Diese können beispielsweise durch Messfehler oder defekte Sensoren auftreten.

↪ **Unvollständige Daten:** Dies ist der Fall, wenn während der Kommunikation Daten verloren gehen oder aber auch, wenn wichtige Daten nicht kommuniziert werden.



(a) Kreisdiagramm (b) Balkendiagramm (c) Liniendiagramm

(d) Spiralgraph

***Abbildung 5.11**: Graphbasierte Darstellung von Abverkaufszahlen*

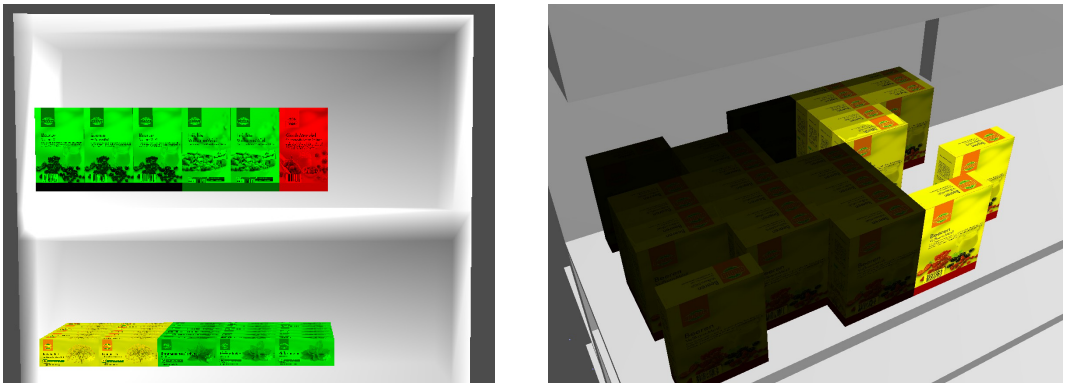
↪ **Zu komplexe Daten:** Dies bedeutet, dass die Daten für das menschliche Verständnis zu komplex sind, so dass ihre Bedeutung nicht innerhalb einer angemessenen Zeitspanne erfasst werden kann.

↪ **Inkonsistenz:** Dies tritt auf, wenn mehrere Quellen Daten liefern, die sich gegenseitig widersprechen.

↪ **Unsicherheiten:** Oftmals sind Messwerte auch mit gewissen Unsicherheiten behaftet. Diese können beispielsweise auftreten, wenn zwischen der Messung und dem Verarbeiten der Informationen eine gewisse Zeitspanne verstrichen ist.

Die Datenqualität kann hierbei in Form von Konfidenzwerten ausgedrückt werden, welche zwischen Eins und Null liegen. Null bedeutet in diesem Fall, dass keine Konfidenz vorliegt und somit die Qualität der Daten nicht bestimmt werden kann, was eine sehr hohe Unsicherheit widerspiegelt. Eins hingegen besagt, dass eine sehr hohe Datenqualität vorliegt und die Daten daher zu 100% verlässlich sind. Zwischen den beiden Werten können alle anderen Zwischenwerte ebenfalls angenommen werden, wodurch eine mess- und darstellbare Art der Unsicherheit erreicht wird. Den gleichen Wertebereich kann man bei quantifizierbaren Parametern durch eine Normalisierung erhalten. Der Konfidenzwert kann entweder als Ziffer dargestellt oder bei der Visualisierung entsprechend berücksichtigt werden. In diesem Zusammenhang wurden in DURMAD mehrere Visualisierungstechniken implementiert, die eine Darstellung von Objekteigenschaften bzw. der Datenqualität ermöglichen. Die implementierten Methoden umfassen farbliche Überlagerungen, Transparenz und Variation der Objektgröße.

Farbliche Überlagerung Eine Art der Repräsentation von Unsicherheiten oder normierten Objekteigenschaften stellt die farbliche Überblendung von Objekten dar. Eine solche Überblendung hat den Vorteil, dass das Aussehen des jeweiligen Objekts noch erkennbar ist, im Gegensatz zu einer Darstellung, bei der die gesamte Textur durch eine Farbe ersetzt wird. Für die farbliche Kodierung werden zwei Funktionen angeboten. Zum einen besteht die Möglichkeit, anhand einer diskreten Einteilung mit Hilfe von Schwellenwerten die entsprechenden Objekte mit unterschiedlichen Farbwerten zu überblenden, wie in Abbildung 5.12(a) dargestellt. Dies bedeutet, dass alle Objekte, deren Konfidenzwert in einem bestimmten Bereich liegt, mit der zugehörigen Farbe überblendet werden. Neben der diskreten Farbdarstellung kann auch ein Farbverlauf zwischen zwei Farben als Darstellung der Konfidenz dienen. In Abbildung 5.12(b) bedeutet eine helle, gelbe Darstellung einen hohen Konfidenzwert, wohingegen eine dunkle, schwarze Überblendung einen geringen Konfidenzwert darstellt.

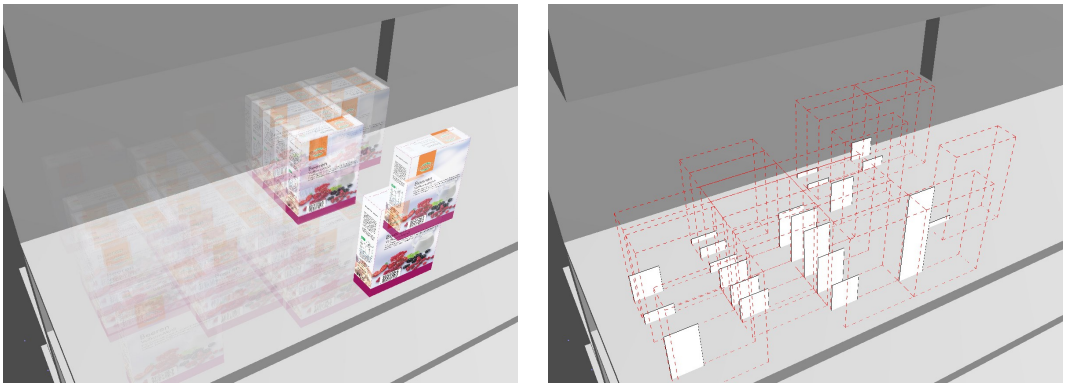


(a) Diskrete Darstellung über Farbwerte (b) Kontinuierliche Darstellung über Farbhelligkeit

***Abbildung 5.12**: Darstellung von Konfidenzwerten durch farbliche Überlagerung*

Transparenz Alternativ zu farblichen Überlagerungen der Texturen können Konfidenzwerte auch mittels Transparenz dargestellt werden. Hierbei besagt ein klar sichtbares Objekt, dass dessen Konfidenzwert hoch ist, wohingegen ein stark transparentes Objekt einen geringen Konfidenzwert widerspiegelt (siehe Abbildung 5.13(a)). Anstelle von transparenten Objekten können auch die Umrisse dieser mit einer Art „Barometer“ in Form einer Säule dargestellt werden, welche den Konfidenzwert repräsentiert. Hierbei wird der zu betrachtende Parameter mittels der Säule repräsentiert und die Transparenz der Produkte verwendet, damit diese Balken gesehen werden können. Je höher die Säule ist, desto höher ist der entsprechende Konfidenzwert, wie in Abbildung 5.13(b) dargestellt. Die Umrandungen helfen hierbei, die Objektdimensionen im Blick zu haben, wobei Balken sehr schnell von Personen erfasst und interpretiert werden.

Größenvariation Schließlich können die Konfidenzwerte auch in die Darstellung der Objektgeometrie mit einfließen. Je geringer der Konfidenzwert ist, desto kleiner wird das entsprechende Objekt dargestellt. Der höchste Konfidenzwert wird hierbei durch die realen Objektdimensionen repräsentiert. Jedoch wird für eine richtige Interpretation entweder ein



(a) Transparente Objektvisualisierung (b) Darstellung anhand von Objektumrandung

***Abbildung 5.13**: Transparente Darstellung von Konfidenzwerten*

Referenzobjekt, welches mit Originaldimensionen dargestellt wird oder das Wissen um die Originalgröße benötigt. Diese Darstellung hat jedoch den Vorteil, dass die einzelnen Objekte leicht identifiziert werden können, da deren Texturen klar ersichtlich sind. Ein Beispiel hierfür ist in Abbildung 5.14 dargestellt.

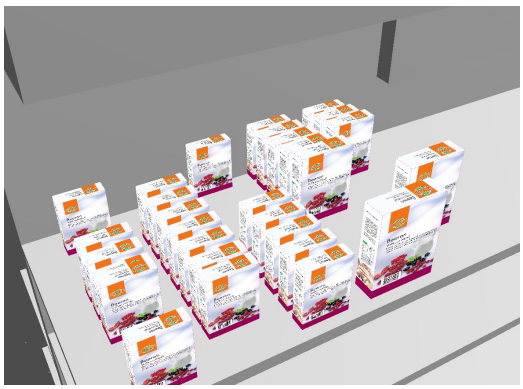


***Abbildung 5.14**: Darstellung der Konfidenz durch Variation der Objektdimension*

Die unterschiedlichen Visualisierungstechniken können auch miteinander kombiniert werden. Dabei können sich die unterschiedlichen Darstellungsarten auf unterschiedliche Parameter beziehen, wie beispielsweise die Größe auf den Energieverbrauch und die farbliche Überblendung auf einen Kostenfaktor. Des Weiteren besteht die Möglichkeit, die Visualisierungstechniken neben der statischen Darstellung auch als dynamische Variante zu integrieren. Dies bedeutet, dass die Darstellungen des Objekts zwischen der regulären Darstellung und der Darstellung des Konfidenzwerts gemäß der jeweiligen Visualisierungstechnik pulsieren. Durch diese stetige Veränderung kann die Aufmerksamkeit des Benutzers auf die entsprechenden Objekte mit geringen Konfidenzwerten gelenkt werden. Als weitere dynamische Darstellungsmethoden wurden zwei Möglichkeiten der Bewegung

von Objekten implementiert. Bei der ersten Variante wird der Konfidenzwert in Form der Geschwindigkeit repräsentiert, mit der sich die Objekte um ihre eigentliche Position bewegen. Je geringer der Konfidenzwert ist, desto schneller die Bewegung. Bei der zweiten Variante ist die Amplitude der Objektschwingung abhängig vom Konfidenzwert. Hierbei beschreibt eine große Amplitude einen geringen Konfidenzwert. Zudem können über die dynamische Darstellung Schwachstellen der statischen Darstellungen erschlossen werden. Beispielsweise ist bei einer transparenten Darstellung mit dem Konfidenzwert 0 kein Objekt sichtbar. In diesem Fall kann der Benutzer nicht erkennen, ob an der freien Stelle kein Objekt oder ein Objekt mit sehr geringer Konfidenz steht. Diese Problematik wird durch das Pulsieren vermieden.

5.3.6 Steuerung & Verarbeitung in DURMAD

Zusätzlich zur Darstellung eines virtuellen Modells dienen die Visualisierungskomponenten der Interaktion mit diesem. Beispiele für Interaktionen, welche in vielen Applikationen verwendet werden, sind die Steuerung der virtuellen Kamera oder die Selektion und Manipulation gewisser Objekte im Modell. Hierzu können entweder Tastatur, Maus oder spezielle Joysticks verwendet werden. Um vordefinierte Kamerapositionen einstellen zu können, wurde in DURMAD die Möglichkeit geschaffen, die aktuelle Kameraposition und -ausrichtung über das Menü zu speichern. Neben der rein statischen Kameraposition können auch gesamte Kamerapfade aufgezeichnet und zum späteren Abspielen gespeichert werden. Jede zu speichernde Kameraposition bzw. jeder Pfad wird unter einem individuellen Namen in einer entsprechenden XML-Datei hinterlegt. Diese wird zum Programmstart ausgelesen, wobei jeder Eintrag als separater Button im Menü (siehe Abschnitt 5.3.2) repräsentiert wird. Um unterschiedliche Blickwinkel und Beweglichkeiten von Kameras zu unterstützen, wurden insgesamt vier Kameratypen implementiert, welche sich durch eine Spezifikation gewisser Parameter, wie z.B. ob die Kamera rotierbar oder frei beweglich ist, von einem generischen Kameratyp ableiten. Der erste Typ ist eine fixe Kamera, welche keine Veränderung ermöglicht. Diese kann beispielsweise zur Repräsentation von fest installierten Kameras oder zur statischen Darstellung gewisser Abschnitte verwendet werden. Der zweite Kameratyp ermöglicht eine Veränderung der Ausrichtung, jedoch nicht der Position. Dies kann unter anderem für die Repräsentation von fest installierten steuerbaren Kameras eingesetzt werden. Die dritte Variante ermöglicht eine Translation in x- und y-Richtung sowie eine Veränderung der Ausrichtung. Die Höhe über dem Boden kann initial angegeben werden und ist anschließend fix. Dieser Kameramodus entspricht der Darstellung aus Sicht einer Person. In diesem Modus wurde zudem integriert, dass Objekte, wie z.B. Wände oder Tische, als Hindernisse gelten und daher nicht durchquert werden können. Der letzte Kameratyp ist eine freie Kamera, welche alle Freiheitsgrade der Bewegung ermöglicht und keine Einschränkungen hinsichtlich der Durchquerung von Hindernissen hat. Bei neu zu erstellenden Kameraobjekten muss einer dieser vier Kameratypen ausgewählt werden.

Neben der Steuerung von Kameras können weitere Interaktionen mit dem Modell erfolgen, worauf entsprechende Reaktionen ausgelöst werden können. So kann die Selektion eines Objekts die Darstellung weiterer Informationen oder eine Möglichkeit, dieses in

gewisser Art zu modifizieren, nach sich ziehen, z.B. dessen Position zu verändern. Die Selektion eines Objekts muss dabei an alle Visualisierungskomponenten weitergeleitet werden, damit diese darauf reagieren können. Zudem kann die Selektion auch für Textdarstellungen und grafische Darstellungen relevant sein. Beispielsweise kann das zugehörige Objektgedächtnis des selektierten Objekts dargestellt werden. Die Verteilung dieser Informationen erfolgt durch den Controller. Neben der Interaktion des Benutzers können auch externe Einflussfaktoren eine Veränderung des Modells verursachen. So müssen Veränderungen, die in der realen Umgebung detektiert werden, unmittelbar im Modell nachgebildet werden, um dem DR-Ansatz gerecht zu werden. Hierzu werden die eintreffenden Informationen vom Controller verarbeitet und als Ausführungsbefehle an die Visualisierungskomponenten gesendet.

Somit kommt es zu einer wechselseitigen Abhängigkeit zwischen dem Controller, welcher für seine Verarbeitung auch auf Informationen aus dem Modell angewiesen sein kann und dem Modell samt aller zugehörigen Visualisierungskomponenten. Aufgrund dieser komplexen Interaktion zwischen den Kernmodulen kann es bei einem gegenseitigen Funktionsaufruf zwischen mehreren Modulen zu einem Deadlock kommen. Dies bedeutet, dass ein Ringschluss gebildet wird, bei dem jedes Modul auf die Antwort des nächsten wartet, um mit diesen Informationen selbst eine Antwort zurückzugeben. Um eine solche Problematik zu vermeiden, wurde bei DURMAD ein synchroner Funktionsaufruf explizit ausgeschlossen. Stattdessen werden die gewünschten Funktionen über eine zentrale Komponente sequenziell aufgerufen. Dieser Ansatz basiert auf der Methode des Remote Process Call (RPC, siehe Kapitel 2.3), welche bei der asynchronen Kommunikation in verteilten Systemen bereits erfolgreich eingesetzt wird. Diese Vorgehensweise wurde mit der Reflexion-API von Java kombiniert, um die entsprechenden Funktionen inklusive der Parameter, welche von den Modulklassen zur Verfügung gestellt werden, zu ermitteln und durch die zentrale Komponente auszuführen [Forman, 2004]. Dabei melden sich die entsprechenden Modulklassen bei einer zentralen Struktur an. Anschließend können andere Klassen, die ebenfalls an dieser Struktur angemeldet sind, deren Funktionen aufrufen. Der Klassenname dient hierbei als Identifikationsschlüssel, welcher in Verbindung mit dem Funktionsnamen, der als zweiter Parameter essentiell wichtig ist, die zugehörige Funktion spezifiziert. Im Anschluss an diese beiden Parameter müssen die Funktionsparameter übergeben werden. Gibt es keine Struktur, die dieser Konstellation von Funktionsname und Parametertypen in der Klasse entspricht, wird eine entsprechende Fehlermeldung ausgegeben. Der Funktionsaufruf mittels Java Reflexion erfolgt über die zentrale Struktur, so dass die aufrufende Klasse normal weiterarbeiten kann. Durch die Verwendung dieser Kommunikationsstruktur wird die Problematik eines Deadlocks umgangen.

5.4 Verarbeitungsautomatismen

Neben der reinen Visualisierung und Interaktionsunterstützung ist DURMAD darauf ausgerichtet, automatisierte Funkionalitäten anzubieten. Diese können auf Informationen zurückgreifen, welche aus dem System ermittelt werden, und diese verarbeiten. Exemplarisch wurden in diesem Kontext ein Regelsystem (siehe Abschnitt 5.4.1) und ein Agentensystem

(siehe Abschnitt 5.4.2) integriert. Des Weiteren wurde eine Möglichkeit für den Benutzer geschaffen, Business Intelligence (BI) Systeme zu erstellen, welche anfallende Informationen direkt verarbeiten und beispielsweise zur Visualisierung an die Graphendarstellung senden können (siehe Abschnitt 5.4.3). Abbildung 5.15 zeigt die exemplarisch implementierten Verarbeitungsautomatismen auf.

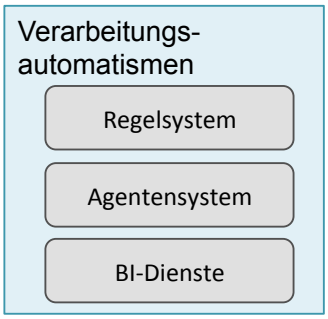


***Abbildung 5.15**: Verarbeitungsautomatismen des Dual Reality Management Dashboards*

5.4.1 Regelsystem

Um einen automatisierten Prozess als Reaktion auf Veränderungen ausführen zu lassen, werden oftmals Regelsysteme eingesetzt. Die zugrundeliegenden Regeln werden durch einen eindeutigen Namen spezifiziert und sind in Prämissen und Konsequenzen unterteilt. Basierend auf einer Wissensbasis wird bei jeder Veränderung für jede Regel geprüft, ob alle Prämissen erfüllt sind. Im positiven Fall wird die Regel gefeuert, was bedeutet, dass die zugehörige Regelkonsequenz ausgeführt wird. In DURMAD wurde Drools der Firma JBoss[3] als Regelsystem integriert. Drools stellt dabei ein Produktionssystem dar, welches nach einem RETE-Algorithmus eine effiziente Verarbeitung der Regeln ermöglicht [Forgy, 1982]. Für die Beschreibung der Regeln gibt es eine eigene Syntax, die Drools Rule Language (DRL). Ein Beispiel einer solchen Regel ist im Anhang in Programmauszug A.6 aufgeführt.

Aufgrund der speziellen Syntax solcher Regeln, welche in gewissem Maße Programmierkenntnisse voraussetzen, können diese im Allgemeinen nur durch Experten erstellt und angepasst werden. Um dies zu vereinfachen, so dass auch andere Personengruppen selbstdefinierte Regeln erstellen können, wurde das Konzept der domänenspezifischen Sprachen (DSL) eingeführt. Hierbei erfolgt im ersten Schritt eine Übersetzung der Regelkonstrukte in natürlichsprachliche Texte durch einen Experten. Dabei wird zwischen den Vorbedingungen (Condition), welche die Prämissen darstellen und den Konsequenzen (Consequence) unterschieden. Nachdem eine Menge an Prämissen und Konsequenzen definiert wurde, können diese in einer DSL-Datei hinterlegt werden. Ein Endanwender kann daraus eigene Regeln zusammenstellen, indem er aus den vorgefertigten Konstrukten eine Vorbedingung und eine Konsequenz zusammensetzt. Diese Konstrukte können auch Variablen enthalten, welche

[3] http://drools.jboss.org

bei der Erstellung der resultierenden DRL-Regeln entsprechend ergänzt werden müssen. Programmauszug A.7 im Anhang zeigt ein Beispiel für eine solche DSL-Datei.

Für die Erstellung und Verwendung von Regeln und DSL-Dateien wurde in DuRMaD ein rollenbasierter Zugriffsmechanismus realisiert, welcher die jeweilige Qualifikation des Anwenders berücksichtigt. Hierbei wird zwischen drei Benutzergruppen unterschieden. Die erste Gruppe stellen Experten für die Erstellung und Anpassung der DSL-Dateien dar. Sie haben Zugriff auf diese Dateien und können sie modifizieren. Damit erstellen sie auch gleichzeitig die Regelbausteine, die später von den Endanwendern verwendet werden können. Ansonsten haben sie keine weiteren Rechte. Die Manager der Umgebung können die DSL-Dateien einsehen, aber nicht verändern. Des Weiteren können Sie neue Regeln erstellen, alle bestehenden Regeln anpassen oder löschen sowie diese aktiv oder inaktiv schalten. Normale Arbeiter können, analog zu den Managern, neue Regeln erstellen. Jedoch haben sie keinen Zugriff auf DSL-Daten und können nur die Regeln verändern, die sie zuvor selbst erstellt haben. Für die Erstellung und Modifikation von Regeln wurde eine grafische Oberfläche entwickelt und in DuRMaD integriert. Diese ist in acht Abschnitte unterteilt, wie in Abbildung 5.16 dargestellt. Auf der linken Seite besteht ein Zugriff auf die Datenbank, so dass Objekte, welche dort hinterlegt wurden, bei der Regelerstellung verwendet werden können. Auf der rechten Seite besteht die Möglichkeit, eigene Labels anzugeben, welche man häufig verwendet und darüber schnell finden kann. Des Weiteren können die bereits erstellten Regeln hier eingesehen und zur Überarbeitung selektiert werden. In der Mitte befinden sich die eigentlichen Editierblöcke, in welchen man den Regelnamen, die Prämisse und die Konsequenz spezifiziert. Die in der zugehörigen DSL-Datei hinterlegten Einträge für Vorbedingungen werden als Prämissenbausteine unten links angezeigt. Analog dazu finden sich die Konsequenzbausteine unten rechts. Zur Erstellung einer Regel kann einfach einer dieser Bausteine in das Feld für die Prämissen bzw. Konsequenzen gezogen werden. Hierbei muss der Benutzer je genau eine der in der DSL-Datei vorgegebenen (komplexen) Prämissen und Konsequenzen wählen und diese bearbeiten. In den Texten vorkommende Variablen werden erkannt und können mittels Drag and Drop durch entsprechende Objekte aus der Datenbank oder aus den eigenen Labels ersetzt werden. Alternativ können sie auch mittels manueller Eingabe editiert werden. Um bei potentiell auftretenden Regelkonflikten, die entstehen, wenn mehrere Regeln feuern können, eine Präferenz der Feuerreihenfolge festzulegen, kann der „Salience“-Parameter eingestellt werden. Ein höherer Wert bedeutet, dass diese Regel eine höhere Feuerpräferenz hat. Schließlich können die Regeln auf dem Server gespeichert und somit automatisch dem Regelsystem hinzugefügt werden. Neben der Erstellung und Modifikation von Regeln kann über diese grafische Oberfläche eingesehen werden, wie oft welche Regel gefeuert hat und wann dies das letzte Mal der Fall war. Dies erfolgt, indem über die vorgegebenen Reiter auf die unterschiedlichen Seiten navigiert werden kann. Schließlich können Regeln aktiviert bzw. deaktiviert werden.

5.4.2 Agentensystem

Neben Regelsystemen bieten Agentensysteme die Möglichkeit einer automatisierten Reaktion auf Veränderungen (siehe Kapitel 2.1.5). In DuRMaD wurde ein Agentensystem integriert, welches auf Jadex basiert [Braubach et al., 2005]. Hierbei wird den einzelnen Objek-

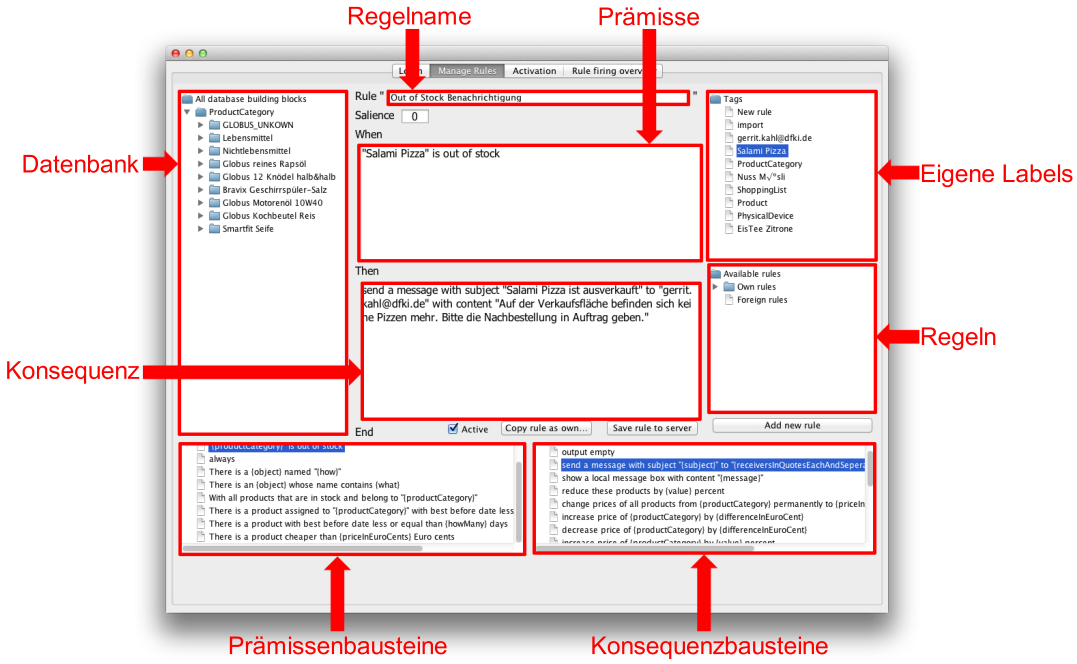
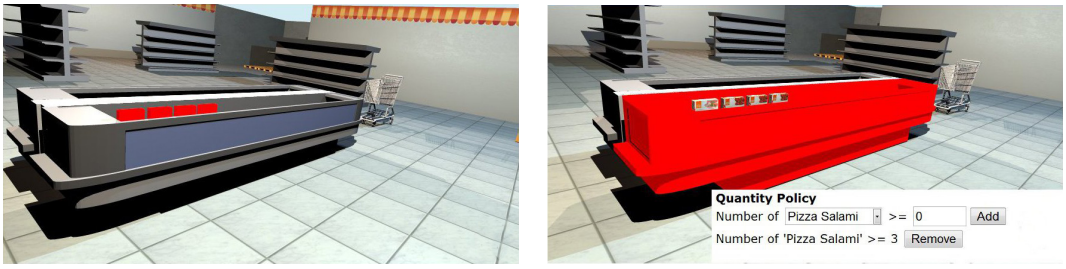


***Abbildung 5.16**: Grafische Oberfläche zur Erstellung und Anpassung von Regeln*

ten der Umgebung jeweils ein Agent zugeordnet, der dessen Funktionsweise integriert. Beispielsweise werden Temperatursensoren durch die Fähigkeit, die Umgebungstemperatur zu ermitteln, aufgefasst. Der Agent kommuniziert hierbei die Temperaturveränderung, so dass weitere Agenten diese Information verarbeiten können [Kahl et al., 2011a]. Veränderungen in der Umgebung werden basierend auf dem DR-Ansatz an DURMAD kommuniziert, welches diese Informationen daraufhin an das Agentensystem weiterleitet. Insbesondere können auch räumliche Faktoren erfasst werden. Zudem können über die Schnittstelle zum OMM Informationen aus den zugehörigen Objektgedächtnissen dem Agentensystem zur Verfügung gestellt werden. Durch eine Kommunikation zwischen den Agenten der einzelnen Objekte kann anschließend ein entsprechender Automatismus angestoßen werden. Beispielsweise kann ermittelt werden, ob sich ein Objekt in einer zu kalten oder warmen Temperaturzone befindet. Hierbei wird die räumliche Zuordnung mittels der Informationen aus dem DURMAD-Modell ermittelt. Aus dem Objektgedächtnis wird die optimale Temperaturzone zur Lagerung des Objekts und über einen Sensor die Umgebungstemperatur geliefert. Mittels des Agentensystems erfolgt der automatische Abgleich beider Temperaturen, falls sich das Objekt innerhalb einer vorgegebenen räumlichen Distanz zum Temperatursensor befindet. Dabei wird die Soll-Temperatur des Objekts mit der Ist-Temperatur des Sensors verglichen. Als Resultat kann entweder eine grafische Hervorhebung in der 3D-Visualisierung erfolgen (siehe Abbildung 5.17(a)) oder eine automatisch generierte Notifikation erstellt und versendet werden. Ein weiteres Beispiel, welches durch ein solches Agentensystem realisiert werden kann, liegt darin, zu bestimmen, ob noch eine gewisse Anzahl an Objekten eines speziellen Typs auf einem Tisch oder in einem Regal liegt. Sobald dieser Grenzwert unterschritten ist, kann eine entsprechende Aktion ausgeführt werden, wie z.B. die farbliche Hervorhebung des Tischs

bzw. Regals (siehe Abbildung 5.17(b)). In der Abbildung ist zudem eine Informationsdarstellung zu sehen, über welche die Grenzwerte angepasst werden können.



(a) Temperatur der Kühtruhe ist zu hoch für die Produktlagerung

(b) Die Mindestmenge an Produkten in der Kühltruhe ist unterschritten

***Abbildung 5.17**: Farbliche Hervorhebung von Objekten zur Darstellung eines durch ein Agentensystem ermittelten Handlungsbedarfs*

5.4.3 BI-Dienste

Eine Ermittlung von spezifischen Objektparametern basierend auf erfassten Informationen erfolgt über Funktionen, welche unter dem Begriff Business Intelligence (BI) Diensten bekannt sind (siehe Kapitel 2.5.3). Neben der Ermittlung von Parametern umfassen BI-Dienste Schnittstellen zu Darstellungskomponenten für eine Repräsentation der Resultate. Um eine einfache Möglichkeit der Erstellung solcher Dienste zu realisieren, wurde in DURMAD ein BI-Editor kreiert und implementiert. Dieser ermöglicht es, auf einer grafischen Oberfläche ähnlich wie bei dem Regelsystem Funktionen per Drag and Drop zusammenzustellen. Die Oberfläche ist in eine Editieroberfläche und eine Darstellung von zur Verfügung stehenden Modulen unterteilt, wie in Abbildung 5.18 dargestellt. Bei den Modulen wird zwischen Ein-, Ausgängen und Operationen unterschieden, welche zur einfachen Differenzierung in unterschiedlichen Farben repräsentiert werden. Die Module müssen einmalig von einem Experten erstellt werden und stehen danach den Anwendern in der grafischen Oberfläche zur Verfügung. Potentielle Eingänge stellen hierbei beispielsweise Informationen aus einer Datenbank oder den Objektgedächtnissen dar. Ebenso können hier Konstanten durch einen Endbenutzer spezifiziert werden, was durch ein entsprechendes Kontextmenü realisiert wurde. Die Operationen umfassen beispielsweise Division und Multiplikation, aber auch komplexere Funktionen, je nach Anwendungsfall. Die Ausgänge dienen schließlich der Repräsentation der Daten. Die ermittelten Ergebnisse können anschließend in eine Datenbank geschrieben oder als direkte Information in eine Darstellungskomponente von DURMAD transferiert werden. Hierbei können die Daten einerseits in Form von Text oder Grafiken repräsentiert werden. Andererseits kann anhand dieser Informationen auch die 3D-Visualisierung angepasst werden, indem Objekte transparent oder mit farblichen Überblendungen dargestellt werden, entsprechend der implementierten Visualisierungstechniken (siehe Abchnitt 5.3.5). Die generierten BI-Dienste werden in einer Datei gespeichert und können somit jederzeit wieder geladen, angepasst oder ausgeführt werden.

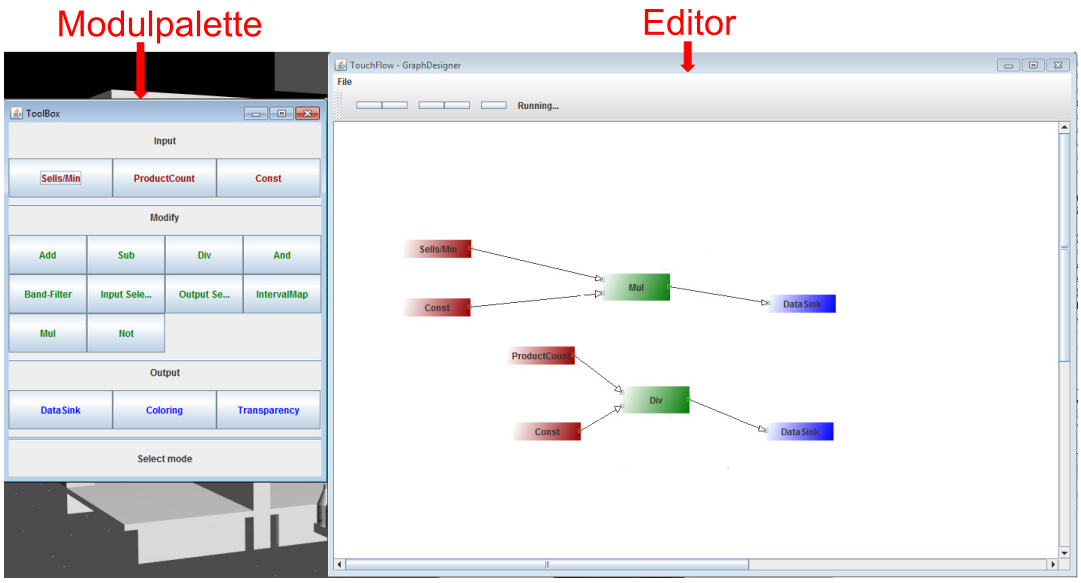


***Abbildung 5.18**: Grafische Oberfläche zum Editieren von BI-Diensten*

5.5 Kontroll- und Steuerungssysteme

Die Verbindung zwischen der CPE und DURMAD wird durch mehrere Funktionalitäten unterstützt (siehe Abbildung 5.19). So wird die Interaktion mit einem Benutzer- und Rechtemanagement gekoppelt, bei dem Benutzer in Gruppen eingeteilt sind, welche dedizierte Rechte zugeteilt bekommen (siehe Abschnitt 5.2.1). Die Rechte werden in einer DB hinterlegt und mit den Anforderungen der einzelnen Module abgeglichen, bevor diese aktiviert werden können (siehe Abschnitt 5.2). Damit ist es möglich, Eingriffe in die CPE mittels DURMAD ausschließlich befugten Personen zu gewähren. Des Weiteren wurde eine VNC-Schnittstelle implementiert, so dass Bildschirminhalte aus der physischen Welt gleichzeitig in der virtuellen Welt auf den Virtualisierungen der jeweiligen Bildschirme dargestellt werden können. Die Bilddaten werden in diesem Fall als Texturen auf die entsprechenden virtuellen Modelle gelegt.

Neben der Möglichkeit, die reale Welt widerzuspiegeln und diese zu beeinflussen, dient DURMAD dem Durchführen von Simulationen gemäß dem DR++-Ansatz (siehe Kapitel 4.2.4). Hierzu wurde unter anderem die Möglichkeit geschaffen, virtuelle Objekte in das Modell zu integrieren. Diese haben keine physische Entsprechung, können aber die gleichen Reaktionen in der virtuellen oder realen Umgebung auslösen. Beispielsweise kann durch eine Veränderung der Position des virtuellen Objekts ein Sensor ausgelöst werden, als wäre dies in der physischen Umgebung erfolgt. Die Auswirkungen können dabei entweder ausschließlich auf die virtuelle 3D-Visualisierung beschränkt oder auch in der CPE widergespiegelt werden. Hierzu wird eine spezielle Kommunikationsschnittstelle verwendet, welche im nächsten Kapitel ausführlich beschrieben wird. Ein Beispiel einer solchen im Virtuellen ablaufenden Simulation stellt die simulierte Bewegung von Personen

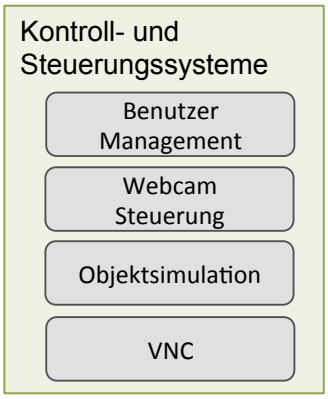


***Abbildung 5.19**: Kontroll- und Steuerungssysteme des Dual Reality Management Dashboards*

dar. Hierzu ist es einerseits möglich, Bewegungspfade aus der realen Welt aufzuzeichnen und anschließend abzuspielen. Andererseits können Kamerapfade durch Interaktionen im Virtuellen aufgezeichnet werden, die anschließend als Bewegungssimulation ablaufen können. Somit ist es möglich, kontrollierte Simulationen der gleichzeitigen Bewegung mehrerer Personen durchzuführen, um Systeme der CPE hinsichtlich Mehrbenutzerverhalten zu testen.

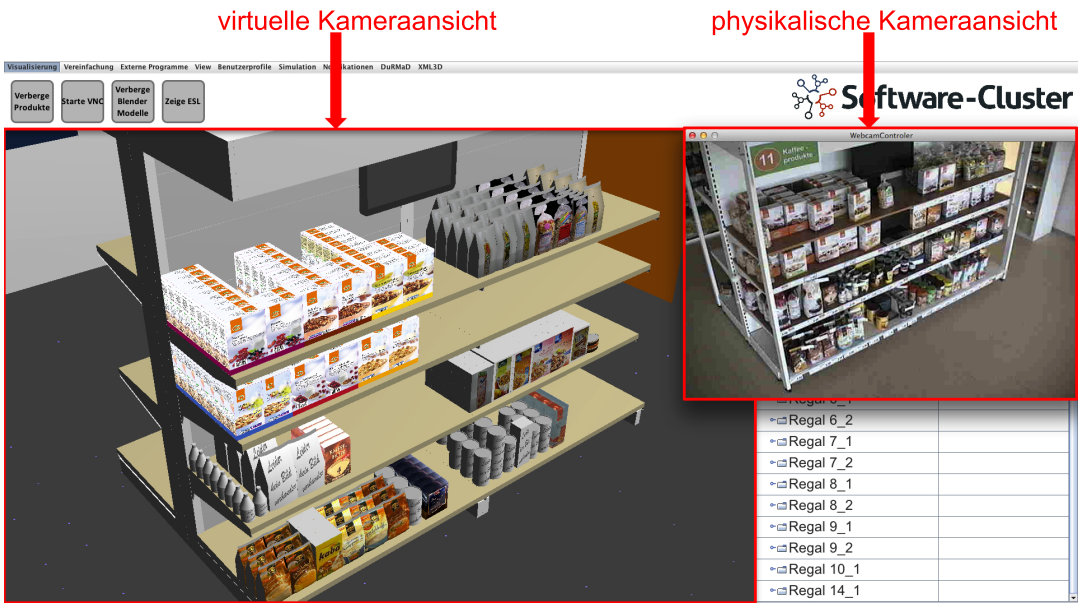


***Abbildung 5.20**: Synchronisation zwischen virtueller und realer Kamera*

Neben der reinen Steuerung von virtuellen Kameras wurde in DuRMAD auch eine Schnittstelle geschaffen, um physische PTZ-Kameras anzusteuern. Hierzu wird eine Virtualisierung dieser Kameras benötigt, welche Position, Initialausrichtung und Bewegungsfähig-

keiten beinhaltet. Wird die virtuelle Kamera bewegt, werden die Veränderungen automatisch über die entsprechende Schnittstelle an die reale Umgebung gesendet, wo die reale Kamera an die entsprechende Position bewegt wird. Das Live-Bild der physischen Kamera wird hierbei in einem separaten Fenster dargestellt (siehe Abbildung 5.20). Somit wird eine Synchronisation zwischen der virtuellen und der physischen Kamera ermöglicht. Beispielsweise ist dies hilfreich, wenn die virtuelle Umgebung mit dem realen Zustand abgeglichen werden muss. Dies ist unter anderem der Fall, wenn nur eine partiell modellierte DR der CPE vorliegt und daher nicht jeder Zustand der physischen Umgebung automatisch im Virtuellen widergespiegelt wird.

5.6 Zusammenfassung

In diesem Kapitel wurde ein Tool zur interaktiven dreidimensionalen Darstellung einer virtualisierten CPE vorgestellt, das *Dual Reality Management Dashboard (*DURMAD*)*. Um eine gegenseitige Beeinflussung zwischen einer CPE und ihrer Virtualisierung zu realisieren, wurden diverse Module implementiert, welche diese Funktionalität in verschiedenen Formen ermöglichen. Basierend auf diesen Modulen und dem DR++-Ansatz dient DURMAD als Monitoring- und Steuerungskomponente für CPE.

Die in den einzelnen Abschnitten detailliert beschriebenen Module bilden die in Abbildung 5.21 dargestellte Gesamtarchitektur von DURMAD. Gestrichelte Verbindungen stellen dabei spezialisierte Ausprägungen und durchgezogene Linien eine Kommunikationsschnittstelle zwischen den Modulen dar. Die Datenquellen fungieren hierbei als Grundlage für die Darstellung. DURMAD unterstützt Objektmodelle, welche in einer Java3D-nativen XML-Syntax oder in XML3D-Syntax vorliegen. Als Grundrisspläne für die Darstellung einer CPE dienen entweder YAMAMOTO-Modelle im yml-Format oder GIS-Modelle im gml-Format. Durch die Kombination der Grundrisspläne mit den Objektmodellen wird die 3D-Visualisierung der CPE automatisch generiert. Eine weitere Datenquelle stellen Ontologien dar, die mittels einer entsprechenden Schnittstelle in DURMAD eingebunden werden können. Die konzipierte Ontologie umfasst unter anderem eine semantische Beschreibung der Sensoren und Aktuatoren.

Zur Anbindung an weitere Datenquellen und für externe Applikationen wurden in DURMAD mehrere Schnittstellen implementiert. Eine Schnittstelle bietet die Möglichkeit, spezifische Informationen aus einer Datenbank abzurufen. Das zugehörige Datenbankschema ermöglicht die Speicherung und das Laden von Informationen zu Objekten und Personen. Bei DURMAD werden diese Daten in der 3D-Visualisierung verwendet. Im Speziellen liegt ein Schwerpunkt auf der Zuweisung von Rollen und Rechten an Personengruppen für eine mögliche Zugriffsbeschränkung auf diverse Funktionen von DURMAD. Schnittstellen zu digitalen Objektgedächtnissen ermöglichen das Abrufen objektbezogener Daten, um diese darzustellen oder sie für Automatismen zu verwenden. Schließlich werden eine SOAP- und eine Kommunikationsschnittstelle angeboten, um Informationen aus DURMAD in externen Applikationen verwenden zu können.

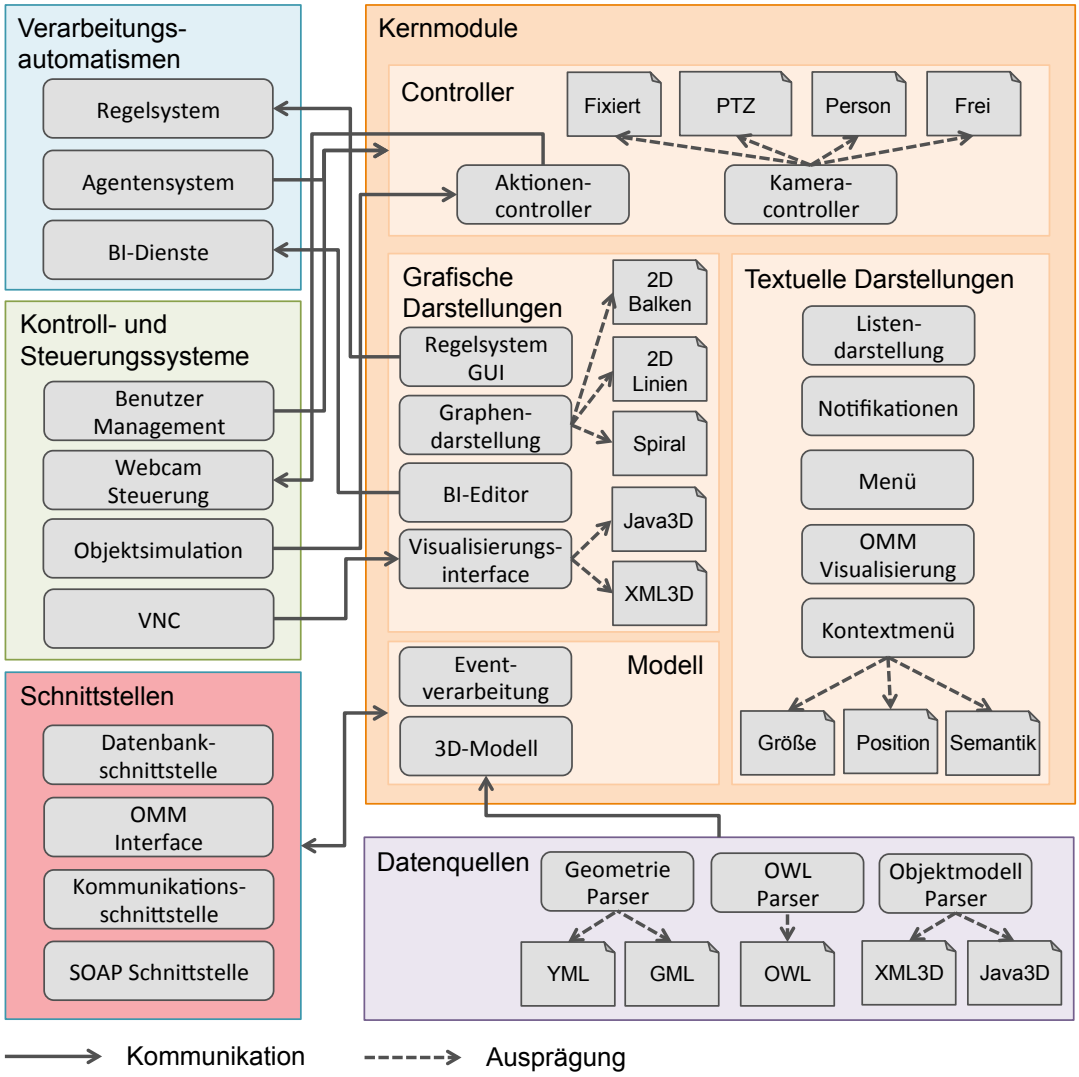


***Abbildung 5.21**: Architektur des Dual Reality Management Dashboards*

Die Kernmodule umfassen mehrere Visualisierungsformen und ermöglichen eine Repräsentation von Informationen. Basierend auf dem Model-View-Controller-Ansatz können verschiedene Visualisierungskomponenten verwendet werden. Exemplarisch wurde eine Darstellung in Form einer Java3D- und einer XML3D-Komponente implementiert, so dass auch eine 3D-Darstellung auf Internetseiten sowie auf mobilen Endgeräten erfolgen kann. Für die Darstellung der gesammelten und verarbeiteten Informationen wurden mehrere Dashboard-Komponenten implementiert, wie textbasierte und grafische Darstellungen. Zudem bieten mehrere Visualisierungstechniken die Möglichkeit, normierte Objektparameter direkt in der 3D-Visualisierung zu repräsentieren. Die Visualisierungstechniken haben hierbei einen direkten Einfluss auf die Darstellung des Objekts in der 3D-Szene und können Transparenz, Farbgebung und Größe des Objekts statisch oder dynamisch verändern.

Über den Controller können Veränderungen im virtuellen Modell erfolgen. Insbesondere werden hierüber die Kamerapositionen verändert, sofern dies durch die implementierten Kameratypen erlaubt ist. Die diversen Kameratypen unterscheiden sich in der Spezifikation diverser Parameter und weisen unterschiedliche Einsatzzwecke auf.

Die in DURMAD integrierten Verarbeitungsautomatismen umfassen ein Regel-, ein Agentensystem und Business Intelligence (BI) Dienste. Sie ermöglichen eine automatisierte Reaktion basierend auf Veränderungen in der CPE. Diese Reaktionen können sich entweder ausschließlich auf die virtuelle Komponente beziehen oder auch eine Veränderung in der CPE nach sich ziehen. Mittels grafischer Oberflächen ist es selbst Nicht-Experten möglich, Regeln einfach zu erstellen und neue BI-Dienste zu konfigurieren.

Die Kontroll- und Steuerungssysteme unterstützen den Datenaustausch zwischen der CPE und ihrer Virtualisierung. Beispielsweise kann mittels der Integration von VNC der Bildschirminhalt aus der realen Umgebung im Virtuellen wiedergegeben werden. Ein Benutzermanagement ermöglicht in DURMAD eine rollen- und rechtespezifische Interaktion und eine personalisierte Darstellungsform. Eine Steuerungskomponente für PTZ-Kameras verbindet die physische Kamera mit ihrer Virtualisierung, um Informationen zwischen beiden Umgebungen austauschen und beide Kameras miteinander synchronisieren zu können. Simulationen, wie beispielsweise Personenbewegungen, können in DURMAD integriert werden, wodurch das DR++-Konzept realisiert wird.

Um einen Datenaustausch zwischen einer CPE und ihrer Virtualisierung zu ermöglichen, werden entsprechende Kommunikationsschnittstellen benötigt. In diesem Zusammenhang wurde eine eigene Kommunikationsinfrastruktur entwickelt, welche im folgenden Kapitel detailliert beschrieben wird. Diese ermöglicht sowohl die Kommunikation zwischen den physischen Komponenten der CPE als auch den Datenaustausch mit DURMAD.

6 EVENT BROADCASTING SERVICE (EBS)

Für eine Verschmelzung einer Cyber-Physischen Umgebung (CPE) mit ihrer Virtualisierung muss eine geeignete Kommunikationsinfrastruktur verwendet werden, so dass eine Synchronisation sowie ein bidirektionaler Datenaustausch gewährleistet werden kann. Im Speziellen stellt hierbei die Berücksichtigung des Konzepts der Erweiterten Dual Reality (DR++) eine Besonderheit dar. Damit die Kommunikationsinfrastruktur zusätzlich in die CPE integriert und dort verwendet werden kann, müssen gewisse Eigenschaften von der Kommunikationsinfrastruktur unterstützt werden. Hierbei müssen die Architektur, die Art und das Format der Daten, welche übertragen werden sollen, spezifiziert werden. Basierend auf diesen Anforderungen wurde im Rahmen dieser Arbeit eine Architektur zur Informationsübertragung in CPE konzipiert und umgesetzt. Die resultierende Kommunikationsinfrastruktur wurde zudem um Filter und Vorverarbeitungsmodule erweitert, welche von den Komponenten vor dem Versand oder nach dem Empfang von Informationen verwendet werden können. Aufgrund der Integration mit dem Dual Reality Management Dashboard (DURMAD) bietet die entwickelte Kommunikationsinfrastruktur eine Möglichkeit, CPE zu monitoren, zu steuern und gleichzeitig Simulationen zu integrieren.

In Kapitel 2.6 wurde bereits eine Abbildung zur „*Architektur im Sinne der ambienten Intelligenz inklusive der Middleware*“ präsentiert, welche den Datenfluss zwischen einer Smarten Umgebung, Diensten und Benutzern darstellt. Aufgrund der Verbindung einer CPE zum Cyberspace muss letzterer über die Kommunikationsinfrastruktur erreichbar sein, um einen Datenaustausch zu ermöglichen. Zudem verschwimmen nach dem Konzept der DR die Grenzen zwischen der realen und virtuellen Umgebung, so dass auch explizite Interaktionen mit physischen Objekten als Teil der CPE relevant sind, um einen direkten Einfluss auf Applikationen auszuüben. Dies ist zwar bereits bei der präsentierten Abbildung 2.5 inkludiert, stellt jedoch einen wichtigen Bestandteil von DR dar und sollte daher explizit in der Architekturskizze berücksichtigt werden. Ein integraler Bestandteil von DR-Systemen ist ebenfalls die visuelle Darstellungskomponente, welche den aktuellen Zustand der Umgebung repräsentiert. Diese Komponente dient somit zur Überwachung und mittels einer interaktiven Darstellung auch als Kontrollsystem für die Umgebung, mit welcher der Benutzer interagieren kann. Basierend auf diesen Ergänzungen wurde die Architekturskizze aus Kapitel 2.6 im Hinblick auf DR in CPE erweitert. Sie ist in Abbildung 6.1 dargestellt, wobei die ergänzten Verbindungen und Komponenten farblich hervorgehoben sind.

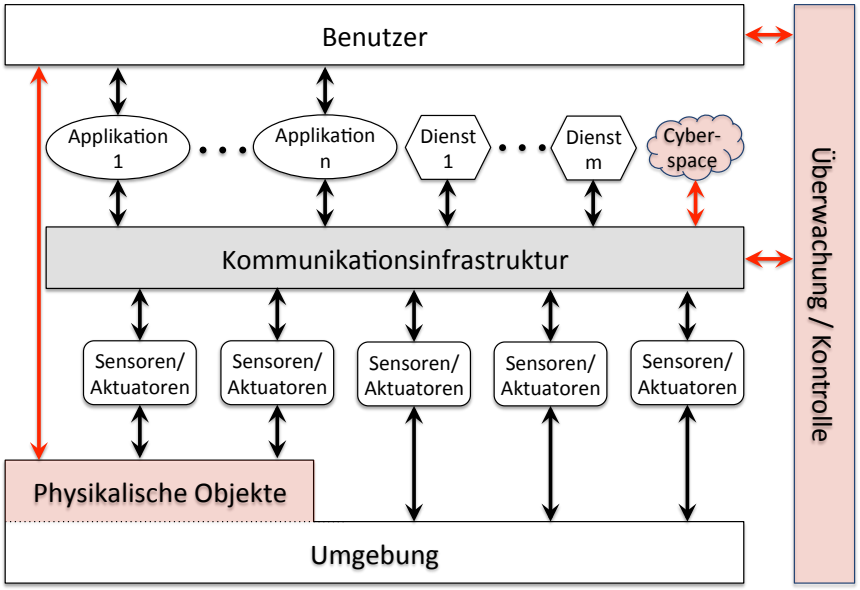


***Abbildung 6.1**: Erweiterung der AmI-Architektur um Dual Reality-Interaktion in CPE*

Wie bereits in Kapitel 2.3 in Tabelle 2.2 erörtert, gibt es mehrere Differenzierungsmerkmale, die bei der Kommunikation in CPE unterschieden werden. Aufgrund der Anzahl an Sensoren und Aktuatoren ist eine Kommunikationsverbindung als Eins-zu-n- oder N-zu-n-Verbindung notwendig. Im ersten Fall würde eine zentrale und andernfalls eine dezentrale Systemverteilung vorliegen. Da Systeme zu jeder Zeit in die Struktur aufgenommen werden müssen, um die Agilität von CPE sicherzustellen, muss die Kommunikationsinfrastruktur eine ad-hoc-Verbindung unterstützen. Ebenso muss die Übertragung asynchron funktionieren, so dass es nicht zu einem kompletten Systemausfall kommen kann, wenn ein Client die Kommunikation blockiert. Schließlich kann noch zwischen einer push- und einer pull-basierten Datenübertragung unterschieden werden. Da die Clients in der pull-basierten Version nicht notifiziert werden, wenn neue Informationen vorhanden sind, müssen diese in kurzen Zeitintervallen beim Server anfragen, um eine zeitgerechte Informationsübermittlung zu gewährleisten. Dies kann eine hohe Auslastung der Infrastruktur durch nicht notwendige Anfragen zur Folge haben. Sind die Zeitabstände der Anfragen zu lang, ist eine zeitnahe Verarbeitung der Informationen jedoch nicht gewährleistet. Push-Systeme bieten daher einen schnelleren Datenaustausch, wobei es zu einer höheren CPU-Auslastung kommt, da die Verbindung zu den Clients die gesamte Zeit offen bleiben muss [Bozdag et al., 2007]. Dies kann wiederum zu einem Verlust von Events führen, die aufgrund der hohen CPU-Auslastung nicht an die Clients versendet werden. Diese Problematik muss daher bereits im Design der Infrastruktur Berücksichtigung finden.

Zusätzlich zu diesen Differenzierungsmerkmalen muss bei der Entwicklung einer Kommunikationsinfrastruktur auf wesentliche Eigenschaften Rücksicht genommen werden, welche im Speziellen bei agilen CPE auftreten. Die folgenden Eigenschaften müssen von einer

Kommunikationsinfrastruktur erfüllt sein, um sie zur einfachen Informationsübertragung einsetzen zu können [Henricksen et al., 2005].

↪ **Unterstützung von Heterogenität**. Sowohl ressourcenschwache Sensoren, Aktuatoren, mobile Clients als auch hochperformante Server müssen die Möglichkeit haben, sich über geeignete Schnittstellen mit der Infrastruktur zu verbinden und mit den anderen Geräten in der Umgebung zu kommunizieren.

↪ **Unterstützung von Mobilität**. Alle Komponenten, im Speziellen Sensoren und Aktuatoren, können auch in einer mobilen Version in Form von mobilen Endgeräten vorliegen. Aus diesem Grund muss die Kommunikationsinfrastruktur eine Flexibilität bei der Informationszustellung unterstützen.

↪ **Skalierbarkeit**. Unabhängig von der Anzahl an Sensoren, Aktuatoren und Verarbeitungskomponenten muss die Infrastruktur einen entsprechenden Informationsdurchsatz aufweisen und daher entsprechend skalieren.

↪ **Unterstützung von Sicherheit**. Informationsflüsse zwischen verteilten Komponenten müssen kontrolliert und geschützt werden, gemäß einer vom Benutzer vorgegebenen Sicherheitsanforderung.

↪ **Verfolgbarkeit und Kontrolle**. Durch die starke Vernetzung der Komponenten, welche zum Teil auch mobil sind, muss eine Verfolgbarkeit des Kommunikationsflusses und eine Eingriffnahme des Benutzers möglich sein, um eine einfache Fehlererkennung und -beseitigung zu realisieren.

↪ **Toleranz gegenüber Komponentenfehlern**. Aufgrund der Fehleranfälligkeit von verteilten Systemen und den darin enthaltenen unterschiedlichsten Komponenten muss die Kommunikationsinfrastruktur auftretende Fehler erkennen und adäquat darauf reagieren, um den weiteren Kommunikationsfluss nicht zu stören.

↪ **Einfache Verwendbarkeit und Konfiguration**. Ein wichtiger Aspekt für die Verwendung einer Kommunikationsinfrastruktur beruht auf einer einfachen Anwendbarkeit mit leichten Konfigurationsmöglichkeiten, welche auch von Nicht-Experten einstellbar sein sollten.

Wie Kapitel 3.3.1 darlegt, gibt es bisher keine Kommunikationsinfrastruktur, welche mehrere Möglichkeiten der Filterung mit Verschlüsselung und automatischer Fehlererkennung kombiniert. Zudem ist im Sinne der Realisierung der Verfolgbarkeit und Kontrolle des Informationsflusses eine visuelle Darstellung der Informationsverteilung wünschenswert. Aus diesen Gründen wurde eine neue Kommunikationsinfrastruktur entwickelt, welche die zuvor aufgeführten Richtlinien und Eigenschaften berücksichtigt. Für die Kommunikation wurde ein eventbasierter Ansatz gewählt.

6.1 Events

Gemäß der Definition 2.9 beschreibt ein Event ein messbares Ereignis in einer Umgebung. Im Fall einer CPE können Sensoren für die Detektion von Kontextparametern der Umgebung

verwendet werden, welche in Form einfacher Events an die weiteren Komponenten der CPE kommuniziert werden können. Neben einfachen Events, wie beispielsweise Temperaturmessungen eines Thermometers, können auch mehrere einfache Events kombiniert und daraus Schlussfolgerungen gezogen werden. Hierdurch entstehen komplexe Events, wie beispielsweise der Rückschluss auf ein Feuer, wenn der Temperatursensor einen hohen Wert ermittelt und zeitgleich Rauch erkannt wird. Die Detektion solcher komplexer Events kann hierbei mittels logischer Sensoren erfolgen, wie in Kapitel 2.2.1 beschrieben. Bezüglich der Art und Funktionsweise können Events in zwei Klassen unterteilt werden, nämlich Nachrichten und Ausführungsaufrufe [Eugster et al., 2003]. Nachrichten dienen der Übermittlung von Informationen, welche zum Monitoren verwendet werden können, indem diese beispielsweise für eine visuelle Darstellung in Form eines Dashboards Verwendung finden. Bei Ausführungsaufrufen werden entsprechende Trigger zum Starten von Aktionen versendet, analog zu RPC (siehe Kapitel 2.3).

Sensorinformationen spiegeln Teile des aktuellen Kontextes einer Umgebung wider. Mittels einer Kombination mehrerer solcher Sensorinformationen kann das Kontextwissen erhöht werden. Die Übermittlung kann hierbei in Form von einfachen Events erfolgen, indem die Sensorinformationen in entsprechende Events transformiert werden. Zur Repräsentation des erfassten Kontextes müssen die generierten Events mehrere Eigenschaften erfüllen [Baldauf et al., 2007]:

↪ **Typ**. Dieser beschreibt die Kategorie des Kontextparameters und kann anschließend für Filterungen verwendet werden. Ein Beispieltyp ist „Temperatur“.

↪ **Wert**. Hierin werden die Rohdaten integriert, wobei anhand des Eventtyps die Einheit verständlich sein muss oder in Form eines zusätzlichen Eintrags hinzugefügt werden sollte.

↪ **Zeitstempel**. Dieses Attribut beschreibt den Zeitpunkt, an dem das Event erfasst wurde, z.B. wann der Sensor den Wert gemessen hat.

↪ **Quelle**. Um zu ermitteln, welcher Sensor oder Dienst die Informationen verschickt hat, muss eine eindeutige Identifizierung der Datenquelle möglich sein, welche den Events hinzugefügt werden sollte.

↪ **Konfidenz**. Da bei Sensoren Unsicherheiten auftreten können, sollten diese Informationen in Form von Konfidenzwerten, deren Wertebereich zwischen 0 und 1 liegt, ebenfalls bei der Eventkommunikation berücksichtigt werden.

Entsprechend dieser Eigenschaften wurde eine Grundstruktur für Events entwickelt, welche je nach Typ mit weiteren spezifischen Eigenschaften ergänzt werden kann. Aufgrund der Tatsache, dass die Kommunikationsinfrastruktur auch komplexe Events verschicken kann, welche aus einem Zusammenschluss mehrerer Daten bestehen, gibt es in der Grundstruktur keine explizit definierte Datenstruktur für den Wert. Anstelle dessen muss jeder Eventtyp eigenständig spezifizieren, welche Werte integriert werden sollten. Neben diesen notwendigen Parametern umfasst die Grundstruktur weitere Attribute, welche zur Erfüllung der zuvor erörterten Eigenschaften von CPE benötigt werden. Die vollständige Liste aller

Parameter in der Eventgrundstruktur inklusive einer kurzen Beschreibung ist in Tabelle 6.1 aufgeführt.

Implementierungstechnisch werden einige der Parameter automatisch vom System ausgefüllt bzw. mit Standardparametern versehen. So richtet sich der Typ eines Events nach dessen Klassennamen, was eine spätere Zuordnung erleichtert. Der Erfassungszeitpunkt wird automatisch bei der Eventerstellung generiert. Ebenso wird die Identifikation der Quelle automatisch ergänzt. Die Session setzt sich aus dem eindeutigen Namen des Clients und dem Startzeitpunkt zusammen, an dem der Client sich mit dem Server verbunden hat. Diese Angabe wird ebenfalls automatisch bei der Erstellung eines Events übernommen. Standardmäßig existiert keine Einschränkung hinsichtlich der Gültigkeit eines Events. Wird nicht explizit ein Adressatenkreis spezifiziert, wird davon ausgegangen, dass alle aktuell verbundenen Clients Ziele des Events darstellen. Standardmäßig sind die Events so eingestellt, dass sie nicht als privat angesehen werden und nicht verschlüsselt werden sollen. Die Priorität ist auf normal festgelegt. Zudem wird die Konfidenz der im Event enthaltenen Informationen als sehr hoch (Konfidenz = 1) angesehen und es wird von real erfassten Daten ausgegangen.

Aufgrund der Möglichkeit, sowohl einfache als auch komplexe Events zu verschicken, muss eine adäquate Informationsbündelung in einem Event Berücksichtigung finden. Wichtig bei der Erstellung entsprechender Events ist, dass diese nicht zu speziell, aber auch nicht zu generell hinsichtlich der enthaltenen Informationstypen, wie beispielsweise Temperatur oder Druck, gehalten werden. So kann ein Sensor, welcher gleichzeitig Temperatur und Druck erfassen kann, beide Werte als separate Events versenden oder die Werte in einem Eventtyp bündeln. Verwendet man nur sehr spezielle, auf jeden Informationstyp ausgerichtete Events, so besteht die Problematik, den Überblick über die daraus resultierende Vielzahl an unterschiedlichen Events zu behalten. Eine sensorspezifische Erstellung von Events mit mehreren Informationen birgt hingegen die Gefahr, dass man diese nicht für andere Sensortypen wiederverwenden kann, wenn diese andere Informationstypen erfassen. Dies erschwert somit den Austausch eines Sensors, da gegebenenfalls in diesem Fall ein neuer Eventtyp erstellt werden muss. Des Weiteren muss berücksichtigt werden, dass Informationen oftmals auch voneinander abhängig sind und es durch eine Aufsplittung in mehrere unterschiedliche Events zu Problemen bei der Verarbeitung kommen kann. In diesem Fall müsste auf alle Teilevents gewartet werden, um diese im Anschluss wieder zu fusionieren. Aus diesem Grund ist die Spezifikation geeigneter Eventtypen eine nicht-triviale Aufgabe. Oftmals bedürfen jedoch zuvor festgelegte Designentscheidungen mit der Zeit einer Änderung, so dass die Kommunikationsinfrastruktur neuen Gegebenheiten angepasst werden muss. Gerade bei sich stetig ändernden Umgebungen, die sehr agil und deren zukünftige Anforderungen und Ausprägungen nicht vorhersehbar sind, kann dieser Überarbeitungsprozess mehrfach notwendig sein. Unter anderem kann eine Separation oder Zusammenführung von Informationstypen sinnvoll sein. In der Regel müssen in einem solchen Fall die einzelnen von der Veränderung betroffenen Systeme entsprechend angepasst werden, so dass diese die neuen Events verarbeiten können. Dies kann relativ aufwändig oder sogar unmöglich sein, zum Beispiel bei Systemen, deren Programmierung nicht verändert werden kann. Um

Parameter	Wert / *Typ*	Beschreibung
Typ	*String*	Der Typ spezifiziert die Art des Events und kodiert dadurch dessen (Daten-)Struktur.
Erfassungs-zeitpunkt	*Zeitstempel*	Das Auftreten eines Events wird als Zeitstempel erfasst und dem Event hinzugefügt.
Gültigkeit	*Zeitstempel*|*ms*	Um die Gültigkeitsdauer eines Events anzugeben, kann entweder ein Zeitpunkt oder Zeitraum festgelegt werden.
Quelle	*String*	Der Sender wird durch eine eindeutige Identifikation spezifiziert, welche dem Event angehängt wird.
Ziel	*String[]*	Um eine gezielte Informationsübermittlung an einen Adressatenkreis zu ermöglichen, können die Empfänger des Events spezifiziert werden. Hierzu wird die jeweilige Identifikation der Empfänger angegeben.
Privat	true|false	Wenn das Event private Informationen enthält kann über dieses Flag sichergestellt werden, dass ausschließlich der Empfängerkreis das Event übermittelt bekommt.
Verschlüsse-lung	true|false	Eine Verschlüsselung kann für jedes Event einzeln eingestellt werden.
Session	*String*	Jeder erneute Start eines Systems wird als neue Session interpretiert. Dies ermöglicht die Erkennung von (automatischen) Systemneustarts.
Priorität	hoch|normal|gering	Zur Spezifikation der Wichtigkeit der Informationen kann eine Priorität angegeben werden. Diese ist in drei Stufen unterteilt: gering, normal und hoch.
Konfidenz	$[0..1]$	Dieser Wert kann die Datenqualität von erfassten Informationen darlegen.
Simulation	true|false	Durch dieses Flag kann spezifiziert werden, ob ein Event auf realen Informationen beruht oder das Ergebnis einer Simulation ist.

***Tabelle 6.1**: Auflistung und Kurzbeschreibung der in der Eventgrundstruktur enthaltenen Parameter*

dem entgegenzuwirken muss die Kommunikationsinfrastruktur eine möglichst einfache Art der Zusammenfügung und Aufsplittung von Events ermöglichen. Dies kann beispielsweise durch ein sogenanntes „Metaevent“ realisiert werden, welches mehrere einfache Events vor der Versendung vereint. Nimmt man einen mobilen Temperatursensor und einen GPS-Sensor als Beispiel, könnte der gemessene Temperaturwert mit der Lokationsinformation in einem „LokationsbasierteTemperaturEvent“ zusammengefasst und gebündelt versendet werden. Nach dem Empfang werden diese zusammengefügten Events in den Clients wieder getrennt.

6.2 Filter und Vorverarbeitung

Aufgrund der einheitlichen Struktur der definierten Events können verschiedene Filtermechanismen verwendet werden. Eine typische Filterung, welche weit verbreitet ist, ist die Aussortierung anhand des Eventtyps, beispielsweise bei Publish/Subscribe-Systemen. Zudem werden oftmals inhaltsbasierte Filterungen bei Publish/Subscribe-Systemen eingesetzt, um eine möglichst flexible Subskription zu ermöglichen [Aguilera et al., 1999]. Neben der Filterung von nicht relevanten Daten, wie beispielsweise der Aussortierung von nicht mehr gültigen Informationen, welches durch eine Verwendung von Gültigkeitszeitstempeln realisiert werden kann, können Filter auch eingesetzt werden, um die Anzahl von redundanten Informationen in der Kommunikation zu reduzieren. Als Beispiel kann ein Temperatursensor betrachtet werden, welcher mit einer hohen Frequenz die aktuelle Temperatur ermitteln kann. Wird jede dieser Erfassung als Event versendet, kommt es zu einer (sehr) hohen Anzahl an versendeten Events, welche sich mit der Anzahl der Empfänger entsprechend erhöht. Dies kann mit Hilfe einer entsprechenden Filterung reduziert werden und damit auch die Auslastung der Kommunikationsinfrastruktur. Alternativ zur Reduzierung der Anzahl von Events kann auch eine Einschränkung der Empfänger zu einer Minimierung der Auslastung herangezogen werden, was ebenfalls durch gezielte Filterung ermöglicht werden kann. Die Zahl der versendeten Events in einem System kann durch Formel 6.1 ermittelt werden. Hierbei wird davon ausgegangen, dass ein Event erst zu einer zentralen Kommunikationsinfrastruktur gesendet und von dieser an alle Empfänger verteilt wird. Aus diesem Grund muss die Anzahl der Empfänger inkrementiert um Eins mit der Anzahl der versendeten Events multipliziert werden.

$$|e^T(System)| = \sum_{s \in Sender} \sum_{t=t_0}^{t_0+T} (|e_t(s)| * (|r(e_t(s))| + 1)) \tag{6.1}$$

$$\begin{aligned} e^T(System) &:= \text{Alle im System versendeten Events im Zeitintervall T} \\ e_t(s) &:= \text{Vom Sender s verschicktes Event } e \text{ zum Zeitpunkt t} \\ r(e_t(s)) &:= \text{Alle Empfänger des versendeten Events } e_t(s) \end{aligned}$$

Neben reinen Filtermechanismen (siehe Abschnitt 6.2.1) ist die Verwendung von Verarbeitungsalgorithmen (siehe Abschnitt 6.2.2) oftmals bereits in der Kommunikationsschnittstelle wichtig, wie beispielsweise zur komplexen Eventverarbeitung. Viele Systeme

umfassen bereits ein gewisses Spektrum an Filtern und Vorverarbeitungsfunktionalität, wie in Kapitel 3.1 beschrieben und in Tabelle 3.1 zusammengefasst. Für den Event Broadcasting Service wurden im Rahmen dieser Arbeit ebenfalls Filter und Vorverarbeitungsfunktionalitäten entwickelt und implementiert, welche sowohl vor der Versendung als auch nach dem Erhalt von Events angewendet werden können [Kahl und Bürckert, 2012]. Ausgangsfilter ermöglichen beispielsweise eine Vermeidung der Versendung redundanter Daten, indem nur Veränderungen kommuniziert werden, auch wenn der Sensor in einer vorgegebenen Frequenz Daten erfasst und diese kommunizieren möchte. Dies ist unter anderem von Vorteil, wenn der zugehörige Programmcode des Sensors oder des Dienstes in Form einer Blackbox funktioniert, so dass der Benutzer nicht eigenständig die Frequenz der Senderate anpassen kann. Die Ein- und Ausgangsfilter sind dabei Teil der Client-Struktur des EBS, wie in Abbildung 6.2 dargestellt.

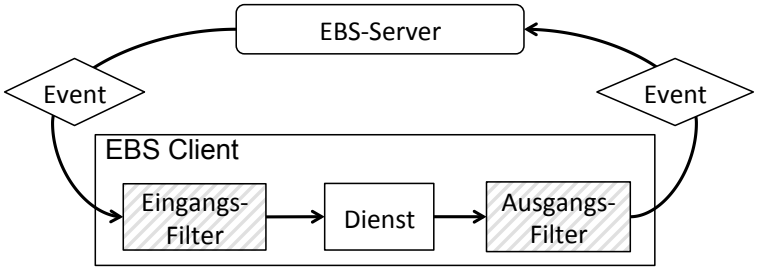


***Abbildung 6.2**: Filter als Teil des EBS-Client*

6.2.1 Filter

Filter beschreiben Verarbeitungen von Events, bei denen der Eventinhalt nicht verändert wird. Filter dienen ausschließlich dazu, gewisse Events durchzulassen und andere auszusortieren. Hierbei können neben inhaltlichen Aspekten auch zeitliche Constraints in Betracht gezogen werden. Insgesamt wurden im EBS drei Filtertypen integriert, welche im Folgenden erläutert werden. E beschreibt hierbei die Menge aller Events e, wobei e_{Typ} den Typ des Events darstellt.

Typfilter

$$f_{Typ}(E) = \{e \in E | e_{Typ} = Typ\}$$

Wie bereits beschrieben ist der Typfilter einer der am häufigsten in Systemen verwendete Filter. Entsprechend des Typs der Events wird herausgefiltert, welche durchgelassen werden und welche nicht. Abbildung 6.3 stellt einen solchen Filter grafisch dar, wobei die Eventtypen durch verschiedene Farben symbolisiert werden. In diesem Fall werden ausschließlich die Events vom Typ T_2 (blau) durchgelassen und die restlichen gefiltert. Die Filterung hat mitunter den Vorteil, dass die Eventtypen und damit auch die Schemata/Datenstrukturen der durchgelassenen Events im Nachgang bekannt sind. Dies kann von der Verarbeitung der darauf folgenden Dienste entsprechend verwendet werden.

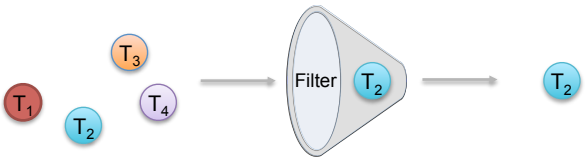


***Abbildung 6.3**: Typfilter*

Inhaltsfilter

$$f_{Ausdruck}(E) = \{e \in E | Ausdruck(e) = wahr\}$$

Neben den Typfiltern werden bei Publish/Subscribe-Systemen oftmals auch inhaltsbasierte Subskriptionen zur Verfügung gestellt. Dies wird im EBS durch einen Inhaltsfilter unterstützt, welcher basierend auf einem Ausdruck, der nach wahr oder falsch evaluiert, eine Filterung der Events ermöglicht. Ein Beispiel eines solchen Ausdrucks ist, dass die im Event enthaltene Temperatur über 7 Grad liegen soll ($e.getTemperature() > 7$). Wird der Ausdruck als wahr evaluiert, kann das Event den Filter passieren, andernfalls wird es aussortiert. Der Ausdruck kann hierbei abhängig vom Einsatz des Filters, spezifiziert werden. Neben einer Auswertung basierend auf statischen Parametern können auch dynamische Inhalte bei der Ausdrucksevaluation verwendet werden. Hierbei können Daten aus den Events zwischengespeichert und bei zukünftigen Evaluationen mit berücksichtigt werden. Dies ermöglicht beispielsweise jeweils eine Notifikation, immer wenn der Besucherzähler in einem Museum an einem Tag den bisherigen Besucherrekord überschreitet. Sobald dies erfolgt, wird der momentane Maximalwert durch den aktuellen Zählerstand ersetzt und dieser damit als neuer Höchststand für die künftigen Abgleiche verwendet. Mit dem Inhaltsfilter kann auch ein „Nicht-Filter“ realisiert werden, wie in Abbildung 6.4 dargestellt. Im Beispiel werden nur Events durchgelassen, deren Wert nicht 3 ist. Eine Spezialisierung des Inhaltsfilters ist der Duplikatfilter, welcher automatisch zwei identische Events erkennt und das als zweites aufgetretene nicht durchlässt.

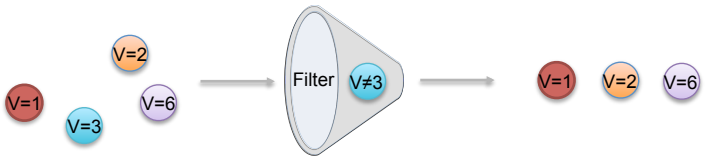


***Abbildung 6.4**: Inhaltsfilter*

Fensterfilter

e^n beschreibt das n-te Element einer geordneten Menge E.

$$f_{Anzahl}(E) = \{e^n | n \mod Anzahl = 0\}$$

$$f_{Zeit_1,Zeit_2}(E) = \{e \in E | Zeit_1 \leq e_{Erstellzeitstempel} \leq Zeit_2\}$$

Zur Verarbeitung von Events in einem vorgegebenen Zeitintervall $[Zeit_1, Zeit_2]$ oder nach einer gewissen Anzahl an eingetroffenen Events kann der Fensterfilter verwendet werden. Die Events werden hierbei entsprechend ihres Auftretens anhand der Zeitstempel sortiert und als geordnete Liste weitergeleitet. Diese kann hergenommen werden, um beispielsweise jedes zweite eintreffende Event den Filter passieren zu lassen, wie in Abbildung 6.5 dargestellt, wobei in diesem Beispiel die Eventeintreffen durch die Zeitstempel $(t_1, \ldots, t_4)$ spezifiziert werden. Ebenso kann ein Zeitintervall spezifiziert werden. Ausschließlich Events, die während dieses Intervalls erstellt wurden, werden vom Filter durchgelassen. Hierbei kann die eine Grenze in Abhängigkeit von der anderen gewählt werden, während diese sich zyklisch ändert. Somit können beispielsweise Events innerhalb der letzten n Minuten durch den Filter gelassen werden.

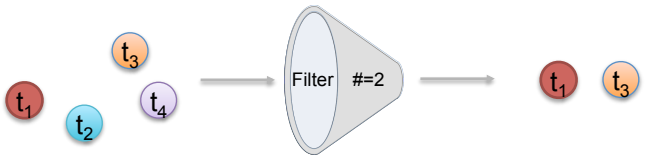


***Abbildung 6.5**: Fensterfilter*

6.2.2 Vorverarbeitung

Im Gegensatz zu den Filtern werden bei Vorverarbeitungen nicht zwangsläufig Events aussortiert. Anstelle dessen werden mehrere Events miteinander fusioniert oder der Inhalt nach gegebenen Vorgaben verändert. Im EBS wurden fünf Vorverarbeitungsmodule implementiert. Damit können kleinere Verarbeitungen erfolgen, ohne etwas an der eigentlichen Implementierung der Komponenten der Cyber-Physischen Umgebung zu verändern.

Akkumulation

$$f_{Akk,Anzahl,i}(E) = \{Akk(e^1, \ldots, e^{Anzahl}) | e^1 = e^i \wedge \cdots \wedge e^{Anzahl} = e^{i+Anzahl}\}$$

$$f_{Akk,Zeit_1,Zeit_2}(E) = \{Akk(e^1, \ldots, e^n) | e^k \in E \wedge Zeit_1 \leq e^k_{Zeitstempel} \leq Zeit_2\}$$

Ein Vergleich mehrerer Events des gleichen Typs kann mittels einer Akkumulation erfolgen. Hierbei kann beispielsweise das Minimum, das Maximum, der Mittelwert oder der Durchschnitt eines Eventparameters ermittelt werden. Das Resultat wird in einem Event des gleichen Typs eingefügt und dieses anschließend weitergeleitet. Die zu betrachtende Menge an Events wird wie beim Fensterfilter mittels einer definierten Anzahl an Events angefangen ab einem spezifizierten Event e^i oder einem Zeitintervall spezifiziert. Beispielsweise kann eine Durchschnittstemperatur von vier Events ermittelt werden, wie in Abbildung 6.6 skizziert, wobei die Zahlen die jeweiligen Temperaturen symbolisieren.

Kombination

$$f_{Typ_1,\ldots,Typ_n,Typ_z}(E) = \{e(e^1, \ldots, e^n) | e^k \in E \wedge e^k_{Typ} = Typ_k \wedge e_{Typ} = Typ_z\}$$

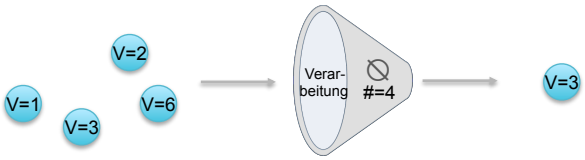


***Abbildung 6.6**: Akkumulation*

Bei der Kombination geht es darum, mehrere Events mit unterschiedlichen Typen miteinander in Verbindung zu bringen und somit eine komplexe Eventdetektion zu realisieren. Hierbei werden die unterschiedlichen Eventtypen spezifiziert, auf die gewartet werden muss. Sobald einer dieser Eventtypen am Filter registriert wird, wird das entsprechende Event für die spätere Verarbeitung zwischengespeichert. Wird erneut ein Event vom selben Typ erfasst, kann dieses entweder das erste ersetzen oder wird verworfen. Dies kann in der Konfiguration der Kombination spezifiziert werden. Sobald alle benötigten Eventtypen registriert wurden, wird der Kombinationsoperator aktiv und kann entweder diese Events durchlassen oder ein neues Event generieren, wie in Abbildung 6.7 angedeutet. Die Reihenfolge der eintreffenden Events ist hierbei egal. Dies kann beispielsweise zur Detektion und der anschließenden Notifikation eines Feuers verwendet werden, wenn ein Temperatursensor und ein Rauchsensor ein entsprechendes Event versenden. Ebenso können hierdurch mehrere unterschiedliche Events in einem gebündelt und somit die Anzahl der versendeten Events reduziert werden.

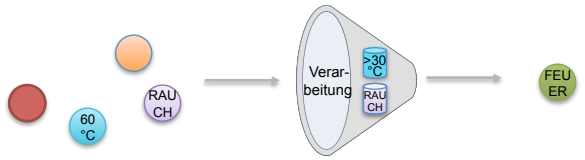


***Abbildung 6.7**: Kombination*

Transformation

$$f_{Transformation}(E) = \{Transformation(e) | e \in E\}$$

Eine Spezialisierung des Kombinationsfilters stellt der Transformationsfilter dar, welcher nur auf ein Event angewendet wird und eine Transformation dessen Informationen ermöglicht. Beispielsweise kann dadurch eine Maßeinheit in eine andere umgewandelt werden, wie in Abbildung 6.8 dargestellt, wobei Grad Celsius in Grad Fahrenheit umgewandelt werden. Ebenso können andere Daten verändert werden, beispielsweise wenn eine Zeitdifferenz zwischen Client und Server vorliegt, kann diese durch einen Transformationsfilter korrigiert werden. Ebenso können hiermit unterschiedliche Zeichenkodierungen bei den diversen Clients automatisch umgewandelt werden.

Aufsplittung

$$f_{Typ_z, Typ_1, \ldots, Typ_n}(E) = \{e^1, \ldots e^n | e \in E \wedge e_{Typ} = Typ_z \wedge e^k_{Typ} = Typ_k\}$$

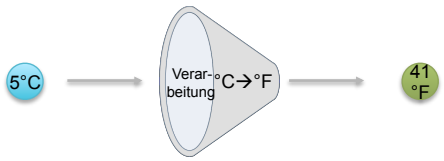


***Abbildung 6.8**: Transformation*

Analog zur Kombination kann bei der Aufsplittung ein Event in mehrere Teilevents separiert werden, wie in Abbildung 6.9 schematisch skizziert. Somit ist es möglich, dass vor dem Versenden durch eine Kombination zusammengefügte Events auf der Empfängerseite wieder in die jeweiligen Einzelevents aufgesplittet werden. Beispielsweise kann man nach Erhalt eines Feuerevents dieses wieder in ein Rauchevent und ein Temperaturevent splitten, in welchem eine Temperatur über 30 Grad Celsius spezifiziert wird. Zudem kann dies verwendet werden, um spezifische Informationen in Form eines Teilevents eines komplexen Events an einen verarbeitenden Dienst weiterzuleiten. Ein solches komplexes Event kann beispielsweise zustande kommen, wenn ein Sensor mehrere Daten erfassen kann, welche in einem Event gebündelt übermittelt werden. Dies kann unter anderem den Austausch von Sensoren in Cyber-Physischen Umgebungen unterstützen, indem die Empfänger der Sensordaten weiterhin die gleichen Events erhalten können, auch wenn der neue Sensor mehr Informationen erfasst.

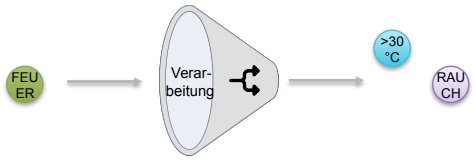


***Abbildung 6.9**: Aufsplittung*

Monitor

e' beschreibt das zuletzt verarbeitete Event.

$$f_{Zeitdifferenz,Fehler}(E) = \begin{cases} Fehler, & e'_{Zeitspempel} \leq Zeit - Zeitdifferenz \\ E, & sonst \end{cases}$$

Entgegen der zuvor beschriebenen Filter und Verarbeitungsmodule, welche Events aussortieren oder deren Inhalt verarbeiten, kann ein Monitoring eingesetzt werden, um Fehlfunktionen zu detektieren. Hierbei wird spezifiziert, in welchem Zeitintervall Events den Filter passieren müssen. Sobald ein Event detektiert wird, wird dieses ohne weitere Verarbeitung weitergeleitet. Gleichzeitig wird der Zeitpunkt erfasst. Wird keins detektiert und das hinterlegte Zeitintervall bis zum letzten erfassten Zeitpunkt überschritten, wird ein entsprechendes Fehlerevent versendet. Beispielsweise kann damit ein defekter Sensor ermittelt werden, wenn dieser keine Events generiert (siehe Abbildung 6.10). Ebenso kann damit erfasst werden, wenn Events bei einem Empfänger erwartet werden, diese jedoch nicht ankommen. Somit

werden potentielle Fehlfunktionen in den entsprechenden Verarbeitungsschritten detektiert und dies entsprechend mittels des Fehlerevents notifiziert.

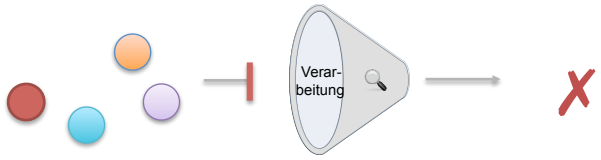


***Abbildung 6.10**: Monitor*

6.2.3 Filter- und Verarbeitungsketten

Neben der Erstellung einzelner Filter und Vorverarbeitungsmodule können diese auch zu Filterketten zusammengeschlossen werden. Eine solche Filterkette wird hierbei in Form eines Graphen modelliert, wobei vorausgesetzt wird, dass es nur einen Startknoten und genau einen Endknoten im Graphen gibt. Zudem muss jeder enthaltene Filter beziehungsweise jedes Vorverarbeitungsmodul vom Startknoten aus erreichbar sein und einen Pfad zum Zielknoten aufweisen. Schließlich dürfen keine Ringschlüsse im Graphen vorkommen, da ansonsten eine unendliche Verarbeitungskette vorliegen würde, welche nicht terminiert. Die Kanten im Graphen spezifizieren den Eventverlauf. Hat ein Filter eine Verknüpfung zu mehr als einem weiteren Filter, so wird das zu übermittelnde Event kopiert und anschließend jeweils eine Kopie an die weiteren Filter übermittelt, um die Verarbeitungsreihenfolge nicht beachten zu müssen. Neben den zuvor erwähnten Filtern und Vorverarbeitungskomponenten können für die Erstellung von Filtergraphen weitere Operatoren, wie „und“, „oder“ und „xor“ eingesetzt werden, um mehrere Stränge im Graphen wieder zusammenzuführen.

Um die Erstellung solcher Filtergraphen zu vereinfachen, wurde im Rahmen dieser Arbeit eine grafische Oberfläche entwickelt, die die Modellierung unterstützt (siehe Abbildung 6.11). Die zur Verfügung stehenden Filter und Vorverarbeitungsmodule werden hierbei in einer Liste angezeigt. Die Selektion eines Eintrags dieser Liste ermöglicht, eine kurze Beschreibung des entsprechenden Filters bzw. des Vorverarbeitungsmoduls zu erhalten. Durch eine einfache Drag’n’Drop-Geste können die Filter anschließend in die eigentliche Modellierungsumgebung gezogen werden, wo sie mittels eines entsprechenden Symbols repräsentiert werden. Ein Interaktionsmenü ermöglicht die Erstellung von Verbindungen zwischen den einzelnen Filtern und Vorverarbeitungsmodulen sowie das Löschen einzelner Elemente des Filtergraphen. Die Selektion eines Filters bzw. Vorverarbeitungsmoduls im Filtergraphen ermöglicht die Anpassung der einzelnen Parameter, wobei sich das Menü an die entsprechenden Filter anpasst. Beispielsweise können hierbei die Evaluationsfunktionen spezifiziert werden oder die zu filternden Eventtypen.

Der resultierende Filtergraph kann in einem XML-Format gespeichert und zur späteren Modifikation geladen werden. Ein Filtergraph kann anschließend entweder direkt bei der Erstellung eines Clients spezifiziert oder mittels eines entsprechenden Events diesem zugesandt werden. Im zweiten Fall wird neben dem eigentlichen Filtergraphen der entsprechen-

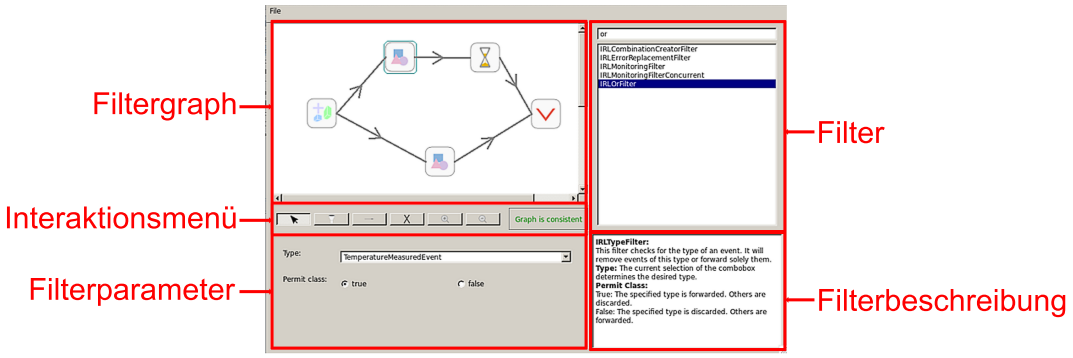


***Abbildung 6.11**: Grafische Oberfläche zur Erstellung und Bearbeitung von Filterketten*

de Adressat und die Information, ob der Filtergraph beim Empfangen von Events oder vor dem Senden verwendet werden soll, an den Client geschickt. Dieser baut automatisch die entsprechende Filterung ein und speichert zwischenzeitlich eintreffende Events. Sobald die Filterkette aktiv ist, werden die zwischengespeicherten Events nach deren Ankunftszeit abgearbeitet. Somit ist eine verlustfreie Adaption von Filterketten zur Laufzeit möglich.

6.3 EBS Architektur

Aufgrund der Eigenschaften von Cyber-Physischen Umgebungen (CPE) (siehe Kapitel 2.2.4) muss eine Kommunikationsinfrastruktur flexibel und möglichst fehlerresistent sein, um darin Einsatz zu finden. Im Speziellen ist die Anzahl der Komponenten, welche in die Kommunikationsinfrastruktur aufgenommen werden sollten, a priori nicht bekannt und kann sich dynamisch verändern. Zusätzlich zur Anzahl der Komponenten erschwert die gegenseitige Beeinflussung während der Kommunikation die Sicherstellung einer korrekten Funktionsweise. Um mögliche Deadlocks und unerwünschte Wartezeiten zu vermeiden, ist daher eine asynchrone Datenübertragung erforderlich. Diese notwendigen Eigenschaften einer Kommunikationsinfrastruktur für CPE wurden bei der Konzeption des *Event Broadcasting Services (EBS)* berücksichtigt. Der EBS ist eine Kommunikationsinfrastruktur, die im Hinblick auf einfache Einrichtung und Verwendung, leichte Erweiterbarkeit und Austauschbarkeit von Komponenten, sowie auf die Möglichkeit einer integrierten Fehlererkennung und -analyse im Rahmen dieser Arbeit entwickelt wurde [Kahl et al., 2012].

Entsprechend der Beschreibung in Kapitel 2.3 können in CPE verschiedene Arten von Kommunikationsverbindungen unterschieden werden. Wie bereits beschrieben, kann entsprechend des Anwendungsfalls eine der jeweiligen Arten von Kommunikationsverbindungen sinnvoll sein. Aus diesem Grund wurde der EBS als modulares System entwickelt, welches je nach Einsatzgebiet unterschiedliche Eigenschaften umfassen kann. Einerseits bietet es die Möglichkeit, als Blackboard-Ansatz verwendet zu werden. Dies ist unter anderem hilfreich, wenn sich die Systeme der CPE nur sporadisch mit der Kommunikationsinfrastruktur verbinden, da sie beispielsweise keinen permanenten Zugriff auf die Kommunikationsinfrastruktur haben. Nach dem Verbinden können die Systeme die

Informationen aus dem Blackboard abrufen und anschließend verarbeiten (pull-Ansatz). Zudem können bei diesem Ansatz die Daten persistent gespeichert werden. Andererseits sind für schnelle Datenübertragungen push-Mechanismen effizienter, da diese direkt von den Zielkomponenten erfasst und verarbeitet werden können (siehe Kapitel 6). Um einen push-Ansatz zu realisieren, kann der EBS auch als Publish/Subscribe-System verwendet werden. Dabei wurde der iROS-Event Heap [Johanson und Fox, 2002] als Vorbild genommen und um weitere Eigenschaften, wie beispielsweise die Modularität, erweitert (siehe Kapitel 3.1.2). Gerade die Server-Client-Struktur ermöglicht es, die Komplexität des Systems zu reduzieren und damit auch eine einfache Einrichtung sowie eine Fehlererkennung zu realisieren. Ebenso kann hierdurch der Kommunikationsfluss zwischen den Systemen von einer zentralen Stelle überwacht und gegebenenfalls reguliert werden. Schließlich kann mittels der im vorherigen Abschnitt beschriebenen Filter- und Verarbeitungsketten eine komplexe Eventverarbeitung (Complex Event Processing, CEP) umgesetzt werden. Die zugrunde liegende Architektur ist in Abbildung 6.12 dargestellt.

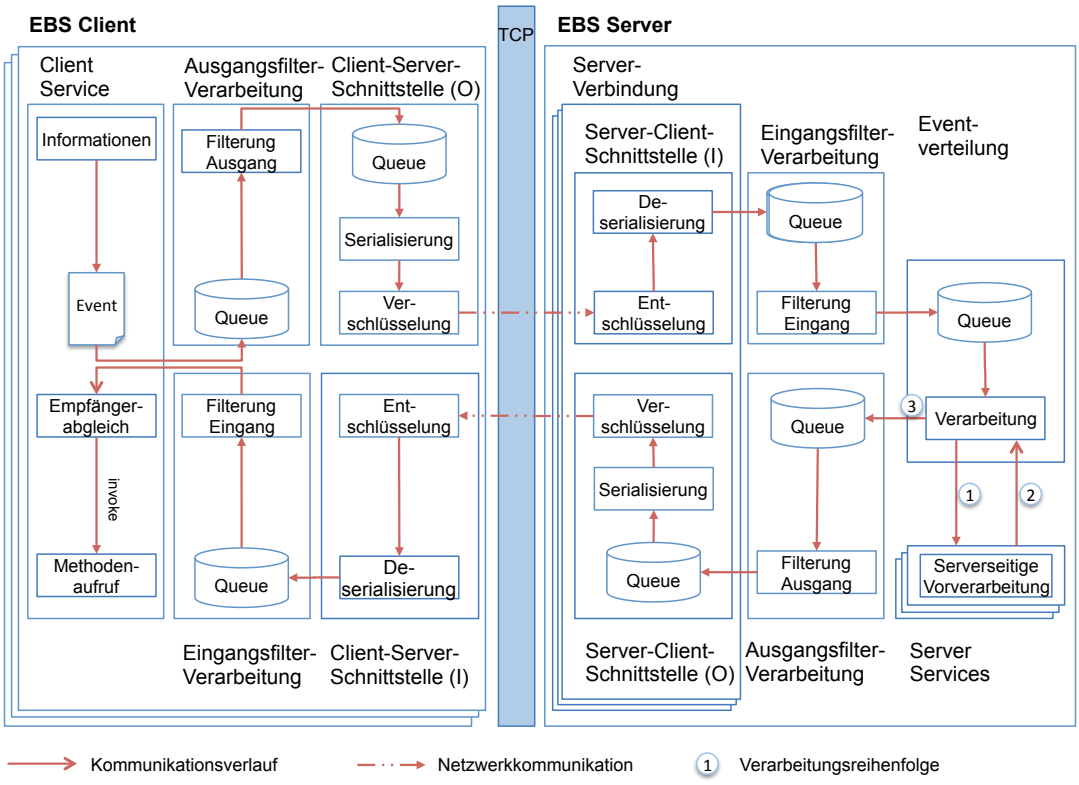


***Abbildung 6.12**: Verarbeitungsablauf und Architektur des Event Broadcasting Service (EBS)*

Im ersten Schritt müssen die zu übermittelnden Informationen in ein Event konvertiert werden. Dies erfolgt, indem spezielle Datencontainer verwendet werden, welche die in Abschnitt 6.1 aufgeführten Informationen zusätzlich zu den Eventtyp-spezifischen Daten beinhalten. Im Anschluss daran erfolgt eine Ausgangsfilterung entsprechend der Konfigura-

tion des Clients gemäß der spezifizierten Filtergraphen. Hierzu werden die Events sowohl vor der Filterung als auch im Anschluss an diese in einer Queue zwischengespeichert, um keine Verluste von Events aufgrund von parallel laufenden Verarbeitungen in den Filtern zu erhalten. Bei der Speicherung wird die spezifizierte Priorität der Events berücksichtigt, so dass Events mit höherer Priorität favorisiert und somit zuerst verarbeitet werden. Beispielsweise führt dies zu einer früheren und damit schnelleren Übertragung wichtiger Nachrichten, selbst wenn die Kommunikationsinfrastruktur ausgelastet ist. Um die Zeit zur Synchronisation zwischen dem Hinzufügen neuer Daten und dem Entnehmen zu reduzieren, ist die Queue-Struktur derart gestaffelt, dass immer zwei parallele Queues existieren, wie in Abbildung 6.13 illustriert. In die eine wird ausschließlich geschrieben und aus der zweiten Queue nur konsumiert. Sind keine weiteren Events in der zweiten Queue enthalten, werden die beiden Queues getauscht, so dass neue Events in die leere geschrieben und die gefüllte abgearbeitet werden kann. Somit wird gerade bei einer hochfrequenten Eventerzeugung eine beschleunigte Datenübertragung erreicht. Ein weiterer Vorteil dieser Queues besteht darin, dass die einzelnen Komponenten unabhängig voneinander funktionieren. Im Speziellen wird dadurch eine asynchrone Übertragung realisiert, bei der Sender und Empfänger voneinander entkoppelt sind.

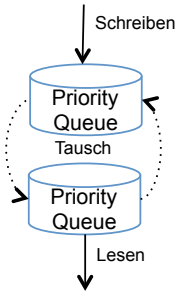


***Abbildung 6.13**: Implementierte Queue-Struktur zum unabhängigen Lesen und Schreiben*

Als weiterer Verarbeitungsschritt der Events steht eine Serialisierung an, welche für eine Übertragung notwendig ist. Hierbei können sowohl stringbasierte als auch binäre Serialisierungsverfahren eingesetzt werden, wobei die Auswahl des passenden Serialisierers in jedem Client eigenständig erfolgen kann. Der Vorteil der flexiblen Serialisierung besteht darin, dass diese an die Gegebenheiten des Systems angepasst wird. Zudem können hiermit unterschiedliche Programmiersprachen und Systeme verwendet werden. Standardmäßig wird eine Serialisierung in die standardisierten stringbasierten Datenformate XML und JSON unterstützt. Für eine binäre Serialisierung kann beispielsweise das Framework Kryo[1] verwendet werden. Während binäre Serialisierungen im Allgemeinen schneller erfolgen und Resultate mit kleineren Datenraten produzieren, können stringbasierte Serialisierungen von Menschen gelesen und auch manuell erstellt werden. Eine JSON-Serialisierung resultiert in einer geringeren Dateigröße als eine XML-Serialisierung und kann ohne weitere Veränderung auch in Form von Javascript interpretiert und somit auf Webseiten verwendet werden.

[1] https://github.com/EsotericSoftware/kryo, Zugriff: November 2014

JSON unterstützt im Gegensatz zur Serialisierung in XML jedoch keine Ringschlüsse in komplexen Events. Zusätzlich zu den bereits integrierten Serialisierungen können weitere ergänzt werden, sofern entsprechende Deserialisierer existieren.

Die serialisierten Events können anschließend verschlüsselt werden, je nachdem ob dies im Event so spezifiziert wurde oder nicht. Hierzu wird eine AES Blockverschlüsselung verwendet. Der entsprechende Schlüssel wird zu Beginn, bei der Registrierung des Clients am Server, durch ein Handshake-Verfahren ausgetauscht, wobei als Schutz dieser Schlüsselübertragung einmalig eine RSA-Verschlüsselung angewendet wird. Das verschlüsselte bzw. unverschlüsselte Event wird daraufhin mittels einer TCP/IP-Verbindung zum Server übertragen. Gemäß der spezifizierten Eventeigenschaften wird dieses gegebenenfalls wieder entschlüsselt und anschließend deserialisiert, so dass eine Verarbeitung des Events auf dem Server erfolgen kann. Hierzu werden die Events erneut in eine Queue geschrieben, von der aus sie über Eingangsfilter vorverarbeitet und gefiltert werden können. Anschließend kann eine weitere Verarbeitung mittels serverseitiger Dienste erfolgen. Ein solcher Dienst wird beispielsweise verwendet, um spezifische Informationen von gewissen Events in eine Datenbank zu schreiben, um eine Eventprotokollierung und den Blackboard-Ansatz zu ermöglichen. Erst nach dieser Verarbeitung werden die Events über eine erneute Zwischenspeicherung an den Ausgangsfilter weitergeleitet, sofern die serverseitige Vorverarbeitung die Events nicht für ungültig erklärt. In diesem Fall werden die Events automatisch ausgefiltert. Dies kann beispielsweise dann eingesetzt werden, wenn die im Event enthaltenen Daten nur für den Server bestimmt sind oder wenn fehlerhafte Inhalte erkannt wurden.

Entsprechend des Serialisierungverfahrens des jeweiligen Clients wird das Event vom Server serialisiert und anschließend gegebenenfalls verschlüsselt, bevor es an die Clients geschickt wird. Dort wird es entsprechend gegebenenfalls wieder entschlüsselt, deserialisiert und mittels eines optionalen Filtergraphen vorverarbeitet. Wie bereits zuvor erwähnt, kann der EBS als Publish/Subscribe-System konfiguriert werden. Somit können sich Clients für bestimmte Eventtypen bei dem Server registrieren. Dies wird bei der serverseitigen Ausgangsfilterung berücksichtigt. Ebenso können Events an bestimmte Empfänger adressiert werden, was ebenfalls Berücksichtigung findet. Soll der Server jedoch auf einem System mit wenig Rechenleistung laufen, könnten solche Verarbeitungen eine zeitnahe Verteilung der Events erschweren. In diesem Fall kann der Server als Broadcasting-Dienst konfiguriert werden, so dass alle eintreffenden Events an alle registrierten Clients weiterverteilt werden. Hierbei erfolgt der Empfängerabgleich auf Seiten der Empfänger. Schließlich werden die Clients automatisch über die eintreffenden Events notifiziert, so dass diese die Daten verarbeiten können.

Durch die Implementierung in Java kann der EBS nativ auch mit mobilen Endgeräten basierend auf Android kommunizieren. Hier erfolgt die Verarbeitung analog zu der auf stationären Rechnern. Für den Verbindungsaufbau zum Server wird wie zuvor beschrieben ein Handshaking-Protokoll zugrunde gelegt. Hierbei wird neben dem AES-Schlüssel auch die Versionsnummer des Systems abgeglichen. Hierzu ist ähnlich zu MQTT [Hunkeler et al., 2008] im EBS eine Versionierung integriert, die zu einer Stabilisierung des Systems beiträgt.

Clients mit einer unterschiedlichen Versionsnummer zu der des Servers werden von diesem nicht akzeptiert. Dies garantiert den stabilen Verlauf des Systems, da mit älteren Clients die Kommunikation zum Server gestört sein kann. Eine Versionsnummeränderung erfolgt ausschließlich bei wichtigen Veränderungen, beispielsweise wenn die Grundstruktur der Events sich geändert hat und es dadurch zu Komplikationen beim Verständnis der Events kommen kann. Gleichzeitig wird bei dieser Initialisierung das verwendete Serialisierungsverfahren des Clients an den Server kommuniziert. Ebenso enthält die Anfangskommunikation Informationen zur ID des Clients sowie zu den Eventtypen, für welche er sich registriert. Die Registrierung der Clients und deren Interessenbekundungen werden bei dem Server gespeichert und können dort über eine entsprechende Schnittstelle abgerufen werden.

Beim Erstellen eines neuen Clients wird zudem die Session-ID generiert, welche aus dem Namen des Clients und dem Zeitstempel des Starts besteht. Wird eine Verbindung aufgrund einer Störung unterbrochen, so wird diese bei nächstmöglicher Gelegenheit automatisch durch den Client wieder aufgebaut. Dadurch wird eine neue Session-ID generiert, so dass solche Neustarts automatisch auch von den Empfängern von Nachrichten erkannt werden, um gegebenenfalls entsprechend darauf reagieren zu können. Während des Neustarts ankommende Events werden zwischengespeichert und direkt nach der erneuten Verbindung des Clients zum Server von diesem übermittelt. Ist die Registrierung des Clients am Server erfolgreich verlaufen, wird vom Server in einem zentralen Repository nachgeschaut, ob ein Filter für den entsprechenden Client zur Verfügung steht und dieser gegebenenfalls an den Client gesandt und dort dynamisch eingebunden. Veränderungen an Filterketten können unabhängig vom Zielclient mit dem im vorherigen Abschnitt präsentierten Tool erfolgen. Die angepassten Filterketten werden anschließend als Event an die Empfänger versendet und dort in die Verarbeitungskette eingebaut. Gleichzeitig kann bei diesem Event spezifiziert werden, ob der Filter bei einem späteren Neustart des Clients wiederverwendet werden soll oder nur für die aktuelle Session gültig ist. Im ersten Fall wird dies beim Server entsprechend interpretiert und der neue Filtergraph im Repository hinterlegt, damit dieser bei der nächsten Verbindung des Clients wieder automatisch dem Client zur Verfügung gestellt werden kann.

Für eine Verbindung mehrerer CPE wurde auch die Möglichkeit des Datenaustauschs zwischen mehreren Servern realisiert. Hierzu kann sich ein spezieller Client mit beiden Servern verbinden und den Datenaustausch regeln. Durch die Verwendung der Filter kann hierbei auch genau spezifiziert werden, welche Daten zwischen den beiden CPE ausgetauscht werden sollen. Durch die Wahl der Filter und serverseitigen Vorverarbeitungsmodule ist eine Modularität des EBS realisiert, so dass dieser je nach Verwendungszweck individuell angepasst werden kann. Beispielsweise kann durch die Speicherung aller empfangenen Events in einer zentralen Datenbank ein Blackboard-Ansatz realisiert werden. Zusätzlich zu den Eventparametern wird hierbei hinterlegt, ob die Events in Bearbeitung sind und welche bereits von einem Client konsumiert wurden, wenn mehrere Systeme parallel die gleichen Berechnungen durchführen können. Zudem kann diese Zwischenspeicherung für spätere Analysen verwendet werden, beispielsweise zur Fehleranalyse, indem die protokollierten Kommunikationspfade ausgewertet werden. Gleichzeitig kann dies zur Aufzeichnung von Eventsequenzen hergenommen werden, welche zu einem späteren Zeitpunkt erneut abge-

spielt werden sollen. Um die Eventhistorie abzurufen und Zugriff hierauf zu ermöglichen, steht ein Datenbankinterface zur Verfügung.

Ein weiterer Vorteil des gewählten Verteilungsmechanismus samt der Filterfunktionalität ist, dass bei einer Weiterentwicklung des Systems bestehende Softwarekomponenten weiterverwendet werden können. Teilweise gibt es Software, die in Form einer Blackbox funktioniert, welche gewisse Eingaben in Form von parametrisierten Funktionsaufrufen erwartet und anschließend entsprechende Resultate ausgibt, ohne dass der jeweilige Berechnungsalgorithmus bekannt ist. Solche Systeme erlauben keine Anpassung der internen Schnittstellen, was es unmöglich macht, die gleiche Funktionalität mittels anderer Anfragen zu erhalten. Zur Ansteuerung solcher Software können die Events interpretiert, die notwendigen Parameter daraus ermittelt und damit die entsprechenden Funktionen ausgeführt werden. Wird an der Eventstruktur etwas verändert, so kann über eine einfache Anpassung an den entsprechenden Verarbeitungsfiltern dies kompensiert werden, so dass ohne eine Anpassung der zuvor definierten Schnittstelle der Dienst trotz neuer Struktur problemlos weiter funktioniert. Ebenso können dank einer solchen Infrastruktur bestimmte Frequenzen als Eingangssignale realisiert werden, was auch eine Simulation und damit auch eine Testumgebung ermöglicht. Beispielsweise kann in einem vordefinierten Zeitrhythmus das zuletzt empfangene Event weitergeleitet werden. Entgegen einer Reduktion von Events kann sowohl vor dem Versenden als auch nach dem Empfang eine höhere Anzahl an Events generiert werden, welche von den verarbeitenden Systemen entsprechend interpretiert werden muss.

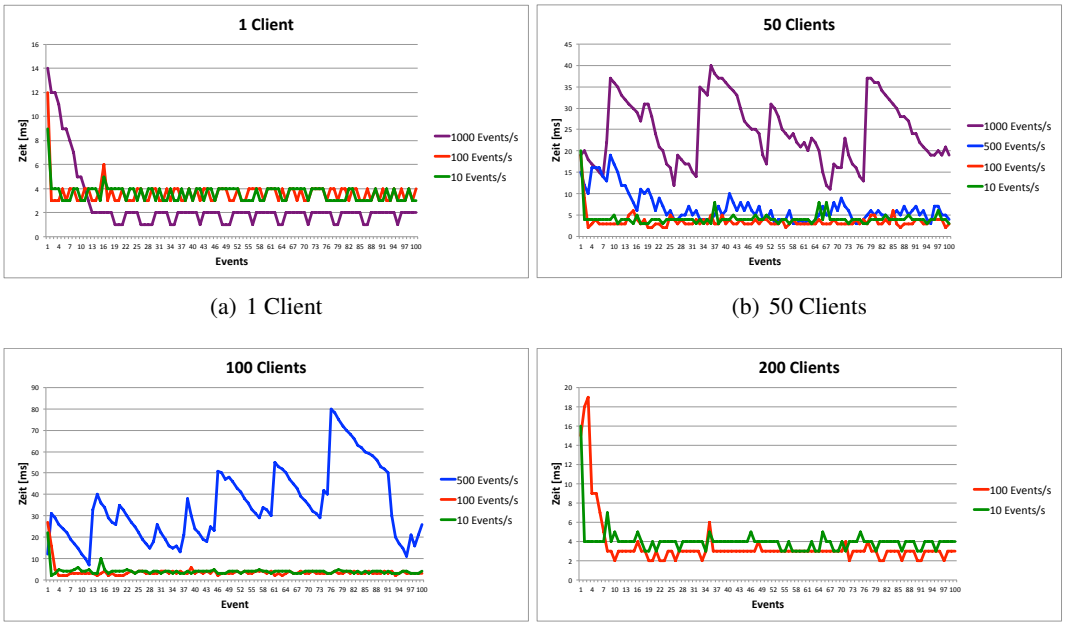
6.4 Evaluation

Zur Validierung der Implementierung des EBS wurde eine Evaluierung hinsichtlich quantifizierbarer Parameter durchgeführt, in Bezug auf Geschwindigkeit (siehe Abschnitt 6.4.1), abhängig von der Anzahl der registrierten Clients, der Anzahl der versendeten Events und der Größe des in einem Event enthaltenen Datenvolumens (siehe Abschnitt 6.4.2). Des Weiteren wurden die Veränderungen des Eventdurchsatzes nach Aktivierung von Filtern am Beispiel einer Temperaturmessung evaluiert (siehe Abschnitt 6.4.3).

6.4.1 Geschwindigkeit

Die Geschwindigkeit bei der Übertragung von Events im EBS hängt von drei Faktoren ab: Anzahl der Empfänger, Anzahl der versendeten Events und verwendetes Serialisierungsverfahren. Aufgrund der gewählten Architektur werden die Informationen vor dem Versenden durch den erzeugenden Client serialisiert, anschließend mittels einer TCP/IP-Übertragung an den Server geschickt, dort deserialisiert, gegebenenfalls vorverarbeitet und anschließend den Empfängern zugesendet. Vor dem Versenden durch den Server wird das Event wiederum mit dem vom Empfängerclient spezifizierten Serialisierungsverfahren serialisiert und auf Seite des Empfängers erneut deserialisiert. Somit umfasst das Übertragen eines Events vom Sender bis zum Empfänger je zwei Serialisierungs- und Deserialisierungsschritte und zwei Übertragungen mittels TCP/IP. Um die entsprechende Übertragungsgeschwindigkeit zu evaluieren, wurden drei Rechner mittels einer Ethernet-Verbindung zusammengeschlossen. Auf

dem einen Rechner wurde der EBS-Server gestartet. Dabei handelte es sich um ein MacBook Air mit einem Intel Core i5 (1.8 GHz) Prozessor. Der zweite Rechner, ebenfalls ein MacBook Air mit Intel Core i5 (1.7 GHz), umfasste einen Dienst, der die Übertragungszeit von Events misst. Hierzu konnte der Dienst so konfiguriert werden, dass er innerhalb einer Sekunde n Events gleichmäßig versendet. Als Eventtyp wurde ein Informationsevent verwendet, welches neben den Standardinformationen eines Events eine textuelle Nachricht beinhaltet. Beim Empfangen von Events wurde der aktuelle Zeitstempel mit dem im Event enthaltenen Erstellzeitstempel verglichen und darüber die Dauer der Eventübertragung erfasst und protokolliert. Auf dem dritten Rechner (MacBook, Intel Core 2 Duo - 2.53 GHz) wurden parallel m Clients erstellt, die sich mit dem Server verbanden, keine Events versendeten, jedoch alle Events empfingen, ohne diese weiter zu verarbeiten. Sobald alle Clients sich mit dem Server verbunden hatten, wurde mit der Eventversendung und der Messung begonnen.



(a) 1 Client

(b) 50 Clients

(c) 100 Clients

(d) 200 Clients

***Abbildung 6.14**: Übertragungsdauer in Abhängigkeit von der Anzahl der versendeten Events und der verbundenen Clients*

Abbildung 6.14 zeigt die Übertragungsdauer der Events in Abhängigkeit von der Anzahl der pro Sekunde versendeten Events und der parallel verbundenen Clients. Hierbei wurde der Kryo-Serialisierer bei allen Clients verwendet (siehe Abschnitt 6.3). Die Events wurden nicht direkt adressiert und daher an alle mit dem Server verbundenen Clients versendet. Dies bedeutet, dass der Server bei mehreren verbundenen Clients das Event mehrfach serialisieren und versenden muss. Die multiple Serialisierung funktioniert aufgrund der Entkapselung in einzelne Threads größtenteils parallel. Bei der ersten Erstellung der Events werden die Serialisierer aktiviert, welche daraufhin durch die Implementierung von Java automatisch opti-

miert werden. Aus diesem Grund sind die Übertragungszeiten zu Beginn der Messung höher. In Abbildung 6.14(a) ist die Übertragungszeit bei nur einem registrierten Client dargestellt, welcher gleichzeitig die zeitliche Analyse durchführt. Wie zu erkennen ist, beträgt die Übertragungszeit eines Events in etwa 3 - 4 Millisekunden. Die Reduktion der Übertragungszeit bei 1000 Events pro Sekunde auf 1 - 2ms ist hierbei durch die verwendete Queue-Struktur zu erklären (siehe Abschnitt 6.3). Durch die hohe Anzahl an Events wird die eine Queue immer gefüllt, während die andere Queue abgearbeitet wird, bevor die beiden getauscht werden. Daher ist die Verarbeitung immer aktiv, was zu einer geringen Leistungssteigerung führt. Beim Versenden von Events in unregelmäßigen Zeitabständen würde dieses Phänomen vermutlich nicht auftreten. Abbildung 6.14(b) stellt die Übertragungsgeschwindigkeit bei 50 verbundenen Clients dar. Hierbei ist zu erkennen, dass bei 1000 Events/s die Übertragung zeitweise überlastet ist und daher die Eventübertragung bis zu 40ms beträgt. Bei bis zu 500 Events/s beträgt die Übertragungszeit in der Regel unter 10ms. Abbildung 6.14(c) stellt die Übertragungsgeschwindigkeit bei 100 verbundenen Clients dar, welche die versendeten Events empfangen. Da bereits bei 50 Clients eine hohe Auslastung bei der Eventübertragung festgestellt wurde, wurde keine Messung mit 1000 Events/s bei 100 Clients durchgeführt. Zudem ist bereits eine starke Auslastung bei 500 Events/s festzustellen. Die Schwankungen bei der Übertragungszeit sind hierbei durch die Funktionsweise der Queue-Struktur, welche die Events sammelt und nach dem Austausch der beiden Queues abarbeitet in Kombination mit den vielen Threads zu erklären. Die zuletzt in diese Queue aufgenommenen Events werden daher schneller verarbeitet als die zuerst aufgenommenen, da die Liste der zu verarbeitenden Events länger wird. Schließlich wurde ein Test mit 200 verbundenen Clients durchgeführt, dessen Ergebnisse in Abbildung 6.14(d) abgebildet sind. Hierbei wurde ebenfalls auf die Versendung von 500 Events/s aufgrund der bereits schlechten Performanz bei 100 Clients verzichtet. Bei bis zu 100 Events/s werden die Events immer noch mit einer durchschnittlichen Zeit von 3 - 4ms übertragen.

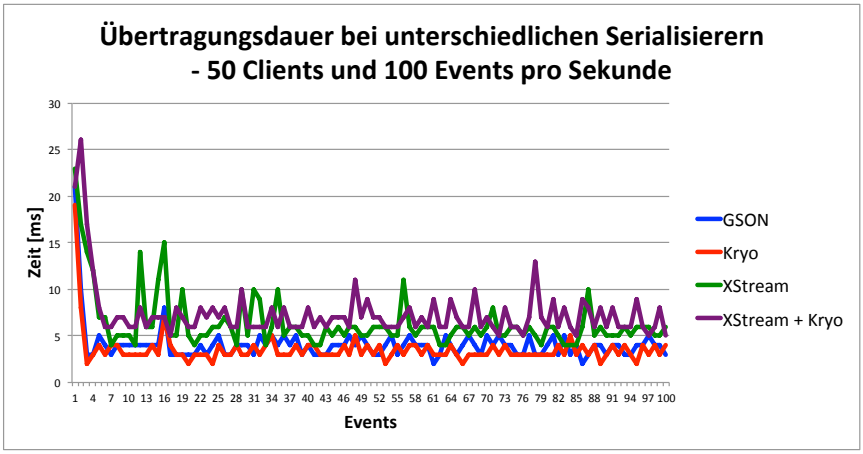
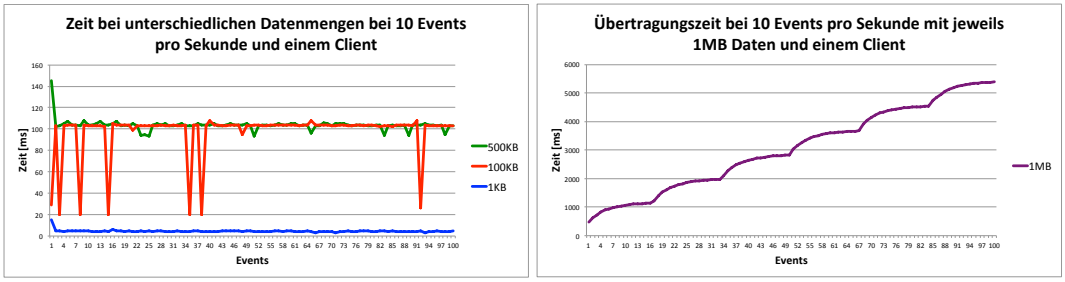


***Abbildung 6.15**: Übertragungsdauer unter Verwendung unterschiedlicher Serialisierer*

Da im EBS zur Übertragung jedes Event einmal vor dem Versenden vom Client an den Server und jeweils einmal pro Empfänger vom Server serialisiert werden muss, wurde eine Analyse der Übertragungsdauer bei Verwendung unterschiedlicher Serialisierer getestet.

Das Ergebnis ist in Abbildung 6.15 dargestellt. Hierbei wurden 50 Clients an den Server angeschlossen und 100 Events/s übertragen, da dieses Setup im zuvor beschriebenen Test eine ausgewogene Mischung an Auslastung und Performanz aufgezeigt hat. Als Serialisierer wurden Kryo[2], GSON[3] und XStream[4] verwendet. Kryo ist hierbei für Java spezialisiert und serialisiert in ein binäres Format. GSON ist ein Serialisierer, welcher als Ausgabe das JSON-Format liefert. XStream konvertiert das Objekt in XML-Syntax. Sowohl bei der Serialisierung mit Kryo als auch mit GSON liegt die Zeit zur Übertragung eines Events vom Senden bis zum Empfangen in der Regel bei unter 5ms, wobei die Kryo-Serialisierung im Durchschnitt minimal schneller ist. XStream hingegen ist etwas langsamer, wobei die Übertragungszeit bis auf wenige Ausreißer bei unter 10ms liegt. Da jeder Client einen anderen Serialisierer verwenden kann und der Server dies vor der Übertragung entsprechend berücksichtigt, wurde zusätzlich ein Testdurchlauf durchgeführt, bei dem der Dienst zum Messen der Kommunikationsgeschwindigkeit die XStream-Serialisierung und die anderen 49 Clients die Kryo-Serialisierung verwendeten. Die Ergebnisse zeigen, dass auch diese Kombination in der Regel eine Übertragungszeit von unter 10ms aufweist.

6.4.2 Durchsatz



(a) 1KB, 100KB und 500KB

(b) 1MB pro Event

***Abbildung 6.16**: Übertragungszeiten in Abhängigkeit vom Datenvolumen bei 10 Events pro Sekunde und einem verbundenen Client*

Einen zweiten kritischen Punkt bei der Übertragungsgeschwindigkeit stellt die Größe der zu übertragenden Events dar. Hierzu wurde eine Übertragung von 10 Events/s mit unterschiedlichen Datenmengen getestet. Es wurde nur der Client mit dem Server verbunden, der gleichzeitig die Messung durchführte. Abbildung 6.16(a) zeigt die Übertragungszeiten pro Event mit einer Datenmenge von 1KB, 100KB und 500KB je Event. Als Serialisierer wurde Kryo verwendet. Während die Übertragungsgeschwindigkeit bei 1KB bei 4 - 5ms liegt, benötigt die Übertragung von 100KB und 500KB großen Events knapp über 100ms. Erhöht man das im Event enthaltene Datenvolumen auf 1MB, resultiert dies im Testsetup in einer Übertragung von 10 MB pro Sekunde in beide Richtungen, Senden und Empfangen. Hierbei ist festzustellen, dass der EBS mit der Übertragung der Events überlastet ist und daher

[2]https://github.com/EsotericSoftware/kryo, Zugriff: November 2014
[3]https://code.google.com/p/google-gson/, Zugriff: November 2014
[4]http://xstream.codehaus.org, Zugriff: November 2014

die Übertragungszeit sukzessive steigt (siehe Abbildung 6.16(b)). Diese Auswertung zeigt, dass kleinere Datenvolumen problemlos über die Eventstruktur versendet werden können, wodurch eine Übertragung von Bildern und Dokumenten möglich ist. Bei größeren Datenmengen muss das Intervall für den Versand von Events entsprechend angepasst werden, um die Netzwerkstruktur nicht zu überlasten. Sobald eine Queue komplett befüllt ist und keine weiteren Events aufnehmen kann, gehen diese verloren. Dadurch wird das restliche System nicht gestört und kann weiterlaufen. Gleichzeitig wird in diesem Fall eine Benachrichtigung ausgegeben, so dass eine verantwortliche Person die Sachlage untersuchen und den Grund der Überfüllung ermitteln kann.

6.4.3 Filtereinsatz

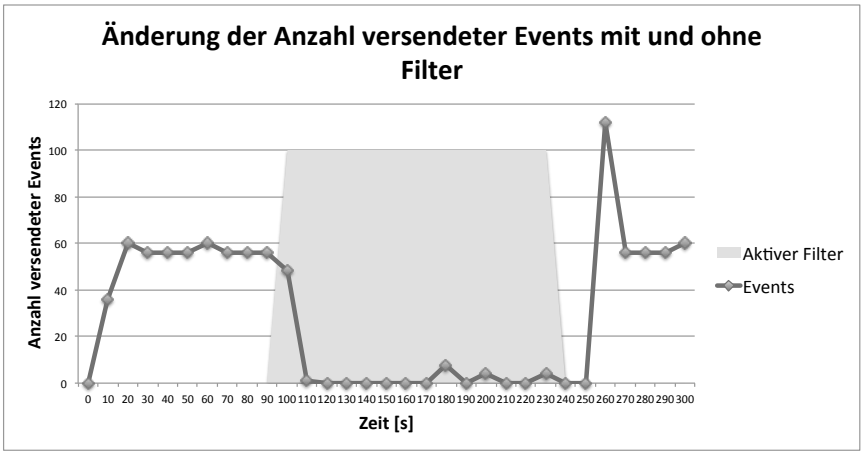


***Abbildung 6.17**: Veränderung der Anzahl versendeter Events bei (De-) Aktivierung eines dynamischen Inhaltsfilters*

Als letzte Evaluation wurde der Einsatz von Filtermechanismen untersucht. Abbildung 6.17 zeigt die Veränderung der Anzahl versendeter Events bei Aktivierung und anschließender Deaktivierung eines dynamischen Inhaltsfilters. Nach der Initialisierungsphase des Sensors versendet dieser ca. 6 Events/s, welche die aktuelle Temperatur beinhalten und eine Referenz zum Sensor, der die Messung durchgeführt hat. Der verwendete Temperatursensor hat eine Messgenauigkeit von 0,5°C. Im Test hat ein zweiter Dienst alle vom Sensor versendeten Events empfangen und alle 10 Sekunden die zwischenzeitlich empfangenen Events gezählt und diesen Wert protokolliert. Laut der gemessenen Daten sendet der Temperatursensor ca. 60 Events pro 10 Sekunden. Nach insgesamt 100 Sekunden wurde ein dynamischer Inhaltsfilter als Event versendet, welcher beim Temperaturdienst als Ausgangsfilter eingebunden wurde. Dieser ließ nur Events passieren, bei denen die Temperatur vom zuletzt versendeten Wert unterschiedlich war. Dadurch reduzierte sich die Anzahl an gesendeten Events auf 0 Events. Lediglich kleinere Temperaturschwankungen von 0.5 Grad führten zur Versendung vereinzelter Events. Nach 240 Sekunden wurde erneut ein Event geschickt, welches den zuvor instantiierten Filter deaktivierte. Während der Filter deaktiviert wurde, wurden die vom Sensor erfassten Temperaturevents zwischengespeichert. Nach der

Deaktivierung wurden diese Events zusätzlich zu den in diesem Zeitraum generierten Events abgearbeitet und versendet, weshalb eine Spitze bei der Eventversendung bei 260 Sekunden in Abbildung 6.17 abzulesen ist. Des Weiteren ist zu erkennen, dass anschließend der Dienst mit gewohnter Frequenz von 6 Hertz weiter Events sendete.

6.5 Dual Reality Framework

Gerade im Zusammenspiel mit dem Dual Reality Management Dashboard (DURMAD) weist der EBS einen großen Mehrwert auf. Beide Systeme zusammen stellen hierbei ein *Dual Reality Framework* dar, welches eine Verschmelzung von realer und virtueller Umgebung ermöglicht. Durch die Verwendung des EBS bereits als Datenübertragung zwischen den Komponenten einer CPE werden die übertragenen Events ohne die Erstellung spezieller Schnittstellen automatisch in DURMAD integriert. Somit werden alle erfassten Sensorinformationen auch gleichzeitig in das virtuelle Modell transferiert, dort interpretiert und entsprechend grafisch repräsentiert. Dies stellt eine Hauptfunktionalität von DR-Applikationen dar, die im Monitoren von Umgebungen besteht, indem alle durch Sensoren erfassten Kontextveränderungen in ein virtuelles Modell transferiert und dort entsprechend verarbeitet werden.

Gleichzeitig kann mittels einer interaktiven Visualisierung eine Steuerung der in der Umgebung befindlichen Aktuatoren erfolgen. Für die Kommunikation zwischen Modell und Umgebung werden in der Regel spezielle Schnittstellen benötigt. Dies hat zur Folge, dass jede Änderung der Eventstrukturen oder der Eventtypen eine zusätzliche Anpassung der entsprechenden Schnittstelle erforderlich macht. Durch die Verwendung des DR-Frameworks hingegen werden keine spezifischen Schnittstellen benötigt, da DURMAD sich als Client am EBS registriert und somit alle versendeten Events direkt empfängt und verarbeiten kann. Somit werden mögliche Anpassungen der Struktur sowie die initiale Einrichtung vereinfacht. Neben der einfachen Verwendung ist ein möglichst fehlerfreier Betrieb der CPE eine notwendige Eigenschaft. Um potentielle Übertragungsschwierigkeiten frühzeitig erkennen und Fehleranalysen durchführen zu können, werden Unterstützungsfunktionen angeboten, wie beispielsweise die dynamische Informationsflussvisualisierung im Modell (siehe Abschnitt 6.5.1). Sollen Simulationen eingebunden werden, müssen weitere Vorkehrungen getroffen werden, um einen realen Betrieb nicht zu beeinflussen. Hierzu kann das DR-Framework in verschiedene Modi versetzt werden, welche sich hinsichtlich des Kommunikationsflusses zwischen der realen Umgebung und dem Modell unterscheiden (siehe Abschnitt 6.5.2). Hierbei stellt im Speziellen die Synergie zwischen realen Sensordaten und einem virtuellen 3D-Modell einen Mehrwert dar. Zusätzlich zu Simulationen ist oftmals eine autonome Reaktion auf Kontextveränderungen erwünscht. Hierzu wurde das in Kapitel 5.4.1 beschriebene Regelsystem in das DR-Framework integriert (siehe Abschnitt 6.5.3) und die in Kapitel 5.1.3 beschriebene Semantik auf die Kommunikationsinfrastruktur ausgeweitet, um eine automatisierte Reaktion auf Kontextveränderungen zu ermöglichen.

6.5.1 Informationsflussvisualisierung

Eine potentielle Fehlerquelle in verteilten Systemen liegt in der Kommunikation zwischen den einzelnen Komponenten, z.B. aufgrund einer falschen Konfiguration. Um den Einblick in die internen Kommunikationswege zu unterstützen und die Validierung der Kommunikationsinfrastruktur sowie das Testen und Finden von Fehlern zu ermöglichen, bedarf es geeigneter Darstellungsformen für den Kommunikationsfluss [Marquardt et al., 2010]. Beispielsweise kann dies anhand einer zweidimensionalen Karte der Umgebung erfolgen, in welcher die entsprechenden Sensoren, Aktuatoren und verarbeitenden Dienste eingezeichnet sind, wie es in *VEE (Visual Environment Explorer)* umgesetzt wurde [Marquardt et al., 2010]. Die Kommunikationspfade werden hierbei durch gerichtete Kanten zwischen den einzelnen Komponenten symbolisiert. Neben der reinen Darstellung von Kommunikationspaketen zwischen den Komponenten kann in VEE auch eine Analyse in Form von Dashboards erfolgen, bei welcher versendete Events in tabellarischer Form oder akkumuliert in einem Liniendiagramm dargestellt werden.

Analoge Ansätze wurden auch bei der Entwicklung des DR-Frameworks im Rahmen dieser Arbeit umgesetzt. Während die Dashboardfunktionalität mittels der beschriebenen grafischen Darstellungen erfolgt (siehe Kapitel 5.3.4), kann der Kommunikationsfluss im 3D-Modell grafisch repräsentiert werden, wie in Abbildung 6.18 dargestellt. Der Server wird hierbei durch eine Position in der Mitte des Modells repräsentiert. Anhand der Subskriptionen ist bekannt, welcher Client welche Eventtypen zugestellt bekommt. Die Positionen der einzelnen Clients im Raum werden hierbei durch das erstellte Modell in Verbindung mit den semantischen Zuweisungen anhand der Ontologie ermittelt (siehe Kapitel 5.1.3). Der Eventfluss zwischen Client und Server wird durch eine Verbindungslinie zwischen diesen repräsentiert, wobei die Kommunikationsrichtung durch konische Pfeile dargestellt wird. Für die Informationsflussvisualisierung können einzelne Eventtypen spezifiziert werden oder ein zu überwachender Client. Werden Probleme in der Kommunikation festgestellt, kann mittels dieser Darstellung verifiziert werden, ob die Ursache bei dem Client liegt oder bereits die Datenzufuhr nicht erwartungsgemäß erfolgt. Hierbei kann entweder ein Eventtyp oder ein Client spezifiziert werden, welcher visuell überwacht werden soll.

6.5.2 Verschmelzung von Real und Virtuell

Lokationsbasierte Informationssysteme in Verbindung mit dem aktuellen Kontext spielen in CPE eine wichtige Rolle (siehe Kapitel 2.1.4). Aus diesem Grund wurden spezielle Kommunikationsinfrastrukturen hierfür entwickelt, wie beispielsweise *Pervaho*, das eine lokationsbasierte Publish/Subscribe-Middleware für mobile Applikationen darstellt [Eugster et al., 2008]. Durch die Verschmelzung von realen Sensordaten und einem virtuellen 3D-Modell können erfasste Informationen mit weiteren Kontextdaten angereichert werden. Beispielsweise kann über einen Sensor, der Objekte in einem Bereich erfassen kann, und ein existierendes Positionsmodell eine potentielle Position eines Objekts kalkuliert und diese an andere Komponenten kommuniziert werden. Prototypisch wurde eine solche Positionsermittlung mit Hilfe von RFID-Sensoren realisiert und in DURMAD integriert. Neben der Bestimmung von Positionen kann das Modell auch verwendet werden, um

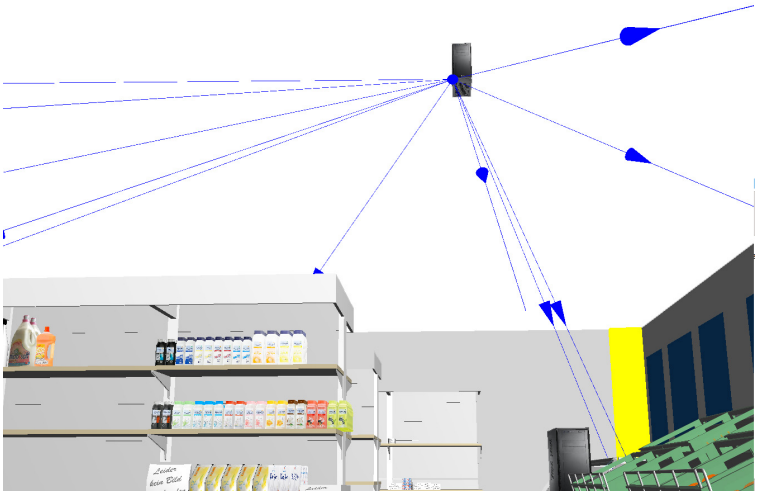


***Abbildung 6.18**: Kommunikationsflussvisualisierung von Events zwischen Clients und Server in* DURMAD

weitere Kontextinformationen zu akquirieren. So können Beziehungen zwischen Objekten und räumliche Verhältnisse zueinander ermittelt werden. Beispielsweise kann bestimmt werden, ob eine Person ein gewisses Objekt von ihrer aktuellen Position aus sehen kann oder die Sicht darauf durch ein anderes Objekt verdeckt ist. Ebenso kann man hierüber alle Objekte in einem gewissen Umkreis zu einer bestimmten Position ermitteln. Diese Lokations- und Kontextinformationen können anschließend kommuniziert und von weiteren Diensten analysiert und verarbeitet werden.

Aufgrund der Möglichkeit, Events in einer Datenbank zu speichern, können die einzelnen Kontextveränderungen der CPE aufgezeichnet werden. Dies ermöglicht ein erneutes Abspielen der Aktionen zu einem späteren Zeitpunkt in Form von Eventabfolgen. Die Events werden dabei entsprechend ihrer ursprünglichen Versendung in der gleichen zeitlichen Abfolge ab einem beliebigen Startzeitpunkt verschickt. Zudem kann hierbei das Senden jederzeit unterbrochen werden oder Event für Event unabhängig vom Zeitpunkt erfolgen. Somit ist eine einfache Möglichkeit zur Fehlerdetektion geschaffen, welche an die Vorgehensweise von Debugging-Tools angelehnt ist, die aus der Programmierung bekannt sind. Gerade in Kombination mit der Informationsflussvisualisierung können dadurch potentielle Fehlerquellen einfach detektiert werden. Um die reale Umgebung während der Abspielphase nicht zu beeinträchtigen, kann im Einzelfall explizit spezifiziert werden, ob die simulierten Events entweder ausschließlich in die reale bzw. virtuelle Umgebung oder parallel in beide kommuniziert werden sollen. Dies erfolgt mit Hilfe von entsprechend vorgeschalteten Filtern.

Zusätzlich zur zuvor beschriebenen Simulationsmöglichkeit können weitere Simulationen durchgeführt werden, wobei eine Beeinflussung der physischen CPE gewollt sein kann, aber auch unterbunden werden könnte. Aus diesem Grund werden zur Verwendung

von Simulationen im DR-Framework insgesamt drei Modi zur Verfügung gestellt. Beim ersten Modus werden ausschließlich real erfasste Informationen, welche aus der CPE kommen, verarbeitet. Zur Ansteuerung von Aktuatoren können basierend auf Interaktionen mit dem 3D-Modell spezielle Kontrollevents versendet werden, beispielsweise zum Öffnen von Türen. Die zweite Variante ermöglicht eine reine Simulation im Virtuellen, so dass alle Informationen von der realen Umgebung ausgefiltert und nicht in das Modell kommuniziert werden. Ebenso erfolgt kein Informationstransfer aus dem Modell in die CPE. Somit existieren hierbei zwei separate Systemkreisläufe. Da oftmals Daten aus einer allgemein zugänglichen Datenbank benötigt werden, welche anhand von Interaktionen und Eventverarbeitungen angepasst wird, wird diese zu Beginn einer solchen Simulation in eine lokale Datenbank kopiert (DURMAD-*Datenbank*). Anschließend operiert DURMAD sowie die Simulationen auf dieser lokalen Kopie, um keine Beeinflussung der realen Umgebung nach sich zu ziehen. Hierbei wird das gleiche Datenbankinterface wie beim Original verwendet, so dass keine Anpassung von Applikationen erfolgen muss und diese wie in der Realität agieren können. Dies vereinfacht zudem eine Überführung einer Simulation in einen realen Dienst, da hierzu keine programmiertechnischen Veränderungen erfolgen müssen. Um keine Synchronisationsprobleme bei gleichzeitigem Schreibzugriff auf die Datenbank zu erhalten, gibt es einen Synchronisationsdienst (*DB Sync*), welcher die Anpassung der Datenbankeinträge übernehmen kann. Dieser ist derselbe, der als serverseitiger Dienst am EBS registriert ist und dort unter anderem die Protokollierung der versendeten Events realisiert. Schließlich gibt es noch den komplexesten Simulationsmodus, welcher eine Vermischung zwischen realen Informationen und Simulationen ermöglicht. Hierbei werden die Events, welche zwischen der realen und der virtuellen Umgebung versendet werden, durch entsprechende Filter sondiert. Dies kann unter anderem zum Testen neuer Dienste eingesetzt werden, welche von der Umgebung erfasste Sensordaten als Eingabe erhalten, deren Resultate jedoch zu Testzwecken ausschließlich im virtuellen Modell widergespiegelt werden. Ebenso kann ein Mehrbenutzersystem in der realen Umgebung getestet werden, indem durch eine Simulation im Modell unterschiedliche Benutzerinformationen generiert werden. Um durch Simulationen veränderte Datenbankeinträge nicht unmittelbar in die reale Datenbank aufzunehmen, wird auch in diesem Modus eine lokale Datenbankkopie zu Beginn der Simulation angelegt. Die drei Modi sind in Abbildung 6.19 schematisch dargestellt.

Aufgrund dieser drei Modi erleichtert das DR-Framework die Integration von Simulationen neben der reinen Erfassung von interaktionsgetriebenen Veränderungen. Durch eine Registrierung für alle Eventtypen werden alle Kontextveränderungen, welche durch Sensoren der CPE erfasst und kommuniziert werden, auch vom DURMAD empfangen und können bei der virtuellen Repräsentation der Umgebung eine Berücksichtigung finden. Diese Struktur ermöglicht somit auch, den Kontext der Umgebung zu erfassen und im Modell zu repräsentieren. Je mehr dieser Events entsprechend verarbeitet werden, desto stärker ist die Synchronisation zwischen der realen und der virtuellen Komponente, was sich im Grad der implementierten DR widerspiegelt (siehe Kapitel 4.2.4). Interaktionen mit dem Modell können wiederum als Events interpretiert werden, welche über die Kommunikationsinfrastruktur an die Umgebung geschickt werden und dort eine Ansteuerung von Aktuatoren nach

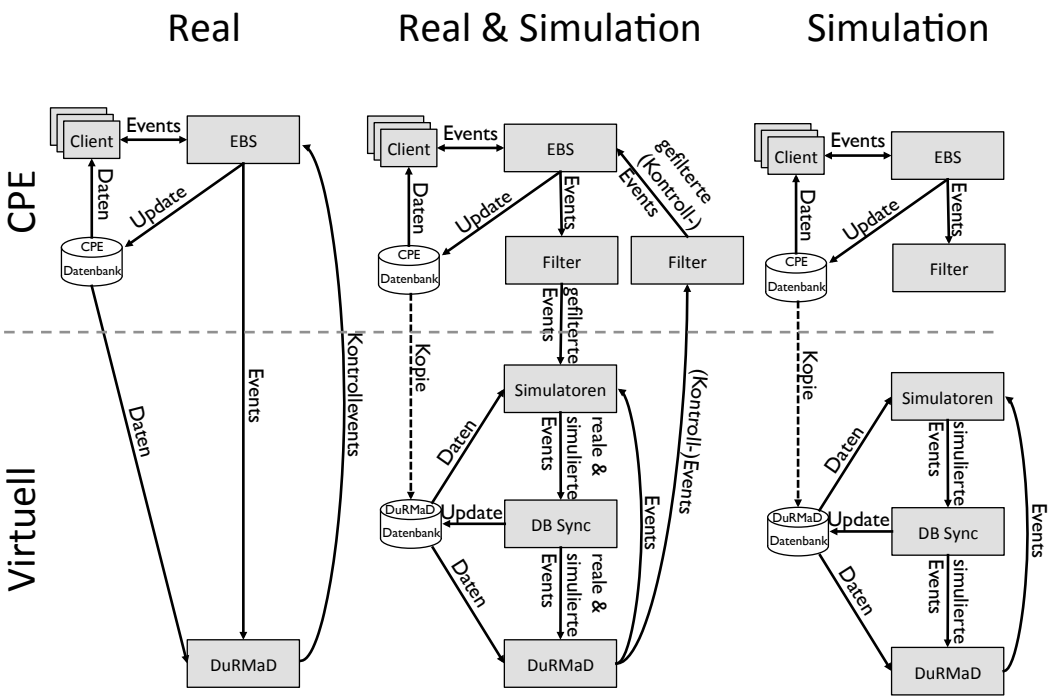


***Abbildung 6.19**: Die drei Modi des Dual Reality Frameworks*

sich ziehen können. Basierend auf Simulationen und spezifischer Implementierung können in DURMAD mehr Funktionen angeboten werden, als in der eigentlichen physischen Umgebung direkt umgesetzt werden können. In diesem Fall wird eine erweiterte DR realisiert. Die Kombination von realen und simulierten Informationen gemäß vorgegebener Regeln, welche durch entsprechende Filter abgebildet werden, entspricht dem DR++-Paradigma.

Gemäß der Definition 4.3 muss bei DR++ die Möglichkeit bestehen, dass nach einer Simulation die CPE in einen Zustand überführt wird, welcher ohne diese zu diesem Zeitpunkt vorliegen würde. Hierzu wird im EBS die Funktion angeboten, einen aktuellen Zwischenstand eines Diensts zu speichern und später wieder zu laden. Dies erfolgt, indem zu Beginn einer Simulation ein entsprechendes Event versendet wird, woraufhin jeder Dienst seinen aktuellen Kontext in Form eines Datencontainers zwischenspeichert. Anschließend erfolgt die Verarbeitung aller realen und/oder simulierten Events. Nach Beendigung der Simulation wird ein weiteres spezifisches Event versendet, nach dessen Erhalt der aktuelle Kontext mit dem zuvor zwischengespeicherten Status überschrieben wird. Im Anschluss daran werden alle realen Events, die während der Simulation versendet wurden, mittels der auf dem Server hinterlegten Historie abgerufen und abgearbeitet. Nimmt man beispielhaft einen Dienst, welcher eine Entnahme eines Produkts aus einem Regal protokolliert, stellt die Anzahl der Entnahmen den aktuellen Kontext dar, welcher zwischengespeichert wird. Anschließend kann dieser Wert durch reale oder simulierte Entnahmen verändert werden. Bei Beendigung der Simulation wird der zuvor zwischengespeicherte Wert wieder geladen

und für jede real stattgefundene Entnahme der Wert entsprechend angepasst, womit die DR++-Anforderung erfüllt ist.

6.5.3 Semantik und Verarbeitung

Die in Kapitel 5.1.3 erläuterte semantische Beschreibung einer CPE umfasst ebenfalls die Semantik der jeweiligen Events. Dadurch wird eine automatisierte Zuordnung von Events zu Sensoren und Aktuatoren gewährleistet. Diese wurde auch in DuRMAD integriert, indem durch einen Rechtsklick auf einen virtualisierten Aktuator oder Sensor ein entsprechendes Kontextmenü geöffnet wird, über welches geeignete Events versendet werden können. Dieses Kontextmenü umfasst alle Events, auf die der Aktuator reagieren bzw. die der Sensor versenden kann. Durch die Selektion eines solchen Eintrags können in einem weiteren Dialogfenster die eventspezifischen Parameter festgelegt und anschließend das Event versendet werden. Neben dieser teils generischen Methode, sensor- oder aktuatorspezifische Events zu versenden, besteht auch die Möglichkeit, in der Ontologie gewisse Standardevents mit festgelegten Parametern vorzudefinieren. Diese werden über einen entsprechenden Namen gekennzeichnet, welcher auch im Kontextmenü dargestellt wird. Durch Anklicken einer solchen Bezeichnung wird automatisch das vordefinierte Event versendet. Dies ist unter anderem sinnvoll, um Kontrollevents schnell und einfach an Aktuatoren zu versenden. Beispielsweise kann hierüber das Öffnen und Schließen von Türen und Toren durch eine einfache Interaktion mit dem 3D-Modell realisiert werden, ohne dass der Anwender die Eventstruktur und die spezifischen Parameter kennen muss. Ebenso können hierüber Sensorevents vereinfacht simuliert werden.

Die Erstellung von Regeln für ein Regelsystem benötigt normalerweise Expertenwissen. Um dies zu vereinfachen, wurden unter anderem Systeme entwickelt, welche dem Benutzer erlauben, die Regelerstellung mit Hilfe von Interaktionen mit realen Objekten durchzuführen, deren Feedback im virtuellen Modell erfolgt [Beckmann und Dey, 2003]. In diesem Fall repräsentieren die physischen Objekte Sensorinformationen bzw. Aktuatoreinstellungen, wie beispielsweise Sonnenschein bzw. eine eingeschaltete Lampe. Für die Prämissenspezifikation werden Sensorwerte, wie beispielsweise die aktuelle Witterung oder die Tageszeit bzw. der Wochentag, mittels Objektkarten repräsentiert. Diese enthalten einerseits eindeutig verständliche Abbildungen und andererseits Erklärungen in Textform. Die Objektkarten können auf eine spezielle Oberfläche gelegt werden, die eine automatische Identifikation ermöglicht. Gleiches gilt für die Aktuatorsteuerung als Regelkonsequenz, die dadurch spezifiziert wird, dass beispielsweise ein Objekt für ein eingeschaltetes Licht oder ein Thermostat mit einer spezifischen Temperatur auf die Oberfläche gelegt wird. Durch diese Interaktion wird automatisch eine Regel spezifiziert, welche dem Regelsystem hinzugefügt wird. Die dadurch erstellten Regeln können in einer grafischen Oberfläche noch einmal kontrolliert werden. Existiert bereits eine Regel mit den gleichen Prämissen, so wird diese durch die neue Spezifikation angepasst. Beispielsweise kann über dieses Interface angegeben werden, dass an einem bestimmten Wochentag, an dem es gleichzeitig regnet, die Raumtemperatur auf einen gewissen Wert gestellt und gleichzeitig das Licht angeschaltet werden soll. Hierzu müssen die entsprechenden Symbole für Regen, den Wochentag, die

Thermostateinstellung und das angeschaltete Licht auf die interaktive Fläche gelegt werden. Die daraus resultierende Regel wird auf einer virtuellen Abbildung des Raums dargestellt, welche auf einem Display visualisiert wird. Beispielsweise wird dort repräsentiert, ob das Licht angeschaltet oder ausgeschaltet ist. Über eine Simulation kann man zudem verschiedene Sensorwerte emulieren und damit die Regelspezifikationen virtuell testen.

Ähnliche Ansätze wurden auch im Rahmen dieser Arbeit im DR-Framework umgesetzt. Wie bereits in Kapitel 5.4.1 erläutert, können über ein entsprechendes Interface in natürlichsprachlichem Text mit diversen Unterstützungsfunktionalitäten Regeln auch von Nichtexperten erstellt werden. Die Spezifikation der Objekte, welche von den Regeln betroffen sein sollen, kann dabei anhand einer Liste oder auch durch eine Selektion im 3D-Modell erfolgen. Auf diese Weise können auch Sensoren und Aktuatoren ausgewählt werden, deren Daten bzw. Funktionen in den Regeln eingebunden werden. Die Einstellung der spezifischen Parameter erfolgt hierbei anhand der in Kapitel 5.4.1 beschriebenen grafischen Oberfläche. Das DuRMAD-Regelsystem ist vollständig in die EBS-Struktur eingebunden, so dass alle versendeten Events auch zu einer Anpassung der Wissensbasis des Regelsystems führen. Somit können auch Regelausführungen mittels Simulationen getestet werden. Hierzu wurden auch DuRMAD-spezifische Aktionen, wie z.B. das visuelle Hervorheben von Objekten, integriert, so dass ein Feuern einer Regel auch eine direkte grafische Darstellung im 3D-Modell nach sich zieht (siehe Kapitel 5.3.5). Ebenso können mittels des integrierten Regelsystems einfache Ansteuerungen von Aktuatoren realisiert werden, indem als Regelkonsequenz entsprechende Events versendet werden.

6.6 Zusammenfassung

Im Rahmen dieser Arbeit wurde die Konzeption und Realisierung des Event Broadcasting Service (EBS) beschrieben, eine Kommunikationsinfrastruktur, welche explizit für den Einsatz gemäß des erweiterten Dual Reality (DR++) Konzepts erstellt wurde. Die modulare Struktur des EBS ermöglicht hierbei eine individuelle Konfiguration im Hinblick auf das jeweilige Anwendungsgebiet. Hierzu gehört insbesondere die potentielle Verwendung des EBS als Blackboard-System, als Publish/Subscribe-System oder als komplexe Eventverarbeitung (CEP). Die Informationsübertragung erfolgt mittels Events, welche ebenfalls individuell konfiguriert werden können. So umfassen sie beispielsweise eine Priorität, die eine bevorzugte Übertragung ermöglicht oder die Möglichkeit, einzelne Events vor dem Versenden zu verschlüsseln.

Einen zweiten Schwerpunkt des EBS stellen die integrierten Filter- und Vorverarbeitungsmodule dar, welche fester Bestandteil der Client-Struktur sind und ebenso serverseitig Anwendung finden. Eine Kurzdarstellung der implementierten Filter und Vorverarbeitungsmodule ist in Tabelle 6.2 zusammengefasst. Es wurden insgesamt drei Filter implementiert, welche eine Aussortierung der Events abhängig von deren Typ, deren Inhalt oder außerhalb eines vorgegebenen Fensters ermöglichen. Des Weiteren umfasst der EBS Vorverarbeitungsmodule, welche eine Veränderung der Eventinhalte nach sich führen können. Insbesondere die Akkumulation von Events, die Transformation von Inhalten, die Kombination mehrerer

Events sowie das Aufsplitten werden durch die Vorverarbeitungsmodule ermöglicht. Ein spezielles Vorverarbeitungsmodul, welches dem proaktiven Notifizieren von potentiellen Fehlerquellen dient, stellt der Monitoringfilter dar. Dieser sendet eine Fehlernotifikation, wenn er in einem gewissen Zeitraum keine Events empfangen hat. Mehrere dieser Filter und Vorverarbeitungsmodule können mittels Verarbeitungs- und Filterketten zusammengeschlossen werden. Diese Filterketten werden in Form eines Graphen repräsentiert, wobei die Erstellung durch eine entwickelte grafische Oberfläche unterstützt wird.

Name	Typ	Kurzbeschreibung
Typfilter	Filter	Filterung nach dem Eventtyp
Inhaltsfilter	Filter	Filterung abhängig von einem Ausdruck
Fensterfilter	Filter	Filterung nach Zeit oder Eventanzahl
Akkumulation	Vorverarbeitungs-modul	Analyse mehrerer Events gleichen Typs
Transformation	Vorverarbeitungs-modul	Veränderung von Eventinhalten
Kombination	Vorverarbeitungs-modul	Verschmelzung mehrerer Events (unterschiedlichen) Typs
Aufsplittung	Vorverarbeitungs-modul	Aufsplittung eines „Metaevents“ in mehrere Events
Monitoring	Vorverarbeitungs-modul	Überwachung von Clientaktivitäten und Notifikation bei Problemen

***Tabelle 6.2**: Kurzdarstellung der im* EBS *integrierten Filter und Vorverarbeitungsmodule*

Die Realisierung des EBS beinhaltet auch dessen Architektur, welche im Rahmen dieser Arbeit konzeptioniert und implementiert wurde. Sie besteht aus mehreren Modulen, die eine individuelle Konfiguration der Kommunikationsinfrastruktur unterstützen. Die verwendeten Events leiten sich von einer Hauptklasse ab, welche spezifische Eventparameter enthält, die eine individuelle Verarbeitung jedes Events ermöglichen. Dies wird beispielsweise für die Verschlüsselung verwendet, die für jedes einzelne Event individuell angewendet werden kann. Mittels einer speziellen Queue-Struktur wird eine asynchrone Datenübertragung und Priorisierung von Events realisiert. Des Weiteren umfasst die EBS-Architektur Module für mehrere Serialisierungsverfahren, welche clientspezifisch unterschiedlich sein können, und serverseitige Verarbeitungen von Events. Bei ungewollten Verbindungsabbrüchen ermöglicht die Architektur eine Neuverbindung der Clients ohne Verlust von Events.

Die Verbindung zwischen dem Dual Reality Management Dashboard (DURMAD) und dem EBS bildet das Dual Reality Framework. Durch diese Kombination beider

Komponenten im Framework werden Vorteile geschaffen, welche unter anderem den Informationsaustausch zwischen einer CPE und ihrer Virtualisierung ohne die Implementierung spezieller Schnittstellen ermöglicht, da die Kommunikation zwischen den einzelnen Komponenten der CPE gleichzeitig im Virtuellen eingebunden wird. Der Informationsfluss zwischen den Komponenten der CPE kann dadurch im 3D-Modell dargestellt werden, um mögliche Fehlerquellen zu detektieren. Zusätzlich umfasst das DR-Framework eine Schnittstelle zur semantischen Beschreibung von Sensoren, Aktuatoren, Diensten und Events sowie ein Regelsystem, um einerseits die Aufgaben des Benutzers zu unterstützen und andererseits einen Automatismus der CPE zu realisieren. Des Weiteren ermöglicht das DR-Framework eine Integration von Simulationen nach dem DR++-Paradigma. Hierzu wurden drei Modi implementiert, welche eine Verwendung von ausschließlich realen, ausschließlich simulierten und sowohl realen als auch simulierten Informationen berücksichtigen. Dadurch können neben Monitoring und Steuerung auch Ideen aus DR++ Verwendung finden. Das Beispiel der Anzahl der Produktentnahmen aus einem Regal zeigt dabei die Umsetzung des DR++-Paradigma mittels des DR-Frameworks, indem nach Beendigung einer Simulation der Dienste die CPE auf den Zustand gebracht wird, den sie ohne die Simulation zu diesem Zeitpunkt hätte.

Im folgenden Kapitel wird das entwickelte DR-Framework im Einsatz in der Handelsdomäne präsentiert. Hierbei wird aufgezeigt, wie die entwickelte Kommunikationsinfrastruktur für die Kommunikation zwischen den Sensoren, Aktuatoren und Diensten eingesetzt werden kann und welche Vorteile sich mit der visuellen Repräsentation der Marktumgebung ergeben. Im Speziellen wird der Einsatz in einer Laborumgebung dargestellt, welche ähnlich wie ein realer Supermarkt aufgebaut ist.

7 Anwendungsszenario in der Einzelhandelsdomäne

Oftmals werden Cyber-Physische Umgebungen mit dem industriellen Umfeld in Verbindung gebracht, was unter dem Begriff Industrie 4.0 bekannt ist [Kagermann et al., 2012]. Doch auch in anderen Branchen werden immer mehr Sensoren und Aktuatoren eingesetzt, um eine Überwachung, Steuerung und Automatisierung zu erreichen. Ein Beispiel hierfür stellt die Einzelhandelsdomäne dar. Basierend auf dem Einsatz von energie- und kostensparenden Sensoren und Aktuatoren können bestehende Prozesse optimiert und damit Geld eingespart werden. Zu den Aktuatoren zählen beispielsweise digitale Preisauszeichnungen (Electronic Shelf Labels, ESL) oder digitale Werbeschilder in Form von Bildschirmen (Digital Signage). Der Einsatz dieser Aktuatoren ermöglicht es, Preise automatisch zu ändern, ohne dass ein weiterer manueller Eingriff erfolgen muss, bzw. schnell Werbung im Geschäft anzupassen. Zudem werden in der Einzelhandelsdomäne optische Sensoren zur Messung von Personenströmen verwendet, die beispielsweise am Ein- und Ausgang angebracht, eine Ermittlung der Kundenanzahl im Geschäft erlauben. Diese Erkenntnis kann anschließend zu einer optimierten Planung der Mitarbeiter verwendet werden.

Ein großer Optimierungsbedarf besteht bei der Befüllung von Regalen. Ist ein Produkt abverkauft, entstehen Gewinneinbußen des Geschäfts. Diese Problematik ist unter dem Begriff *„Out-of-Stock“ (OoS)* bekannt [Gruen et al., 2002]. Zum einen verliert das Geschäft beim OoS die Gewinnmargen, da vor der nächsten Lieferung keine weiteren Produkte mehr verkauft werden können. Zum anderen führt dies zur Verärgerung der Kunden und schlimmstenfalls auch dazu, dass diese zukünftig in einem anderen Geschäft einkaufen gehen. Der Gewinnverlust und die Reaktion der Kunden auf OoS-Situationen wurden in mehreren Studien untersucht. Unter anderem wurde dabei festgestellt, dass 32% der Kunden in diesem Fall eine andere Größe oder ein anderes Produkt des gleichen Herstellers gekauft haben [Emmelhainz et al., 1991]. 41% der Kunden kauften ein Konkurrenzprodukt, 14% sind zu einem anderen Händler gegangen, um das Produkt zu kaufen. Die restlichen 13% verschoben den Kauf des Produkts auf einen späteren Zeitpunkt. Des Weiteren wurde gezeigt, dass eine dauerhafte OoS-Problematik dazu führte, dass eine hohe Prozentzahl an Personen künftig in einem anderem Geschäft einkaufen geht [Verbeke et al., 1998]. Hierbei ist es auch nicht relevant, ob in der Nähe kein weiterer Einzelhändler mit einem vergleichbaren Produktsortiment vorhanden ist, da die Kunden heutzutage sehr mobil sind.

Um eine aufkommende OoS-Situation frühzeitig zu erkennen und diese durch entsprechende Maßnahmen zu verhindern, können Sensoren eingesetzt werden. Hierzu ist oftmals die Radiofrequenz Identifikation (RFID) Technik in Studien eingesetzt worden [Metzger, 2009]. Diese Technologie ermöglicht eine automatisierte Inventur, bei welcher der Warenbestand auf der Verkaufsfläche erfasst und mit einem Soll-Bestand abgeglichen wird. Neben der theoretischen Diskussion und Darstellung von prototypischen Umsetzungen gibt es auch erste Ansätze der Umsetzung in realen Einzelhandelsumgebungen [Thiesse et al., 2009]. Beispielsweise wurde in Form eines Pilotprojekts der Textilbereich eines Marktes der Firma Kaufhof mit RFID-Antennen ausgestattet. Unter anderem wurden die Kleiderkabinen, die Verbindung zwischen Lager und Verkaufsraum, Rollgleiten und Aufzüge instrumentiert. Als Dienste wurden mehrere Assistenzsysteme für die Kunden angeboten, welche eine Darstellung der vorrätigen Größen und Farben von ausgewählten Produkten ermöglichten. Die Waren wurden anhand von RFID-Labels identifiziert und somit automatisch an den entsprechenden Positionen im Markt erkannt. Zudem wurde mittels der Instrumentierung eine automatisierte Inventur und ein Diebstahlschutz umgesetzt. Die Instrumentierung von Kleidungsstücken mittels RFID-Tags ist aufgrund der physikalischen Gegebenheiten - keine Abschirmung durch Metall und Flüssigkeit - sowie der erzielten Gewinnmargen interessant. Die Zusatzkosten für RFID-Etiketten rechnen sich bei billigerer Ware heutzutage noch nicht, weshalb neben RFID auch weitere Möglichkeiten der Erkennung einer OoS-Situation erprobt werden. Ein Beispiel hierfür stellen drucksensitive Oberflächen dar, die das Gewicht der Ware erfassen und mittels mathematischer Modelle eine Regalauslastung ermitteln können [Metzger, 2009].

Neben der Erfassung des IST-Bestands von Ware auf der Verkaufsfläche stellen die aktuell erfassten und historischen Sensorinformationen weitere Möglichkeiten der Analyse zur Verfügung. Beispielsweise kann diese Informationsvielfalt als mögliches Messinstrument verwendet werden, um bereits im Vorfeld Abverkaufszahlen von Produkten abhängig von ihrer Position im Markt zu prognostizieren. Hierzu werden Informationen aus unterschiedlichen Wissensquellen, wie beispielsweise vergangene Abverkaufszahlen, zusammengeschlossen und verarbeitet. Die zentrale Schnittstelle, welche als Datengrundlage dient, ist das sogenannte Datenlager (Data Warehouse, DWH). Die Aufgaben des DWH umfassen hierbei das Kopieren der Informationen aus externen Quellen mit anschließender Aufbereitung und Speicherung in einem einheitlichen Format. Diese Daten werden aggregiert, um Handlungsträgern, wie Analysten oder Managern, als Entscheidungsunterstützung zu dienen [Chaudhuri und Dayal, 1997]. Aufgrund dieser Funktionsweise kann ein DWH folgendermaßen definiert werden:

Definition 7.1 (Datenlager) *Ein Datenlager ist eine themenorientierte, integrierte, chronologisierte und persistente Sammlung von Daten, um das Management bei seinen Entscheidungsprozessen zu unterstützen [Inmon, 1992, Seite 33].*

Zusammengefasst kann die Funktionsweise eines DWH durch drei Schritte dargestellt werden: Datenbeschaffung, Verwaltung und Auswertung. Da für die Auswertung oftmals auch historische Daten wichtig sind, werden die Informationen persistent im

DWH hinterlegt. Hieraus ergeben sich sehr große Datenmengen, die verarbeitet werden müssen. Um die Relation zwischen den einzelnen Informationen abzubilden, werden in der Regel mehrdimensionale Datenschemata verwendet, die entweder in einer entsprechenden Datenbank oder in mehrdimensionalen Objekten hinterlegt werden. Die Anfragen an das DWH erfolgen über erweiterte SQL-Abfragen oder entsprechende Operatoren. Nimmt man die Abverkäufe als Beispiel, so sind der Zeitpunkt des Verkaufs, die Abteilung, der Verkäufer und das Produkt mögliche Dimensionen, die betrachtet werden sollten [Chaudhuri und Dayal, 1997]. Aus diesem Grund müssen die Strukturen und Relationen bereits im Vorfeld geplant werden, um möglichst performant Ergebnisse liefern zu können. Die Integration neuer Datensätze erfolgt oftmals in einem definierten Zeitraum (z.B. nachts), währenddessen die Verarbeitung im Datenlager angehalten wird. Die Entwicklungen zeigen jedoch, dass aktuelle Informationen bereits möglichst in Echtzeit im DWH übernommen werden, um frühzeitige Analysen zu ermöglichen. Die resultierenden Ergebnisse werden an Darstellungskomponenten versendet, um diese dem Entscheider möglichst einfach zu präsentieren. Dabei stellen Tabellenkalkulationsprogramme das meist genutzte Visualisierungstool dar. Eine weitere Visualisierungsform ist die Darstellung in Dashboards (siehe Kapitel 2.5.4).

Weitere Veränderungen im Handelssektor sind durch den Verkauf über Onlineportale, sogenannte Webshops, möglich. Im Onlinehandel kann man sich den Vorteil der Digitalisierung zunutze machen, indem man adaptive Werbung und Vorschläge abhängig vom aktuellen Benutzerverhalten anzeigt, z.B. abhängig davon, welche Produkte der Benutzer sich anschaut und welche sich in seinem Einkaufswagen befinden. Die meisten Webshops weisen ein ähnliches Design auf, wodurch Kunden die Strukturen schnell verstehen und sich einfach orientieren können. Ebenso gibt es oftmals explizit auf mobile Endgeräte angepasste Darstellungen, so dass der Wechsel zwischen einem stationären Computer und einem mobilen Endgerät problemlos erfolgen kann. Eine solche flexible Anpassung ist insbesondere auch bei Mischformen relevant. Viele Händler bieten die Möglichkeit, sowohl auf einer physischen Fläche als auch über einen Webshop einkaufen zu können, um unterschiedliche Kundensegmente anzusprechen. Wichtig hierbei ist, dass eine Harmonisierung der angebotenen Kanäle erfolgt und somit eine Unterstützung für die Kunden realisiert wird. Dies ist unter den Begriffen Omni-Channel, Multi-Channel, Cross-Channel oder No-Line-Commerce bekannt [Fost, 2014]. Im Online-Handel wurden bereits Ansätze untersucht, bei denen die Einkäufe von verschiedenen Webshops in einem zentralen Einkaufswagen hinterlegt werden [Puglia et al., 2000]. Dies vereinfacht und harmonisiert den Bezahlprozess, der auf diesem Weg immer gleich abgebildet wird und stellt somit eine händlerübergreifende Darstellung sicher.

Aufgrund der fortlaufenden Entwicklung kostengünstiger Sensoren wird die stationäre Handelsumgebung auch zukünftig eine Veränderung erfahren. Speziell Sensoren zur Erfassung des Füllstatus eines Regals oder zur Erkennung und Interpretation von Kundeninteraktionen, wie z.B. einer Zeigegestik, werden zukünftig an Bedeutung gewinnen [Krüger et al., 2011]. Die Sensoren und Aktuatoren ermöglichen hierbei Konzepte, welche bereits im Onlinehandel erprobt sind, wie Produktempfehlungen oder -vergleiche auf die physische Fläche zu bringen. Dabei können die Vorteile des „Offlinehandels“, welcher unter anderem in der

sozialen Komponente liegt, durch digitale Assistenzfunktionen ergänzt werden. Dabei ist es auch wichtig, dass eine direkte, personalisierte Kundenansprache erfolgt, da eine Massenkommunikation, wie z.B. mittels Faltblattwerbung, in den letzten Jahren an Attraktivität verloren hat [Krüger et al., 2014]. Um eine solche personalisierte Ansprache zu realisieren, bedarf es den aktuellen Kontext des Kunden zu erfassen, was ausschließlich durch eine vernetzte Umgebung realisiert werden kann. Die erfassten Kontextinformationen können anschließend durch individuell angepasste Dienste verarbeitet und zur Nutzung in adaptiven Assistenzsystemen, wie beispielsweise Dialogberatern, oder zur Generierung kontextbasierter Werbeinformationen eingesetzt werden [Kahl et al., 2010]. Diese und weitere Ansätze wurden in einer instrumentierten Handelsumgebung prototypisch umgesetzt, dem Innovative Retail Laboratory (IRL).

7.1 Innovative Retail Laboratory (IRL) - Instrumentierte Einzelhandelsumgebung

Das IRL ist eine agile Cyber-Physische Handelsumgebung, welche in Kooperation mit der deutschen Einzelhandelskette Globus SB-Warenhaus Holding GmbH & Co. KG und dem Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI GmbH) betrieben wird [Spassova et al., 2009; Krüger et al., 2010]. Seit dem Beginn im Oktober 2007 wird in einer Laborumgebung eine Supermarktatmosphäre nachempfunden, die mit Sensoren, Aktuatoren und entsprechenden Diensten angereichert ist. In dieser Laborumgebung werden Fragestellungen aufgegriffen, welche den zukünftigen Einkauf umfassen. Insbesondere werden neuartige Interaktionsparadigmen in öffentlichen Umgebungen mittels eingebetteter Sensorik in Form von Demonstratoren untersucht. Hierbei spielt neben der Informationsakquise die personalisierte Aufbereitung und Darbietung von Informationen eine wichtige Rolle. In diesem Zusammenhang werden Grundvoraussetzungen für die Akzeptanz und einen effizienten Einsatz von Assistenzsystemen anhand von Prototypen evaluiert. Wichtig hierbei ist, dass die Systeme Zugriff auf alle relevanten Sensorinformationen haben, um den aktuellen Kontext der Umgebung sowie des Benutzers zu erfassen und damit eine entsprechende Auswertung und Reaktion zu realisieren.

Betrieben wird das IRL als sogenanntes Living Lab, was bedeutet, dass immer wieder neue Aspekte und Demonstratoren hinzukommen, worauf die existierenden Dienste möglicherweise angepasst werden müssen. Hierdurch entsteht eine agile Struktur von Komponenten, Daten und Kommunikationskanälen, welche mit der Zeit erweitert und verändert wird. Das IRL umfasst eine Vielzahl von Sensoren und Aktuatoren (siehe Abschnitt 7.1.1), die den Kontext der Umgebung und somit auch Veränderungen erfassen bzw. zu Änderungen des Umgebungszustands führen. Die erfassten Daten werden durch entsprechende Dienste (siehe Abschnitt 7.1.2) ausgewertet, welche ebenso die Aktuatoransteuerung übernehmen. Abbildung 7.1 zeigt einige dieser Komponenten. Teilweise umfassen physische Geräte sowohl Sensoren als auch Aktuatoren. Im Speziellen sind hierbei Smartphones zu erwähnen. Diese werden im IRL wie einzelne Sensor- und Aktuatorknoten behandelt, die auf einer physikalischen Plattform vereint sind.

***Abbildung 7.1**: Innovative Retail Lab (IRL) - eine Cyber-Physische Handelsumgebung*

7.1.1 Sensoren und Aktuatoren im IRL

Zur Erfassung von Benutzerinteraktionen und Kontextveränderungen umfasst das IRL eine Vielzahl unterschiedlicher Sensoren, welche elektromagnetische Gegebenheiten messen, mittels optischer Verfahren Veränderungen wahrnehmen, magnetische Eigenschaften messen, sowie in Form weiterer kleiner Sensorknoten Veränderungen erfassen. Eine Auswahl der vielfältigen und heterogenen Sensortypen ist in folgender Auflistung dargestellt.

↪ **RFID & NFC:** Die Funktechnologie ermöglicht eine eineindeutige Identifikation von Objekten. Im IRL wird dies zur Erkennung sowie zur Lokalisation von Produkten verwendet. Zur Identifikation sind an entsprechenden (mobilen) Komponenten, wie z.B. Regalen und Einkaufswagen, Lesegeräte angebracht. Es wird sowohl RFID im hochfrequenten (HF) als auch im ultrahochfrequenten (UHF) Bereich eingesetzt. Zudem wird eine Spezialisierung von RFID, die Nahfeldkommunikation (Near Field Communication, NFC), verwendet, insbesondere wenn es um die Identifikation oder Verifikation von Benutzern geht.

↪ **Bluetooth Beacons:** Als weitere Positionierungsmethode werden im IRL Sensoren verwendet, welche auf dem neuen Standard Bluetooth Low Energy (BLE) beruhen. Im Speziellen wird das Quuppa[1]-System eingesetzt, welches eine Lokalisation im 3D-Raum ermöglicht.

↪ **Kameras:** Sowohl statische als auch dreh- und neigbare Kameras werden im IRL

[1] http://quuppa.com, Zugriff: November 2014

zur optischen Aufnahme der Umgebung eingesetzt. Insbesondere werden auch Kameras verwendet, welche eine dreidimensionale Erfassung ermöglichen, wie z.B. die Microsoft Kinect[2]. Zur Erkennung von Objekten auf einer Fläche (z.B. einem Regalboden) wird im IRL ein neuartiger optischer Sensor eingesetzt, der einfallendes Licht durch eine Struktur von eingebetteten Glasfasern an eine Kamera weiterleitet.

↪ **Biometrische Sensoren:** Ebenso sind im IRL biometrische Sensoren, wie beispielsweise Fingerabdruckscanner, zu finden, welche eine Identifikation von Personen ermöglichen.

↪ **Digitale Kompasse:** Zur Bestimmung von Orientierungen werden im IRL digitale Kompasssysteme eingesetzt.

↪ **Temperatursensoren:** Für die Qualität eines Produkts spielt oftmals die Lagerungstemperatur eine große Rolle. Aus diesem Grund werden im IRL Temperatursensoren, wie beispielsweise μParts, eingesetzt [Beigl et al., 2005].

↪ **Beschleunigungssensoren:** Beschleunigungssensoren werden im IRL verwendet, um die aktuelle Orientierung von Produkten zu ermitteln.

Neben den Sensoren wurden im IRL auch mehrere Aktuatoren integriert, welche digitale Werbedarstellungen und Auszeichnungen, erweiterte Visualisierungen, akustische Ausgaben und elektronische Tore und Türen umfassen. Diese können digital über entsprechende Schnittstellen angesteuert werden. Folgende Liste stellt einen Auszug der installierten Aktuatoren im IRL dar.

↪ **Displays:** Zur Darstellung von allgemeinen Informationen oder Werbung befinden sich mehrere Displays im IRL. Eine spezielle Form von Displays stellen elektronische Preisschilder (Electronic Shelf Labels, ESL) dar. Smartphone Displays, die sich im IRL befinden, können ebenfalls zur Darstellung verwendet werden.

↪ **(Steuerbare) Projektoren:** Neben der Informationspräsentation auf Displays werden im IRL auch Projektoren verwendet. Insbesondere bieten steuerbare Projektoren eine Möglichkeit, Informationen auf beliebigen planaren Flächen darzustellen. Hierzu wird das FluidBeam-System verwendet [Spassova, 2004, 2011].

↪ **See-Through Devices:** Eine weitere Möglichkeit der Erweiterung von physischen Objekten durch digitale Informationen stellen im IRL sogenannte „See-Through Devices“ dar. Als Beispiel hierfür sind Smartphones oder Tablets zu nennen, welche mittels ihrer Kamera die reale Umgebung erfassen und diese gleichzeitig auf ihrem Display mit digitalen Inhalten anreichern können.

↪ **Lautsprecher:** Zur Wiedergabe von Sprachnachrichten, Musik oder Hinweistönen befinden sich mehrere Lautsprecher im Labor.

[2] http://www.microsoft.com/en-us/kinectforwindows/, Zugriff: November 2014

↪ **Akustische Alarme:** Zusätzlich zu den Lautsprechern können akustische Warnhinweise auch in speziellen Geräten, wie beispielsweise Gates zur Warensicherung in Kaufhäusern, ausgegeben werden, was unter anderem zur Aufmerksamkeitslenkung (z.B. von Mitarbeitern und Sicherheitspersonal) eingesetzt wird.

↪ **Schranken:** Elektronische Schranken und Tore ermöglichen eine Ein- und Ausgangskontrolle.

↪ **Automaten:** Schließlich befinden sich Automaten im IRL, welche automatisch Produkte auswerfen können.

Wie zuvor erwähnt gibt es zur Ansteuerung dieser Aktuatoren spezifische Schnittstellen, welche von Diensten implementiert werden können. Diese umfassen in der Regel ein einfaches Signal, welches an die physische Komponente weitergeleitet wird, z.B. „Tür öffnen“. Teilweise ist es jedoch nicht sinnvoll, jeden einzelnen Schritt für eine Veränderung zu versenden. In diesem Fall umfasst die Schnittstelle neben der reinen Weiterleitung eine zusätzliche Verarbeitungskomponente. Beispielsweise kann hierüber eine Kamera oder ein Beamer innerhalb einer spezifizierten Zeit von seiner aktuellen zu einer vorgegebenen Position gefahren werden. Die einzelnen Zwischenschritte bis zu diesem Ziel können in diesem Fall mittels eines Interpolators ermittelt und zu den entsprechenden Zeitpunkten an das physische Gerät geschickt werden. Somit wird die Sequenz an Befehlen durch einen Metabefehl, welcher an die Schnittstelle kommuniziert wird, gekapselt. Ebenso kann hierdurch zwischen absoluten und relativen Positionsbefehlen konvertiert werden. Durch die direkte Kommunikation mit dem physischen Endgerät ist der Schnittstelle ihr aktueller Zustand bekannt und sie kann hierüber die Konvertierung von extern eintreffenden Befehlen durchführen.

7.1.2 Dienste im IRL

Die einzelnen Sensoren und Aktuatoren finden in mehreren Diensten Verwendung. Wie zuvor erwähnt, umfassen diese Assistenzfunktionalitäten für Kunden und Mitarbeiter in der Einkaufsdomäne. Ganzheitlich betrachtet beginnt und endet der Einkaufsprozess im Heimbereich, und zwar mit der Vor- und Nachbereitung des eigentlichen Einkaufs im Supermarkt. Einen wichtigen Punkt bei der Vorbereitung stellt das Erstellen einer Einkaufsliste dar. Bei der Nachbereitung ist unter anderem das Einräumen der gekauften Produkte relevant. Aus diesem Grund umfasst das IRL neben einer Einkaufsumgebung auch einen entsprechend instrumentierten Heimbereich.

Einkaufsvor- und -nachbereitung Die Einkaufsvor- und -nachbereitung erfolgt in der Regel zu Hause, wobei der Kühlschrank meist im Mittelpunkt steht. Dieser ist oftmals ein sozialer Knotenpunkt in häuslichen Gemeinschaften, zu dem alle Bewohner Zugriff haben. Am IRL wurde der Kühlschrank mit einem Touchdisplay, welches in die Tür integriert wurde, und RFID-Antennen im Inneren instrumentiert. Sobald ein RFID-Tag von den Lesern erkannt wird, wird dies als Event über den EBS kommuniziert. Ebenso wird das Verschwinden eines RFID-Tags aus dem Lesefeld eines Readers weitergegeben. Durch einen

externen Dienst wird darüber die aktuelle Produktposition ermittelt und als Event versendet. Diese Angaben ermöglichen es, alle im Kühlschrank befindlichen Produkte zu erfassen, die mit mindestens einem RFID-Tag versehen wurden. Wird ein Positionsevent empfangen, also ein neues Produkt in den Kühlschrank gelegt, wird dessen digitales Objektgedächtnis vom Kühlschrank geladen. Basierend auf diesen Informationen werden Dienste, wie die Darstellung aller im Kühlschrank befindlichen Produkte und die Visualisierung grafisch aufbereiteter digitaler Produktgedächtnisse, angeboten. Des Weiteren findet ein automatischer Abgleich der Mindesthaltbarkeitsdaten (MHD) mit dem aktuellen Datum statt, dessen Ergebnis ebenfalls visuell auf dem Kühlschrankdisplay dargestellt wird. Neben dem MHD wird auch die empfohlene Lagerungstemperatur aus dem Objektgedächtnis geladen, um abzugleichen, ob die Produkte in der geeigneten Temperaturzone gelagert werden und andernfalls eine pro-aktive Warnung anzuzeigen. Zudem kann über eine digitale Version eines Faltblatts eine Einkaufsliste erstellt und angepasst werden, wie in Abbildung 7.2(a) dargestellt. Diese wiederum wird im entsprechenden Benutzerprofil hinterlegt, so dass sie von mobilen Endgeräten abgerufen und dargestellt werden kann (siehe Abbildung 7.2(b)). Jede Veränderung der Liste wird hierbei als Event verschickt, woraufhin die Liste im Benutzerprofil automatisch angepasst wird. Wird ein solches Einkaufslistenevent von einem verarbeitenden Dienst empfangen, kann dieser die aktualisierte Liste laden. Somit ist eine Synchronisation der Einkaufsliste auf unterschiedlichen Endgeräten, wie beispielsweise Kühlschrank und Smartphone, möglich.

(a) Erstellung der Einkaufsliste am Kühlschrank

(b) Synchronisation der Einkaufsliste mit einem mobilen Endgerät

***Abbildung 7.2**: Instrumentierter Kühlschrank zur Erstellung einer Einkaufsliste*

Eine weitere Möglichkeit der Erstellung einer Einkaufsliste wurde mittels Handschrifterkennung umgesetzt [Liwicki et al., 2011]. Hierbei können die Einträge einer Einkaufsliste handschriftlich erfolgen, indem die Anoto[3]-Technologie eingesetzt wird. Diese besteht aus einem speziellen Stift, welcher das Geschriebene mittels einer Kamera erfasst und via Bluetooth an einen Rechner schickt. Dabei wird spezielles Papier verwendet, welches ein vordefiniertes Muster beinhaltet. Wichtig hierbei ist, dass die Erfassung automatisch erfolgt

[3]http://www.anoto.com, Zugriff: November 2014

und der Benutzer wie gewohnt seine Einkaufsliste handschriftlich notieren kann. Auf dem Computer werden die geschriebenen Begriffe mittels einer standardmäßigen Schrifterkennung von Microsoft in computerverständlichen Text umgewandelt. Die ermittelten Einträge werden in einem weiteren Schritt mit der Produktontologie, welche im IRL verwendet wird, abgeglichen und die erkannten Begriffe automatisch als Einkaufslistenevents versendet, woraufhin neue Einträge zur Einkaufsliste im entsprechenden Benutzerprofil hinzugefügt werden. Wie auch bei der Veränderung der Einkaufsliste am Kühlschrank werden die Dienste, welche die Einkaufsliste benötigen, über die Änderungen notifiziert. Zum Verbessern der Erkennungsrate wurden zudem Verfahren implementiert, welche die Ähnlichkeit zwischen Wörtern ermitteln und dadurch eine Fehlerkorrektur bei der Digitalisierung ermöglichen.

IRL SmartCart Im Supermarkt ist der Einkaufswagen ein allgegenwärtiges Objekt, das als Technologieträger prädestiniert ist, um Einkaufsassistenz zu ermöglichen. Unter anderem sind hierbei Dienste, wie die Darstellung der Einkaufsliste, die Visualisierung der sich im Wagen befindlichen Produkte samt Preisinformationen und ein beschleunigtes Checkout an der Kasse von Bedeutung [Kourouthanassis et al., 2002]. Diese Konzepte wurden im Rahmen dieser Arbeit anhand des *IRL SmartCart* umgesetzt [Kahl et al., 2011a]. Dieser umfasst funkbasierte Sensorik in Form von RFID und NFC, einen Fingerabdrucksensor, einen digitalen Kompass sowie ein Touchdisplay. Diese Sensoren und Aktuatoren sind an ein integriertes Netbook angeschlossen und werden durch Batterien mit Strom versorgt. Zudem ist ein Sender zur Positionserkennung auf Basis von BLE-Technologie am Wagen angebracht (Quuppa Tag). Der Prototyp ist in Abbildung 7.3 dargestellt.

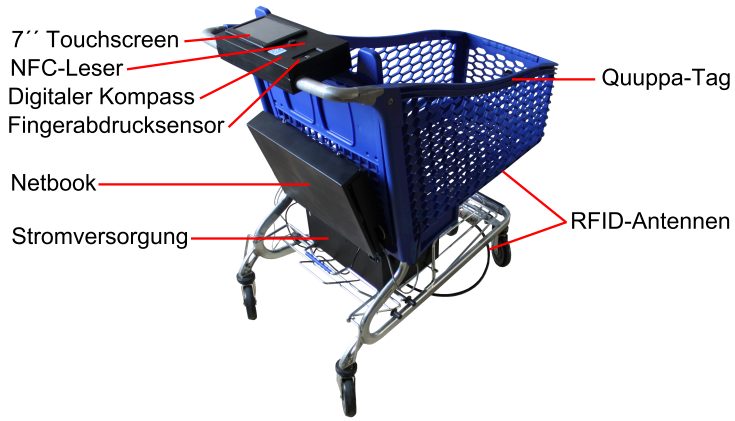
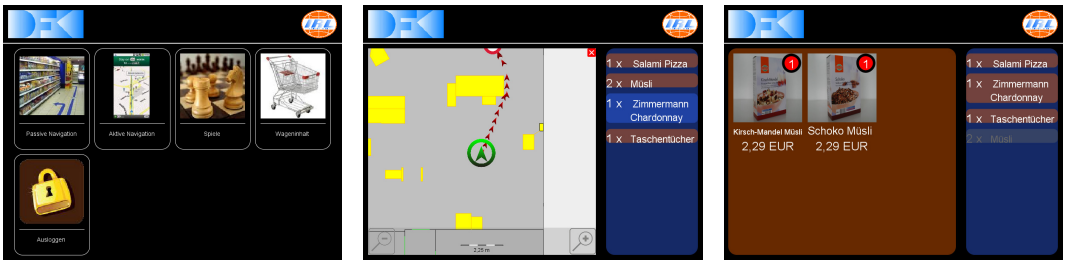


***Abbildung 7.3**: Instrumentierung des IRL SmartCart*

Basierend auf dieser Instrumentierung werden mehrere Dienste angeboten. Durch die Verknüpfung mit dem Warenwirtschaftssystem können die Produktpositionen im Markt erfasst werden. In Kombination mit der aktuellen Position des Wagens kann dies genutzt werden, um eine Navigation zu Produkten im Markt zu realisieren. Hierbei werden die Einträge der am IRL SmartCart geladenen Einkaufsliste als Ziele verwendet. Um die Einkaufsliste

auf den Einkaufswagen zu laden, muss der Benutzer sich lediglich an diesem authentifizieren, woraufhin sein Benutzerprofil inklusive der erstellten Liste automatisch geladen wird. Basierend auf eintreffenden Einkaufslistenevents wird diese Einkaufsliste aktualisiert. Des Weiteren wird die Position des Einkaufswagens mittels des Quuppa-Systems konstant ermittelt und bei einer Positionsveränderung als Event an den Einkaufswagen gesendet. Diese Informationen werden in Diensten verwendet, welche dem Benutzer nach dem Einloggen zur Verfügung stehen (siehe Abbildung 7.4(a)). Neben einer aktiven Navigation zu einem Produkt (siehe Abbildung 7.4(b)) gibt es auch die Möglichkeit einer passiven Navigation. Im letzteren Fall werden keine Pfeile zum Ziel dargestellt. Anstelle dessen wird die Umgebung des Kunden präsentiert, welche mit entsprechenden Markierungen angereichert wird, wenn Produkte seiner Einkaufsliste in diesem Kartenausschnitt zu finden sind. Hierbei dient die Visualisierung als Erinnerungsfunktion. Diese Navigationsunterstützung hat zudem den Vorteil, dass die Kunden nicht von der Umgebung abgelenkt werden, da sie nicht ständig darauf fokussiert sind, Navigationsanweisungen zu folgen. In einer Studie wurde gezeigt, dass selbst wenn Produkte oder andere Landmarken als Informationen für die Supermarktnavigation verwendet werden, die Umgebung während der Navigation von den Kunden nicht bewusst wahrgenommen wird [Nurmi et al., 2011]. Dieses ist jedoch für den Handel wichtig, da zusätzliche Umsätze durch Spontaneinkäufe von Produkten gemacht werden. Neben der visuellen Hervorhebung sich in räumlicher Nähe befindender Produkte der Einkaufsliste wurde im IRL die Möglichkeit einer akustischen Notifikation realisiert [Jung et al., 2011a]. Hierbei wird die als Event versendete Position des Einkaufswagens verwendet, um die Distanz zu den entsprechenden Produkten zu ermitteln. Unterschreitet diese Distanz einen vorgegebenen Schwellenwert, wird ein Event versendet, woraufhin in die aktuell abgespielte Musik des Supermarkts ein vom Kunden spezifizierter Notifikationshinweis integriert wird. Dieser Hinweis kann nur vom jeweiligen Kunden interpretiert werden, sodass dadurch kein Einfluss auf andere Kunden stattfindet. Beispielsweise kann dies durch die Verstärkung der Lautstärke eines speziellen Musikinstruments erfolgen [Jung, 2009].



(a) Angebotene Dienste (b) Darstellung der Navigation (c) Anzeige des Wageninhalts und der Einkaufsliste

***Abbildung 7.4**: Benutzeroberfläche beim IRL SmartCart*

Die Position des Einkaufswagens sowie dessen aktuelle Geschwindigkeit können zudem als Parameter verwendet werden, um adaptiv Musikstücke aus einer personalisierten Playliste abzuspielen [Jung et al., 2011b]. Die Playliste kann gleichzeitig mit der Erstellung der Einkaufsliste generiert werden. Hierbei werden die Musikstücke anhand ihres Genres den Bereichen im Supermarkt zugewiesen. Begibt sich der Kunde in einen bestimmten Bereich, wird

ein entsprechendes Musikstück auf dessen Einkaufswagen gestreamt. Neben dem Bereich wird bei der Auswahl des Musikstücks auch die Bewegungsgeschwindigkeit berücksichtigt, da diese ein Indiz für den aktuellen Stresslevel darstellt. Bewegt sich der Kunde schnell und befindet sich somit wahrscheinlich im Stress, kann ein ruhiges Lied ihn höchstwahrscheinlich zusätzlich frustrieren. Daher wird in diesem Fall automatisch ein Lied mit angepasstem Tempo ausgewählt. Die Aufforderung zum Abspielen eines bestimmten Songs wird als Event versendet. Das Abspielen kann entweder über den Einkaufswagen oder über ein mobiles Endgerät erfolgen. Hierdurch wird eine personalisierte musikalische Begleitung im Supermarkt realisiert. Als weiteren Dienst bietet der IRL SmartCart die Darstellung des Wageninhalts samt Preisinformationen an, wie in Abbildung 7.4(c) zu sehen. Schließlich existiert das Konzept, Spiele auf dem Display zu ermöglichen [Kahl et al., 2009]. Diese können Produkte mit einbeziehen, die sich aktuell im Wagen befinden, und sind für Kleinkinder gedacht, die im Kindersitz des Wagens sitzen. Dies hat den Vorteil, dass die Kinder die Produkte und damit Objekte im Spiel bereits kennen oder spielerisch kennenlernen.

Regal- und Kiosksysteme Um frühzeitig eine Out-of-Stock-Situation zu erkennen und weitere Mehrwertfunktionalitäten für Kunden und Marktmitarbeiter zu bieten, wurden mehrere Regale mit Sensoren und Aktuatoren instrumentiert. Beispielsweise wird mittels RFID eine Entnahme von Produkten erkannt und als Event kommuniziert, woraufhin auf einem am Regal angebrachten Display produktbezogene Informationen, wie z.B. Inhaltsstoffe und Nährwertangaben, dargestellt werden. Die Darstellung erfolgt mittels einer Webseite, deren Link als Event an den Client des Displays geschickt wird. Die Darstellung am Display erleichtert den Kunden die Suche nach bestimmten Produktinformationen, die sie ansonsten auf der Produktverpackung suchen müssten, was speziell für ältere Menschen aufgrund der kleinen Schriftgröße und dem teilweise schlechten Kontrast ein Problem darstellen kann. Wird mehr als ein Produkt aus dem Regal genommen, werden die zugehörigen Nährwertangaben in einer tabellarischen Form gegenübergestellt, wie in Abbildung 7.5(a) dargestellt. Weitere Sensoren an den Produkten, wie beispielsweise μParts [Beigl et al., 2005], ermöglichen die Ermittlung zusätzlicher Produktinformationen (Temperatur). Diese werden als Events versendet und können ebenfalls bei der Produktentnahme auf dem Display dargestellt werden [Schmitz et al., 2008]. Die Einbindung von Filtern ermöglicht hierbei, dass trotz einer regelmäßigen Temperaturerfassung durch den Sensor nur bei Abweichungen von über einem Grad zum letzten kommunizierten Wert ein neues Event versendet wird. Zudem können die Produktinformationen auch in akustischer Form mittels Audiodateien oder Text-to-Speech ausgegeben werden, wobei der Befehl zur Ausgabe mittels spezifischer Events erfolgt. Eine zweite Form der Überwachung des Füllstands von Regalen stellt das FibreShelf dar (siehe Abbildung 7.5(b)) [Krüger et al., 2011]. Hierbei wird eintreffendes Licht durch im Regal integrierte Glasfasern zu einer Kamera transportiert, wodurch die Umrisse der im Regal befindlichen Produkte erkannt werden. Durch einen Abgleich der aktuell erkannten Regalbelegung mit einer vordefinierten Befüllung kann hierbei abgeleitet werden, ob Produkte fehlen oder an einer falschen Position stehen. Dies kann den Mitarbeitern mittels Events kommuniziert werden, so dass diese sich gegebenenfalls um die Wiederbefüllung bzw. die Aufräumung des Regals kümmern können. Neben der Instrumentierung des Regals ist bei diesem Ansatz keine weitere Instrumentierung der Produkte notwendig.

(a) Instrumentiertes Einkaufsregal zur Darstellung von Produktinformationen nach Produktentnahme

(b) FibreShelf

***Abbildung 7.5**: Instrumentierte Regalsysteme*

Durch die eineindeutige Identifikation von Produkten können neben Nährwertangaben und Inhaltsstoffen auch weitere Informationen, wie z.B. Mindesthaltbarkeitsdaten (MHD), abgerufen und in den Diensten verwendet werden. Beispielsweise kann dies dazu verwendet werden, um die Preise abhängig vom MHD der Produkte dynamisch anzupassen. Um solche Änderungen direkt in die Umgebung zu transferieren, werden im IRL elektronische Preisauszeichnungen (ESL) eingesetzt, welche vom System verändert werden können. Dazu wird von einem Dienst aus dem Warenwirtschaftssystem heraus die Datengrundlage entnommen, die zur Erstellung des Inhalts eines Preisetiketts notwendig ist [Kahl, 2013a]. Anschließend wird anhand eines Templates das Etikett erstellt und als Bild zum ESL transferiert. Eine Preisänderung erfolgt durch die Versendung eines entsprechenden Events, woraufhin automatisch das neue Preisschild generiert und an das entsprechende ESL verschickt wird. Neben der Darstellung von Preisinformationen können ESL auch zur Darstellung weiterer Informationen eingesetzt werden, wie beispielsweise als Navigationshinweis, indem ein Pfeil in Richtung des gesuchten Produkts visualisiert wird, was ebenfalls durch die Versendung von Events geregelt wird.

Frischetheken stellen eine besondere Form von Regalsystemen dar. Bei der Kommunikation zwischen Kunden und Verkäufern werden bei Theken oftmals Zeigegesten verwendet, um gewünschte Produkte zu spezifizieren. Im IRL werden solche Zeigegesten erkannt. Basierend auf einem 3D-Modell der Theke, welches auch die einzelnen Produktplatzierungen umfasst, wird das Produkt ermittelt, auf welches der Kunde gerade zeigt. Dieses wird als Event an die elektronische Waage versendet, woraufhin das Produkt sowohl dem Kunden als auch dem Verkäufer auf den beiden Displays der Waage präsentiert wird. Neben Produktinformationen bekommt der Verkäufer weiterführende Daten, anhand derer er den Kunden kompetent beraten kann, wie beispielsweise eine Weinempfehlung zu einem Käse.

Zur transparenten Darstellung von Produktinformationen wurde zudem ein System entwickelt, welches es ermöglicht, den Inhalt von digitalen Produktgedächtnissen visuell zu präsentieren. Der sogenannte *SemProM-Browser* ermöglicht es Kunden, Einblicke in produktspezifische Informationen aus digitalen Produktgedächtnissen zu bekommen [Kröner et al., 2009]. Hierzu werden die Daten, welche eigentlich zum Verarbeiten durch Maschinen ausgelegt sind, in eine verständliche Darstellung transferiert, bei der der Kunde in einer Zeitleiste die einzelnen erfassten Stationen des Produktlebenszyklus dargestellt bekommt. Durch eine entsprechende Visualisierung werden Zusatzinformationen zu jedem einzelnen Event aus dem Produktlebenszyklus angezeigt. Somit ist es möglich, alle Informationen in einer Browser-Metapher zu durchforsten. Die Erkennung des Produkts, dessen Produktgedächtnis visualisiert werden soll, erfolgt über eine sensitive Oberfläche, welche mittels RFID die Produktinformationen ausliest und diese als Event an den Dienst zur Darstellung des Objektgedächtnisses sendet. Ein Beispielszenario ist die Darstellung des im Lebenszyklus eines Produkts erzeugten CO_2-Ausstoßes, welcher bei der Herstellung, dem Transport und der Lagerung anfällt [Kröner et al., 2010]. Dieser kann im Produktgedächtnis hinterlegt und mit Hilfe des SemProM-Browsers visualisiert werden. Solche Informationen sind für Kunden interessant, welche auf klimatische Belastungen Rücksicht nehmen.

Mobile Assistenzdienste Neben den im Markt installierten Sensoren und Aktuatoren bringen Kunden weitere Komponenten in Form ihrer Smartphones mit. Der Vorteil dieser Geräteklasse ist, dass diese privat und stark personalisiert ist. Während eine Darstellung privater Informationen auf einem Display in einem öffentlichen Einkaufsregal einen Eingriff in die Privatsphäre der Kunden darstellen kann, da diese Informationen auch für andere Kunden sichtbar sind, können Displays von Smartphones als private Bereiche angesehen werden, welche nicht von Dritten eingesehen werden können. Aus diesem Grund kann ein Abgleich von allgemeinen Produktinformationen mit dem persönlichen Benutzerprofil erfolgen und Zusatzinformationen mittels erweiterter Realität präsentiert werden, wie in Abbildung 7.6(a) zu sehen [Löchtefeld et al., 2010]. Beispielsweise können Produkte, die dem Benutzerprofil widersprechen virtuell durchgestrichen oder durch eine Warnung hervorgehoben werden, wohingegen passende Produkte durch eine Überblendung in Form eines Häkchens kenntlich gemacht werden können. Ebenso ermöglichen mobile Endgeräte eine Klassifizierung der Kunden. Selbst wenn diese vom Supermarkt zur Verfügung gestellt werden und somit keine persönlichen Informationen des Benutzers beinhalten, kann beispielsweise anhand der Sprache eines Kunden dessen Geschlecht und Altersgruppe ermittelt werden, worüber eine individualisierte Einkaufsberatung erfolgen kann [Feld und Kahl, 2008]. Mittels Text-to-Speech oder Links zu Webseiten können Aktuatoren von Smartphones mittels Events angesteuert werden. Des Weiteren kann das mobile Endgerät auch als Schlüssel zur Identifikation verwendet werden, was mittels der NFC-Technologie erfolgt. In Abbildung 7.6(b) erfolgt der Login des Kunden mittels NFC-Schnittstelle, woraufhin seine Einkaufsliste an den Einkaufswagen übertragen wird. Basierend auf einem Event, welches den Loginprozess kommuniziert, kann für die anderen Dienste eine Verknüpfung des Einkaufswagens mit dem Smartphone erfolgen, um Informationen an eine der beiden Plattformen oder beide gleichzeitig zu schicken.

(a) Personalisierte Darstellung von Informationen auf einem mobilen Endgerät

(b) Übertragung der Einkaufsliste vom Smartphone auf den Einkaufswagen

***Abbildung 7.6**: Mobile Endgeräte als Teil der IRL-Umgebung*

Kassenzone Der Kassenbereich stellt in der Regel den Abschluss eines Einkaufs im Supermarkt dar. Die Produkterfassung am *Easy Checkout* im IRL erfolgt aufgrund der Instrumentierung der Produkte mit RFID-Tags automatisch. Der Kunde bekommt auf einem Display an der Kasse die gescannten und zu zahlenden Produkte angezeigt, wie in Abbildung 7.7 dargestellt. Bezahlt werden kann über biometrische Sensorik (Fingerabdruck) oder über NFC, basierend auf dem Verfahren einer kontaktlosen Kreditkarte, welche beispielsweise im Autoschlüssel des Kunden integriert sein kann. Die unterschiedlichen Bezahlverfahren werden durch ihre Kodierung in Events gleich gestaltet, so dass eine Erweiterung durch zusätzliche Bezahlverfahren einfach möglich ist. Nach der Bezahlung werden Events generiert, welche das Drucken des Kassenbons und das Öffnen des Ausgangstors anstoßen. Neben der Bezahlung in einem festgelegten Bereich (Kassenzone) können die Produkte auch mittels mobiler Bezahlverfahren (Mobile Payment) im Markt bezahlt werden [Kahl und Paradowski, 2013]. Hierbei wird der zu zahlende Betrag entweder über die NFC-Schnittstelle oder mittels eines Events vom Einkaufswagen an das mobile Endgerät des Kunden gesendet. Dort kann die Überweisung des entsprechenden Betrags ausgelöst werden, woraufhin ein Bestätigungscode zurückgeschickt wird, welcher anschließend vom Smartphone an den Einkaufswagen transferiert wird. Hierzu dient die NFC-Schnittstelle der beiden Geräte. Daraufhin wird ein Event versendet, welches die entsprechenden Produkte als gekauft markiert, was in dessen digitalem Objektgedächtnis vermerkt wird, und ihre Sicherheitsetiketten deaktiviert, so dass beim Verlassen des Marktes kein Alarm ausgelöst wird [Kahl et al., 2013a]. Die Zuordnung der bezahlten Produkte erfolgt mittels der eineindeutigen Identifikation der Objekte.

***Abbildung 7.7**: Easy Checkout*

7.2 Integration des DR-Frameworks in das IRL

Um den Kontext des IRL und Benutzerinteraktionen zu erfassen und zu verarbeiten, müssen die zur Verfügung stehenden Dienste Zugriff auf alle relevanten Sensorinformationen haben. Ebenso ist als Reaktion auf diese Verarbeitung eine mögliche Ansteuerung der entsprechenden Aktuatoren seitens der Dienste erforderlich. Für die Kommunikation dieser vernetzten Umgebung wurde im Anfangsstadium am IRL der iROS Event Heap von der Stanford Universität verwendet, ein eventbasiertes Publish/Subscribe-System (siehe Kapitel 3.1.2) [Johanson und Fox, 2002]. Um auch Smartphones als Teil der Kommunikationsumgebung zu unterstützen und Vorverarbeitungen in die Kommunikationsinfrastruktur integrieren zu können, wurde die iROS-Infrastruktur später durch den im Rahmen dieser Arbeit entwickelten Event Broadcasting Service (EBS) ersetzt. Dieser implementiert auch die direkten Kommunikationsschittstellen zwischen Diensten, welche zunächst nach dem Server-Client-Prinzip aufgebaut wurden. Aufgrund dieser Server-Client-Struktur musste in der Anfangsphase des IRL bei den Systemstarts eine vorgegebene Reihenfolge eingehalten werden, damit die Server vor den Clients gestartet wurden, was durch die Umstellung auf die eventbasierte Kommunikation nicht mehr notwendig ist. Im Zuge der Umstellung wurden alle Sensoren, Aktuatoren und Dienste als separate Clients im EBS integriert, so dass diese einzeln angesprochen werden können.

Die verteilten Sensoren im IRL dienen als potentielle Eingabequellen für die diversen Systeme und Assistenzapplikationen. Beispielhaft kann hier das Regalberatungssystem betrachtet werden, welches nach der Entnahme eines Produkts zusätzliche Informationen zum Produkt auf einem Display präsentiert. Wird das entnommene Produkt im Anschluss in den Einkaufswagen gelegt, so wird davon ausgegangen, dass eine Kaufentscheidung getroffen wurde. Daher wird die Informationspräsentation nicht mehr benötigt und nicht

mehr auf dem Display dargestellt. Neben der Präsentation auf dem öffentlichen Display kann die Darstellung auch auf dem mobilen Endgerät des Kunden erfolgen. Um festzustellen, welcher Kunde die Interaktion durchführt, werden die Kundenpositionen benötigt. Dies kann anhand der Positionierung des Einkaufswagens erfolgen, an dem sich der Kunde eingeloggt hat. Dadurch kann die Kundenposition über die Position dieses Einkaufswagens approximiert und in Verbindung mit der Interaktion am Regal gebracht werden [Kahl et al., 2013b]. Die Positionserfassung von Objekten mittels RFID wird durch einen entsprechenden Dienst realisiert, welcher auf dem Server läuft. Dieser Dienst empfängt die Events, wenn RFID-Tags in Lesefeldern von RFID-Antennen erscheinen oder verschwinden. Da Produkte mehrere RFID-Tags umfassen können, ist ein Verschwinden eines Tags aus dem Lesefeld eines Readers nicht mit einer Entnahme des Produkts gleichzusetzen. Aus diesem Grund wurde dieser spezielle Dienst entwickelt, welcher ermittelt, wann tatsächlich ein Produkt entnommen oder zurück gestellt wird. Die erfasste Produktposition ist hierbei abhängig von der Anbringung der entsprechenden Reader, zum Beispiel an einem Regalboden oder am Einkaufswagen. Die daraus resultierenden Positionsangaben können auf unterschiedlichen Detaillierungsstufen ermittelt und kommuniziert werden. Beispielsweise kann eine Produktposition in Bezug auf einen Regalboden oder das zugehörige Regal erfolgen. Im ersten Fall liegt jedoch eine höhere Fehleranfälligkeit vor, da aufgrund von größeren Antennenreichweiten die Position nicht exakt ermittelt werden kann und es daher zu einer Fehlinformation kommt. Dieses wird entsprechend über den Konfidenzwert im Event übermittelt. Ebenso erfasst dieser Positionierungsdienst, wenn ein Objekt von mehreren RFID-Antennen erkannt wird und kann darüber genauere Rückschlüsse auf dessen Position weitergeben. Ein solches Management der unterschiedlichen RFID-Lesepunkte ist hierbei essenziell, um möglichst genaue Aussagen über Produktpositionen treffen zu können [Lindsay et al., 2003].

Wie bereits in Kapitel 5.2.2 beschrieben, ist in DURMAD eine Schnittstelle zu den digitalen Objektgedächtnissen (OMM) integriert. Basierend auf den Sensorinformationen der CPE werden mittels des DR-Frameworks die jeweiligen Objektgedächtnisse im IRL automatisch fortgeschrieben. Hierzu wurde ein entsprechender Dienst entwickelt, welcher alle ermittelten Daten, die über Events im IRL versendet werden und den jeweiligen Objekten zugeordnet sind, in die entsprechenden digitalen Objektgedächtnisse integriert. Dadurch wird beispielsweise Interaktion mit Produkten dokumentiert. Jedes Mal, wenn ein Produkt aus einem Regal entnommen und wieder zurückgestellt wird, wird ein entsprechender Zähler in dessen Objektgedächtnis inkrementiert. Aufgrund der Anzahl solcher Entnahmen und Rückstellungen können spezifische Rückschlüsse über die Produkte getroffen und diese dem Produzenten zurückgemeldet werden. Beispielsweise zeigt eine häufige Entnahme, dass das Produkt im Allgemeinen für Kunden interessant ist. Wird es des Öfteren wieder zurück gestellt, zeigt dies, dass gewisse Faktoren, wie beispielsweise die Inhaltsstoffe oder der Preis, nicht den Vorstellungen der Kunden entsprechen. Wird das Produkt gar nicht entnommen, ist eventuell dessen äußeres Erscheinungsbild oder die Position innerhalb des Markts nicht geeignet.

Bei der Integration des DR-Frameworks im IRL wurden DURMAD (siehe Abschnitt 7.2.1) und der EBS (siehe Abschnitt 7.2.2) erweitert, um für das Anwendungsge-

biet spezifische Funktionalitäten zu umfassen. Zudem wurden spezielle BI-Dienste und Geschäftsprozesse in das Framework integriert (siehe Abschnitt 7.2.3). Exemplarisch wird der Einsatz des DR-Frameworks im IRL an der Beschreibung eines Produktberaters aufgezeigt (siehe Abschnitt 7.2.4).

7.2.1 DuRMaD im IRL

Die Objekte des IRL, wie beispielsweise Regale, Regalböden, Tische, Einkaufswagen und Produkte, wurden modelliert und im XML3D-Format exportiert. Basierend auf einem Grundrissplan des IRL, welcher in Form von gml-Dateien vorliegt, wird nach einem Systemstart automatisch das entsprechende 3D-Modell erzeugt (siehe Abbildung 7.8, links). Hierbei beinhalten die gml-Dateien zusätzliche Informationen, wie beispielsweise die Anzahl von Regalbrettern bei einem Regal und deren Positionen, so dass die Virtualisierung des Regals darüber automatisiert erfolgen kann. Zudem enthalten die gml-Dateien die Ausmaße der Objekte, so dass durch eine Skalierung der XML3D-Modelle diese auch für unterschiedliche Typen verwendet werden können, wie beispielsweise Tische in verschiedenen Ausmaßen. In einem nächsten Schritt werden Produktpositionen aus einer Datenbank geladen, wobei diese nur durch Regal und Regalboden spezifiziert sind. Darauf basierend werden die entsprechenden Produktmodelle nach einem vordefinierten Algorithmus in den Regalen positioniert (siehe Abbildung 7.8, rechts). Dieser Algorithmus beachtet einerseits die Anzahl der Produkte im Regal, die zum Kunden hin ausgerichtet sind, was unter dem Begriff *Facing* bekannt ist. Andererseits berücksichtigt er auch die maximale Anzahl von Stapelungen und die Dimensionen des Regalbodens und der Produkte. Durch die hinterlegte Form der Produkte wird gleichzeitig auch ermittelt, ob diese stapelbar sind oder nicht. Produkte des gleichen Typs werden zusammen platziert und die Positionierung der Objekte beginnt von vorne nach hinten. Dies hat den Hintergrund, dass für ein ansprechendes Layout die Produkte in der Regel möglichst weit vorne positioniert werden, um so am besten vom Kunden erkennbar zu sein. Die Funktionalität wurde hierbei generisch implementiert, so dass es möglich ist, auch andere Positionierungsalgorithmen zu integrieren. Beispielhaft wurde das Verfahren implementiert, Produkte auch von hinten nach vorne zu platzieren, wie es normalerweise beim Nachfüllen mit neuen Produkten durch Mitarbeiter im Markt erfolgt. Während das 3D-Modell erstellt wird, wird auch die Ontologie geladen, welche die semantische Beschreibung der einzelnen Objekte des IRL beinhaltet, sowie die weiteren Visualisierungsformen und das Datenlager mit den entsprechenden Daten gefüllt. Nachdem alle Produkte platziert wurden, ist die Initialisierungsphase abgeschlossen.

Um mit DuRMaD interagieren zu können, muss der Benutzer sich authentifizieren, woraufhin seine Benutzerrolle und die zugehörigen Rechte geladen werden. Diese legen unter anderem fest, ob Preise von Produkten aus DuRMaD heraus verändert werden dürfen. Des Weiteren werden die Login-Daten verwendet, um die zugehörigen Aufgaben zu ermitteln sowie die automatische Authentifizierung und das zugrundeliegende Rechtemanagement bei der Verwaltung der Regeln zu ermöglichen (siehe Kapitel 5.4.1). Neben der Selektion einzelner Produkte, um weiterführende Informationen darüber zu erhalten, können mittels der Zuordnung von Virtualisierung zu semantischer Annotation auch Aktuatoren angesprochen werden. So können beispielsweise durch Interaktion mit dem Modell ein im

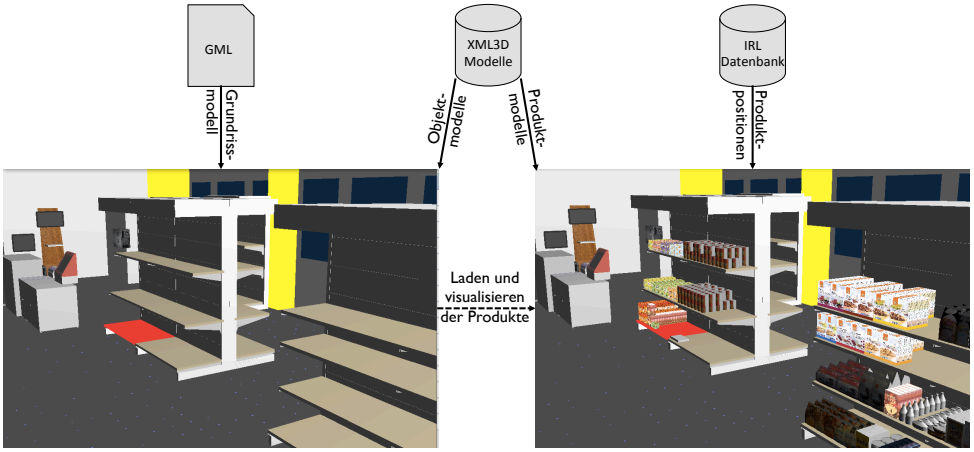


***Abbildung 7.8**: Initialisierungsphase bei* DURMAD *im IRL*

IRL installiertes Schnelllauftor oder das Ausgangstor im Kassenbereich einfach geöffnet und geschlossen werden. Ebenso ist es darüber möglich, Alarme zu triggern oder Textnachrichten mittels Text-to-Speech in der Umgebung oder auf mobilen Endgeräten ausgeben zu lassen. Neben den in der Ontologie fest vordefinierten Events können in DURMAD ebenfalls manuell Events erstellt werden, sofern diese von dem jeweiligen Aktuator unterstützt werden. Hierzu wird ein entsprechendes grafisches Kontextmenü zur Verfügung gestellt, das durch einen Rechtsklick auf die Virtualisierung des Aktuators im 3D-Modell geöffnet wird (siehe Abbildung 7.9). Neben möglichen Objektmanipulationen kann hierüber eine Ansteuerung des Aktuators erfolgen, sofern diese a priori in der Ontologie spezifiziert wurde. Einerseits können bereits über dieses Menü Events direkt versendet werden, welche in der Ontologie als relevant spezifiziert wurden. Andererseits lässt sich darüber ein zweites Menü öffnen, welches alle in der Ontologie vordefinierten Eventausprägungen sowie die vom Aktuator unterstützten Eventtypen beinhaltet. Diese können vom Benutzer ausgewählt werden, woraufhin sie zu einer Eventsequenz hinzugefügt werden. Diese kann schließlich verwendet werden, um die hinterlegten Events zu versenden und damit den Aktuator zu steuern.

In der textuellen Darstellung von Produktinformationen kann eine Anpassung des Preises erfolgen, vorausgesetzt, der aktuelle Benutzer hat eine entsprechende Berechtigung. Diese Anpassung wird mittels eines Events versendet, so dass alle Dienste diese Veränderung registrieren und entsprechend darauf reagieren. Beispielsweise wird über die Schnittstelle zu den elektronischen Preisschildern der neue Preis direkt in der physischen Umgebung angezeigt. Ebenso wird der Eintrag im Warenwirtschaftssystem angepasst, so dass dieser auch bei der Bezahlung berücksichtigt wird. Gleichzeitig wird auch auf der Virtualisierung des ESL im 3D-Modell der neue Preis dargestellt. Neben der Synchronisation der Darstellung von ESL ist auch die Visualisierung von Bildschirminhalten mittels einer VNC-Schnittstelle realisiert. So kann im virtuellen Modell der gleiche Inhalt wie in der realen Umgebung dargestellt werden. Schließlich werden im IRL erkannte Aktionen im 3D-Modell abgebildet. Beispielsweise wird die Entnahme eines Produkts durch das Entfernen seiner Virtualisierung

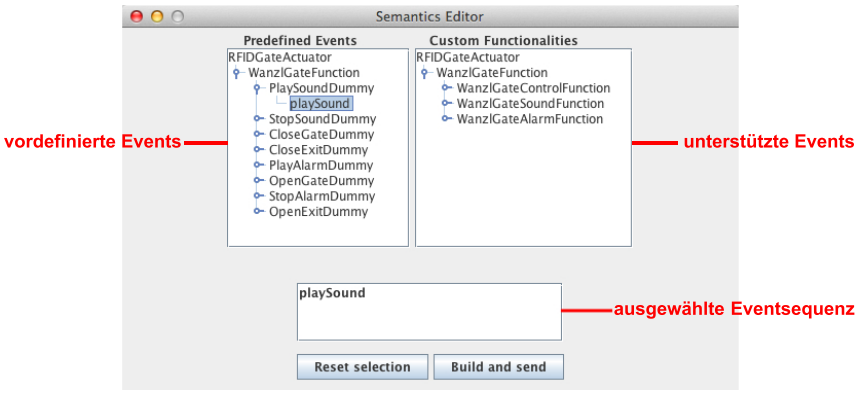


***Abbildung 7.9**: Grafischer Editor zur Aktuatoransteuerung aus* DURMAD *in Abhängigkeit von der definierten Ontologie*

realisiert. Das Produkt erscheint erst dann wieder an einer entsprechenden Position, sobald es erneut durch Sensoren erfasst wird, wenn es beispielsweise wieder ins Regal zurückgestellt wird. Analog wird die Position des Einkaufswagens bei DURMAD im Modell automatisch aktualisiert, wenn der Positionierungsdienst eine neue Lokation meldet.

Als Simulation und Erweiterung des aktuellen Zustands der Umgebung kann die Position des Einkaufswagens im virtuellen Modell verändert werden, um darüber lokationsbasierte Dienste evaluieren zu können. Ebenso ist es möglich, virtuell Produkte zu verräumen. Dies ist beispielsweise hilfreich, um eine Regalbestückung in Form eines Planogramms zu erstellen. Planogramme sind Darstellungen, wie ein Regal befüllt sein sollte, und ermöglichen es, eine automatisierte Produktbestellung zu realisieren oder die Informationen an externe Personen weiterzugeben, die daraufhin die Regale im Markt mit Produkten bestücken können [Murray et al., 2010]. Applikationen, welche die virtuelle Befüllung von Regalen mit unterschiedlichsten Assistenzfunktionalitäten unterstützen, werden als „Spaceplanner“ bezeichnet (siehe Kapitel 1.1.2) [Hübner und Kuhn, 2012]. Neben der zuvor beschriebenen manuellen Befüllung von virtuellen Regalen wurde in DURMAD eine automatisierte Spaceplanning-Funktionalität integriert. Hierfür wird anhand einer Abverkaufsprognose der Ertrag eines Produkts abhängig von dessen Position im Markt ermittelt. Somit ist es möglich, den erwarteten Gesamtertrag eines kompletten Regals zu kalkulieren. Ziel ist es, diesen Gesamtgewinn zu maximieren, wobei einzelne Voraussetzungen erfüllt sein müssen. Beispielsweise sollten genügend Produkte verräumt sein, so dass vor der nächsten Lieferung möglichst keine Out-of-Stock-Situation eintritt. Die Abverkaufsprognose ist neben den vorherigen Abverkäufen auch von weiteren Faktoren abhängig. Beispielsweise werden Produkte in Augenhöhe eher verkauft als andere [Dreze et al., 1995]. Je mehr Fläche des Produkts der Kunde sieht, desto eher nimmt er dieses wahr und kauft es. Mehr Fläche wird hierbei durch Erhöhung des Facings ermöglicht. Da Produkte unterschiedlich breit sind, wird der eingenommene Platz sowohl durch die Facinganzahl als auch durch die Größe der Produktverpackung bestimmt. Da ab einer gewissen Fläche eine Erhöhung der Aufmerksamkeit des Kunden auf dieses Produkt nicht mehr so relevant ist, wird zusätzlich die vom Kunden

sichtbare Fläche betrachtet und nicht ausschließlich die Facinganzahl. Dies ist unter dem Begriff *Raumelastizität* (Space Elasticity) bekannt, welches die relative Flächenveränderung beschreibt, wenn eine weiteres Facing des Produkts hinzugenommen wird [Curhan, 1973]. Ebenso hat die prominente Darstellung eines Produkts einen nachteiligen Einfluss auf andere Produkte der gleichen Art. Nimmt ein Müsli des Herstellers A beispielsweise eine größere Fläche ein als das Müsli von Hersteller B, wird A eher von den Kunden wahrgenommen und daher auch mit einer größeren Wahrscheinlichkeit gekauft, wobei auch die Position der beiden Produkte betrachtet werden muss (Cross Space Location Elasticity).

Für das automatische Spaceplanning müssen in DURMAD in einem ersten Schritt die Regalbretter selektiert werden, welche berücksichtigt werden sollen. In einem zweiten Schritt werden die zu platzierenden Produkte über eine grafische Oberfläche spezifiziert, welche in Abbildung 7.10 dargestellt ist. In dieser Oberfläche können weitere produktbezogene Parameter für das Spaceplanning angegeben werden, wie beispielsweise das minimale und das maximale Facing eines jeden Produkts. Über einen Trendfaktor, wie beispielsweise einen aktuell beworbenen Artikel, kann eine erhöhte Abverkaufsprognose indiziert werden. Ebenso ist es möglich, die maximale Anzahl an Produkten, die übereinander gestapelt werden sollen, anzugeben. Anhand ihrer äußeren Form wird bei gewissen Produkten, wie beispielsweise Flaschen, eine Stapelung bereits von der Applikation ausgeschlossen. Als letzter Parameter kann angegeben werden, ob ab einer bestimmten Produktmenge ein Rabatt auf den Einkauf gewährt wird, da dieses die Gewinnmarge des Produkts um den entsprechenden Rabatt erhöht und somit den erwarteten Gesamtgewinn des Regals beeinflusst. Um eine optimale Befüllung des Regals zu ermitteln, bedarf es eines zu großen Zustandsraums, welcher alle möglichen Facinganzahlen und Positionen der Produkte umfasst, als dass dieser vollständig abgedeckt werden kann. Aus diesem Grund wurden Heuristiken implementiert, welche eine Generierung einer neuen optimierten Befüllung in Abhängigkeit von dem zuletzt ermittelten Befüllungszustand ermöglichen. Beispielsweise kann das Facing eines Produkts verringert und dafür das des Nachbarn erhöht werden. Die integrierten Heuristiken können hierbei manuell zu- oder abgeschaltet werden. Schließlich können Optimierungsparameter angegeben werden. Da bisher in keiner Studie nachgewiesen werden konnte, dass die horizontale Position von Produkten einen Einfluss auf den Abverkauf hat [Dreze et al., 1995], besteht die Möglichkeit, Produktgruppen in einem Regalbrett untereinander zu vertauschen. Dies kann in einem nachgelagerten Prozessschritt erfolgen, mit Hilfe dessen eine Sortierung nach Marke oder Art ermöglicht wird. Die maximale Laufzeit des Systems kann durch den Benutzer festgelegt werden, wobei eine größere Berechnungszeit das Expandieren mehrerer Knoten und damit mehrerer möglicher Regalbefüllungen ermöglicht. Zudem können über Parallelisierung der Anfragen mehrere mögliche Regalbefüllungen zur gleichen Zeit betrachtet und analysiert werden. Über Skalierungsparameter kann angegeben werden, wie stark die Space Elasticity, Cross Space Location Elasticity, die Anzahl der Facings und die Positionierung (Augenhöhe oder nicht) bei der Berechnung des erwarteten Gesamtgewinns einbezogen werden sollen. Zudem kann eine bereits manuell erfolgte Bestückung des Regals bei der Automatisierung übernommen werden, so dass ein Benutzer gewisse Produkte bereits platziert, da diese Produktposition beispielsweise vom Hersteller bezahlt wurde. Über Menüknöpfe kann die Berechnung

gestartet, gestoppt und zurückgesetzt werden. In der Regel werden Produkte überspezifiziert. Dies bedeutet, dass mehr Produkte ins Regal verräumt als verkauft werden. Daher bietet der Algorithmus nicht nur eine Befüllung des Regals an, sondern n viele, welche den gleichen erwarteten Gesamtgewinn aufweisen. Diese unterscheiden sich nur in geringem Maße und bieten daher noch Alternativen für Entscheidungsträger, da z.B. eine ästhetisch ansprechende Bestückung ausgesucht werden kann. Schließlich kann das gewünschte Modell in Form einer Webseite als Planogramm exportiert werden. Hierzu wird die virtuelle Kamera automatisch so im Modell ausgerichtet, dass die Regale möglichst senkrecht zur Sichtachse liegen und komplett sichtbar sind. Dieser Bildausschnitt wird der Webseite hinzugefügt. Ebenso werden automatisch Tabellen erzeugt, welche die genauen Produktpositionen inklusive wichtiger Produktinformationen, wie Name, EAN, Facing und Stapelung und die logistischen Maße, beinhalten. Die Planogrammdarstellung orientiert sich an Planogrammen, wie sie von der Warenhauskette Globus verwendet werden. Ein resultierendes Bild und eine entsprechende Tabelle sind im Anhang zu finden (siehe Abbildung A.2 und Tabelle A.1).

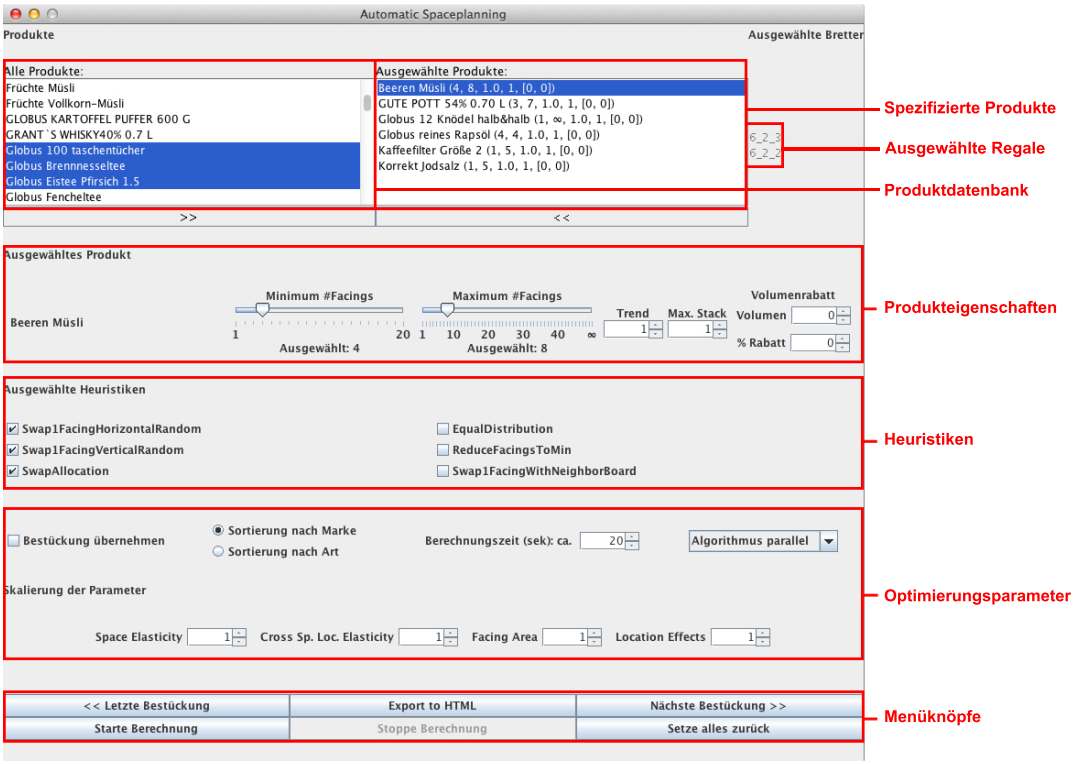


***Abbildung 7.10**: Grafische Oberfläche zur Eingabe der Parameter für ein automatisches Spaceplanning*

Neben der Erweiterung von DURMAD wurden auch Ergänzungen am EBS vorgenommen, welcher spezifisch auf die Bedürfnisse des Innovative Retail Laboratory angepasst wurde. Diese werden im folgenden Abschnitt erläutert.

7.2.2 Spezialisierung des EBS

Durch die Integration des EBS ins IRL wurden Monitoring- und Steuerungsmöglichkeiten auf verschiedenen Ebenen realisiert. Eine Spezialisierung ist die Betrachtung von physischen Rechnern als einzelne Geräte, die mehrere Dienste umfassen können. Hierbei stellen die an einen Rechner angeschlossenen Sensoren und Aktuatoren ebenfalls einzelne Dienste dar. Sowohl der Rechner als auch die einzelnen Dienste werden jeweils mittels eines Clients mit dem Server verbunden, wobei der Dienst des Rechners besonders ist, da er direkten Zugriff auf die anderen Dienste hat. Er kann diese sowohl starten als auch stoppen. Somit wurde ein agiles System realisiert, welches Anpassungen zur Laufzeit erlaubt. Als eine Schnittstelle für diese Steuerung dient ein Webservice, der Informationen dazu bereitstellt, welche Rechner aktuell angemeldet sind und welche Dienste jeweils auf diesen laufen. Durch eine Schnittstelle zu diesem Webservice kann beispielsweise der Verbindungsstatus der einzelnen Systeme auf einer speziellen Internetseite dargestellt werden. Über diese Seite können zudem Dienste (neu-)gestartet oder gestoppt werden. Ebenso ermöglicht die Seite das Hoch- und Herunterfahren einzelner Rechner. Beides erfolgt über Events, wobei das Hochfahren als Wake-on-Lan (WoL) Aufruf durch einen serverseitigen Dienst realisiert wird und das Herunterfahren autonom durch die Clients nach Erhalt eines „Shut-down“-Events erfolgt. Ebenso können Befehle versendet werden, die eine automatische Aktualisierung der Systeme mit anschließendem Neustart zur Folge haben. Hierbei werden die aktualisierten Dienste und Schnittstellen aus einem Versionierungsdienst (Subversion) geladen. Dies erfolgt auch automatisch, wenn bei der Initialisierung festgestellt wird, dass die Version des Clients nicht zu der des Servers passt. Somit wird die gesamte CPE automatisiert auf den aktuellen Status gebracht.

Des Weiteren umfassen die Clients im IRL größtenteils automatisierte Verfahren zum Umgang mit Fehlfunktionen. Beispielsweise wird eine Notifikation versendet, sobald ein Problem bei der Kommunikation mit einem RFID-Reader erkannt wird. Diese erscheint auch in DuRMAD, so dass der Manager der Umgebung entsprechende Handlungen einleiten kann. Zur Erkennung von Fehlfunktionen dienen unter anderem die Monitoring-Filter, die Fehlverhalten automatisch in Form von Notifikationen kommunizieren. Alle versendeten Events werden in einer Datenbank protokolliert. Zur Darstellung der letzten Einträge wird ein Webinterface bereitgestellt, so dass Kommunikationsfehler darüber ermittelt werden können. Zusätzlich kann der Kommunikationsverlauf in DuRMAD grafisch dargestellt werden (siehe Kapitel 6.5.1). Ein weiteres Webinterface ermöglicht das Erstellen und Versenden von Events unabhängig von einer Implementierungsumgebung. Die Erzeugung von Simulationsevents ist vor allem zum Testen von Applikationen hilfreich. Durch die Verwendung verschiedener Serialisierer kann auch über eine REST-Schnittstelle das Event als formatierter Text übergeben werden. Dieser Text wird von einem entsprechenden Dienst automatisch in ein Event umgewandelt und anschließend versendet. Somit können auch externe Applikationen Events versenden, welche keine Schnittstelle zum EBS implementiert haben. Bisher integrierte Beispiele zur Übertragung von Events in textueller Form sind XML- und JSON-Kodierungen. Durch den Einsatz der Filter wurde die Anzahl der versendeten Events reduziert. Beispielsweise kommunizieren Positionierungsdienste und Temperatursensoren durch einen einfachen Inhaltsfilter nur Veränderungen und somit

weniger redundante Daten.

Die Integration des EBS im IRL vereinfacht die Verwendung von multimodalen und autonomen Applikationen. Durch die Versendung der einzelnen heterogenen Sensorinformationen ist es möglich, unterschiedliche Sensorkombinationen und Sensorfusionen zu realisieren. Durch die Codierung in Events können unterschiedliche Modalitäten (siehe Kapitel 2.1.1) als gleichwertige Eingabe für verarbeitende Dienste eingebunden werden, wie beispielsweise Gesten, Schrift oder Sprache. Ebenso können darüber Reaktionen über diverse Ausgabemodalitäten wiedergegeben werden, z.B. mittels Text, Sprache oder Gesten. Die Wahl der jeweiligen Modalität kann hierbei explizit durch den Dienst festgelegt oder von den Aktuatoren selbstständig gewählt werden. Somit kann die gesamte IRL-Umgebung mit dem Zusammenspiel der einzelnen Komponenten als ein Agentensystem verstanden werden (siehe Kapitel 2.1.5). So kann beispielsweise eine automatische Preisanpassung abhängig von Mindesthaltbarkeitsdaten oder der Frische von Produkten erfolgen [Kahl, 2013a]. Die neu kalkulierten Preise werden in diesem Fall als Event versendet, woraufhin das aktualisierte Preisschild generiert und auf das zugehörige ESL transferiert wird. Ebenso kann die CPE autonom auf einen auftretenden Alarm reagieren, indem es eine entsprechende Kamera mittels Events auf das System ausrichtet, welches den Alarm ausgelöst hat, und ein Foto als Event an das mobile Endgerät eines zuständigen Marktmitarbeiters sendet. Dieser kann daraufhin geeignete Maßnahmen einleiten, wie das Verschließen von Türen, was ebenfalls über die Eventkommunikation realisiert werden kann, sofern dieses Vorgehen nicht bereits durch ein automatisiertes Verfahren realisiert wurde. Tabelle 7.1 zeigt die Verknüpfung der in diesem Kapitel beschriebenen Dienste im IRL anhand der wichtigsten Events auf. Die ausgehenden Events werden, sofern nicht an spezifische Clients adressiert, von allen anderen Diensten empfangen, die sich für diese Events registriert haben. In der Tabelle wird dies durch entsprechende eingehende Events dargestellt. Somit entsteht eine lose Verknüpfung zwischen den Diensten, welche über Events miteinander kommunizieren können.

	ShoppingListEvent	RFIDEvent	ObjectLocationEvent	ErrorEvent	QuuppaEvent	AcousticEvent	URLEvent	ObjectOrientationEvent	TemperatureEvent	ActuatorOnOffEvent	MoveCameraEvent	OMMLUpdateEvent	PriceChangedEvent	MessageEvent	ESLEvent	PointingEvent
Einkaufsvor- und -nachbereitung	⇒ ⇐	⇐	⇒	⇐		⇒										
IRL SmartCart	⇒	⇐	⇒	⇐	⇒	⇐					⇐	⇐				
Regal- und Kioskysteme		⇐	⇒	⇐		⇐	⇐	⇐ ⇒	⇐ ⇒	⇒	⇐ ⇒	⇐	⇐ ⇒	⇐	⇐ ⇒	⇐
Mobile																
Assistenzdienste	⇒					⇒	⇒									
Kassenzone		⇐	⇒	⇐		⇒						⇐		⇒		
Server	⇐ ⇒	⇐ ⇒	⇐ ⇒	⇒	⇐	⇒		⇒	⇒	⇐		⇐ ⇒	⇒	⇒		
DURMAD	⇒	⇒	⇒	⇒	⇒	⇐ ⇒	⇐ ⇒	⇒	⇐ ⇒	⇐	⇐ ⇒	⇐ ⇒	⇐ ⇒	⇐	⇒	⇒

	UserPaledEvent	MotionWalletEvent	GateEvent	RollerShutterEvent	ReportEvent	TaskEvent	ProcessTaskEvent	RuleEvent	RuleFiredEvent	StartServiceEvent	StopServiceEvent	ShutDownEvent
Einkaufsvor- und -nachbereitung										⇒	⇒	⇒
IRL SmartCart	⇒ ⇐			⇐						⇒	⇒	⇒
Regal- und Kioskysteme										⇒	⇒	⇒
Mobile	⇐	⇐		⇐						⇒	⇒	⇒
Assistenzdienste	⇒	⇒	⇐	⇐	⇐ ⇒	⇐ ⇒	⇒	⇐ ⇒	⇐ ⇒	⇒	⇒	⇒
Kassenzone		⇒	⇐		⇐ ⇒	⇐ ⇒		⇐	⇐	⇐	⇐	⇐
Server	⇐ ⇒	⇒	⇐ ⇒	⇐ ⇒	⇐ ⇒	⇐ ⇒	⇒	⇐ ⇒	⇒	⇐ ⇒	⇐ ⇒	⇐ ⇒
DURMAD	⇐ ⇒	⇒	⇐ ⇒	⇐ ⇒	⇐ ⇒	⇐ ⇒	⇒	⇐ ⇒	⇒	⇐ ⇒	⇐ ⇒	⇐ ⇒

⇓: Eingehende Events
⇑: Ausgehende Events

***Tabelle 7.1**: Eventkommunikation zwischen den Diensten im IRL*

7.2.3 Integration von Geschäftsprozessen und BI-Diensten

Eine spezielle Erweiterung des DR-Frameworks stellt die Integration von Geschäftsprozessen dar. Prozessmodelle, die im XPDL-Format (XML Process Definition Language) vorliegen, können in DURMAD gelesen und entsprechend dargestellt werden. Ein Auszug einer XPDL-Datei ist im Anhang in Programmauszug A.8 zu finden. Das XPDL-Format ist graphenorientiert und setzt sich aus Aktivitäten (Activity) und Verbindungen (Transition) zusammen. Aktivitäten sind hierbei einzelne Prozessschritte (Tasks), die über Kanten miteinander verbunden sind und somit einen Ablauf widerspiegeln. Neben eindeutigen Identifikationsschlüsseln und Namen umfassen Aktivitäten auch Informationen zu ihrer visuellen Darstellung. Diese werden von DURMAD ausgelesen und in ein Prozess-Diagramm konvertiert, welches parallel zum 3D-Marktmodell angezeigt werden kann, wie in Abbildung 7.11 dargestellt.

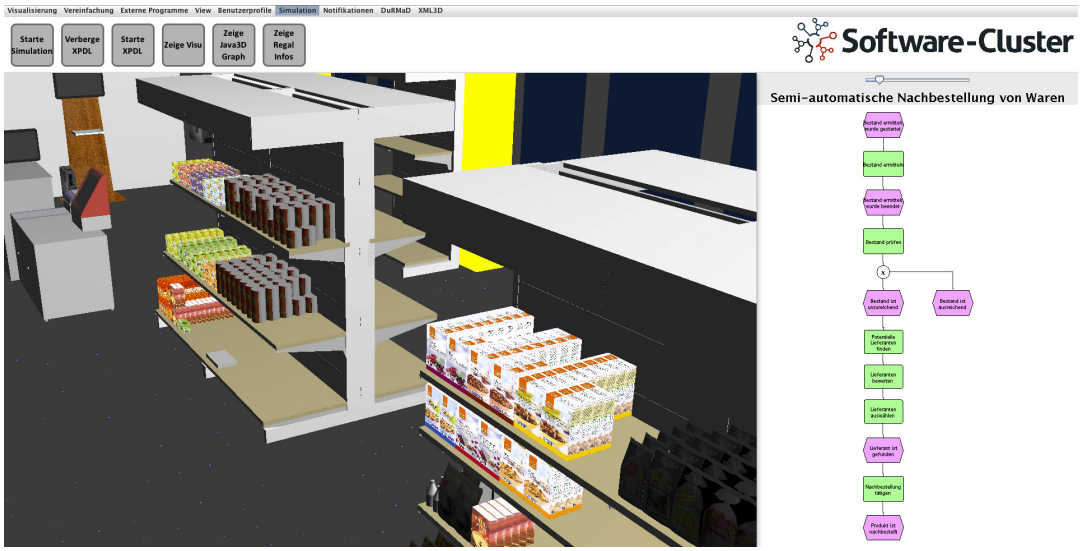


***Abbildung 7.11**: Darstellung eines Prozessmodells in* DURMAD

Mit Hilfe dieser Darstellung ist es auch möglich, den Task hervorzuheben, welcher gerade aktiv ist. Dies wird durch eine Verbindung zur Ausführungsumgebung von Prozessmodellen realisiert. Diese kommuniziert die Bearbeitung eines Prozessschritts als Event, welches in DURMAD entsprechend verarbeitet wird und den kommenden Prozessschritt visuell hervorhebt. Ebenso ist es aus DURMAD heraus möglich, automatisiert einen Prozess zu starten. Hierfür kann z.B. das Regelsystem eingesetzt werden, welches, sobald eine Mindestmenge an Produkten in einem Regal unterschritten wird, den entsprechenden Nachbestellprozess startet. Zur Ermittlung des aktuellen Bestands wird in diesem Fall die Schnittstelle zu DURMAD von der Prozessausführungsumgebung verwendet, wodurch die tatsächliche Produktmenge auf der Fläche und im Lager ermittelt werden kann. Hierbei werden auch mögliche Zweitplatzierungen berücksichtigt, so dass bei ausreichendem, aber ungleich verteiltem Bestand eine Information an die zuständigen Mitarbeiter versendet wer-

den kann, damit diese die Ware entsprechend umplatzieren. Durch diese Implementierung ist es möglich, die Reaktionen der Umgebung auf Kontextveränderungen auch in Form von automatisierten Prozessen zu modellieren. Die einzelnen Prozessschritte können hierbei durch das Versenden von Events Aktuatoren in der Umgebung ansteuern.

Dieses Vorgehen wurde in der Realisierung von Eventabläufen umgesetzt. Um solche Reaktionen automatisch ausführen zu können, wurde ein System entwickelt, mit dem eine Abfolge von Eventversendungen spezifiziert werden kann. Hierbei kann eine Zeitspanne zwischen versendeten Events hinterlegt werden oder ein Trigger, welcher ein Fortführen veranlasst. So können Eventabläufe manuell mit Hilfe einer grafischen Oberfläche zusammengestellt werden. Als weitere Alternative besteht die Möglichkeit, erfasste Sequenzen von Events, welche in der Historie hinterlegt sind, in einen solchen Eventablauf zu überführen. Dieser steht ab diesem Zeitpunkt mittels einer Abspielfunktion zur Verfügung und kann zu jeder Zeit erneut abgespielt werden. Somit ist es beispielsweise möglich, nacheinander mehrere Interaktionssequenzen in der realen Umgebung durchzuführen und diese gleichzeitig zu speichern. In diesem Fall werden die Handlungen aufgenommen und nicht manuell modelliert, was oftmals ein einfacheres Vorgehen bedeutet. Anschließend können die resultierenden Eventabläufe parallel ausgeführt werden, wodurch eine Simulation mehrerer Personen realisiert wird. Die Eventabläufe werden als Prozesse aufgefasst und abgearbeitet.

Gerade im Bereich des Handels spielen pro-aktive Handlungsempfehlungen und die Aufbereitung von gesammelten Daten eine große Rolle, wie beispielsweise eine frühzeitige Warennachbestellung oder Information an die Mitarbeiter zum Nachfüllen von Produkten. Aus diesem Grund werden Business Intelligence (BI) Systeme eingesetzt, um potentielle Handlungsnotwendigkeiten zu erkennen (siehe Abschnitt 2.5.3). Klassische BI-Dienste werden normalerweise für die (grafische) Aufbereitung und somit den Report von gesammelten Informationen eingesetzt. In Kombination mit modernen Ansätzen, wie beispielsweise CEP-Systemen, besteht die Zielstellung darin, dass die zugrunde liegenden Informationen mit aktuellen Daten aus der Umgebung verknüpft werden können, um daraus Handlungsempfehlungen zu generieren oder sogar automatisch Aktionen ausführen zu lassen [Wormus, 2008]. Ziel dabei ist, dass so viele Handlungsanweisungen wie möglich automatisiert werden und dort, wo menschlicher Handlungsbedarf besteht, eine Unterstützung hinsichtlich der potentiellen Handlungsschritte gegeben wird. Dazu müssen frühzeitig Anomalien erkannt und erfasst werden. Beispielsweise wird dies in der Finanzdomäne eingesetzt, um unübliche Transaktionen zu identifizieren. BI in Kombination mit entsprechenden CEP-Systemen bietet daher pro-aktive, automatisierte, kontextbasierte Handlungs- und Entscheidungsunterstützung und nimmt damit einen operationalen Einfluss im Gegensatz zu klassischen BI-Diensten, die lediglich analytische Aufgaben erfüllen. Mögliche Handlungen sind hierbei das Versenden von Notifikationen, das Zwischenspeichern von Informationen an geeigneter Stelle zur weiterführenden Auswertung oder die Durchführung anderer automatisierter, domänenspezifischer Aktionen. Im IRL wird ein BI-Dienst unter anderem verwendet, um Produkte darzustellen, deren Bestand niedrig ist und solche, deren Mindesthaltbarkeitsdatum bald überschritten sein wird. Dies kann dazu verwendet werden, um einen Anstoß der „Auto-Disposition“ zu ermöglichen und um zu einer Reduktion der „Abschriftenquote“ beizutragen.

Des Weiteren finden BI-Dienste Einsatz beim automatisierten Starten von Prozessen. Insbesondere wurden im IRL BI-Dienste mit einem Regelsystem verknüpft, um entsprechende Visualisierungen und pro-aktive Reaktionen zu ermöglichen.

7.2.4 Beispielhafte Integration eines Produktberaterdienstes in das DR-Framework

In Abschnitt 7.1.2 wurden Dienste des IRL beschrieben, welche ins DR-Framework eingebunden wurden. Exemplarisch wird an dieser Stelle aufgezeigt, wie ein Produktberaterdienst in das DR-Framework eingebunden ist. Hierbei wird auf die verwendeten Events zur Ein- und Ausgabe, auf die Verbindung zu anderen Diensten und auf die 3D-Darstellung in DURMAD eingegangen.

Funktionsweise und Instrumentierung: Der Beraterdienst erkennt aufgrund von RFID-Technologie, sobald eine Weinflasche aus dem zugehörigen instrumentierten Regal entnommen wird und zeigt daraufhin entsprechende Informationen zu diesem Wein auf einem Display an. Gleichzeitig wird eine vordefinierte Sprachdatei, welche eine Erklärung zum Produkt beinhaltet, akustisch wiedergegeben. Durch eine Instrumentierung der Weinflaschen mit Temperatur- und Beschleunigungssensoren werden zusätzlich ihre Temperatur und Orientierung erkannt und ebenfalls bei der Darstellung der Informationen berücksichtigt. Das Zurückstellen in das Regal oder in den IRL SmartCart bewirkt, dass die Informationsdarstellung auf dem Display und die akustische Ausgabe gestoppt werden.

Events: Sowohl das Erkennen der Entnahme eines RFID-Tags aus dem Lesefeld der am Regal angebrachten RFID-Antenne als auch die Informationsdarstellung auf dem Display und das Abspielen von Sprachdateien wird beim Produktberatungsdienst mittels entsprechender Events gesteuert. Parallel dazu versendet ein weiterer Dienst, welcher die Temperatur- und Beschleunigungssensoren der Weinflaschen auswertet, die gemessenen Daten als Events. Hierbei werden Filter vorgeschaltet, die ausschließlich Veränderungen kommunizieren (siehe Kapitel 6.4.3). Da die Entnahme eines Produkts durch ein Event kommuniziert wird, welches ausschließlich das Entfernen eines RFID-Tags aus dem Lesefeld der Antenne notifiziert (RFIDEvent), ermittelt ein zweiter Dienst auf dem Server, ob das zugehörige Produkt entfernt wurde oder nicht und versendet diese Information als ein weiteres Event (LokationsEvent) (siehe Abschnitt 7.2). Ist eine Produktentnahme erkannt und als Event kommuniziert worden, ermittelt der Beratungsdienst, welche Informationen dargestellt werden sollen und versendet das Ergebnis als Event (URLEvent), woraufhin die Darstellung der Informationen auf dem zugehörigen Bildschirm erfolgt. Kommt ein Temperatur- oder ein Orientierungsevent beim Beratungsdienst an, adaptiert dieser die auf dem Display darzustellenden Informationen und sendet dies ebenfalls als Event heraus. Die Darstellung auf dem Display erfolgt in Form einer Webseite, wobei die Informationsparameter in der Adressleiste kodiert werden. Parallel zum URLEvent wird ein SprachEvent versendet, welches das Abspielen einer Sprachdatei nach sich zieht. Dieses Event ist im Gegensatz zum URLEvent unabhängig von der Temperatur und Orientierung der Weinflasche und wird versendet, sobald die Weinflasche aus dem Regal entnommen wird. Wird die Weinflasche in das Regal zurückgestellt oder den IRL SmartCart

gelegt, wird dies wiederum als Event notifziert, woraufhin der Regalberater ein Event zum Stoppen der Sprachausgabe und zum Entfernen der Darstellung auf dem Display versendet. Die Zusammenarbeit der einzelnen Komponenten ist in Abbildung 7.12 dargestellt.

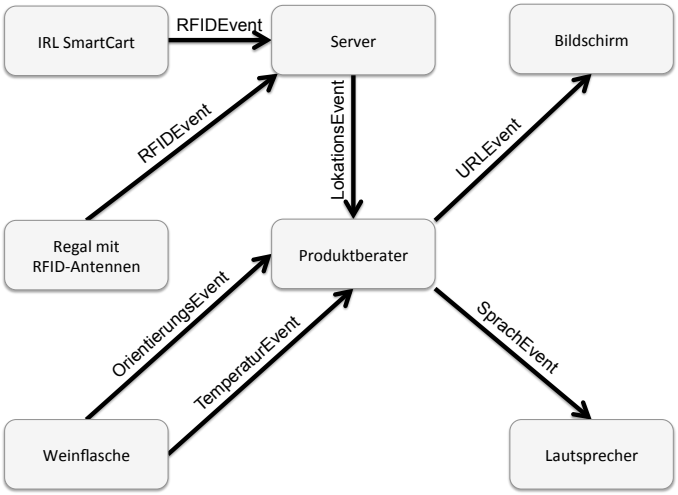


***Abbildung 7.12**: Verbindung der einzelnen Komponenten beim Regalberaterdienst mittels entsprechender Events*

Implementierung Insgesamt umfasst der Produktberater sechs Clients, welche auf einem Rechner laufen: drei Sensoren (Temperatur, Beschleunigung, RFID), zwei Aktuatoren (Lautsprecher, Display) und den eigentlichen Beratungsdienst. Diese Dienste können separat ein- und ausgeschaltet werden, wobei sie teilweise voneinander abhängen. Beispielsweise wird ohne die Erfassung einer Produktentnahme keine Ansteuerung der Aktuatoren erfolgen. Jeder dieser Clients verbindet sich mit dem Server. Um Events zu empfangen, muss eine Funktion *onEvent* implementiert werden, die als Parameter ein Objekt des zugehörigen Eventtyps umfasst. Sollen mehrere Eventtypen erfasst werden, müssen entsprechend viele *onEvent*-Methoden implementiert werden. In Programmauszug 7.1 werden beispielsweise *ObjectLocationEvents* verarbeitet. Sobald ein solches Event an den Client gesendet wird, wird automatisch die entsprechende Funktion aufgerufen. Im Fall des Produktberaters wird geprüft, ob das Event ein Zurückstellen oder ein Entnehmen notifiziert und die interne Liste der Produkte angepasst, die sich aktuell in der Hand des Kunden befinden. Hierbei wird ein Platzieren in den IRL SmartCart genauso behandelt wie das Zurückstellen in das Regal. Bei der Entnahme der Produkte muss der RFID-Leser des zugehörigen Regals die Produktentnahme detektiert haben. Anschließend wird ein Event ausgesendet, welches die Ausgabe der entsprechenden Sprachdatei veranlasst (*AcousticActuatorEvent*). Beim Zurückstellen wird diese Sprachwiedergabe gestoppt. Parallel dazu wird eine URL erzeugt, welche die aktuell entnommenen Produkte als Parameter beinhaltet. Diese URL wird mittels eines Events versendet und somit auf dem zugehörigen Display angezeigt (*URLActuatorEvent*). Die Darstellung der Informationen ist hierbei in der aufgerufenen php-Datei enthalten. Sowohl die *URLActuatorEvents* als auch die *AcousticActuatorEvents* werden ausschließlich an die

Aktuatoren geschickt, die für den Berater zuständig sind. Dies wird über die Spezifikation der Ziele (*speechTargets*, *browserTagets*) realisiert.

```
public void onEvent(ObjectLocationEvent event) {
  if (event.getType().equals(ObjectLocationEvent.Type.APPEARED)) {
    customerHand.remove(product);
    send(new AcousticActuatorEvent
      (AcousticActuatorEvent.ActionCommand.predefinedText,
      product.getName(), AcousticActuatorEvent.ActionType.stop),
      speechTargets);
  } else {
    if (event.getReader().startsWith("DigitalSommelier")) {
      customerHand.add(product);
      send(new AcousticActuatorEvent
        (AcousticActuatorEvent.ActionCommand.predefinedText,
        product.getName(), AcousticActuatorEvent.ActionType.play),
        speechTargets);
    }
  }
  onHandChange();
}

private void onHandChange() {
  if (customerHand.isEmpty()) {
    update("http://172.16.151.15/digisom.php");
  } else {
    String catIds = "";
    for (Product p : customerHand) {
      catIds += "," + p.id;
    }
    catIds = catIds.substring(1);
    update("http://172.16.151.15/digisom.php?products=" + catIds);
  }
}

private void update(String newpage) {
  send(new URLActuatorEvent("Sommelier", newpage), browserTargets);
}
```

Programmauszug 7.1: Empfangen und Senden von Events

Neben dem Empfangen und dem Versenden von Events beinhaltet der Produktberater einen Filtermechanismus, um die Anzahl der versendeten Events zu reduzieren, indem keine redundanten Daten versendet werden. Insbesondere ist hierbei der Temperaturfilter zu erwähnen, welcher nur Events durchlässt, deren Temperaturwert sich vom zuletzt versendeten Event unterscheidet. In Programmauszug 7.2 ist der entsprechende Programmcode zum Erstellen eines solchen Filters dargestellt. Zuerst muss der Ausdruck, nach welchem die Filterung des dynamischen Inhaltsfilters funktioniert, angegeben werden. Über die beiden Variablen e und f kann auf das Event bzw. den Filter zugegriffen werden. In diesem Fall wird die im Event hinterlegte Temperatur mit dem Wert, der im Filterslot 0 hinterlegt

```
Expression ifExpression = ExpressionParser.createExpression
  ("((float) e.getTemperature())!= ((float)f.getSlot(0))");
Expression thenExpression = ExpressionParser.createExpression
  ("f.setSlot(0, (float) e.getTemperature())");
IRLDynamicContentFilter contentFilter =
  new IRLDynamicContentFilter(ifExpression, thenExpression);
contentFilter.setSlot(0, Float.MIN_VALUE);
SetOutputFilterEvent e = new SetOutputFilterEvent
  ("DigitalSommelier.UPartConnectorService", filterGraph, true);
send(e);
```

Programmauszug 7.2: Erstellung und Versenden eines Filters

ist, abgeglichen. Im Fall, dass die beiden Werte ungleich sind, wird ein weiterer Ausdruck ausgeführt. Dieser schreibt den neuen Temperaturwert in den Slot 0. Anschließend wird das Event versendet. Als Ausgangswert wird dem Filter auf Slot 0 ein Initialwert zugewiesen. Da der Filter als Ausgangsfilter agieren soll, wird ein entsprechendes *SetOutputFilterEvent* erzeugt, welches den Zielclient - in diesem Fall den *UPartConnectorService* auf dem Rechner *DigitalSommelier* -, den erstellten Filtergraphen und die Information, ob der Filtergraph auch nach einem Neustart des Clients erneut verwendet werden soll, enthält. Schließlich wird das Event versendet. Anhand des Ziels im Event wird der Filter ausschließlich auf dem dafür vorgesehenen System aktiviert. Zum Deaktivieren eines Filters kann man als Parameter für den Filtergraphen $null$ übergeben. Alternativ zur manuellen Erstellung des Filters könnte auch die grafische Oberfläche verwendet werden.

Verbindung zum Dashboard: Mittels der Versendung von LokationsEvents zu den Produkten wird die Darstellung des entsprechenden Regals in DuRMaD automatisch angepasst. Wird beispielsweise eine Flasche aus einem Regal entnommen, wird dies durch das Entfernen der zugehörigen Virtualisierung aus dem virtuellen Regal entsprechend abgebildet. Wird ein Produkt im Regal platziert, welches nicht in dieses Regal gehört, kann dieses in DuRMaD farblich hervorgehoben werden. Ebenso kann durch eine farbliche Hervorhebung kenntlich gemacht werden, wenn eine Mindestmenge an Produkten unterschritten wird. Diese Prüfung erfolgt immer, sobald ein Produkt ins Regal gestellt oder herausgenommen wird. Neben der farblichen Hervorhebung kann bei Unterschreitung einer bestimmten Menge von Produkten auch automatisiert der Nachbestellprozess ausgelöst werden (siehe Abschnitt 7.2.3). Zusätzlich besteht die Möglichkeit, durch Interaktion mit dem Regal in DuRMaD, die Information auf dem Display in der Umgebung zu verändern oder eine akustische Nachricht in der realen Umgebung ausgeben zu lassen, um beispielsweise die verwendete Sprache der Darstellung manuell zu ändern.

7.3 Zusammenfassung

In diesem Kapitel wurde die praktische Verwendung des Dual Reality Frameworks in der Handelsdomäne vorgestellt. Neben der Datenkommunikation und der virtuellen Repräsentation nimmt die physische Umgebung eine wichtige Rolle in DR-Anwendungen

ein. Eine Handelsumgebung, die mit Sensoren, Aktuatoren und verarbeitenden Diensten instrumentiert ist, stellt das vorgestellte Innovative Retail Laboratory (IRL) dar. Dieses umfasst eine Vielzahl heterogener Sensorsysteme, welche in diversen Modalitäten Inhalte zur späteren Kontextinterpretation erfassen können. Darunter fallen optische, thermische und elektromagnetische Sensoren, wie z.B. Temperatursensor, digitale Kompasse und RFID-Sensoren. Ebenso bieten die verwendeten Aktuatoren die Möglichkeit, Veränderungen in unterschiedlichsten Modalitäten wiederzugeben oder darauf zu reagieren, beispielsweise mittels Displays, Lautsprechern oder elektronischen Türen. Die Sensorverarbeitung und Aktuatoransteuerung erfolgt durch entsprechende Dienste, die im IRL unter anderem Assistenzfunktionen für Kunden bereitstellen. Im IRL besteht eine enge Verbindung zwischen den einzelnen Sensoren, Aktuatoren und Diensten, die über eine gemeinsame Infrastruktur miteinander verbunden sein müssen, um Mehrwertfunktionalitäten bieten zu können. Beispielsweise findet eine Kommunikation zwischen instrumentiertem Einkaufswagen und Regalassistenten statt. Die Kommunikationsschnittstelle wird hierbei durch den EBS realisiert, welcher eine Entkopplung der einzelnen Komponenten und gleichzeitig eine Verbindung zur gegenseitigen Informationsübermittlung ermöglicht. Dies wurde unter anderem mit der Verbindung zwischen dem Heimbereich und der Supermarktumgebung dargestellt, welche Informationen auch umgebungsübergreifend austauschen können. Durch die Möglichkeit, mehrere EBS-Server miteinander zu verknüpfen, können unterschiedliche Cyber-Physische Umgebungen miteinander verbunden werden. Ebenso ermöglicht die Verwendung des EBS den Einsatz von mobilen Endgeräten mit ihren integrierten Sensoren und Aktuatoren im IRL.

Durch die Transformation der heterogenen Sensorinformationen in eine einheitliche Eventstruktur wird eine Verarbeitung von Eingaben in unterschiedlichen Modalitäten ermöglicht. Somit können kontext- und lokationsbasierte Dienste mit entsprechenden Informationen versorgt werden. Zudem werden potentielle Automatismen ermöglicht, welche bei bestimmten Kombinationen unterschiedlicher Sensordaten angestoßen werden können. Beispielsweise ist durch die Integration eines Regelsystems oder von BI-Diensten eine Automatisierung der Umgebung umsetzbar. Die Verbindung einer physischen Umgebung mit DuRMaD bietet Potenziale, welche am Beispiel des IRL dargestellt wurden. Hierzu wurde ein virtuelles Modell des IRL in Form einer partiell erweiterten Virtualisierung realisiert (siehe Kapitel 4.2.2). Das Grundrissmodell liegt hierbei in Form von gml-Dateien vor und die Objektmodelle im XML3D-Format. Bei der Initialisierung wird basierend auf dem Grundrissplan und Informationen aus der Datenbank die entsprechende 3D-Szene automatisch erstellt. Durch die Verbindung des 3D-Modells mit der Darstellung in XML3D kann die virtuelle Supermarktumgebung im Browser dargestellt werden und ermöglicht dadurch weitere Anwendungsmöglichkeiten. Hierdurch wird die Darstellung auf mobilen Geräten ermöglicht und somit eine mobile Monitoring- und Steuerungskomponente umgesetzt. Beispielsweise kann diese Umgebung zu Trainingszwecken, z.B. für zukünftige Marktmitarbeiter, eingesetzt werden, wie z.B. bei der Zielstellung vom „Virtuellen Supermarkt[4]“. Aufgrund der generischen Struktur des Modells können hierbei realweltliche Supermärkte als Vorbild für die virtuellen Modelle verwendet werden.

[4]http://www.virtuellersupermarkt.de, Zugriff: November 2014

Die Verbindung von der physischen Umgebung mit dem virtuellen Modell in DuRMaD wurde anhand des Beispiels der Entnahme von Produkten umgesetzt. Dabei ermöglicht die Interaktion mit dem Modell die Ansteuerung der Aktuatoren. Insbesondere die in DuRMaD integrierte Schnittstelle zur Ontologie und die Darstellung der Kontextmenüs erleichtern Managementaufgaben. Des Weiteren ist es möglich, in DuRMaD virtuell weitere Produkte im Modell zu ergänzen, die nicht physisch vorhanden sind. Damit ist es unter anderem möglich, eine virtuelle Regalbefüllung zu realisieren, welche anschließend als Handlungsanweisung in Form von Planogrammen weitergegeben werden kann. Die Erstellung von Planogrammen wurde mittels einer Spaceplanning-Funktionalität dargestellt, welche in DuRMaD integriert ist. Über eine grafische Oberfläche können Regalbretter, Produkte und spezifische Parameter spezifiziert werden, anhand deren ein Algorithmus eine optimierte Regalbestückung ermittelt. Hierbei wird versucht, anhand von prognostizierten Abverkaufzahlen in Abhängigkeit von den Produktpositionen den potenziellen Gesamtgewinn des Regals zu optimieren. Das resultierende Planogramm kann anschließend als Webseite exportiert werden. Hinsichtlich der bi-direktionalen Verbindung mit möglichen Erweiterungen im virtuellen Modell wurde somit eine partielle DR geschaffen.

Neben der Verwendung von BI-Diensten stellen Geschäftsmodelle eine zweite wichtige Standardkomponente von Handelsumgebungen dar. Hierbei werden Arbeitsschritte durch Prozesse abgebildet, die ein standardisiertes Vorgehen ermöglichen. Aus diesem Grund wurde eine Schnittstelle im DR-Framework implementiert, die eine Integration von Geschäftsprozessen ermöglicht. Durch diese Integration ist es möglich, die Reaktionen der Umgebung auf Kontextveränderungen in einem prozessualen Vorgehen zu modellieren und zu realisieren. Hierbei ermöglicht die Verarbeitung von Sensordaten eine automatische Ausführung von Prozessen. Gleichzeitig ermöglicht eine Visualisierung des Prozessmodells in DuRMaD das Monitoring von Prozessausführungen, indem der aktuell bearbeitete Prozesstask visuell hervorgehoben wird. Neben dem Monitoring können Prozessmodelle zur Ansteuerung von Aktuatoren in der CPE verwendet werden, indem die entsprechenden Prozesstasks Events versenden. Schließlich können Handlungen in der Umgebung mittels des DR-Frameworks aufgezeichnet, in einen Prozess umgewandelt und anschließend abgespielt werden.

Im folgenden Kapitel werden die Ergebnisse der Arbeit noch einmal zusammengefasst und die wissenschaftlichen Fragestellungen anhand dieser Ergebnisse beantwortet. Zudem werden weitere Anwendungsszenarien für das DR-Framework sowie dessen Weiterentwicklungspotenziale aufgezeigt.

Teil IV

Diskussion

8 RESÜMEE UND AUSBLICK

In diesem Kapitel werden die entwickelten Konzepte und Realisierungen abschließend zusammengefasst. Im Speziellen werden hierbei die wissenschaftlichen Beiträge hervorgehoben und weitere mögliche Einsatzgebiete für die Ergebnisse dieser Arbeit präsentiert. Schließlich werden Erweiterungspotenziale beschrieben, welche mögliche weiterführende Forschungsthemen darstellen, die auf den Ergebnissen dieser Arbeit aufbauen.

8.1 Zusammenfassung

Im Rahmen der vorliegenden Promotionsarbeit wurde ein *Dual Reality (DR) Framework* entwickelt, welches durch die Verbindung einer interaktiven 3D-Visualisierung einer *Cyber-Physischen Umgebung (CPE)* mit einer modularen Kommunikationsinfrastruktur das Monitoring und die Steuerung dieser CPE ermöglicht. Für die 3D-Darstellung wurde hierbei das *Dual Reality Management Dashboard (*DURMAD*)* entwickelt, eine Komponente, die basierend auf Veränderungen der Umgebung die Visualisierung automatisch anpasst, um ein Monitoring der CPE zu gewährleisten. Um die Kontextveränderungen der physischen Umgebung erfassen und diese an die Komponenten der CPE weitergeben zu können, wurde eine eventbasierte Kommunikationsinfrastruktur, über welche Sensordaten, Ergebnisse von Diensten und Ansteuerungsbefehle für Aktuatoren versendet werden, entwickelt, die als *Event Broadcasting Service (*EBS*)* bezeichnet wird. Die Events, die für die Kommunikation der Komponenten in der CPE generiert und versendet werden, werden gleichzeitig auch an DURMAD übermittelt, so dass diese dort entsprechend ausgewertet, aufbereitet und visualisiert werden können. Personen, die für das Management der CPE zuständig sind, haben Zugriff auf diese Daten und erhalten darüber eine Unterstützung, um Entscheidungen treffen zu können. In diesem Zusammenhang wurde ein Rechte- und Rollenmanagement im DR-Framework integriert, welches eine Zugriffskontrolle auf die Daten ermöglicht. Durch Interaktion mit dem Modell in DURMAD ist es zudem möglich, die physische Umgebung zu beeinflussen, so dass eine bi-direktionale Einflussnahme möglich ist, entsprechend dem *DR*-Gedanken. Für die Auswertung wurden Schnittstellen und Systeme in Form von *Business Intelligence (BI)* Diensten konzipiert und in das DR-Framework integriert. Gleichzeitig ermöglicht eine Schnittstelle zu *digitalen Objektgedächtnissen (OMM)* die Anreicherung der aus der Umgebung erfassten Informationen mit historischen, produktbezogenen Daten. Des Weiteren wurde ein Modul entwickelt und im DR-Framework integriert, welches eine Ansteuerung

von Aktuatoren in Form von *Geschäftsprozessen (BPM)* ermöglicht, während deren Ausführung zudem in DuRMAD überwacht werden kann. Das entwickelte Framework wurde in eine Laborumgebung integriert, die einem Supermarkt nachempfunden ist (Innovative Retail Laboratory, IRL). Anhand dieser Umsetzung wurden die Vorteile des DR-Frameworks aufgezeigt und am Beispiel des IRL präsentiert. Das Zusammenspiel der einzelnen Komponenten ist in Abbildung 8.1 illustriert.

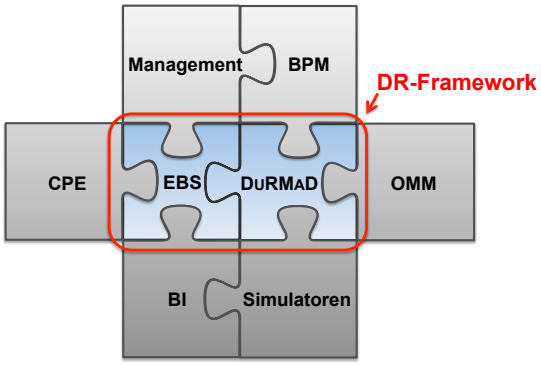


***Abbildung 8.1**: Zusammenspiel der einzelnen Komponenten mit dem DR-Framework, der Kombination von* EBS *und* DuRMAD

8.2 Wissenschaftliche Beiträge

Im Rahmen dieser Arbeit wurden mehrere theoretische (siehe Abschnitt 8.2.1) und ingenieurwissenschaftliche (siehe Abschnitt 8.2.2) Beiträge geleistet, die im Folgenden aufgeführt werden und die in Kapitel 1.2 aufgeführten Forschungsfragen beantworten. Die erzielten Beiträge wurden zudem in Form mehrerer wissenschaftlicher Veröffentlichungen publiziert (siehe Abschnitt 8.2.3).

8.2.1 Theoretische Beiträge

Die theoretischen Beiträge in diesem Abschnitt umfassen Konzepte und Definitionen, die in dieser Arbeit erstmalig thematisiert und aufgestellt wurden.

↪ **Formalisierung von Virtualisierung und Dual Reality.**

Dual Reality (DR) wurde von Lifton als Zusammenspiel zwischen einer realen und einer virtuellen Welt definiert, welche sich basierend auf einem Sensor-Aktuator-Netzwerk gegenseitig beeinflussen können (Definition 1.2). In dieser Arbeit wurde erstmals eine mathematische Formalisierung von DR vorgenommen (Kapitel 4.2.4). Diese basiert auf einer Formalisierung von Virtualisierung, die ebenfalls erstmalig im Rahmen dieser Arbeit erstellt wurde (Kapitel 4.2.2). Hierzu wurden Grundlagen und Konzepte zur Visualisierung von CPE aufgezeigt, welche im Rahmen dieser Arbeit untersucht wurden (Kapitel 4.2.1). Basierend auf erstellten Definitionen wurde hierbei eine Unterscheidung zwischen Modellierung und Virtualisierung spezifiziert

(Definitionen 4.1 und 4.2). Bei der Formalisierung von DR wird zwischen einer vollständigen, einer partiellen und einer erweiterten DR unterschieden, je nach Grad der möglichen Synchronisierung beider Welten. Dies ermöglicht eine Klassifizierung von DR-Applikationen.

↪ **Entwicklung eines Konzepts zur Erweiterung von DR um Simulationen.**

Die ursprüngliche Definition von DR umfasst ausschließlich die gegenseitige Beeinflussung von realer und virtueller Welt. Darauf aufbauend wurde in dieser Arbeit eine Definition entwickelt, die zusätzlich Simulationen betrachtet. Dies wird unter dem Begriff Erweiterte Dual Reality (DR++) zusammengefasst (Definition 4.3). Hierbei spielt eine wichtige Rolle, welchen Einfluss eine Simulation auf die physische Umgebung hat und welcher Zustand nach Beendigung einer Simulation in der Umgebung vorliegt. Mittels der Kombination von realer und virtueller Welt ist es möglich, Simulationen zu integrieren, die Daten aus beiden Welten verwenden und Informationen in beide Welten senden können (Kapitel 6.5.2). Durch Einbindung entsprechender Filter in die im Rahmen dieser Arbeit entwickelte Kommunikationsinfrastruktur und durch Verarbeitung von Events ist eine Verschmelzung einer physischen Umgebung mit deren Virtualisierung realisierbar. Dadurch ist eine erste Umsetzung des DR++-Konzepts erfolgt (Kapitel 6.2).

↪ **Semantische Erweiterung von CPE zur einfachen Steuerung von Aktuatoren.**

Im Rahmen dieser Arbeit wurde eine Methode entwickelt, wie mit Hilfe semantischer Annotationen von Sensoren, Diensten, Aktuatoren und Events vereinfachte Interaktionsformen im 3D-Modell bereitgestellt werden können, die eine Beeinflussung der realen Umgebung zur Folge haben. Hierzu wurde eine Schema für eine Ontologie entworfen, welche eine Verbindung zwischen den Komponenten einer CPE ermöglicht (Kapitel 5.1.3). Gleichzeitig kann diese semantische Annotation in einer Virtualisierung verwendet werden, um beispielsweise Simulationen automatisiert durchführen zu können. In dieser Arbeit wurden exemplarisch die Komponenten einer instrumentierten Handelsumgebung anhand dieser semantischen Beschreibung annotiert. Dadurch wurde die Möglichkeit der Ansteuerung ihrer Aktuatoren geschaffen, indem deren Funktionalitäten automatisch erfasst und dem Benutzer im 3D-Modell zur Verfügung gestellt werden (Kapitel 6.5.3).

↪ **Prozessbasierte Steuerung von CPE.**

In dieser Arbeit wurde erstmalig erörtert, wie standardisierte Beschreibungen in Form von Prozessmodellen verwendet werden können, um eine (automatisierte) Steuerung von CPE zu ermöglichen (Kapitel 7.2.3). Präsentiert wird dieses Konzept anhand der Integration von Prozessmodellen in die entwickelte Kommunikationsinfrastruktur. Hierbei wird bei der Ausführung von Prozessmodellen eine Schnittstelle zum EBS verwendet, um Aktuatoren anzusprechen. Zudem können protokollierte Eventsequenzen in Prozessmodelle transformiert und zu einem späteren Zeitpunkt abgespielt werden und bieten damit eine Möglichkeit, Simulationen zu erstellen. Des Weiteren wurde in dieser Arbeit eine Möglichkeit präsentiert, ein Prozessmodell als Reaktion auf Sensorerfassungen eventbasiert zu starten. Dieses Konzept wurde anhand einer automati-

sierten Erkennung eines Produktbedarfs mit entsprechendem Starten eines zugehörigen Prozessmodells zur Produktnachbestellung getestet. Der zugehörige Prozess kann sowohl Daten aus der Umgebung akquirieren als auch Veränderungen in dieser durchführen.

8.2.2 Ingenieurwissenschaftliche Beiträge

Neben theoretischen Beiträgen wurden auch Entwicklungen und prototypische Umsetzungen von Konzepten im Rahmen dieser Arbeit getätigt, welche im Folgenden beschrieben werden.

↪ **Entwicklung einer interaktiven 3D-Visualisierung zum Monitoren und Steuern von CPE.**

Als eine Plattform zum Monitoren und gleichzeitig auch zum Steuern von CPE wurde das Dual Reality Management Dashboard (DURMAD) konzipiert und umgesetzt (Kapitel 5). Hierbei wird das virtuelle 3D-Modell einer CPE basierend auf Grundrissplänen automatisch aufgebaut. Durch die dreidimensionale Darstellung kann der aktuelle IST-Zustand der CPE visualisiert werden (Kapitel 5.3.1). Hierbei werden Sensordaten aus der Umgebung direkt in die Visualisierungskomponenten übertragen, um eine Synchronisation zwischen der realen und der virtuellen Umgebung zu erreichen (Kapitel 6.5). Gleichzeitig kann basierend auf semantisch annotierten Aktuatoren (Kapitel 6.5.3) und durch gezielte Interaktion mit der 3D-Visualisierung eine direkte Steuerung der CPE mittels der Ansteuerung der Aktuatoren aus der virtuellen Umgebung heraus erfolgen (Kapitel 5.5). Die Entwicklung sowie die Funktionalitäten wurden hierbei durch eine prototypische Umsetzung in einer instrumentierten Handelsumgebung aufgezeigt (Kapitel 7.2.1).

↪ **Unterstützung von Entscheidungsträgern durch Mehrwertdienste.**

Neben der reinen Darstellung des aktuellen Zustands der CPE wurde in DURMAD die Möglichkeit geschaffen, weiterführende Verarbeitungs- und Visualisierungsroutinen zu integrieren. Insbesondere wurden Systeme zur Automatisierung in Form von Agenten- und Regelsystemen entwickelt. Zudem ermöglichen BI-Dienste und Dashboard-Visualisierungen speziell aufbereitete Darstellungen von Informationen (Kapitel 5.4.3). Durch die Integration von Objektgedächtnissen stehen weitere objektbezogene Informationen zur entsprechenden Darstellung zur Verfügung (Kapitel 5.2.2). Die Darstellung aufbereiteter Daten ermöglicht es, Entscheidungsträgern CPE-relevante Informationen mittels unterschiedlicher Darstellungsformen zu präsentieren, wodurch diese bei ihrer Entscheidungsfindung unterstützt werden (Kapitel 5.3).

↪ **Konzeptionierung und Implementierung einer modularen Kommunikationsinfrastruktur.**

Je nach Anwendungszweck ist der Einsatz von Blackboard-Architekturen, Publish/Subscribe-Systemen oder komplexer Eventverarbeitung (CEP) vorteilhaft. Aus diesem Grund wurde der Event Broadcasting Service (EBS) entwickelt, welcher modular angepasst werden und somit alle zuvor genannten Funktionsweisen abdecken kann (Kapitel 6). In Verbindung mit Filtern und Vorverarbeitungsmodulen

vor dem Versenden oder nach dem Empfang von Events werden gewisse Verarbeitungsschritte bereits in die Kommunikationsinfrastruktur integriert und somit die Effizienz der Datenübertragung gesteigert (Kapitel 6.2). Dadurch ist eine einzigartige, modulare Kommunikationsinfrastruktur entwickelt worden, die schnell und einfach an das jeweilige Anwendungsszenario angepasst werden kann.

↪ **Erweiterungsfähige CPE.**

Die Verarbeitung von multimodalen Ein- und Ausgaben kann im EBS durch die Kodierung von Informationen in Events mit generischen Eventtypen realisiert werden (Kapitel 6.1). Gemäß spezieller, in die Kommunikationsinfrastruktur integrierter Verarbeitungsmodule können unterschiedliche Events miteinander kombiniert und bei Bedarf auch wieder aufgesplittet werden (Kapitel 6.2.2). Dies ermöglicht die Kombination von unterschiedlichen Informationen aus verschiedenen Quellen zu sogenannten „Metaevents“. Zudem bietet dies die Möglichkeit, Sensoren mit zusätzlichen Erfassungsparametern ohne großen Aufwand in die Infrastruktur integrieren zu können und ohne notwendigerweise eine Anpassung der Kommunikationsinfrastruktur vornehmen zu müssen. Analog kann durch entsprechende Transformationsfilter auch ein Austausch von Aktuatoren auf einfache Art und Weise realisiert werden.

↪ **Darstellung von heterogenen Sensorinfrastrukturen und Datenströmen.**

Anhand der Visualisierung einer CPE basierend auf ihrer Virtualisierung in DuRMAD und der Verbindung mit semantisch annotierten Events wurde die Möglichkeit geschaffen, mögliche Zusammenhänge zwischen Sensoren, Aktuatoren und Diensten visuell darzustellen. Insbesondere kann der Informationsfluss im 3D-Modell dargestellt werden (Kapitel 6.5.1). Durch eine integrierte Protokolliermöglichkeit der versendeten Events kann auch im Nachhinein der Informationsfluss, welcher in einer gewissen Zeitspanne erfolgt ist, visuell dargestellt werden (Kapitel 6.3).

↪ **Erkennung und Protokollierung auftretender Fehlfunktionen in einer Cyber-Physischen Umgebung.**

Zusätzlich zur visuellen Darstellung des Informationsflusses, welcher bereits einen Anhaltspunkt zur Erfassung potentieller Fehlerquellen bietet, wurde ein Regelsystem zur Erkennung von Fehlfunktionen implementiert und in einer instrumentierten Handelsumgebung getestet (Kapitel 5.4.1). Des Weiteren wurde ein spezieller Monitoringfilter entwickelt, welcher eine frühzeitige Erkennung von Fehlfunktionen ermöglicht und diese entsprechend zwecks Ursachenbehebung signalisiert (Kapitel 6.2.2).

↪ **Verbindung einer Kommunikationsinfrastruktur und einer Visualisierungskomponente nach dem DR++-Konzept.**

Zum ersten Mal wurde die Thematik einer Kommunikationsinfrastruktur für CPE und einer Visualisierung dieser Umgebung als gemeinsame Einheit betrachtet und mittels einer Verschmelzung beider Ansätze ein DR-Framework entwickelt (Kapitel 6.5). Diese Umsetzung ermöglicht es, eine Anwendung des DR++-Konzepts zu realisieren, ohne spezifische Schnittstellen zur Visualisierungskomponente implemen-

tieren zu müssen. Des Weiteren bietet die 3D-Darstellung im Webbrowser eine mobile Überwachungs- und Steuerungskomponente für CPE.

↪ **Prototypische Umsetzung in einer Einzelhandelsumgebung.**

Die Vorteile des DR-Frameworks, das eine Visualisierung des 3D-Modells der Umgebung mit einer Kommunikationsinfrastruktur verknüpft, wurden anhand einer voll funktionsfähigen Implementierung in der Einzelhandelsdomäne präsentiert (Kapitel 7.2). Hierbei wurde auch gezeigt, wie die bereitgestellten Schnittstellen von DURMAD und EBS in Bezug auf das spezielle Einsatzgebiet noch erweitert werden können.

8.2.3 Publikationen

Die im Rahmen dieser Arbeit geleisteten theoretischen und ingenieurwissenschaftlichen Beiträge wurden in Form diverser Veröffentlichungen publiziert.

Buchkapitel

Kahl, G., Magerkurth, C., Preißinger, J., Gebhard, P., und Weyl, B. (2013a). Enhancement of Consumer Support in Retail Scenarios by Utilization of Semantic Product Memories. In Wahlster, W., editor, SemProM, Cognitive Technologies, pages 329–347. Springer Berlin Heidelberg

Full Papers

Jung, R., Spassova, L., und Kahl, G. (2011b). Product-Awareness Through Smart Audio Navigation in a Retail Environment. In Proceedings of the Seventh International Conference on Intelligent Environments, pages 186–191. IEEE Computer Society

Liwicki, M., Thieme, S., Kahl, G., und Dengel, A. (2011). An Intelligent Shopping List - Combining Digital Paper with Product Ontologies. In König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R., und Jain, L., editors, Knowledge-Based and Intelligent Information and Engineering Systems, volume 6884 of Lecture Notes in Computer Science, pages 187–194. Springer Berlin Heidelberg

Nurmi, P., Salovaara, A., Bhattacharya, S., Pulkkinen, T., und Kahl, G. (2011). Influence of Landmark-based Navigation Instructions on User Attention in Indoor Smart Spaces. In Proceedings of the 16th international conference on Intelligent user interfaces, IUI '11, pages 33–42, New York, NY, USA. ACM

Kahl, G. und Bürckert, C. (2012). Architecture to Enable Dual Reality for Smart Environments. In Eighth International Conference on Intelligent Environments, pages 42–49, Guanajuato, México

Short Papers / Notes / Poster Papers

Feld, M. und Kahl, G. (2008). Integrated Speaker Classification for Mobile Shopping Applications. In Nejdl, W., Kay, J., Pu, P., und Herder, E., editors, Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, volume 5149 of Lecture Notes in Computer Science, LNCS, pages 288–291. L3S Research Center, Hannover, Germany, Springer

Spassova, L., Schöning, J., Kahl, G., und Krüger, A. (2009). Innovative Retail Laboratory. In Roots for the Future of Ambient Intelligence (AmI'09), pages 18–21

Kahl, G., Spassova, L., Schöning, J., Gehring, S., und Krüger, A. (2011a). IRL SmartCart - A User-adaptive Context-aware Interface for Shopping Assistance. In Proceedings of the 16th International Conference on Intelligent User Interfaces, IUI '11, pages 359–362, New York, NY, USA. ACM

Jung, R., Kahl, G., und Spassova, L. (2011a). The Shopping Sound Experience. The Journal of the Acoustical Society of America, 130(4):2545–2545

Kahl, G. und Paradowski, D. (2013). A Privacy-aware Shopping Scenario. In Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion, IUI '13 Companion, pages 107–108, New York, NY, USA. ACM

Kahl, G. (2013b). A Visual Monitoring and Management Tool for Smart Environments. In Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion, IUI '13 Companion, pages 93–94, New York, NY, USA. ACM

Workshop Papers

Kahl, G., Wasinger, R., Schwartz, T., und Spassova, L. (2008). Three Output Planning Strategies for Use in Context-aware Computing Scenarios. In Proceedings of the AISB 2008 Symposium on Multimodal Output Generation (MOG 2008), pages 46–49. Online-Proceedings

Kahl, G., Leichtenstern, K., Schöning, J., Spassova, L., und Krüger, A. (2009). A Contextual Learning Game for Toddlers Installed on an Interactive Display Attached to a Shopping Cart. In Workshop on Pervasive Computing Education (PerED-09). o.A

Kröner, A., Gebhard, P., Spassova, L., Kahl, G., und Schmitz, M. (2009). Informing Customers by Means of Digital Product Memories. In Schneider, M., Kröner, A., Olivier, P., und Stephan, P., editors, Proceedings of the 1st international Workshop on Digital Object Memories, volume 4 of Ambient Intelligence and Smart Environments, AISE, pages 21–26. IOS Press, Amsterdam

Kröner, A., Kahl, G., Spassova, L., Feld, T., Mayer, D., Magerkurth, C., und Dada, A. (2010). Demonstrating the Application of Digital Product Memories in a Carbon Footprint Scenario. In Callaghan, V., Kameas, A., Egerton, S., Satoh, I., und Weber, M., editors, Proceedings of the Sixth International Conference on Intelligent Environments (IE'10), pages 164–169. IEEE Computer Society

Kahl, G., Spassova, L., und Krüger, A. (2010). User-adaptive advertisement in retail environments. In Pervasive Advertising and Shopping 2010. International Conference on Pervasive Computing (Pervasive-2010), 8th International Conference on Pervasive Computing, May 17, Helsinki, Finland. Springer

Kahl, G., Warwas, S., Liedtke, P., Spassova, L., und Brandherm, B. (2011b). Management Dashboard in a Retail Scenario. In Kahl, G., Schwartz, T., Nurmi, P., Dim, E., und Forsblom, A., editors, Workshop on Location Awareness in Dual and Mixed Reality (LAMDa 11), pages 22–25. Online-Proceedings

Kahl, G., Bürckert, C., Spassova, L., und Schwartz, T. (2012). Event Broadcasting Service – An Event-Based Communication Infrastructure. In Proceedings of the Second International Workshop on Location Awareness for Mixed and Dual Reality (LAMDa'12), pages 9–12. online

Kahl, G., Schwartz, T., und Brandherm, B. (2013b). Fusion of Multiple Datasets to Support Location-based Services in Retail Applications. In Proceedings of the 3rd International Workshop on Location Awareness in Mixed and Dual Reality (LAMDa'13), pages 9–12. DFKI

Kahl, G. (2013a). A Plugin Framework to Control Electronic Shelf Labels. In Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, UbiComp '13 Adjunct, pages 1007–1014, New York, NY, USA. ACM

8.3 Potentielle Anwendungsbereiche

Instrumentierte Umgebungen finden immer mehr Einsatz in unserem täglichen Leben. Sowohl im privaten Umfeld als auch in öffentlichen Umgebungen werden Sensoren und Aktuatoren eingebracht, die eine Überwachung und Kontrolle ermöglichen (siehe Kapitel 2.2.3). Als Beispiele sind hierbei im privaten Bereich Autos und Wohnungen zu nennen. Dabei ist das Einsatzgebiet primär in der Überwachung der Systemkomponenten (Maintanance) und der Unterstützung der Benutzer basierend auf Aktivitätserkennung zu sehen [Chan et al., 2008]. Im öffentlichen Bereich werden Arbeitsstätten, wie Fabriken, öffentlicher Transport oder Flughäfen zunehmend instrumentiert. Neben der Benutzerunterstützung wird die Instrumentierung hierbei auch eingesetzt, um Vorgaben und Vorschriften kontrollieren zu können. Neben der in dieser Promotionsarbeit aufgezeigten Einsatzmöglichkeit im Einzelhandelskontext kann das entwickelte Konzept des DR-Frameworks auch in solchen Umgebungen eingesetzt werden. Dies setzt voraus, dass neben einem Grundrissmodell die einzelnen Komponenten der Umgebung als Virtualisierung vorliegen, um eine Zuordnung der physischen mit den virtuellen Objekten herstellen zu können. Der EBS kann in diesem Fall als primäre oder als zusätzliche Kommunikationsschnittstelle verwendet werden, um ein zusätzliches Monitoring zu bewerkstelligen bzw. um eine weitere Steuerungskomponente zu ermöglichen, oder sogar die Datenübertragung zwischen den Komponenten zu übernehmen. Im Folgenden werden die Potenziale des DR-Frameworks in unterschiedlichen Anwendungsbereichen kurz aufgezeigt.

↪ **Automotive:**

In modernen Automobilen werden ähnliche Ansätze für die Datenübertragung verwendet, wie sie beim EBS implementiert wurden. Der CAN-Bus ermöglicht beispielsweise den Informationsaustausch zwischen den verschiedenen Komponenten in einem modernen Auto, wobei die einzelnen Komponenten eigenständig entscheiden, welche dieser Daten sie verarbeiten [Wolf et al., 2004]. Durch eine Schnittstelle zwischen diesem Bus-System und dem EBS könnten die Informationen zusätzlich in einem virtuellen 3D-Modell des Autos verarbeitet und dargestellt werden, ohne die eigentliche Kommunikation zu stören. Durch die integrierte Fehlererkennung in der Kommunikationsinfrastruktur können potentiell auftretende Probleme erkannt und im 3D-Modell dargestellt werden. Ebenso können dem Benutzer Verschleißinformationen in einer visuell aufbereiteten Form präsentiert werden. Durch die Kodierung von Eingaben in Events, unabhängig von der verwendeten Modalität, können zusätzliche Schnittstellen zur Ansteuerung gewisser Aktuatoren im Auto, wie beispielsweise Radio, leicht ergänzt werden. Mittels Filtern kann hierbei sichergestellt werden, dass keine kritischen Parameter, wie z.B. die aktuelle Fahrgeschwindigkeit, aufgrund fehlerhafter Events seitens des EBS beeinflusst werden.

↪ **Häuslicher Dienstleistungssektor:**

Bereits heutzutage finden sich viele Aktuatoren in Wohnungen in Form von fernsteuerbaren Haushaltsgeräten wieder, wie z.B. Fernseher[1], Steckdosen[2] oder Beleuchtung[3]. Diese werden in der Regel durch Applikationen geregelt, welche speziell auf Mobilfunkgeräte ausgelegt sind. Durch Sensoren können zudem Benutzeraktivitäten erfasst und automatisch darauf reagiert werden, wie dies beispielsweise bei MavHome gezeigt wird [Youngblood et al., 2005]. Basierend auf einem Agentensystem können hierbei die von Bewegungssensoren und Kamerasystemen erfassten Bewegungen der Bewohner analysiert werden, so dass diese mittels einer Aktivitätserkennung zum automatischen Auslösen pro-aktiver Reaktionen führen, wie z.B. die Realisierung einer adaptiven Lichtsteuerung. Mit Hilfe eines 3D-Simulators können die real erfassten Sensorwerte in einem 3D-Modell der Umgebung dargestellt und von dort Veränderungen in der Umgebung hervorgerufen werden. Beispielsweise können darüber Lampen an- und ausgeschaltet werden [Das und Cook, 2005]. Durch den Einsatz des DR-Frameworks könnten Aktuatoren über eine mobile 3D-Darstellung der Hausumgebung angesteuert werden. Hierbei bestünde die Möglichkeit, von einer Benutzeroberfläche heraus alle Aktuatoren anzusteuern. Gleichzeitig könnten Handlungsempfehlungen vom Benutzer in der Darstellung kenntlich gemacht werden, anstatt alle Aktionen automatisch durchführen zu lassen, so dass der Benutzer die letztendliche Entscheidung selbstständig treffen kann. Die Semantik und die visuellen Darstellungsformen für die einzelnen Systemkomponenten könnten beispielsweise über entsprechende digitale Objektgedächtnisse mitgeliefert werden. Durch eine Erweiterung einer instrumentierten Haushaltsumgebung mit dem DR-Framework könnte der Benutzer eine stärkere Kon-

[1] http://www2.philips.de/heimnetzwerk/app/apps.html, Zugriff: November 2014
[2] http://power-switch.eu, Zugriff: November 2014
[3] http://www.philips.de/e/hue/hue.html, Zugriff: November 2014

trolle über die Umgebung erlangen. Gleichzeitig kann der Status der Umgebung auch aus der Ferne abgerufen und im Browser visuell dargestellt werden.

↪ **Industrie:**

Durch den intelligenten Zusammenschluss Cyber-Physischer Systeme entstehen Fabrikumgebungen der Zukunft, sogenannte *Smart Factories* [MacDougall, 2013]. Hierbei steht im Fokus, neue Strukturen zu schaffen und damit modulare Fabriksysteme zu generieren, die eine individuelle Produktion bis auf Losgröße 1 ermöglichen. Eine solche Fabrik stellt die *SmartFactory*$^{\mathbf{KL}}$ dar [Zühlke, 2010]. Mittels diverser Sensoren kann diese Anlage aus der Ferne überwacht werden. Hierbei spielt die Integration von mobilen Endgeräten, wie beispielsweise Smartphones, eine wichtige Rolle, um eine flexible Steuerung der Anlage zu ermöglichen. Dabei müssen die Steuerungsverfahren so ausgelegt sein, dass zwischen der Arbeit an einem stationären Arbeitsplatz und der mobilen Anwendung möglichst kein Unterschied in der Funktionsweise besteht, so dass zwischen beiden Arbeitsplätzen gewechselt werden kann. Gleichzeitig ist die Trennung der einzelnen Sensoren, Aktuatoren und Maschinen wichtig, um die Flexibilität der Anlage zu erhalten. Hierbei umfassen Fabrikanlagen mehrere unterschiedliche Sensor- und Aktuatortypen, die berücksichtigt werden müssen. Durch den Einsatz des DR-Frameworks wären die Anforderungen für zukünftige Fabriken erfüllt. Dabei könnten die Sensordaten mit den Vorverarbeitungsmethoden im EBS überarbeitet werden, bevor diese an die weiteren Komponenten kommuniziert werden. Des Weiteren spielt insbesondere der Monitoring-Filter zur Überwachung der ordnungsgemäßen Funktionsweise der Anlage eine wichtige Rolle. Die 3D-Darstellung kann zum Monitoren und Darstellen potentieller Probleme verwendet werden. Durch die Unterstützung von mobilen Endgeräten kann zudem eine Anlagensteuerung direkt an der Anlage erfolgen. Bereits existierende Simulationen könnten in DuRMAD eingebunden werden und somit den Leitern der Fabrik visuelle Entscheidungsunterstützungen an die Hand geben.

↪ **Smart Spaces:**

Schließlich kann die Monitoring-Komponente des DR-Frameworks zur Überwachung von öffentlichen Umgebungen eingesetzt werden, beispielsweise um Besucherbewegungen in Zoos oder in Freizeitparks darzustellen [Solmaz et al., 2012]. Anhand der Visualisierungsmöglichkeit von Besucherströmen könnten pro-aktiv Informationen an Mitarbeiter versendet werden, um beispielsweise einen dynamischen Personaleinsatz zu ermöglichen. Gleichzeitig kann der Betrieb der Anlagen bzw. die Aktivität der Tiere überwacht und verantwortlichen Personen dargestellt werden [Gervasi et al., 2006]. Die Sensordaten könnten zudem verwendet werden, um Analysen zu generieren und potentielle Prognosen zu eruieren, um darüber Handlungsempfehlungen ableiten zu können. Neben den Darstellungen für die Betreiber der Umgebung können gewisse Informationen auch für die Besucher verfügbar gemacht werden, um dadurch die Attraktivität der Umgebung zu steigern. Beispielsweise kann ein 3D-Umgebungsmodell online präsentiert werden, über welchem man ein reales Kamerabild aus der Umgebung dargestellt bekommt, dessen Ausrichtung über Interaktion mit dem Modell verändert werden kann. Eine weitere öffentliche Umgebung stellen Flughäfen dar. Hierbei

kann ebenfalls der Betreiber durch die Darstellung der Kundenströme einen Vorteil gewinnen, um beispielsweise das Sicherheitspersonal entsprechend einteilen zu können. Gleichzeitig können auch die Fluggäste durch Integration ihrer Smartphones in die Kommunikationsinfrastruktur pro-aktive Hinweise empfangen, wie z.B. Veränderungen der Abfluggates oder der Abflugzeiten. Die Darstellung der Flughafenumgebung kann des Weiteren als Navigationsgrundlage dienen.

8.4 Möglichkeiten für zukünftige Forschungsthemen

Aufbauend auf den Ergebnissen dieser Arbeit können weitere Forschungsthemen untersucht und im DR-Framework umgesetzt werden.

↪ **Automatisierung:**

Basierend auf den semantischen Beschreibungen von Events könnten weitere Verfahren entwickelt werden, die eine automatische Verarbeitung unterstützen, ähnlich eines Ansatzes von Teymourian et al.. Hierbei werden die Informationen aus den Events mit Daten aus externen Ontologien verknüpft, um darüber automatisierte Reaktionen durchführen zu können [Teymourian et al., 2012]. Dieser Ansatz wurde durch die Integration von digitalen Objektgedächtnissen bereits teilweise realisiert und könnte durch weitere Verknüpfungen mit Diensten ausgeweitet werden. Als Resultat könnte in diesem Fall die gesamte Umgebung als intelligenter Agent angesehen werden, welcher mittels DURMAD überwacht und gesteuert werden kann.

↪ **Automatische Auswahl geeigneter Ausgabemodalitäten:**

Aufgrund der Ansteuerung von Aktuatoren mittels Events können Ausgaben in unterschiedlichen Modalitäten durch die gleiche Datenstruktur erfolgen. Dies kann dazu verwendet werden, um abhängig vom jeweiligen Kontext automatisch geeignete Ausgabegeräte sowie Ausgabemodalitäten zu ermitteln [Kahl et al., 2008]. Beispielsweise können Ausgaben in der Umgebung oder auf dem mobilen Endgerät des Benutzers textuell oder sprachlich erfolgen, wobei die Auswahl der Ausgabemodalität und -plattform automatisch durch die Umgebung bestimmt wird.

↪ **Semantische Schnittstellen zu Sensoren und Aktuatoren:**

Die Anbindung von Sensoren und Aktuatoren an den EBS erfolgt aktuell noch durch entsprechende Schnittstellen, die speziell implementiert werden müssen. Eine Integration von semantisch annotierten Schnittstellen, wie sie im UCH (Universal Control Hub) verwendet werden, würde hierbei einen Mehrwert bieten [Zimmermann und Vanderheiden, 2007; Frey et al., 2011] (siehe Kapitel 3.2.6). Damit könnten automatisch neue Sensoren in die Kommunikationsinfrastruktur eingebunden werden, ohne eine Anpassung ihrer Schnittstellen durchführen zu müssen.

↪ **Integration von Simulationen:**

Im Sinne von DR++ besteht die Möglichkeit, Simulationstools ans DR-Framework anzuschließen. Im Rahmen dieser Arbeit wurde dies exemplarisch für das Zählen der Pro-

duktentnahmen präsentiert. Aufbauend darauf könnten weitere Simulationstools angeschlossen werden, wie beispielsweise DiaSim [Bruneau und Consel, 2012], welches einen Simulator für Pervasive Computing Applikationen darstellt. Ähnliche Simulationstools existieren bereits für den Heimbereich, wie z.B. der eHomeSimulator [Armac und Retkowitz, 2007], oder im Fabrikumfeld [Westkämper und Jendoubi, 2003]. Durch eine Verbindung mit der physischen Umgebung können die Simulatoren zusätzlich erweitert werden, indem auch reale Sensordaten als Eingabe verwendet werden können. Zudem können auch Ergebnisse der Simulationen einen direkten Einfluss auf die reale Umgebung ausüben, indem die entsprechenden Aktuatoren angesprochen werden.

↪ **Selbstlernende Umgebung:**

Eine smarte Umgebung kann automatisch erfassen, welche Ereignisse zu welchen Zeitpunkten auftreten und welche Reaktionen darauf ausgeführt werden. Durch maschinelles Lernen kann basierend auf gelernten Daten abgeleitet werden, welche Aktionen nach dem Erfassen von gewissen Sensorinformationen ausgeführt werden sollen. Dadurch würde eine selbstlernende Umgebung entstehen, die mittels des Sensor-Aktuator-Dienst-Netzwerks sich möglichst automatisch selbst verwalten und eigenständige Reaktionen auf Zustandsänderungen ausführen kann. Unter anderem wäre dies zur Erkennung von unnatürlichen Handlungen sinnvoll, deren Ursache ein Systemfehler oder ein unbefugtes Eingreifen sein kann, um daraufhin entsprechende Maßnahmen einzuleiten.

↪ **Einsatz als Lern- und Testumgebung:**

Durch die realgetreue Darstellung der Umgebung auch in Internetbrowsern kann mehreren Personen gleichzeitig ein Zugriff auf die interaktive Visualisierung geboten werden. Dies kann beispielsweise dazu verwendet werden, um Platzierungsstudien online durchzuführen und damit Kundenpräferenzen bezüglich der Produktplatzierung zu ermitteln. Gleichzeitig kann die Umgebung als Trainingsplatz für Mitarbeiter verwendet werden, wie es beispielsweise bei der virtuellen Lernplattform für den Einzelhandel erfolgt[4]. Durch eine Integration von Gamification-Ansätzen könnte hierbei der Anreiz zum Lernen weiter gestärkt werden, indem Anreize durch spielerische Aufgaben geschaffen werden [Raymer, 2011].

[4] http://www.virtuellersupermarkt.de/, Zugriff: November 2014

Teil V

Anhang

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1_01" class="java.beans.XMLDecoder">
  <void class="com.sun.j3d.utils.geometry.GeometryInfo" id="Tea-gi0">
    <object class="com.sun.j3d.utils.geometry.GeometryInfo" field="TRIANGLE_ARRAY"></object>
    <void property="coordinates">
      <array class="float" length="108">
        <void index="0">
          <float>7.84999990463</float>
        </void>
        ...
        <void index="107">
          <float>7.30000019073</float>
        </void>
      </array>
    </void>
    <void property="textureCoordinates2">
      <array class="float" length="72">
        <void index="0">
          <float>0.727324187756</float>
        </void>
        ...
        <void index="71">
          <float>0.269791901112</float>
        </void>
      </array>
    </void>
  </void>
  <void class="com.sun.j3d.utils.geometry.NormalGenerator">
    <void property="creaseAngle">
      <object class="java.lang.Math" field="PI"></object>
    </void>
    <void method="generateNormals">
      <object idref="Tea-gi0"></object>
    </void>
  </void>
  <void class="com.sun.j3d.utils.geometry.Stripifier">
    <void method="stripify">
      <object idref="Tea-gi0"></object>
    </void>
  </void>
  <void idref="Tea-gi0">
    <void property="geometryArray" id="Tea-ga0"></void>
  </void>
  <object class="javax.media.j3d.Shape3D">
    <void property="userData">
      <string>glo_2075282_kuvert_path_small.jpg</string>
    </void>
    <void property="geometry">
      <object idref="Tea-ga0"></object>
    </void>
    <void property="appearance">
      <object class="javax.media.j3d.Appearance">
        <void property="material">
          <object class="javax.media.j3d.Material">
            <object class="javax.vecmath.Color3f">
              <float>0.40000000596</float>
              <float>0.40000000596</float>
              <float>0.40000000596</float>
            </object>
            <object class="javax.vecmath.Color3f">
              <float>0.0</float>
              <float>0.0</float>
              <float>0.0</float>
            </object>
            <object class="javax.vecmath.Color3f">
              <float>0.800000011921</float>
              <float>0.800000011921</float>
              <float>0.800000011921</float>
            </object>
            <object class="javax.vecmath.Color3f">
              <float>1.0</float>
              <float>1.0</float>
              <float>1.0</float>
            </object>
            <float>6.40000009537</float>
          </object>
        </void>
      </object>
    </void>
  </object>
</java>
```

Programmauszug A.1: Modellexport in Java-Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<xml3d xmlns="http://www.xml3d.org/2009/xml3d">
  <defs id="mainDef">
    <transform id="t_Brennnessel" rotation="1.000000 0.000000 0.000000 0.000000" scale="1.000000 1.000000
        1.000000" translation="0.000000 0.000000 0.000000"/>
    <data id="mesh_Tea_Material">
      <float3 name="position">
        15.700001 6.400000 0.000000 15.700001 -0.000000 0.000000 0.000000 0.000000 0.000000 0.000001 6.400001
              0.000000 15.700001 6.400000 7.300000 0.000000 6.400000 7.300000 0.000000 0.000000 7.300000
              15.699999 -0.000001 7.300000 15.700001 6.400000 0.000000 15.700001 6.400000 7.300000 15.699999
              -0.000001 7.300000 15.700001 -0.000000 0.000000 15.700001 -0.000000 0.000000 15.699999 -0.000001
              7.300000 0.000000 0.000000 7.300000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
               0.000000 7.300000 0.000000 6.400000 7.300000 0.000001 6.400001 0.000000 15.700001 6.400000
              7.300000 15.700001 6.400000 0.000000 0.000001 6.400001 0.000000 0.000000 6.400000 7.300000
      </float3>
      <float3 name="normal">
        0.000000 0.000000 -1.000000 0.000000 0.000000 -1.000000 0.000000 0.000000 -1.000000 0.000000 0.000000
              -1.000000 0.000000 -0.000000 1.000000 0.000000 -0.000000 1.000000 0.000000 -0.000000 1.000000
              0.000000 -0.000000 1.000000 1.000000 -0.000000 0.000000 1.000000 -0.000000 0.000000 1.000000
              -0.000000 0.000000 1.000000 -0.000000 0.000000 -0.000000 -1.000000 -0.000000 -0.000000 -1.000000
              -0.000000 -0.000000 -1.000000 -0.000000 -0.000000 -1.000000 -0.000000 -1.000000 0.000000 -0.000000
               -1.000000 0.000000 -0.000000 -1.000000 0.000000 -0.000000 -1.000000 0.000000 -0.000000 0.000000
              1.000000 0.000000 0.000000 1.000000 0.000000 0.000000 1.000000 0.000000 0.000000 1.000000 0.000000
      </float3>
      <float2 name="texcoord">
        0.727324 0.502799 0.727324 0.706892 0.240744 0.706892 0.240744 0.502799 0.727324 0.269792 0.240744
              0.269792 0.240744 0.068139 0.727324 0.068139 0.727324 0.502799 0.951820 0.502799 0.951820 0.706892
               0.727324 0.706892 0.727324 0.706892 0.727324 0.940607 0.240744 0.940607 0.240744 0.706892
              0.240744 0.706892 0.016906 0.706892 0.016906 0.502799 0.240744 0.502799 0.727324 0.269792 0.727324
               0.502799 0.240744 0.502799 0.240744 0.269792
      </float2>
      <int name="index">
        0 1 2 2 3 0 4 5 6 6 7 4 8 9 10 10 11 8 12 13 14 14 15 12 16 17 18 18 19 16 20 21 22 22 23 20
      </int>
    </data>
    <shader id="Ground" script="urn:xml3d:shader:phong">
      <float3 name="diffuseColor">
        0.800000 0.800000 0.800000
      </float3>
      <float name="ambientIntensity">
        0.0
      </float>
      <float3 name="specularColor">
        0.500000 0.500000 0.500000
      </float3>
      <float name="shininess">
        0.09784735812133072
      </float>
    </shader>
    <shader id="Material" script="urn:xml3d:shader:phong">
      <texture name="diffuseTexture">
        <img src="../textures/glo_2075282_kuvert_path_small.jpg"/>
      </texture>
      <float3 name="diffuseColor">
        1 1 1
      </float3>
      <float name="ambientIntensity">
        0.0
      </float>
      <float3 name="specularColor">
        0.100000 0.100000 0.100000
      </float3>
      <float name="shininess">
        0.03913894324853229
      </float>
    </shader>
  </defs>
  <group id="Brennnessel" transform="#t_Brennnessel">
    <group shader="#Material">
      <mesh src="#mesh_Tea_Material" type="triangles"/>
    </group>
  </group>
</xml3d>
```

Programmauszug A.2: Modellexport in XML3D-Syntax

```
<?xml version="1.0" encoding="utf-8"?>
<Model name="IRL" rootURL=".\globus.yml" rootObject="Polygon: 19.02.2008 16:10:17" xShiftAsSubModel="12"
     yShiftAsSubModel="49" scaleAsSubModel="0,898045348015458" rotationAsSubModel="12,6642560345013">
  <Scale x1="407" y1="0" x2="672" y2="0" realLength="12,2" />
  <GeoReference axGeo="0" ayGeo="0" bxGeo="0" byGeo="0" cxGeo="0" cyGeo="0" axModel="0" ayModel="0" bxModel="0"
       byModel="1024" cxModel="1024" cyModel="0" metersAboveSeaLevel="0">
    <ModelToGeo m00="n. def." m01="n. def." m02="n. def." m03="n. def." m10="n. def." m11="n. def." m12="n. def.
         " m13="n. def." m20="n. def." m21="n. def." m22="n. def." m23="n. def." m30="n. def." m31="n. def."
         m32="n. def." m33="n. def." />
    <GeoToModel m00="n. def." m01="n. def." m02="n. def." m03="n. def." m10="n. def." m11="n. def." m12="n. def.
         " m13="n. def." m20="n. def." m21="n. def." m22="n. def." m23="n. def." m30="n. def." m31="n. def."
         m32="n. def." m33="n. def." />
  </GeoReference>
  <Coord id="ID0" ref="model" x="672" y="951" z="0" />
  ...
  <Coord id="ID43" ref="model" x="409" y="557" z="0" projector="steerableProjector1" />
  ...
  <Coord id="ID251" ref="model" x="672" y="286" z="0" />
  <Connection label="" a="ID0" b="ID1" passability="None" attributes="Wall" obstacle="False" />
  <Connection label="" a="ID2" b="ID3" passability="Both" attributes="Door" obstacle="False" />
  ...
  <Connection label="" a="ID251" b="ID82" passability="None" attributes="Window, i=104, j=243, l=15, frameColor
       =#E0E0E0" obstacle="False" />
  <Layer layerid="0" layername="0" xOrigin="0" yOrigin="0" zOrigin="0" width="1024" height="1024" style="opaque"
       wallHeight="330" doorHeight="200" muralHeight="100">
    <Image URL=".\irl.png" x="0" y="0" />
    <Object name="Polygon: 25.10.2007 17:01:39" type="indoor" surfacetype="Room" accessability="Pedestrian">
      <CoordIndex>0#1#2#3#4#5#6#7#8</CoordIndex>
      <Triangle a="ID0" b="ID7" c="ID8" />
      <Triangle a="ID7" b="ID0" c="ID6" />
      <Triangle a="ID6" b="ID0" c="ID5" />
      <Triangle a="ID5" b="ID0" c="ID4" />
      <Triangle a="ID4" b="ID0" c="ID3" />
      <Triangle a="ID3" b="ID0" c="ID2" />
      <Triangle a="ID2" b="ID0" c="ID1" />
      <SubModel URL="" />
    </Object>
    ...
    <Object name="Polygon: 24.06.2010 03:35:35" type="indoor" surfacetype="Room" accessability="Pedestrian">
      <CoordIndex>244#251#250#249</CoordIndex>
      <Triangle a="ID251" b="ID249" c="ID244" />
      <Triangle a="ID249" b="ID251" c="ID250" />
      <SubModel URL="" />
    </Object>
    <Item type="rfbeaconpassive" name="e0040100175a6e96" info="enter info">
      <Sphere ref="model" x="414" y="89" z="0" radius="2" longitude="n. def." latitude="n. def." />
    </Item>
    ...
    <Item type="rfbeaconpassive" name="E0040100175A9185" info="enter info">
      <Sphere ref="model" x="503" y="550" z="0" radius="2" longitude="n. def." latitude="n. def." />
    </Item>
    <Device name="StaticProjector1" x="635,3339" y="453,7051" z="39,09836" h="4 " w="7 " d="10 ">
      <StaticProjector beam_angle="25" aspect_ratio="1,333333" initial_azimuth="90" initial_zenith="90"
           mount_azimuth="0" mount_zenith="0" range="135" height="135" />
    </Device>
    <Display name="wineDisplay" server="" port="5900" password="" staticURL="" pointable="False" x="597,17" y="
         317" z="36,057" orientation="0" pitch="90" size="10,67114" landscape="True" aspect="_4to3" />
    ...
    <Display name="DresserDisplay" server="" port="5900" password="" staticURL="" pointable="False" x="428,65" y
         ="549,06" z="31,4959" orientation="180" pitch="90" size="12,59836" landscape="True" aspect="_16to9" />
    <Box name="_ShelfGroup1-1" type="Group" x="477,754" y="325,207" z="0" orientation="90" width="126,5 cm"
         height="217 cm" depth="72,5 cm" color="#FFFFFF" textureFront="" textureTop="">
      <Box name="Base1-1-1" type="Box" x="-13,5" y="1,2" z="0" orientation="0" width="2 cm" height="16 cm" depth
           ="12,5 " color="#FFFFFF" textureFront="" textureTop="" />
      <Box name="Base1-1-2" type="Box" x="13,5" y="1,2" z="0" orientation="0" width="2 cm" height="16 cm" depth=
           "12,5 " color="#FFFFFF" textureFront="" textureTop="" />
      <Box name="_Shelf1-1" type="Group" x="0" y="1" z="3,557047" orientation="0" width="28 " height="180 cm"
           depth="12 " color="#FFFFFF" textureFront="" textureTop="">
        <Box name="Shelfboard1-1-1" type="Box" x="0" y="-1,2" z="0" orientation="0" width="126,5 cm" height="3
             cm" depth="70 cm" color="#FFFFFF" textureFront="" textureTop="" />
        <Box name="Shelfboard1-1-2" type="Box" x="-0,706" y="1,15" z="10,7844" orientation="0" width="120 cm"
             height="2 cm" depth="10,5 " color="#B88601" textureFront="" textureTop="" />
        <Box name="Shelfboard1-1-3" type="Box" x="-0,706" y="1,15" z="20,7844" orientation="0" width="120 cm"
             height="2 cm" depth="10,5 " color="#B88601" textureFront="" textureTop="" />
        <Box name="Shelfboard1-1-4" type="Box" x="-0,706" y="1,15" z="30,34396" orientation="0" width="120 cm"
             height="2 cm" depth="10,5 " color="#B8860B" textureFront="" textureTop="" />
      </Box>
      <Box name="Cover1-1" type="Box" x="0" y="-0,25" z="42,5" orientation="0" width="126,5 cm" height="21 cm"
           depth="70,2 cm" color="#FFFFFF" textureFront="" textureTop="" />
      <Box name="Back1-1" type="Box" x="0" y="7,66" z="0" orientation="0" width="126,5 cm" height="215 cm" depth
           ="0,5 " color="#FFFFFF" textureFront="" textureTop="" />
      <Box name="MetalRod1-1" type="Box" x="13,5" y="-7,4" z="4,223993" orientation="0" width="0,5 " height="176
           cm" depth="1 " color="#FFFFFF" textureFront="" textureTop="" />
    </Box>
    ...
  </Layer>
</Model>
```

Programmauszug A.3: Grundrissplan im yml-Format

```
<?xml version="1.0" encoding="utf-8" ?>
<ogr:FeatureCollection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://ogr.
      maptools.org/ Shelves.xsd" xmlns:ogr="http://ogr.maptools.org/" xmlns:gml="http://www.opengis.net/gml">
  <gml:boundedBy>
    <gml:Box>
      <gml:coord><gml:X>0.4780000000000086</gml:X><gml:Y>0.09113645383202204</gml:Y></gml:coord>
      <gml:coord><gml:X>11.71600000000001</gml:X><gml:Y>13.96500000000003</gml:Y></gml:coord>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <ogr:Shelves fid="F0">
      <ogr:geometryProperty>
        <gml:Polygon>
          <gml:outerBoundaryIs>
            <gml:LinearRing>
              <gml:coordinates>
                6.625,6.925999999999988 6.625,7.566000000000003 9.125000000000057,7.566000000000003
                    9.125000000000057,6.925999999999988 6.625,6.925999999999988
              </gml:coordinates>
            </gml:LinearRing>
          </gml:outerBoundaryIs>
        </gml:Polygon>
      </ogr:geometryProperty>
      <ogr:Id>0</ogr:Id>
      <ogr:Align>0.00000000000</ogr:Align>
      <ogr:CentroidX>7.93500000000</ogr:CentroidX>
      <ogr:CentroidY>7.24600000000</ogr:CentroidY>
      <ogr:Area>1.60000000000</ogr:Area>
      <ogr:Contents>Hygieneartikel</ogr:Contents>
      <ogr:XExtent>2.50000000000</ogr:XExtent>
      <ogr:YExtent>0.64000000000</ogr:YExtent>
      <ogr:ShelfId>13</ogr:ShelfId>
      <ogr:Type>normal</ogr:Type>
      <ogr:PartNo>2</ogr:PartNo>
      <ogr:Parts>2</ogr:Parts>
      <ogr:OpenBoth>0</ogr:OpenBoth>
    </ogr:Shelves>
  </gml:featureMember>
  ...
</ogr:FeatureCollection>
```

Programmauszug A.4: Ausschnitt aus einer gml-Datei

```
<?xml version="1.0" encoding="utf-8" ?>
<ogr:FeatureCollection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://ogr.
      maptools.org/ ShelfMeters.xsd" xmlns:ogr="http://ogr.maptools.org/" xmlns:gml="http://www.opengis.net/gml"
      >
  <gml:boundedBy>
    <gml:Box>
      <gml:coord>
        <gml:X>1.175000000000011</gml:X>
        <gml:Y>4.556000000000004</gml:Y>
      </gml:coord>
      <gml:coord>
        <gml:X>10.92599999999999</gml:X>
        <gml:Y>13.26800000000006</gml:Y>
      </gml:coord>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <ogr:ShelfMeters fid="F0">
      <ogr:geometryProperty>
        <gml:Point>
          <gml:coordinates>10.557000000000016,4.556000000000004</gml:coordinates>
        </gml:Point>
      </ogr:geometryProperty>
      <ogr:ShelfID>13</ogr:ShelfID>
      <ogr:shelfPart>2</ogr:shelfPart>
      <ogr:No>99</ogr:No>
    </ogr:ShelfMeters>
  </gml:featureMember>
  ...
</ogr:FeatureCollection>
```

Programmauszug A.5: Ausschnitt aus einer zweiten gml-Datei

```
rule "Alle Salamipizzen um 20 Cent reduzieren"
when
$pizza: Pizza(typ="Salamipizza", $preis:price)
then
$pizza.price = $price-0.2
end
```

Programmauszug A.6: Beispiel einer Regel in DRL-Syntax

```
[condition][] Der Preis aller Pizzen des Typs "{pizzaType}"=
$pizza: Pizza(typ="{pizzaTyp}", $preis:price)
[consequence][] soll um {"reduction"} Euro reduziert werden=
$pizza.price = $price-{"reduction"}
```

Programmauszug A.7: Beispiel einer Regel in DSL-Syntax

***Abbildung A.2**: Resultierendes Bild einer virtuellen Verräumung als Teil eines Planogramms*

Regal	Meter	Brett	Platzierungsnummer	Name	Stacking	EAN	Hersteller	Breite	Höhe	Tiefe
6	2	3	1	Globus reines Rapsöl	1	4304218714596	Globus	0.065	0.288	0.07
6	2	3	2	Globus reines Rapsöl	1	4304218714596	Globus	0.065	0.288	0.07
6	2	3	3	Globus reines Rapsöl	1	4304218714596	Globus	0.065	0.288	0.07
6	2	3	4	Globus reines Rapsöl	1	4304218714596	Globus	0.065	0.288	0.07
6	2	3	5	GUTE POTT 54% 0.70 L	1	4001731152798	Henkell & Co.	0.07	0.285	0.07
6	2	3	6	GUTE POTT 54% 0.70 L	1	4001731152798	Henkell & Co.	0.07	0.285	0.07
6	2	3	7	GUTE POTT 54% 0.70 L	1	4001731152798	Henkell & Co.	0.07	0.285	0.07
6	2	3	8	GUTE POTT 54% 0.70 L	1	4001731152798	Henkell & Co.	0.07	0.285	0.07
6	2	3	9	GUTE POTT 54% 0.70 L	1	4001731152798	Henkell & Co.	0.07	0.285	0.07
6	2	3	10	GUTE POTT 54% 0.70 L	1	4001731152798	Henkell & Co.	0.07	0.285	0.07
6	2	3	11	Beeren Müsli	1	4304218711700	Globus	0.164	0.22	0.054
6	2	3	12	Beeren Müsli	1	4304218711700	Globus	0.164	0.22	0.054
6	2	3	13	Beeren Müsli	1	4304218711700	Globus	0.164	0.22	0.054
6	2	2	1	Korrekt Jodsalz	1	4304218713728	Globus	0.067	0.143	0.046
6	2	2	2	Korrekt Jodsalz	1	4304218713728	Globus	0.067	0.143	0.046
6	2	2	3	Korrekt Jodsalz	1	4304218713728	Globus	0.067	0.143	0.046
6	2	2	4	Korrekt Jodsalz	1	4304218713728	Globus	0.067	0.143	0.046
6	2	2	5	Korrekt Jodsalz	1	4304218713728	Globus	0.067	0.143	0.046
6	2	2	6	Kaffeefilter Größe 2	1	2000088958154	Globus	0.122	0.173	0.042
6	2	2	7	Globus 12 Knödel halb&halb	1	4304218712578	Globus	0.135	0.183	0.026
6	2	2	8	Globus 12 Knödel halb&halb	1	4304218712578	Globus	0.135	0.183	0.026
6	2	2	9	Beeren Müsli	1	4304218711700	Globus	0.164	0.22	0.054
6	2	2	10	Beeren Müsli	1	4304218711700	Globus	0.164	0.22	0.054
6	2	2	11	Beeren Müsli	1	4304218711700	Globus	0.164	0.22	0.054

***Tabelle A.1**: Tabellarische Darstellung der Verräumung als Teil eines Planogramms*

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Package xmlns="http://www.wfmc.org/2009/XPDL2.2" xmlns:adapt="http://www.wfmc.org/2009/XPDL2.2" xmlns:ext="http
      ://www.wfmc.org/2009/XPDL2.2/ext" Id="838008c0-7a3f-11e1-53ca-782bcb9c92d9" Name="Semi-automatische
      Nachbestellung von Waren">
  <PackageHeader>
    <XPDLVersion>2.2</XPDLVersion>
    <Vendor>Software AG</Vendor>
    <Created>Thu Aug 01 12:29:45 CEST 2013</Created>
    <Description/>
  </PackageHeader>
  <RedefinableHeader>
    <Author>system</Author>
    <Countrykey>DE</Countrykey>
  </RedefinableHeader>
  <Participants/>
  <Applications/>
  <Pages>
    <Page Id="838008c0-7a3f-11e1-53ca-782bcb9c92d9PAGE" Name="Semi-automatische Nachbestellung von Waren"/>
  </Pages>
  <Pools>
    <Pool BoundaryVisible="true" Id="838008c0-7a3f-11e1-53ca-782bcb9c92d9" Name="Semi-automatische
          Nachbestellung von Waren" Process="838008c0-7a3f-11e1-53ca-782bcb9c92d9WORKFLOWPROCESS">
      <Lanes>
        ...
      </Lanes>
    </Pool>
  </Pools>
  <WorkflowProcesses>
    <WorkflowProcess Id="838008c0-7a3f-11e1-53ca-782bcb9c92d9WORKFLOWPROCESS" Name="Semi-automatische
          Nachbestellung von Waren">
      <ProcessHeader>
        <Created>Mar 30, 2012 10:08:20 AM</Created>
        <Description/>
      </ProcessHeader>
      <Participants/>
      <Applications/>
      <ActivitySets/>
      <Activities>
        <Activity Id="4fe536e3-5fd9-11e2-5c02-005056c00008" Name="XOR-Regel">
          <adapt:Font size="0" type=""/>
          <Description/>
          <Event>
            <StartEvent Trigger="None"/>
          </Event>
          <NodeGraphicsInfos>
            <NodeGraphicsInfo BorderColor="0,0,0" FillColor="255, 255, 255" Height="100" LaneId="838008c0-7a3f
                  -11e1-53ca-782bcb9c92d9" PageId="838008c0-7a3f-11e1-53ca-782bcb9c92d9PAGE" Width="100">
              <Coordinates XCoordinate="760" YCoordinate="1450"/>
            </NodeGraphicsInfo>
          </NodeGraphicsInfos>
        </Activity>
        <Activity Id="eed69cac-ca9b-11e2-55ac-0050569d399c" Name="Potentielle Lieferanten finden">
          <adapt:Font size="8" type="Arial"/>
          <Description/>
          <Event>
            <StartEvent Trigger="None"/>
          </Event>
          <NodeGraphicsInfos>
            <NodeGraphicsInfo BorderColor="0,0,0" FillColor="173,255,173" Height="180" LaneId="838008c0-7a3f-11
                  e1-53ca-782bcb9c92d9" PageId="838008c0-7a3f-11e1-53ca-782bcb9c92d9PAGE" Width="300">
              <Coordinates XCoordinate="661" YCoordinate="1951"/>
              <Coordinates XCoordinate="961" YCoordinate="1951"/>
              <Coordinates XCoordinate="961" YCoordinate="2131"/>
              <Coordinates XCoordinate="661" YCoordinate="2131"/>
            </NodeGraphicsInfo>
          </NodeGraphicsInfos>
        </Activity>
        ...
      </Activities>
      <Transitions>
        <Transition From="4fe536e3-5fd9-11e2-5c02-005056c00008" Id="4fe536ea-5fd9-11e2-5c02-005056c00008" To="4
              fe536e6-5fd9-11e2-5c02-005056c00008">
          <Description/>
          <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo BorderColor="51,51,51" PageId="838008c0-7a3f-11e1-53ca-782bcb9c92d9PAGE"
                  ToolId="ARIS">
              <Coordinates XCoordinate="810" YCoordinate="1550"/>
              <Coordinates XCoordinate="810" YCoordinate="1643"/>
            </ConnectorGraphicsInfo>
          </ConnectorGraphicsInfos>
        </Transition>
        ...
      </Transitions>
    </WorkflowProcess>
  </WorkflowProcesses>
  <ext:Kpis/>
</Package>
```

Programmauszug A.8: Auszug aus einer XPDL-Datei

Abbildungsverzeichnis

1.1 Darstellung der Verteilung von Computersystemen nach der Vorlage von Mark Weiser 6
1.2 Illustration einer instrumentierten Supermarktumgebung 7
1.3 Lichtschalterlayout im klassischen Stil untereinander und in Abhängigkeit vom Raummodell 10
1.4 Leitstand in einem Kraftwerk 10
1.5 3D-Planungskomponenten für Raumausstattung 12
1.6 DURMAD als Verbindung von Monitoring, Steuerung und Visualisierung . . 13

2.1 Verarbeitungskette von Daten in Smarten Umgebungen 18
2.2 Struktur eines Objektgedächtnisses basierend auf dem OMM-Format 21
2.3 Unterteilung von Sensoren mit Beispielen in Anlehnung an Baldauf et al. . . 26
2.4 Verarbeitungskreislauf in Smarten Umgebungen 28
2.5 Architektur im Sinne der ambienten Intelligenz inklusive der Middleware . . 29
2.6 Unterteilung von Smarten Umgebungen inklusive Beispiele 30
2.7 Cyber-Physische Umgebungen als Zusammenschluss mehrerer Cyber-Physischer Systeme 32
2.8 Cyber-Physisches System als Zusammenschluss von Sensoren, Aktuatoren, Objekten und Diensten 33
2.9 Virtualisierung im Sinne des Internet der Dinge und Dienste 34
2.10 Differenzierung von verteilten Systemen anhand ihrer Kommunikationsverbindungen 36
2.11 Entwicklung der Kommunikationsinfrastrukturen 39
2.12 Mögliche Darstellungsvarianten für Informationen 45
2.13 2D- und 3D-Darstellungen von Umgebungen 46
2.14 Einteilung der verschiedenen Welten und deren Verschmelzungen 47
2.15 Beispiel eines Flugzeugsimulators 48
2.16 Darstellung des Aktivitätslevels mittels Morphismus eines Avataren 48
2.17 Darstellung zweier Dashboards mit mehreren Darstellungskomponenten . . . 56
2.18 Darstellung in den Betriebszentralen von Verkehrsbetrieben 57

3.1 Zwei Alternativen für Gateways in MQTT-SN 67

3.2 Grafische Oberfläche zur Erstellung und Bearbeitung von Verarbeitungsgraphen in Aurora . . . 70
3.3 Grafische Oberfläche von STREAM . . . 74
3.4 Grafische Oberfläche von Cayuga . . . 76
3.5 Darstellung der instrumentierten Steckerleiste von Lifton . . . 78
3.6 Virtueller Datenteich mit Avatar zur Repräsentation von erfassten Kontextfaktoren und reales Gegenstück . . . 79
3.7 Second Life als virtuelle Testumgebung für Systeme zur Positionserkennung . 81
3.8 Darstellung des Eolus One Park in Second Life . . . 82
3.9 Darstellung der Schokoladenfabrik als reales Bild und virtuelles Modell in Multiverse . . . 83
3.10 Virtuelle Darstellung einer instrumentierten Wohnung bei iWorlds . . . 84
3.11 Ausschnitt aus dem Modellierungstool YAMAMOTO . . . 85
3.12 Vergleich real virtuell bei YAMAMOTO . . . 86

4.1 Darstellung einer modellierten Umgebung . . . 98
4.2 Kreislauf der Modellierung mit virtuellen Simulationen und Validierung reeller Prototypen . . . 100
4.3 Bearbeitungsoberfläche von Blender . . . 101
4.4 Darstellung eines Modells in YAMAMOTO mit Regalen und RFID-Tags in Form von Punktobjekten . . . 103
4.5 Interaktion zwischen Sensoren und Aktuatoren in Dual Reality Systemen . . 109
4.6 Dual Reality in Cyber-Physischen Umgebungen . . . 110

5.1 Architekturskizze des Dual Reality Management Dashboards . . . 120
5.2 Datenquellen des Dual Reality Management Dashboards . . . 120
5.3 Darstellung der OWL-Semantik für Sensoren und Aktuatoren in DURMAD . 123
5.4 Schnittstellen des Dual Reality Management Dashboards . . . 125
5.5 Ausschnitt aus dem Datenbankschema von DURMAD . . . 126
5.6 Kernmodule des Dual Reality Management Dashboards . . . 129
5.7 Grafische Oberfläche von DURMAD in der Java3D-Version . . . 130
5.8 Visualisierung von Notifikationen mittels des Kontextmenüs von DURMAD . 132
5.9 Objektbezogene Informationsdarstellung im 3D-Raum . . . 133
5.10 Darstellung derselben Szene in zwei unterschiedlichen Visualisierungskomponenten . . . 134
5.11 Graphbasierte Darstellung von Abverkaufszahlen . . . 135
5.12 Darstellung von Konfidenzwerten durch farbliche Überlagerung . . . 136
5.13 Transparente Darstellung von Konfidenzwerten . . . 137
5.14 Darstellung der Konfidenz durch Variation der Objektdimension . . . 137
5.15 Verarbeitungsautomatismen des Dual Reality Management Dashboards . . . 140
5.16 Grafische Oberfläche zur Erstellung und Anpassung von Regeln . . . 142
5.17 Farbliche Hervorhebung von Objekten zur Darstellung eines durch ein Agentensystem ermittelten Handlungsbedarfs . . . 143
5.18 Grafische Oberfläche zum Editieren von BI-Diensten . . . 144
5.19 Kontroll- und Steuerungssysteme des Dual Reality Management Dashboards 145

5.20 Synchronisation zwischen virtueller und realer Kamera 145
5.21 Architektur des Dual Reality Management Dashboards 147

6.1 Erweiterung der AmI-Architektur um Dual Reality-Interaktion in CPE 150
6.2 Filter als Teil des EBS-Client 156
6.3 Typfilter 157
6.4 Inhaltsfilter 157
6.5 Fensterfilter 158
6.6 Akkumulation 159
6.7 Kombination 159
6.8 Transformation 160
6.9 Aufsplittung 160
6.10 Monitor 161
6.11 Grafische Oberfläche zur Erstellung und Bearbeitung von Filterketten 162
6.12 Verarbeitungsablauf und Architektur des Event Broadcasting Service (EBS) . 163
6.13 Implementierte Queue-Struktur zum unabhängigen Lesen und Schreiben . . . 164
6.14 Übertragungsdauer in Abhängigkeit von der Anzahl der versendeten Events und der verbundenen Clients 168
6.15 Übertragungsdauer unter Verwendung unterschiedlicher Serialisierer 169
6.16 Übertragungszeiten in Abhängigkeit vom Datenvolumen bei 10 Events pro Sekunde und einem verbundenen Client 170
6.17 Veränderung der Anzahl versendeter Events bei (De-) Aktivierung eines dynamischen Inhaltsfilters 171
6.18 Kommunikationsflussvisualisierung von Events zwischen Clients und Server in DuRMaD 174
6.19 Die drei Modi des Dual Reality Frameworks 176

7.1 Innovative Retail Lab (IRL) - eine Cyber-Physische Handelsumgebung 185
7.2 Instrumentierter Kühlschrank zur Erstellung einer Einkaufsliste 188
7.3 Instrumentierung des IRL SmartCart 189
7.4 Benutzeroberfläche beim IRL SmartCart 190
7.5 Instrumentierte Regalsysteme 192
7.6 Mobile Endgeräte als Teil der IRL-Umgebung 194
7.7 Easy Checkout 195
7.8 Initialisierungsphase bei DuRMaD im IRL 198
7.9 Grafischer Editor zur Aktuatoransteuerung aus DuRMaD in Abhängigkeit von der definierten Ontologie 199
7.10 Grafische Oberfläche zur Eingabe der Parameter für ein automatisches Spaceplanning 201
7.11 Darstellung eines Prozessmodells in DuRMaD 205
7.12 Verbindung der einzelnen Komponenten beim Regalberaterdienst mittels entsprechender Events 208

8.1 Zusammenspiel der einzelnen Komponenten mit dem DR-Framework, der Kombination von EBS und DuRMaD 216

A.2 Resultierendes Bild einer virtuellen Verräumung als Teil eines Planogramms . 233

Tabellenverzeichnis

2.1 Unterscheidungsmerkmale der Gruppen von Smarten Umgebungen 31
2.2 Differenzierungsmerkmale der Kommunikation in Cyber-Physischen Umgebungen 38

3.1 Vergleich der Kommunikationsinfrastrukturen mit dem Event Broadcasting Service (EBS) 90
3.2 Vergleich der Dual Reality Systeme mit dem Dual Reality Management Dashboard (DURMAD) 93

4.1 Differenzierungsmerkmale zwischen den Visualisierungskomponenten 115
4.2 Unterscheidungskriterien bei Darstellungsvarianten 116

6.1 Auflistung und Kurzbeschreibung der in der Eventgrundstruktur enthaltenen Parameter 154
6.2 Kurzdarstellung der im EBS integrierten Filter und Vorverarbeitungsmodule . 179

7.1 Eventkommunikation zwischen den Diensten im IRL 204

A.1 Tabellarische Darstellung der Verräumung als Teil eines Planogramms 234

Definitionsverzeichnis

Definition 1.1 Smarte Umgebung 5
Definition 1.2 Dual Reality 12
Definition 2.1 Modalität 18
Definition 2.2 Kontext 22
Definition 2.3 Lokationsbasierte Informationssysteme 22
Definition 2.4 Sensor 25
Definition 2.5 Aktuator 27
Definition 2.6 Blackboard 39
Definition 2.7 Tupel 40
Definition 2.8 Tupelraum 40
Definition 2.9 Event 42
Definition 2.10 Datenstrom 44
Definition 2.11 Fenster 44
Definition 2.12 Business Intelligence Systeme 55
Definition 2.13 Dashboard 55
Definition 4.1 Modellierung 97
Definition 4.2 Virtualisierung 104
Definition 4.3 Erweiterte Dual Reality 112
Definition 5.1 Management 119
Definition 7.1 Datenlager 182

Literaturverzeichnis

Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., und Zdonik, S. (2003). Aurora: a new model and architecture for data stream management. The VLDB Journal, 12(2):120–139.

Abowd, G. D. und Sterbenz, J. P. (2000). Final report on the inter-agency workshop on research issues for smart environments. Personal Communications, IEEE, 7(5):36–40.

acatech, editor (2011). Cyber-Physical Systems: Innovationsmotor für Mobilität, Gesundheit, Energie und Produktion. acatech POSITION. Springer, Berlin.

Adaikkalavan, R. und Chakravarthy, S. (2006). SnoopIB: Interval-based event specification and detection for active databases. Data & Knowledge Engineering, 59(1):139–165.

Aguilera, M. K., Strom, R. E., Sturman, D. C., Astley, M., und Chandra, T. D. (1999). Matching Events in a Content-based Subscription System. In Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing, PODC '99, pages 53–61, New York, NY, USA. ACM.

Aitenbichler, E. (2006). System Support for Ubiquitous Computing. PhD thesis, Darmstadt University of Technology.

Aitenbichler, E., Kangasharju, J., und Muhlhauser, M. (2005). Experiences with MundoCore. In Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOMW '05, pages 168–172, Washington, DC, USA. IEEE Computer Society.

Aitenbichler, E., Kangasharju, J., und Mühlhäuser, M. (2007). MundoCore: A Light-weight Infrastructure for Pervasive Computing. Pervasive Mob. Comput., 3(4):332–361.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., und Cayirci, E. (2002). Wireless sensor networks: a survey. Computer networks, 38(4):393–422.

Anderson, T., Peterson, L., Shenker, S., und Turner, J. (2005). Overcoming the Internet Impasse through Virtualization. Computer, 38(4):34–41.

Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., und Widom, J. (2004). STREAM: The Stanford Data Stream Management System. Technical Report 2004-20, Stanford InfoLab.

Arasu, A., Babcock, B., Babu, S., Datar, M., Ito, K., Nishizawa, I., Rosenstein, J., und Widom, J. (2003). STREAM: The Stanford Stream Data Manager (Demonstration Description). In Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pages 665–665. ACM.

Arasu, A., Babu, S., und Widom, J. (2006). The CQL continuous query language: semantic foundations and query execution. The VLDB Journal—The International Journal on Very Large Data Bases, 15(2):121–142.

Armac, I. und Retkowitz, D. (2007). Simulation of Smart Environments. In Proceedings of the IEEE International Conference on Pervasive Services 2007 (ICPS'07), pages 257–266. IEEE Press.

Atzori, L., Iera, A., und Morabito, G. (2010). The Internet of Things: A survey. Comput. Netw., 54(15):2787–2805.

Augusto, J. C., Nakashima, H., und Aghajan, H. (2010). Ambient Intelligence and Smart Environments: A State of the Art. In Handbook of Ambient Intelligence and Smart Environments, pages 3–31. Springer.

Babcock, B., Babu, S., Datar, M., Motwani, R., und Widom, J. (2002). Models and Issues in Data Stream Systems. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 1–16. ACM.

Back, M., Kimber, D., Rieffel, E., Dunnigan, A., Liew, B., Gattepally, S., Foote, J., Shingu, J., und Vaughan, J. (2010a). The Virtual Chocolate Factory: Building a Real World Mixed-reality System for Industrial Collaboration and Control. In Multimedia and Expo (ICME), 2010 IEEE International Conference on, pages 1160–1165.

Back, M., Kimber, D., Rieffel, E., Dunnigan, A., Liew, B., Gattepally, S., Foote, J., Shingu, J., und Vaughan, J. (2010b). The virtual chocolate factory: Mixed reality industrial collaboration and control. In Proceedings of the International Conference on Multimedia, MM '10, pages 1505–1506, New York, NY, USA. ACM.

Baldauf, M., Dustdar, S., und Rosenberg, F. (2007). A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing, 2(4):263–277.

Baldoni, R. (2005). The Publish/Subscribe Communication Paradigm and its Application to Mobile Systems. MINEMA Summer School.

Beckmann, C. und Dey, A. (2003). Siteview: Tangibly programming active environments with predictive visualization. In Adjunct Proceedings of the Fifth International Conference on Ubiquitous Computing, pages 167–168.

Beigl, M., Decker, C., Krohn, A., Riedel, T., und Zimmer, T. (2005). μParts: Low Cost Sensor Networks at Scale. In International Conference on Ubiquitous Computing, Demonstration.

Bernsen, N. O. (2002). Multimodality in Language and Speech Systems - From Theory to Design Support Tool. In Multimodality in language and speech systems, pages 93–148. Springer.

Birrell, A. D. und Nelson, B. J. (1984). Implementing Remote Procedure Calls. ACM Trans. Comput. Syst., 2(1):39–59.

Bleser, G., Becker, M., und Stricker, D. (2007). Real-time vision-based tracking and reconstruction. Journal of Real-Time Image Processing, 2(2-3):161–175.

Bonato, P. (2003). Wearable sensors/systems and their impact on biomedical engineering. Engineering in Medicine and Biology Magazine, IEEE, 22(3):18–20.

Bozdag, E., Mesbah, A., und van Deursen, A. (2007). A comparison of push and pull techniques for ajax. In Proceedings of the 2007 9th IEEE International Workshop on Web Site Evolution, WSE '07, pages 15–22, Washington, DC, USA. IEEE Computer Society.

Brandherm, B. und Schwartz, T. (2005). Geo Referenced Dynamic Bayesian Networks for User Positioning on Mobile Systems. In Strang, T. und Linnhoff-Popien, C., editors, Location- and Context-Awareness, volume 3479 of Lecture Notes in Computer Science, pages 223–234. Springer Berlin Heidelberg.

Brandherm, B., Ullrich, S., und Prendinger, H. (2008a). Simulation Framework in Second Life with Evaluation Functionality for Sensor-based Systems. In Stevenson, G., Neely, S., und Kray, C., editors, Proceedings of the 2nd Workshop on Ubiquitous Systems Evaluation (USE '08), volume 393, page 68. CEUR Workshop Proceedings.

Brandherm, B., Ullrich, S., und Prendinger, H. (2008b). Simulation of sensor-based tracking in second life. In AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, pages 1689–1690, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Braubach, L., Pokahr, A., und Lamersdorf, W. (2005). Jadex: A BDI-Agent System Combining Middleware and Reasoning. In Unland, R., Calisti, M., und Klusch, M., editors, Software Agent-Based Applications, Platforms and Development Kits, Whitestein Series in Software Agent Technologies, pages 143–168. Birkhäuser Basel.

Brenna, L., Demers, A., Gehrke, J., Hong, M., Ossher, J., Panda, B., Riedewald, M., Thatte, M., und White, W. (2007). Cayuga: A High-performance Event Processing Engine. In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD '07, pages 1100–1102, New York, NY, USA. ACM.

Bruneau, J. und Consel, C. (2012). DiaSim: a simulator for pervasive computing applications. Software: Practice and Experience, pages 885—909.

Butz, A. und Krüger, A. (2006). Applying the Peephole Metaphor in a Mixed-Reality Room. Computer Graphics and Applications, IEEE, 26(1):56–63.

Carrier, N. und Gelernter, D. (1998). Linda and Friends. Distributed Shared Memory: Concepts and Systems, 21:177–185.

Carzaniga, A., Rosenblum, D. S., und Wolf, A. L. (2001). Design and Evaluation of a Wide-Area Event Notification Service. ACM Transactions on Computer Systems (TOCS), 19(3):332–383.

Chakravarthy, S. (1997). Sentinel: An Object-Oriented DBMS With Event-Based Rules. In ACM SIGMOD Record, volume 26, pages 572–575. ACM.

Chakravarthy, S. und Mishra, D. (1994). Snoop: An expressive event specification language for active databases. Data & Knowledge Engineering, 14(1):1–26.

Chan, A. T. S., Chuang, S.-N., Cao, J., und Leong, H.-V. (2004). An Event-Driven Middleware for Mobile Context Awareness. The Computer Journal, 47(3):278–288.

Chan, M., Estève, D., Escriba, C., und Campo, E. (2008). A review of smart homes—Present state and future challenges. Computer Methods and Programs in Biomedicine, 91(1):55–81.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., und Shah, M. (2003a). TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD '03, pages 668–668. ACM.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S. R., Reiss, F., und Shah, M. A. (2003b). TelegraphCQ: Continuous Dataflow Processing. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pages 668–668. ACM.

Chandrasekaran, S. und Franklin, M. (2004). Remembrance of Streams Past: Overload-Sensitive Management of Archived Streams. In Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, pages 348–359. VLDB Endowment.

Chaudhuri, S. und Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. SIGMOD Rec., 26(1):65–74.

Chen, G., Kotz, D., et al. (2000). A Survey of Context-Aware Mobile Computing Research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

Coleman, B. (2009). Using Sensor Inputs to Affect Virtual and Real Environments. Pervasive Computing, IEEE, 8(3):16–23.

Cook, D. und Das, S. (2004). Smart Environments: Technology, Protocols and Applications. Wiley-Interscience.

Cook, D. J. (2009). Multi-agent smart environments. Journal of Ambient Intelligence and Smart Environments, 1(1):51–55.

Cook, D. J. und Das, S. K. (2007). How smart are our environments? An updated look at the state of the art. Pervasive Mob. Comput., 3(2):53–73.

Cruz-Neira, C., Sandin, D. J., und DeFanti, T. A. (1993). Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '93, pages 135–142, New York, NY, USA. ACM.

Cugola, G. und Margara, A. (2009). RACED: an Adaptive Middleware for Complex Event Detection. In Proceedings of the 8th International Workshop on Adaptive and Reflective MIddleware, ARM '09, pages 5:1–5:6, New York, NY, USA. ACM.

Cugola, G. und Margara, A. (2010). TESLA: A Formally Defined Event Specification Language. In Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS '10, pages 50–61, New York, NY, USA. ACM.

Cugola, G. und Margara, A. (2012a). Complex Event Processing with T-REX. J. Syst. Softw., 85(8):1709–1728.

Cugola, G. und Margara, A. (2012b). Processing flows of information: From data stream to complex event processing. ACM Comput. Surv., 44(3):15:1–15:62.

Curhan, R. C. (1973). Shelf Space Allocation and Profit Maximization in Mass Retailing. The Journal of Marketing, pages 54–60.

Das, S. K. und Cook, D. J. (2005). Designing Smart Environments: A Paradigm Based on Learning and Prediction. In Pal, S., Bandyopadhyay, S., und Biswas, S., editors, Pattern Recognition and Machine Intelligence, volume 3776 of Lecture Notes in Computer Science, pages 80–90. Springer Berlin Heidelberg.

Davies, M. und Callaghan, V. (2012). iworlds: Generating artificial control systems for simulated humans using virtual worlds and intelligent environments. JAISE, 4(1):5–27.

DeLillo, B. P. (2010). WebGLU Development Library for WebGL. In ACM SIGGRAPH 2010 Posters, SIGGRAPH '10, pages 135:1–135:1, New York, NY, USA. ACM.

Demers, A., Gehrke, J., Hong, M., Riedewald, M., und White, W. (2006). Towards Expressive Publish/Subscribe Systems. In Proceedings of the 10th International Conference on Advances in Database Technology, EDBT'06, pages 627–644, Berlin, Heidelberg. Springer-Verlag.

Dey, A. K. (2001). Understanding and Using Context. Personal Ubiquitous Comput., 5(1):4–7.

Dey, A. K., Abowd, G. D., und Salber, D. (2000). A Context-Based Infrastructure for Smart Environments. In Managing Interactions in Smart Environments, pages 114–128. Springer.

Dey, A. K., Hamid, R., Beckmann, C., Li, I., und Hsu, D. (2004). a cappella: programming by demonstration of context-aware applications. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 33–40. ACM.

Dreze, X., Hoch, S. J., und Purk, M. E. (1995). Shelf Management and Space Elasticity. Journal of Retailing, 70(4):301–326.

Eckert, M. und Bry, F. (2009). Complex Event Processing (CEP). Informatik-Spektrum, 32(2):163–167.

Emmelhainz, M. A., Stock, J. R., und Emmelhainz, L. W. (1991). Consumer responses to retail stock-outs. Journal of Retailing.

Endres, C. (2003). Towards a Software Architecture for Device Management in Instrumented Environments. In Adjunct Proceedings of UbiComp-The Fifth International Conference on Ubiquitous Computing, pages 245–246, Seattle, Washington USA.

Erl, T. (2004). Service-Oriented Architecture. Prentice Hall Englewood Cliffs.

Erman, L., Hayes-Roth, F., Lesser, V., und Reddy, D. (1980). The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. Computing Surveys, 12(2):213–253.

Eugster, P., Garbinato, B., und Holzer, A. (2008). Design and Implementation of the Pervaho Middleware for Mobile Context-Aware Applications. In e-Technologies, 2008 International MCETECH Conference on, pages 125–135. IEEE.

Eugster, P. T., Felber, P. A., Guerraoui, R., und Kermarrec, A.-M. (2003). The Many Faces of Publish/Subscribe. ACM Comput. Surv., 35(2):114–131.

Feld, M. und Kahl, G. (2008). Integrated Speaker Classification for Mobile Shopping Applications. In Nejdl, W., Kay, J., Pu, P., und Herder, E., editors, Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, volume 5149 of Lecture Notes in Computer Science, LNCS, pages 288–291. L3S Research Center, Hannover, Germany, Springer.

Few, S. (2006). Information Dashboard Design: The Effective Visual Communication of Data. O'Reilly Media, Inc.

Few, S. (2007). Dashboard Confusion Revisited. Perceptual Edge.

Fichtner, M., Großmann, A., und Thielscher, M. (2003). Intelligent Execution Monitoring in Dynamic Environments. Fundam. Inf., 57(2-4):371–392.

Fishkin, K., Philipose, M., und Rea, A. (2005). Hands-on rfid: wireless wearables for detecting use of objects. In Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on, pages 38–41.

Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence, 19(1):17–37.

Forman, I. R. (2004). Java Reflection in Action. In Action Series. Manning.

Fost, M. (2014). Handelsstrukturen von produzierenden Unternehmen. In E-Commerce-Strategien für produzierende Unternehmen, pages 33–51. Springer Fachmedien Wiesbaden.

Frey, J., Neßelrath, R., und Stahl, C. (2011). An Open Standardized Platform for Dual Reality Applications. In Proceedings of the Second International Workshop on Location Awareness for Mixed and Dual Reality. ACM.

Funkwerk Information Technologies (2009). Betriebsinformationssystem: Aktuelle Betriebsinformationen - jederzeit und überall verfügbar. Online (http://www.funkwerk-it.com/wDeutsch/downloads/prospekte/PDB_LeiBIT_dt_140409_Neu.pdf).

Galton, A. und Augusto, J. C. (2002). Two Approaches to Event Definition. In Database and Expert Systems Applications, pages 547–556. Springer.

Gelernter, D. und Bernstein, A. J. (1982). Distributed Communication via Global Buffer. In Proceedings of the first ACM SIGACT-SIGOPS symposium on Principles of distributed computing, pages 10–18. ACM.

Gelernter, D. und Carriero, N. (1992). Coordination Languages and their Significance. Communications of the ACM, 35(2):96.

Gershon, N. (1999). Knowing What We Don't Know: How to Visualize an Imperfect World. SIGGRAPH Comput. Graph., 33(3):39–41.

Gervasi, V., Brunberg, S., Swenson, J. E., und Bowman, J. (2006). An Individual-Based Method to Measure Animal Activity Levels: A Test on Brown Bears. Wildlife Society Bulletin, 34(5):1314–1319.

Ghezzi, C., Mandrioli, D., und Morzenti, A. (1990). TRIO: A Logic Language for Executable Specifications of Real-time Systems. J. Syst. Softw., 12(2):107–123.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2):199–220.

Gruen, T., Corsten, D., Bharadwaj, S., und of America, G. M. (2002). Retail Out-of-stocks: A Worldwide Examination of Extent, Causes and Consumer Responses. Grocery Manufacturers of America (GMA).

Grün, J., Gerber, T., und Herfet, T. (2012). Multi-Reality Interfaces - Remote Monitoring and Maintenance in modular Factory Environments. In Proceedings14th IFAC Symposium on Information Control Problems in Manufacturing, pages 987–992. IFAC.

Gubbi, J., Buyya, R., Marusic, S., und Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. Future Gener. Comput. Syst., 29(7):1645–1660.

Haupert, J. (2013). DOMeMan: Repräsentation, Verwaltung und Nutzung von digitalen Objektgedächtnissen, volume 339 of Dissertationen zur Künstlichen Intelligenz (DISKI). Akademische Verlagsgesellschaft, Berlin.

Heckmann, D. (2006). Ubiquitous User Modeling. Dissertationen zur Künstlichen Intelligenz - DISKI. Akademische Verlagsgesellschaft.

Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., und Wilamowitz-Moellendorff, M. (2005). Gumo – The General User Model Ontology. In Ardissono, L., Brna, P., und Mitrovic, A., editors, User Modeling 2005, volume 3538 of Lecture Notes in Computer Science, pages 428–432. Springer Berlin Heidelberg.

Henricksen, K., Indulska, J., McFadden, T., und Balasubramaniam, S. (2005). Middleware for Distributed Context-aware Systems. In Proceedings of the 2005 Confederated International Conference on On the Move to Meaningful Internet Systems, OTM'05, pages 846–863, Berlin, Heidelberg. Springer-Verlag.

Hübner, A. H. und Kuhn, H. (2012). Retail category management: State-of-the-art review of quantitative research and software applications in assortment and shelf space management. Omega, 40(2):199–209.

Hühn, A. E., Khan, V.-J., Lucero, A., und Ketelaar, P. (2012). On the Use of Virtual Environments for the Evaluation of Location-Based Applications. In Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems, pages 2569–2578. ACM.

Hunkeler, U., Truong, H. L., und Stanford-Clark, A. (2008). MQTT-S – A Publish/Subscribe Protocol For Wireless Sensor Networks. In Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on, pages 791–798.

Infso D.4 Networked Enterprise & RFID, Infso G.2 Micro & Nanosystems, und in cooperation with the Working Group RFID of the ETP EPoSS (2008). Internet of Things in 2020: A Roadmap for the Future. Information Society and Media, Tech. Rep, Version 1.1.

Inmon, W. H. (1992). Building the Data Warehouse. QED Information Sciences, Inc., Wellesley, MA, USA.

ISO (2008). Information Technology – User Interfaces – Universal Remote Console – 5 parts. ISO ISO/IEC 2452, International Organization for Standardization.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., und Fitzgibbon, A. (2011). Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11, pages 559–568, New York, NY, USA. ACM.

Johanson, B. und Fox, A. (2002). The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, WMCSA '02, pages 83–93, Washington, DC, USA. IEEE Computer Society.

Johanson, B., Fox, A., und Winograd, T. (2002). The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. IEEE Pervasive Computing, 1(2):67–74.

Jung, R. (2009). Ambiente Audionotifikation - Ein System zur kontext-sensitiven Integration von nicht-intrusiven Notifikationssignalen in emotionsklassifizierten Soundscapes. Phd-thesis, Saarland University, Dept. of Computer Science, Saarbrücken.

Jung, R., Kahl, G., und Spassova, L. (2011a). The Shopping Sound Experience. The Journal of the Acoustical Society of America, 130(4):2545–2545.

Jung, R., Spassova, L., und Kahl, G. (2011b). Product-Awareness Through Smart Audio Navigation in a Retail Environment. In Proceedings of the Seventh International Conference on Intelligent Environments, pages 186–191. IEEE Computer Society.

Kagermann, H., Lukas, W. D., und Wahlster, W. (2011). Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution. VDI nachrichten, 13.

Kagermann, H., Wahlster, W., und Helbig, J., editors (2012). Bericht der Promotorengruppe Kommunikation – Im Fokus: Das Zukunftsprojekt Industrie 4.0, Handlungsempfehlungen zur Umsetzung. Forschungsunion Wirtschaft – Wissenschaft, Berlin.

Kahl, G. (2013a). A Plugin Framework to Control Electronic Shelf Labels. In Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, UbiComp '13 Adjunct, pages 1007–1014, New York, NY, USA. ACM.

Kahl, G. (2013b). A Visual Monitoring and Management Tool for Smart Environments. In Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion, IUI '13 Companion, pages 93–94, New York, NY, USA. ACM.

Kahl, G. und Bürckert, C. (2012). Architecture to Enable Dual Reality for Smart Environments. In Eighth International Conference on Intelligent Environments, pages 42–49, Guanajuato, México.

Kahl, G., Bürckert, C., Spassova, L., und Schwartz, T. (2012). Event Broadcasting Service – An Event-Based Communication Infrastructure. In Proceedings of the Second International Workshop on Location Awareness for Mixed and Dual Reality (LAMDa'12), pages 9–12. online.

Kahl, G., Leichtenstern, K., Schöning, J., Spassova, L., und Krüger, A. (2009). A Contextual Learning Game for Toddlers Installed on an Interactive Display Attached to a Shopping Cart. In Workshop on Pervasive Computing Education (PerED-09). o.A.

Kahl, G., Magerkurth, C., Preißinger, J., Gebhard, P., und Weyl, B. (2013a). Enhancement of Consumer Support in Retail Scenarios by Utilization of Semantic Product Memories. In Wahlster, W., editor, SemProM, Cognitive Technologies, pages 329–347. Springer Berlin Heidelberg.

Kahl, G. und Paradowski, D. (2013). A Privacy-aware Shopping Scenario. In Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion, IUI '13 Companion, pages 107–108, New York, NY, USA. ACM.

Kahl, G., Schwartz, T., und Brandherm, B. (2013b). Fusion of Multiple Datasets to Support Location-based Services in Retail Applications. In Proceedings of the 3rd International Workshop on Location Awareness in Mixed and Dual Reality (LAMDa'13), pages 9–12. DFKI.

Kahl, G., Spassova, L., und Krüger, A. (2010). User-adaptive advertisement in retail environments. In Pervasive Advertising and Shopping 2010. International Conference on Pervasive Computing (Pervasive-2010), 8th International Conference on Pervasive Computing, May 17, Helsinki, Finland. Springer.

Kahl, G., Spassova, L., Schöning, J., Gehring, S., und Krüger, A. (2011a). IRL SmartCart - A User-adaptive Context-aware Interface for Shopping Assistance. In Proceedings of the 16th International Conference on Intelligent User Interfaces, IUI '11, pages 359–362, New York, NY, USA. ACM.

Kahl, G., Warwas, S., Liedtke, P., Spassova, L., und Brandherm, B. (2011b). Management Dashboard in a Retail Scenario. In Kahl, G., Schwartz, T., Nurmi, P., Dim, E., und Forsblom, A., editors, Workshop on Location Awareness in Dual and Mixed Reality (LAMDa 11), pages 22–25. Online-Proceedings.

Kahl, G., Wasinger, R., Schwartz, T., und Spassova, L. (2008). Three Output Planning Strategies for Use in Context-aware Computing Scenarios. In Proceedings of the AISB 2008 Symposium on Multimodal Output Generation (MOG 2008), pages 46–49. Online-Proceedings.

Kahn, J. M., Katz, R. H., und Pister, K. S. J. (1999). Next Century Challenges: Mobile Networking for "Smart Dust". In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, MobiCom '99, pages 271–278, New York, NY, USA. ACM.

Kanade, T., Rander, P., und Narayanan, P. (1997). Virtualized Reality: Constructing Virtual Worlds from Real Scenes. MultiMedia, IEEE, 4(1):34–47.

Kato, H. und Billinghurst, M. (1999). Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In Proceedings of the 2Nd IEEE and ACM International Workshop on Augmented Reality, IWAR '99, pages 85–94. IEEE, IEEE Computer Society.

Kim, H., Lee, W., und Woo, W. (2009). CAMAR tag framework: Context-Aware Mobile Augmented Reality Tag Framework for Dual-Reality Linkage. In Ubiquitous Virtual Reality, 2009. ISUVR'09. International Symposium on, pages 39–42. IEEE.

Kim, S. und Dey, A. K. (2009). Simulated Augmented Reality Windshield Display As a Cognitive Mapping Aid for Elder Driver Navigation. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09, pages 133–142, New York, NY, USA. ACM.

Kimber, D., Shingu, J., Vaughan, J., Arendash, D., Lee, D., und Back, M. (2012). Through the Looking-Glass: Mirror Worlds for Augmented Awareness & Capability. In Proceedings of the 20th ACM international conference on Multimedia, pages 1269–1270. ACM.

Klein, F., Rubinstein, D., Sons, K., Einabadi, F., Herhut, S., und Slusallek, P. (2013). Declarative AR and Image Processing on the Web with Xflow. In Proceedings of the 18th International Conference on 3D Web Technology, Web3D '13, pages 157–165, New York, NY, USA. ACM.

Klein, F., Sons, K., Rubinstein, D., Byelozyorov, S., John, S., und Slusallek, P. (2012). Xflow - Declarative Data Processing for the Web. In Proceedings of the 17th International Conference on Web 3D Technology, Web3D '12, pages 37–45, Los Angeles, California. ACM.

Klusch, M., editor (2008). On Agent-Based Semantic Service Coordination. Universität des Saarlandes.

Koubarakis, M. (2003). Multi-agent Systems and Peer-to-Peer Computing: Methods, Systems, and Challenges. In Klusch, M., Omicini, A., Ossowski, S., und Laamanen, H., editors, Cooperative Information Agents VII, volume 2782 of Lecture Notes in Computer Science, pages 46–61. Springer Berlin Heidelberg.

Kourouthanassis, P., Spinellis, D., Roussos, G., und Giaglis, G. M. (2002). Intelligent Cokes and Diapers: MyGROCER Ubiquitous Computing Environment. In First International Mobile Business Conference, pages 150–172.

Kreylos, O., Bethel, E., Ligocki, T., und Hamann, B. (2003). Virtual-Reality Based Interactive Exploration of Multiresolution Data. In Farin, G., Hamann, B., und Hagen, H., editors, Hierarchical and Geometrical Methods in Scientific Visualization, Mathematics and Visualization, pages 205–224. Springer Berlin Heidelberg.

Kröner, A., Gebhard, P., Spassova, L., Kahl, G., und Schmitz, M. (2009). Informing Customers by Means of Digital Product Memories. In Schneider, M., Kröner, A., Olivier, P., und Stephan, P., editors, Proceedings of the 1st international Workshop on Digital Object Memories, volume 4 of Ambient Intelligence and Smart Environments, AISE, pages 21–26. IOS Press, Amsterdam.

Kröner, A., Haupert, J., Hauck, C., Deru, M., und Bergweiler, S. (2013). Fostering Access to Data Collections in the Internet of Things. In Narzt, W. und Gordon-Ross, A.,

editors, UBICOMM 2013, The Seventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pages 65–68. IARIA, IARIA.

Kröner, A., Haupert, J., Seißler, M., Kiesel, B., Schennerleinen, B., Horn, S., Schreiber, D., und Barthel, R. (2011). Object Memory Modeling - W3C Incubator Group Report. Online (http://www.w3.org/2005/Incubator/omm/XGR-omm-20111026/).

Kröner, A., Kahl, G., Spassova, L., Feld, T., Mayer, D., Magerkurth, C., und Dada, A. (2010). Demonstrating the Application of Digital Product Memories in a Carbon Footprint Scenario. In Callaghan, V., Kameas, A., Egerton, S., Satoh, I., und Weber, M., editors, Proceedings of the Sixth International Conference on Intelligent Environments (IE'10), pages 164–169. IEEE Computer Society.

Krüger, A., Maaß, W., Paradowski, D., und Janzen, S. (2014). Empfehlungssysteme und integrierte Informationsdienste zur Steigerung der Wertschöpfung im stationären Handel, pages 273–291. W. Kohlhammer, Stuttgart.

Krüger, A., Schöning, J., und Olivier, P. (2011). How Computing Will Change the Face of Retail. IEEE Computer, 44(4):84–87.

Krüger, A., Spassova, L., und Jung, R. (2010). Innovative Retail Laboratory—Investigating Future Shopping Technologies. it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik, 52(2):114–118.

Lehman, T. J., McLaughry, S. W., und Wycko, P. (1999). T Spaces: The Next Wave. In Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8 - Volume 8, HICSS '99, pages 1–9, Washington, DC, USA. IEEE Computer Society.

Leichtenstern, K., André, E., und Rehm, M. (2010). Using the Hybrid Simulation for Early User Evaluations of Pervasive Interactions. In Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries, pages 315–324. ACM.

Leung, C. und Salga, A. (2010). Enabling WebGL. In Proceedings of the 19th International Conference on World Wide Web, WWW '10, pages 1369–1370, New York, NY, USA. ACM.

Lifton, J. (2007). Dual Reality: An Emerging Medium. Ph.D. Dissertation, Massachusetts Institute of Technology, Department of Media Arts and Sciences.

Lifton, J., Laibowitz, M., Harry, D., Gong, N.-W., Mittal, M., und Paradiso, J. A. (2009). Metaphor and Manifestation Cross-Reality with Ubiquitous Sensor/Actuator Networks. Pervasive Computing, IEEE, 8(3):24–33.

Lifton, J. und Paradiso, J. A. (2009). Dual Reality: Merging the Real and Virtual. In Proceedings of the First International ICST Conference on Facets of Virtual Environments (FaVE), pages 12–28.

Lifton, J., Seetharam, D., Broxton, M., und Paradiso, J. (2002). Pushpin Computing System Overview: A Platform for Distributed, Embedded, Ubiquitous Sensor Networks. In Pervasive Computing, pages 139–151. Springer.

Lindsay, A., Downs, D., und Lunn, K. (2003). Business processes–attempts to find a definition. Information & Software Technology, 45(15):1015–1019.

Liwicki, M., Thieme, S., Kahl, G., und Dengel, A. (2011). An Intelligent Shopping List - Combining Digital Paper with Product Ontologies. In König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R., und Jain, L., editors, Knowledge-Based and Intelligent Information and Engineering Systems, volume 6884 of Lecture Notes in Computer Science, pages 187–194. Springer Berlin Heidelberg.

Löchtefeld, M., Gehring, S., Schöning, J., und Krüger, A. (2010). ShelfTorchlight: Augmenting a Shelf using a Camera Projector Unit. In Adjunct Proceedings of the Eighth International Conference on Pervasive Computing. Workshop on personal projection (UBIPROJECTION-2010), pages 20–23.

Lok, B. C. (2004). Toward the Merging of Real and Virtual Spaces. Commun. ACM, 47(8):48–53.

Luckham, D. C. (2001). The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Luckham, D. C. und Frasca, B. (1998). Complex Event Processing in Distributed Systems. Computer Systems Laboratory Technical Report CSL-TR-98-754. Stanford University, Stanford, 28.

MacDougall, W. (2013). Industrie 4.0: Smart Manufacturing for the Future. Germany Trade and Invest, Gesellschaft für Außenwirtschaft und Standortmarketing mbH, Berlin.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., und Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, WSNA '02, pages 88–97, New York, NY, USA. ACM.

Mansouri-Samani, M. (1995). Monitoring of Distributed Systems. PhD thesis, University of London.

Mansouri-Samani, M. und Sloman, M. (1997). GEM: A Generalized Event Monitoring Language for Distributed Systems. Distributed Systems Engineering, 4:96–108.

Marquardt, N., Gross, T., Carpendale, S., und Greenberg, S. (2010). Revealing the Invisible: Visualizing the Location and Event Flow of Distributed Physical Devices. In Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction, TEI '10, pages 41–48, New York, NY, USA. ACM.

Medenica, Z., Kun, A. L., Paek, T., und Palinko, O. (2011). Augmented Reality vs. Street Views: A Driving Simulator Study Comparing Two Emerging Navigation Aids. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, MobileHCI '11, pages 265–274, New York, NY, USA. ACM.

Metzger, C. (2009). High Fidelity Shelf Stock Monitoring: A Framework for Retail Replenishment Optimization. Lightning Source Incorporated.

Milgram, P. und Kishino, F. (1994). A Taxonomy of Mixed Reality Visual Displays. IEICE TRANSACTIONS on Information and Systems, 77(12):1321–1329.

Murray, C. C., Talukdar, D., und Gosavi, A. (2010). Joint Optimization of Product Price, Display Orientation and Shelf-Space Allocation in Retail Category Management. Journal of Retailing, 86(2):125–136.

National Communications System Technology & Standards Division (1996). Federal Standard 1037C: Telecommunications: Glossary of Telecommunication Terms. Technical report, General Services Administration Information Technology Service, Arlington, USA.

Negash, S. (2004). Business intelligence. Communications of the Association for Information Systems, 13(1):177–195.

Negash, S. und Gray, P. (2008). Business Intelligence. In Handbook on Decision Support Systems 2, International Handbooks Information System, pages 175–193. Springer Berlin Heidelberg.

Newman, N. (2014). Apple iBeacon technology briefing. Journal of Direct, Data and Digit Marketing Practice, 15(3):222–225.

Norman, D. A. (2002). The Design of Everyday Things. Basic books.

Nurmi, P., Salovaara, A., Bhattacharya, S., Pulkkinen, T., und Kahl, G. (2011). Influence of Landmark-based Navigation Instructions on User Attention in Indoor Smart Spaces. In Proceedings of the 16th international conference on Intelligent user interfaces, IUI '11, pages 33–42, New York, NY, USA. ACM.

Occello, M. und Demazeau, Y. (1994). Building Real Time Agents using Parallel Blackboards and its use for Mobile Robotics. In Systems, Man, and Cybernetics, 1994. IEEE International Conference on'Humans, Information and Technology', volume 2, pages 1610–1615. IEEE.

Oh, Y., Kang, C., und Woo, W. (2009). U-VR Simulator Linking Real and Virtual Environments based on Context-Awareness. Proc. of IWUVR'09, pages 52–55.

Pflegera, K. und Hayes-Roth, B. (1997). An Introduction to Blackboard-Style Systems Organization. Knowledge Systems Laboratory, Stanford University, Stanford KSL-98-03, 29.

Prendinger, H., Brandherm, B., und Ullrich, S. (2009). A Simulation Framework for Sensor-Based Systems in Second Life. Presence: Teleoperators and Virtual Environments, 18(6):468–477.

Puglia, S., Carter, R., und Jain, R. (2000). MultECommerce: A Distributed Architecture for Collaborative Shopping on the WWW. In Proceedings of the 2nd ACM Conference on Electronic Commerce, EC '00, pages 215–224, New York, NY, USA. ACM.

Qin, W., Suo, Y., und Shi, Y. (2006). Camps: A Middleware for Providing Context-Aware Services for Smart Space. In Advances in Grid and Pervasive Computing, pages 644–653. Springer.

Rander, P., Narayanan, P. J., und Kanade, T. (1997). Virtual*ized* Reality: Constructing Time-Varying Virtual Worlds From Real World Events. In Visualization '97., Proceedings, pages 277–283.

Raymer, R. (2011). Gamification: Using Game Mechanics to Enhance eLearning. eLearn, 2011(9).

Risden, K., Czerwinski, M. P., Munzer, T., und Cook, D. B. (2000). An Initial Examination of Ease of Use for 2D and 3D Information Visualizations of Web Content. International Journal of Human-Computer Studies, 53(5):695–714.

Russell, S. J. und Norvig, P. (1995). Artificial Intelligence: A Modern Approach, volume 74. Prentice-Hall, Inc.

Russomanno, D. J., Kothari, C. R., und Thomas, O. A. (2005). Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In IC-AI, pages 637–643.

Satyanarayanan, M. (2001). Pervasive Computing: Vision and Challenges. Personal Communications, IEEE, 8(4):10–17.

Schilit, B., Adams, N., und Want, R. (1994). Context-Aware Computing Applications. In Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on, pages 85–90. IEEE.

Schmitz, M., Baus, J., und Dörr, R. (2008). The Digital Sommelier: Interacting with Intelligent Products. In The Internet of Things, pages 247–262. Springer.

Seoane, F., Ferreira, J., Alvarez, L., Buendia, R., Ayllón, D., Llerena, C., und Gil-Pita, R. (2013). Sensorized garments and textrode-enabled measurement instrumentation for ambulatory assessment of the autonomic nervous system response in the atrec project. Sensors, 13(7):8997–9015.

Shah, M. A., Hellerstein, J. M., und Brewer, E. (2004). Highly Available, Fault-Tolerant, Parallel Dataflows. In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pages 827–838. ACM.

Shaw, C., Liang, J., Green, M., und Sun, Y. (1992). The Decoupled Simulation Model for Virtual Reality Systems. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '92, pages 321–328, New York, NY, USA. ACM.

Shreiner, D. und The Khronos OpenGL ARB Working Group (2009). OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1. Addison-Wesley Professional, 7th edition.

Smisek, J., Jancosek, M., und Pajdla, T. (2013). 3d with kinect. In Fossati, A., Gall, J., Grabner, H., Ren, X., und Konolige, K., editors, Consumer Depth Cameras for Computer Vision, Advances in Computer Vision and Pattern Recognition, pages 3–25. Springer London.

Solmaz, G., Akbaş, M. I., und Turgut, D. (2012). Modeling Visitor Movement in Theme Parks. In Proceedings of the 2012 IEEE 37th Conference on Local Computer Networks (LCN 2012), LCN '12, pages 36–43, Washington, DC, USA. IEEE Computer Society.

Sons, K., Klein, F., Rubinstein, D., Byelozyorov, S., und Slusallek, P. (2010). XML3D – Interactive 3D Graphics for the Web. In Proceedings of the 15th International Conference on Web 3D Technology, Web3D '10, pages 175–184. ACM.

Spassova, L. (2004). Fluid Beam - A Steerable Projector and Camera Unit. In 8th IEEE International Symposium on Wearable Computers (Proceedings of the Student Colloquium), pages 56–58.

Spassova, L. (2011). Interactive Ubiquitous Displays Based on Steerable Projection. Phd-thesis, Saarland University, Dept. of Computer Science, Saarbrücken.

Spassova, L., Schöning, J., Kahl, G., und Krüger, A. (2009). Innovative Retail Laboratory. In Roots for the Future of Ambient Intelligence (AmI'09), pages 18–21.

Stahl, C. (2009). Spatial Modeling of Activity and User Assistance in Instrumented Environments. Phd-thesis, Saarland University, Dept. of Computer Science, Saarbrücken.

Stahl, C., Frey, J., Alexandersson, J., und Brandherm, B. (2011). Synchronized Realities. Journal of Ambient Intelligence and Smart Environments, 3(1):13–25.

Stahl, C. und Haupert, J. (2006). Taking Location Modelling to New Levels: A Map Modelling Toolkit for Intelligent Environments. In Hazas, M., Krumm, J., und Strang, T., editors, Location- and Context-Awareness, volume 3987 of Lecture Notes in Computer Science, pages 74–85. Springer Berlin Heidelberg.

Tennenhouse, D. (2000). Proactive computing. Commun. ACM, 43(5):43–50.

Teymourian, K., Rohde, M., und Paschke, A. (2012). Knowledge-based Processing of Complex Stock Market Events. In Proceedings of the 15th International Conference on Extending Database Technology, pages 594–597. ACM.

The STREAM Group (2003). STREAM: The Stanford Stream Data Manager. Technical Report 2003-21, Stanford InfoLab.

Thiesse, F., Al-Kassab, J., und Fleisch, E. (2009). Understanding the value of integrated RFID systems: a case study from apparel retail. EJIS, 18(6):592–614.

Tnazefti-Kerkeni, I., Arantes, L., und Marin, O. (2002). A Multi-Blackboard Approach to the Control/Monitoring of APS. In WSEAS International Conference on Applied Mathematics and Computer Science (AMCOS'02), pages 1991–1996.

Ullrich, S., Brandherm, B., und Prendinger, H. (2008). Simulation framework with testbed for sensor-based systems in second life. In Proceedings of 10th International Conference on Ubiquitous Computing (UbiComp 2008) Demo Session, pages 468–477.

Verbeke, W., Farris, P., und Thurik, R. (1998). Consumer response to the preferred brand out-of-stock situation. European Journal of Marketing, 32(11/12):1008–1028.

Virrantaus, K., Markkula, J., Garmash, A., Terziyan, V. Y., Veijalainen, J., Katasonov, A., und Tirri, H. (2001). Developing GIS-Supported Location-Based Services. In International Conference on Web Information Systems Engineering, pages 66–75.

Wahlster, W. (2003). Towards Symmetric Multimodality: Fusion and Fission of Speech, Gesture, and Facial Expression. In in Proceedings of the 26th German Conference on Artificial Intelligence, September 2003, pages 1–18. Springer.

Wasinger, R. und Krüger, A. (2006). Modality Preferences in an Instrumented Environment. In Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI), pages 336–338, Sydney, Australia.

Wasinger, R., Krüger, A., und Jacobs, O. (2005). Integrating Intra and Extra Gestures Into a Mobile and Multimodal Shopping Assistant. In Pervasive Computing, pages 297–314. Springer.

Wasinger, R. und Wahlster, W. (2006). The anthropomorphized product shelf: Symmetric multimodal interaction with instrumented environments. In Aarts, E. und Encarnação, J. L., editors, Chapter in: True Visions: The Emergence of Ambient Intelligence, pages 291–306. Springer-Verlag.

Weiser, M. (1991). The Computer for the 21st Century. Scientific american, 265(3):94–104.

Weppner, J. und Lukowicz, P. (2013). Bluetooth Based Collaborative Crowd Density Estimation with Mobile Phones. In Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on, pages 193–200.

Westkämper, E. und Jendoubi, L. (2003). Smart Factories - Manufacturing Environments and Systems of the Future. In Proceedings of the 36th CIRP International Seminar on Manufacturing Systems, pages 13–16.

Wolf, M., Weimerskirch, A., und Paar, C. (2004). Security in Automotive Bus Systems. In Workshop on Embedded IT-Security in Cars, pages 11–12.

Wormus, T. (2008). Complex Event Processing. Business Intelligence Journal, 13(4):53–58.

Wyckoff, P., McLaughry, S. W., Lehman, T. J., und Ford, D. A. (1998). TSpaces. IBM System Journal, 37(3):454–474.

Youngblood, G. M., Heierman, E. O., Holder, L. B., und Cook, D. J. (2005). Automation Intelligence for the Smart Environment. In International Joint Conference On Artificial Intelligence, volume 19, pages 1513–1514. Citeseer.

Zhu, Y., Toth, Z., Wobus, R., Richardson, D., und Mylne, K. (2002). The Economic Value of Ensemble-based Weather Forecasts. Bulletin of the American Meteorological Society, 83(1):73–83.

Zimmermann, A., Lorenz, A., und Oppermann, R. (2007). An Operational Definition of Context. In Modeling and using context, pages 558–571. Springer.

Zimmermann, G. und Vanderheiden, G. (2007). The Universal Control Hub: An Open Platform for Remote User Interfaces in the Digital Home. In Proceedings of the 12th International Conference on Human-computer Interaction: Interaction Platforms and Techniques, HCI'07, pages 1040–1049, Berlin, Heidelberg. Springer-Verlag.

Zühlke, D. (2010). Smartfactory—towards a factory-of-things. Annual Reviews in Control, 34(1):129–138.