Bachelorarbeit

Ahidjo Ayeva

Verwendung von Agentensystemen zur Optimierung der Steuerung auf der Basis von JADE
(Am Beispiel des Gebäudemanagements)

Prof. Dr. Bernd Kahlbrandt
Prof. Dr. Michael Neitzke

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Ahidjo Ayeva

Verwendung von Agentensystemen zur Optimierung der Steuerung auf der Basis von JADE

(Am Beispiel des Gebäudemanagements)

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Techniche Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Bernd Kahlbrandt
Zweitgutachter : Prof. Dr. Michael Neitzke

Abgegeben am 10. April 2014

Ahidjo Ayeva

Thema der Arbeit

Verwendung von Agentensystemen zur Optimierung der Steuerung auf der Basis von JADE *(Am Beispiel des Gebäudemanagements)*

Stichworte

Agentensystem, CIM, Gebäudemanagement, JADE, SBLIM, verteilte Systeme, Reinforcement Learning, WBEM.

Kurzzusammenfassung

Diese Arbeit untersucht den Einsatz des Reinforcement Learning in einer nicht-deterministischen Umwelt. Mit dem Ziel, die Steuerung in einem intelligenten Gebäude zu optimieren, werden nach der Untersuchung der Umwelt Methoden und Algorithmen des Reinforcement Learning analysiert und ein Modell abgeleitet. Die Implementierung des Agentensystems wird auf dem **"Java Agent Development Framework"(JADE)** erfolgen.

Ahidjo Ayeva

Title of the paper

Use of agent systems in order to optimize the control on the basis of JADE *(Example of building management)*.

Keywords

Agent system, CIM, building management, JADE, SBLIM, distributed System, Reinforcement Learning, WBEM.

Abstract

This paper investigates the use of reinforcement learning in a non-deterministic environment. With the aim to optimize the control in an intelligent building, a model will be derived after the examination of the environment and according to the analysis of methods and algorithms of reinforcement learning. The implementation of the agent system is made on the **"Java Agent Development Framework"(JADE)**

Abkürzungsverzeichnis

ACHE Adaptive Control of Home Environment

ACL Agent Communication Language

CIM Common Information Model

CIMOM CIM Object Manager

DF Directory Facilitator

DMTF Distributed Management Task Force

DP Dynamic Programming

FIPA Foundation for Intelligent Physical Agents

IBA Internationale Bauausstellung

JADE Java Agent Development Framework

JSR Java Specification Request

MC Monte Carlo

MDP Markov Decision Processes

MOF Managed Object Format

RL Reinforcement Learning

SBLIM Standards Based Linux Instrumentation for Manageability

SFCB Small Footprint CIM Broker

TD Temporal Difference

WBEM Web-Based Enterprise Management

Abbildungsverzeichnis

1.1 Die ACHE-Architektur. . . . 2

2.1 Agent und seine Umwelt. (Wooldridge, 2002, 16). . . . 4
2.2 3-Phasen-Zyklus 5
2.3 Modell eines lernenden Agenten (Russell und Norvig, 2003, 53) 8
2.4 Agent-Umwelt-Interaktion (Wolter, 2008, 3). . . . 10
2.5 Die Aktor-Kritik-Architektur (Sutton und Barto, 1998). . . . 16

3.1 Einflussgrößen auf den Gebäudebetrieb (VDI-Gesellschaft, 2003) 18
3.2 Beurteilung von Produkten im Bereich des intelligenten Hauses (Angaben in %) (Heusinger, 2004, 165) 22
3.3 Überblick über episodische und kontinuierliche Aufgaben 26
3.4 Zustände 27
3.5 Belohnungsstrategie. . . . 32
3.6 Approximation der Anzahl der neuen Zustände pro Zeiteinheit. . . . 34
3.7 Beziehungen zwischen Hauptelementen der Architektur. (Bellifemine u. a., 2007, 33) 38
3.8 Steuerung nach (Wang, 2010, 113). . . . 40
3.9 Beispielszenario 42

4.1 Toplevel des Frameworks. . . . 45
4.2 Modell des Gebäudes. . . . 46
4.3 Modell des intelligenten Gebäudes; Vererbung der Eigenschaften der Klasse CIM_ManagedSystemElement. . . . 47
4.4 CIM-Schema. . . . 48
4.5 RMA-GUI; Beispiel einer Implementation eines Gebäudes auf der JADE-Plattform. . . . 49
4.6 Lernmodell innerhalb einer Zone 51

5.1 Flussdiagramm des Operator-Programms. . . . 56
5.2 Client-GUI; 59
5.3 Operator-GUI 60
5.4 Simulator-GUI 60

5.5 GUI des Lernelementes. 60

Tabellenverzeichnis

3.1 Liste der möglichen Situationen für ein einfaches Lichtsystem 27
3.2 Beschreibung der Situationen. 29
3.3 Belohnungsfunktion; mögliche rewards in Situationen $s_1 = (AUS, 0, 0)$. . . . 32
3.4 Belohnungsfunktion; mögliche rewards in Situationen $s_8 = (AN, 1, 1)$. 32
3.5 Situationen und Last eines einfaches Lichtsystem 33
3.6 Position des Aktuators und mögliche Aktionen. 40
3.7 Mögliche Situationen und Mengen der erlaubten Aktionen 41
3.8 Mögliche Situationen und Mengen der erlaubten Aktionen (Folgerung aus Tab. 3.7). 41
3.9 Beispiel zur Berechnung der Q-Werte; $\alpha = 0.3$ und $\gamma = 0.9$ 42
3.10 Unterteilung einer Zone nach Umwelteigenschaften 44

5.1 Situationen 61

Inhaltsverzeichnis

Abkürzungsverzeichnis V

Abbildungsverzeichnis VI

Tabellenverzeichnis VIII

1 Einführung 1
- 1.1 Motivation . . . 1
- 1.2 Ziel . . . 3
- 1.3 Aufbau der Arbeit . . . 3

2 Intelligente Agentensysteme 4
- 2.1 Agent und Umwelt . . . 4
  - 2.1.1 Einleitung . . . 4
  - 2.1.2 Eigenschaften der Umwelt . . . 5
  - 2.1.3 Agenteneigenschaften . . . 6
- 2.2 Agentenarten . . . 7
  - 2.2.1 Reflex-Agenten . . . 7
  - 2.2.2 Zielbasierte Agenten . . . 7
  - 2.2.3 Nutzwertbasierte Agenten . . . 7
  - 2.2.4 Lernende Agenten . . . 8
- 2.3 Reinforcement learning . . . 9
  - 2.3.1 Übersicht . . . 9
  - 2.3.2 Exploration vs. Exploitation . . . 10
  - 2.3.3 Markov-Entscheidungsprozesse . . . 11
  - 2.3.4 Bewertungsfunktionen . . . 12
  - 2.3.5 Optimale Bewertungsfunktionen . . . 13
  - 2.3.6 On- vs. Off-Policy . . . 13
  - 2.3.7 Temporal Difference Algorithmen . . . 14

3 Systemspezifikation 18
- 3.1 Analyse der realen Welt . . . 18

3.1.1 Das Gebäude . . . 18
3.1.2 Einfluss der Natur . . . 19
3.1.3 Nutzerverhalten . . . 19
3.1.4 Versorgungsgüter . . . 19
3.1.5 Technische Infrastrukturen und Anlagen . . . 20
3.2 Systemziele . . . 21
3.2.1 Ressourcenverbrauch . . . 21
3.2.2 Erhöhung der Lebensqualität . . . 22
3.3 Kostenermittlung . . . 22
3.3.1 Energiekosten . . . 23
3.3.2 Nutzerbeschwerden . . . 23
3.3.3 Gesamtkosten . . . 23
3.4 Optimierungsstrategien . . . 23
3.4.1 Minimieren der Leistung . . . 24
3.4.2 Antizipatives Verhalten . . . 24
3.4.3 Kooperation . . . 24
3.4.4 Nutzung des erworbenen Wissen . . . 24
3.5 Ermittlung der Lernparameter . . . 25
3.5.1 Situationen . . . 25
3.5.2 Zustandsübergänge . . . 27
3.5.3 Aktionen . . . 27
3.5.4 Verbotene Aktionen . . . 29
3.5.5 Belohnungsfunktion . . . 29
3.5.6 Bewertungsfunktionen . . . 32
3.6 Eigenschaften des Lernens . . . 33
3.6.1 Strategie . . . 33
3.6.2 Lernrate . . . 34
3.6.3 Auswahl der Bewertungsfunktion(en) . . . 34
3.7 Umgebung des Agentensystems . . . 35
3.7.1 Der Operator . . . 35
3.7.2 Modell des intelligenten Gebäudes . . . 35
3.7.3 Das Web-Based Enterprise Management (WBEM) . . . 36
3.8 Java Agent Development Framework (JADE) . . . 37
3.8.1 Eigenschaften . . . 37
3.8.2 Architektur . . . 38
3.8.3 Agentenkommunikation . . . 38
3.8.4 Agentenmanagement . . . 39
3.9 Beispielszenario eines Dimmers . . . 39
3.9.1 Aufgabenstellung . . . 40
3.9.2 Annahmen . . . 40

3.9.3 Einsatz des Reinforcement Learning (RL) . . . 41
3.10 Zusammenfassung . . . 44

4 Entwurf **45**
4.1 Systemkomponenten . . . 45
4.2 Modellierung der Umgebung . . . 46
4.2.1 Modell des intelligenten Gebäudes . . . 46
4.3 Lernmodell . . . 47
4.3.1 Der Operator . . . 48
4.3.2 Lernender Agent . . . 49

5 Implementation **52**
5.1 Umgebung . . . 52
5.1.1 Gebäudemodell . . . 52
5.1.2 Umsetzung in Java . . . 53
5.1.3 Der WBEM-Client . . . 54
5.2 Agentensystem auf der Basis von JADE . . . 54
5.2.1 Agentenkommunikation & Protokolle . . . 54
5.2.2 Agententask . . . 54
5.2.3 Operator . . . 55
5.3 Die Lernkomponente . . . 57
5.3.1 Abstrakte Situation / Situation . . . 57
5.3.2 Abstrakte Aktion / Aktion . . . 57
5.3.3 Aktion_Situation_Paare . . . 57
5.3.4 Das Problem . . . 58
5.4 Realisierung des Frameworks . . . 59
5.5 Simulation . . . 61
5.5.1 Vorhaben . . . 61
5.5.2 Simulator . . . 62
5.6 Entwicklungsumgebung . . . 63
5.6.1 JADE-Plattform . . . 63
5.6.2 CIM-Server . . . 63

6 Ergebnisse **64**
6.1 Auswertung der Modelle . . . 64
6.1.1 Gebäudemodell . . . 64
6.1.2 Lernmodell . . . 64
6.2 Einsatz von JADE im Gebäudemanagement . . . 65
6.3 Reinforcement Learning im Gebäudemanagement . . . 66

7 Zusammenfassung **67**

7.1 Fazit 67
7.2 Ausblick 68

Anhänge **70**
Literaturverzeichnis 70
Listing 73

Stichwortverzeichnis **74**

Versicherung über Selbstständigkeit **75**

1 Einführung

1.1 Motivation

Für eine stetige Verbesserung der *Lebensqualität* wird der Mensch immer häufiger bei im Alltag anfallenden Aufgaben durch moderne Technologien unterstützt. Bei Senioren und behinderten Menschen ist es zum Beispiel in erster Linie relevant, ihre Autonomie zu erhalten oder durch technische Hilfestellung wiederzuerlangen. Daneben gewinnt die Reduktion des Energieverbrauches aufgrund knapper Ressourcen und ökonomischer Aspekte und die flexible Auslegung einer Einrichtung während der Nutzungsdauer immer mehr an Bedeutung. Sicherheitsaspekte *(Brand, Wasserschaden oder Einbrüche)* im Gebäudemanagement sind auch für viele Nutzer und Bauherrn ein wichtiger Parameter bei der Planung einer Einrichtung.

Der Begriff *"intelligente Gebäude"* wird heutzutage sehr oft verwendet. Dabei werden unterschiedliche Bezeichnungen benutzt: *"smart House"*, *"Home-Automation"*, *"Gebäudeautomatisierung"*, ... (Heusinger, 2004, 12). Da die bestehende Technologie einen Einfluss auf die Beschreibung eines intelligenten Gebäudes hat, ist seine Bedeutung seit den Frühzeiten der Heimautomatisierung nicht immer dieselbe gewesen. In (Harper, 2003, 34-35) führt die Definition des intelligenten Gebäudes zu den fünf folgenden hierarchischen Klassen: *Gebäude mit intelligenten Objekten*, *Gebäude mit intelligenten und kommunizierenden Objekten*, *vernetzte Gebäude*, *lernende Gebäude* und *attentive Gebäude*. Diese Gebäudeklassen bieten für den Betrieb des intelligenten Hauses unterschiedliche Optimierungspotentiale. Eine Zusammenführung dieser Eigenschaften bei der Entwicklung eines intelligenten Gebäudes ist vom Stand der Technik her realisierbar und stellt aufgrund der zusätzlichen Optimierungsmöglichkeiten einen interessanten Aspekt dar. Dies bietet außerdem eine Flexibilität bei der Kostenplanung, da Bereiche im Gebäude, je nach Priorität der Funktion und / oder Intelligenzbedarf, durch eine bestimmte Klasse bzw. durch eine Mischung der Klassen beschrieben werden können.

Im Rahmen dieser Arbeit wird ein intelligentes Gebäude als ein vernetztes und attentives Gebäude mit intelligenten und kommunizierenden Objekten mit der Fähigkeit zu Lernen definiert. Bei der Internationalen Bauausstellung (Internationale Bauausstellung) 2013 in Hamburg wurde beispielsweise das Projekt *"Hybrid House"* vorgestellt *(iba, 2013)*. Diese Art von Gebäuden bietet eine Flexibilität im Betrieb, indem während ihres Lebenszyklus ihr Typ nach Wunsch verändert werden kann. Ein Arbeitsplatz könnte zum Beispiel im Tagesverlauf, für eine gewollte Dauer, konfliktfrei auch als Wohnraum genutzt werden.

Die Nutzungsart eines intelligenten Gebäudes sowie die Erwartungen können von Nutzer zu Nutzer stark variieren und in manchen Fällen auch zum Konflikt führen. Dies kann auf Eigenschaften wie zum Beispiel *Kultur*, *Lebensstil*, *Interesse* und *Tagesablauf*, die das Verhalten der Nutzer beeinflußen, zurückgeführt werden. Deshalb wird eine Lösung bevorzugt, die

unabhängig vom Individuum eine optimale Flexibilität im System gewährt. Dieser Ansatz sollte aus Kostengründen und für eine erfolgreiche Markteinführung eines Gebäudemanagementsystems berücksichtigt werden.

In seinem Artikel "*An intelligent environment must be adaptive*" (Mozer, 1999) stellt der Autor fest, dass die Notwendigkeit besteht, intelligente Gebäude mit einer selbstanpassenden Fähigkeit zu entwickeln. Laut dem Autor bleibt der Programmieraufwand bei der Fertigstellung von intelligenten Häusern mit festgeschriebenen Funktionalitäten für individuelle Haushalte kostenintensiv, selbst wenn die Preise der technischen Ausrüstungen über die Jahre fallen. In diesem Sinne stellt der Autor den Prototyp des Adaptive Control of Home Environment (ACHE) vor. ACHE ist ein Projekt, welches die Optimierung von intelligenten Umgebungen umsetzt. Abbildung 1.1 zeigt ein fortgeschrittenes Modell der ACHE-Architektur, das in (Mozer, 2005) beschrieben wurde.

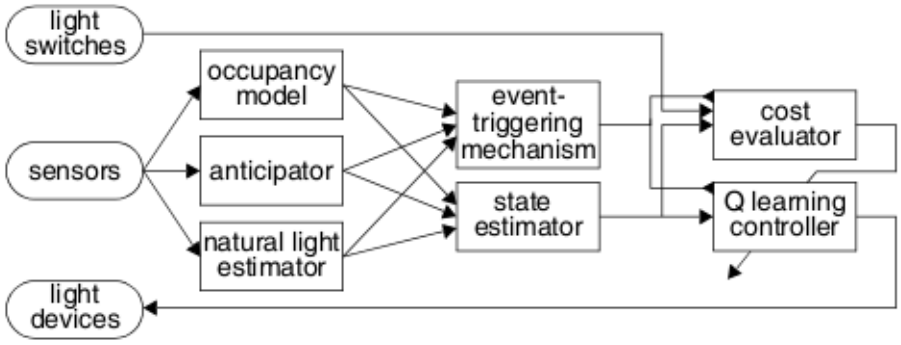

Abb. 1.1: Die ACHE-Architektur.

Agentensysteme bieten vielen technischen Systemen eine gewisse Autonomie. Sie unterscheiden sich jedoch nach Aufgabengebiet und eingesetzten Technologie. Eine Reihe von Agenten-Frameworks unterstützen die Entwickler bei der Implementation von Agentensystemen. Das von der Foundation for Intelligent Physical Agents (FIPA) entwickelte Agenten-Framework JADE stellt, aufgrund ihrer verteilten Architektur, einen interessanten Aspekt in der Gebäudeautomatisierung dar. Allerdings können physische *(vernetzte)* Objekte unabhängig von ihrer Lage angesprochen und die auf der Abstraktionsebene möglichen Modularisierungen eines Hauses mit Containern gekapselt werden. Die Art der Agenten in einem Agentensystem ist stark an die Art der Aufgaben gebunden. Für ein intelligentes Gebäude sind aufgrund des zuvor erwähnten Nutzerverhaltens Agenten mit adaptiven Verhalten von nennenswerter Bedeutung. Dies bedeutet den Einsatz von Agenten mit der Fähigkeit, Neues zu lernen. Auf dem Gebiet der künstlichen Intelligenz bietet das *maschinelle Lernen* eine Reihe von Frameworks, die das System mit Lernfähigkeiten aufrüsten können. In einer nicht-deterministischen Umwelt *(im Falle eines intelligenten Hauses)* bringt das *verstärkende Lernen (engl.: reinforcement*

learning) die Möglichkeit mit sich, ohne Vorkenntnisse zu Lernen und sich somit an neue Situationen anzupassen.

1.2 Ziel

Anhand eines Beispieles eines intelligenten Gebäudes soll geprüft werden, ob und wie der Einsatz von Agentensystemen zur Optimierung von Steuerungen beitragen kann. Dafür wird auf JADE ein Framework realisiert, welches das Lernen auf der Basis des Reinforcement Learning als Optimierungsstrategie einsetzt. Das Framework soll als Eingabe das Modell eines beliebigen Gebäudes übernehmen können. Es soll ein Gebäudemodell entworfen werden, das sich durch ein ebenfalls entworfenes Lernmodell steuern lässt. Beim Entwurf der Modelle werden Faktoren wie Komplexität, Ressourcenverbrauch, Aufwand bei Wartungen, Aufwand bei wachsender Nutzeranzahl bzw. bei Nutzerwechsel und Aufwand bei Änderung einer Komponente des verwalteten Gebäudes berücksichtigt. Am Ende dieser Arbeit werden die Ergebnisse analysiert und deren Auslegung bewertet.

1.3 Aufbau der Arbeit

Kapitel 2 bietet eine kurze Einführung in die Theorie der Agentensysteme und erläutert das Problem des Reinforcement Learning. Die Systemspezifikation erfolgt in Kapitel 3. Nachdem ein Lernmodell im Rahmen dieser Arbeit im 4. Kapitel abgeleitet wurde, wird in Kapitel 5 die Implementierung des Modells beschrieben. Um die Vorteile dieses Ansatzes zu ermitteln, werden in Kapitel 6 die Ergebnisse analysiert. Kapitel 7 bietet zum Schluss eine Zusammenfassung der Arbeit.

2 Intelligente Agentensysteme

Das Agentenkonzept wird in den letzten Jahren verstärkt u.a. im Fachgebiet der künstlichen Intelligenz untersucht. (Wooldridge, 2002) und (Gilbert, 2008) situieren den Durchbruch der agentbasierten Modellierung Mitte bzw. Anfang der 90er Jahre. Die Einführung von intelligenten Agentensystemen in vielen Ingenieurwissenschaften ermöglicht die Simulation bzw. die Beschleunigung von aufwändigen Prozessen und bietet ebenfalls einen Lösungsansatz für Systeme *bzw.* Teilsysteme, die mit einer gewissen Autonomie mit ihrer Umwelt interagieren. Dieser Abschnitt erläutert das Konzept der Agentensysteme und führt in das Problem des Reinforcement Learning ein.

2.1 Agent und Umwelt

2.1.1 Einleitung

Die Interaktionen zwischen einem Agenten und seiner Umwelt sind ein grundlegender Bestandteil des Agentenkonzeptes. Wie in Abb. 2.1 gezeigt führt der Agent eine Aktion durch und erhält über seine(n) Sensor(en) den Zustand der Umgebung. Auf der Basis der gewonnenen Informationen durch kontinuierliche Interaktion mit seiner Umgebung kann ein Agent durch autonom getroffene Entscheidungen die in seiner Umwelt durchzuführenden Aktionen festlegen.

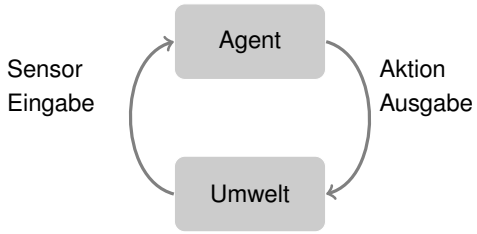

Abb. 2.1: Agent und seine Umwelt. (Wooldridge, 2002, 16).

Es existiert in der Literatur keine einheitliche Definition eines Agenten. Diese kann je nach Anforderungen und von Autor zu Autor leicht variieren.
Laut (Wooldridge, 2002, 15) ist ein Agent *ein Rechnersystem, das in einer Umwelt situiert ist und die Fähigkeit besitzt, autonome Aktionen in seiner Umwelt auszuführen, um sein Ziel zu erreichen*.
Es gibt jedoch eine allgemeine Abstraktion der Arbeitsweise eines Agenten in einem *3-Phasen-Zyklus* (siehe Abb. 2.2). In der ersten Phase nimmt der Agent über seine Sensoren Informationen über seine Umwelt auf. Diese Informationen werden in der 2. Phase je nach Art des Agenten verarbeitet und die dabei getroffene Entscheidung (die nächste Aktion) wird anschließend in der Phase 3 ausgeführt.

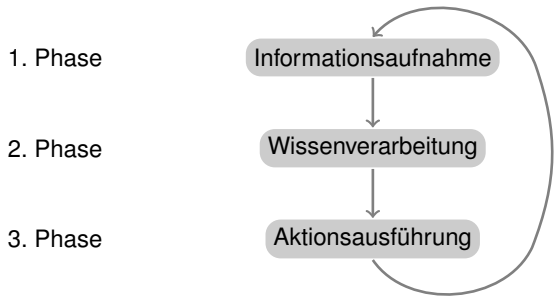

Abb. 2.2: 3-Phasen-Zyklus

2.1.2 Eigenschaften der Umwelt

Die Art der Umgebung stellt ein entscheidendes Kriterium für den Entwurf eines Agentensystems dar. Die Umgebung kann nach Eigenschaften klassifiziert werden und wird in (Russell und Norvig, 2003) wie folgt umgesetzt:

- **Vollständig / Teilweise beobachtbar**
  Erhält ein Agent über seine(n) Sensor(en) alle Informationen seiner Umwelt, ist sie damit *vollständig beobachtbar*.

- **Deterministisch / Stochastisch**
  Eine Umgebung ist deterministisch, falls ihr nächster Zustand von dem aktuellen Zustand und der ausgeführten Aktion vollständig bestimmt werden kann. Ist dies nicht der Fall, ist die Umgebung *stochastisch*.

- **Episodisch / Sequenziell**
  In einer *episodischen* Umgebung sind die Erfahrungen des Agenten in unabhängigen Episoden verteilt. Dabei setzt sich eine Episode aus einer Beobachtung der Umgebung und aus der Ausführung einer Aktion zusammen. Im Gegensatz zu einer *episodischen* Umgebung können aktuelle Entscheidungen in einer *sequenziellen* Umgebung alle künftigen Entscheidungen des Agenten beeinflußen. Dies schreibt dem Agenten, den Besitz der Fähigkeit vorauszudenken, zu.

- **Dynamisch / Statisch**
  Wenn sich der Zustand der Umgebung während eines Entscheidungsprozesses verändern kann, dann ist die Umgebung *dynamisch*. Sonst ist sie *statisch*. Es gibt Situationen, die trotz Veränderung der *rewards* konstant bleiben. Eine Umgebung, in der derartige Situationen eintreten könnte ist *halbdynamisch (engl.: semidynamic)* (Russell und Norvig, 2003, 42).

- **Kontinuierlich / Diskret**
  Kontinuierliche bzw. *diskrete* Eigenschaften können auf den Zustand der Umgebung und

auf die Empfindungen und Aktionen des Agenten angewendet werden. Eine Umgebung mit einer endlichen Anzahl von Zuständen und einer endlichen Menge von Beobachtung-Aktion-Paare ist *diskret*.

- **Einzel-** / **Multi-Agenten**
  Diese Entscheidung hängt von der Art der Aufgabe ab und ist das Ergebnis der Analyse des Agentenverhaltens gegenüber anderen Objekten, die vom Agenten als *Agent* bzw. *Objekte* mit stochastischen Verhalten behandelt werden.

Die Komplexität einer Umgebung kann auf der Basis dieser Eigenschaften ermittelt werden. Im *worst-case* ist die Umgebung *teilweise beobachtbar*, *stochastisch*, *sequenziell*, *dynamisch*, *kontinuierlich* und *multi-Agent*.

2.1.3 Agenteneigenschaften

In (Padgham und Winikoff, 2004) ist ein intelligenter Agent ein Teil eines Computerprogrammes, welches folgende Eigenschaften besitzt:

- **Situiert**
  Der Agent existiert in einer Umgebung, in der er tätig werden soll. Zu jeder Zeit kann sich der Agent anhand der Situation in einem bestimmten Teil der Umgebung befinden.

- **Autonomie**
  Diese Eigenschaft ermöglicht eine unabhängige Handlung des Agenten ohne externe Einflüsse.

- **Reaktivität**
  Reaktionen auf Veränderungen der Umgebung werden innerhalb eines angemessenen Zeitraumes getätigt.

- **Proaktivität**
  Bei der Verfolgung seiner Ziele sind Fehlentscheidungen nicht ausgeschlossen. Das Ziel eines Agenten ist dabei dauernd. Die *Proaktivität* ermöglicht eine durchgehende Verfolgung der Ziele trotz Fehlentscheidungen und unterstützt somit die *Robustheit*.

- **Flexiblität**
  Der Agent hat mehrere Wege, um seine Ziele zu erreichen und kann in bestimmten Situationen, wegen Optimierungsbedarf oder Fehlentscheidungen, den Pfad zum Ziel *flexibel* austauschen.

- **Robustheit**
  Diese Eigenschaft bietet dem Agenten die Möglichkeit, Fehlentscheidungen aufzudecken.

- **Sozialeigenschaft**
  Durch diese Eigenschaft bekommt der Agent die Fähigkeit, in seiner Umwelt zu handeln.

Intelligente Agenten unterscheiden sich von einfachen Agenten durch den Besitz der folgenden Eigenschaften: *Reaktivität*, *Proaktivität* und *Sozialeigenschaften* (Wooldridge, 2002, 23).

2.2 Agentenarten

2.2.1 Reflex-Agenten

Diese Art von Agenten sind in zwei folgenden Kategorien unterteilt:

- **Simpel Reflex-Agent**
  Der Agent trifft in diesem Fall seine Entscheidungen nur unter Berücksichtigung des aktuellen Zustands. Alle vergangenen Ereignisse haben darauf keinen Einfluss. Der einfache Reflex-Agent benutzt beim Eintritt einer Situation eine Sammlung von Regeln[1], um seine Entscheidung zu treffen.

- **Modellbasierter Reflex-Agent**
  Diese Art von Agenten eignen sich für teilweise beobachtbare Umgebungen. Der Agent speichert seine Beobachtungen und hat somit die Möglichkeit, in jeder Situation weitere Aspekte der Umwelt aufzudecken. Dafür hält der Agent einen internen Zustand, der im Laufe der Zeit anhand eines Modells der Umgebung akualisiert wird.

2.2.2 Zielbasierte Agenten

Agenten dieser Art verfolgen ein oder mehrere definierte Ziel(e). Zusätzlich zu den Informationen über den Zustand ermöglicht die Definition von Zielen dem Agenten, sich für gewünschte Aktionen in gegebenen Situationen zu entscheiden.

2.2.3 Nutzwertbasierte Agenten

Die Nutzwertfunktion (engl.: utility function) ist die Abbildung eines Zustands der Umwelt auf einen realen Wert. Zustände können somit anhand ihres Nutzwertes verglichen werden. Dies ermöglicht dem Agenten durch die Auswahl von Aktionen unter Berücksichtigung des Nutzwertes des nächsten Zustands, einen optimalen Pfad zum Ziel zu finden.

[1] Bedingung-Aktion- Regeln bzw. Situation-Aktion-Regeln (Russell und Norvig, 2003, 46).

2.2.4 Lernende Agenten

Ein lernender Agent kann mit den folgenden Komponenten zusammengefasst werden:

- **Lernelement**
  Diese Komponente ist für die Optimierungen zuständig. Während des Lernens bekommt das *Lernelement* kontinuierlich ein Feedback von der *Kritik* (vgl. Abb. 2.3). Diese Feedbacks enthalten Informationen über die Fortschritte des Agenten und werden im Lernelement bei der Ermittlung von möglichen Optimierungen des *Leistungselements* eingesetzt.

- **Leistungselement**
  Über das Leistungselement interagiert der Agent mit seiner Umwelt. Es ist für das Auswählen von Aktionen und die Wahrnehmung der Umwelt zuständig.

- **Kritik**
  In dieser Komponente werden die Handlungen des Agenten anhand eines Leistungsstandards bewertet.

- **Problemgenerator**
  Diese Komponente gewährt dem Agenten durch ihre explorative Eigenschaft ein dauerhaftes Lernverhalten. Unabhängig von der verfolgten Strategie schlägt sie dem Agenten Aktionen vor.

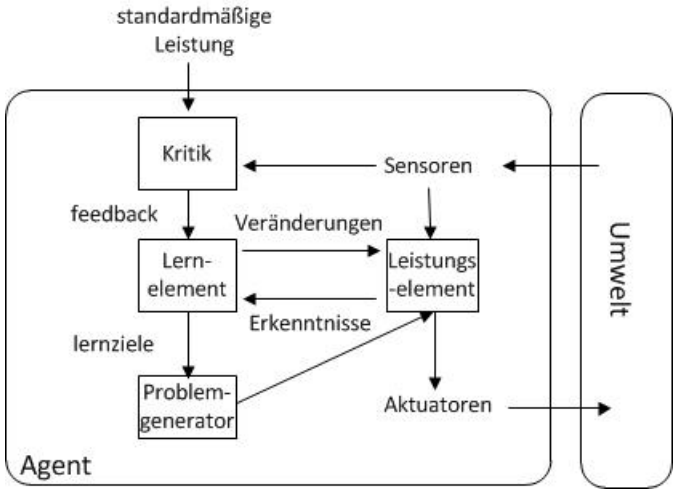

Abb. 2.3: Modell eines lernenden Agenten (Russell und Norvig, 2003, 53)

Lernmethoden in Agentensystemen werden anhand der Art der Rückmeldungen an den lernenden Agenten gruppiert. Dabei wird im Allgemeinen zwischen *überwachten* und *unüberwachten* Lernmethoden unterschieden.

Überwachtes Lernen

In diesem Fall bekommt der Agent (der Lernende) von einem *Lehrer* für alle Beispieleingaben die richtige Antwort. Auf der Basis dieses Wissens ist der Agent bei beliebigen Eingaben in der Lage, die richtige Entscheidung abzuleiten.

Unüberwachtes Lernen

Aus Beispieleingaben fasst der Lernende alle enthaltenen Regularitäten zusammen und lernt daraus ein Modell.

Verstärkendes Lernen (Reinforcement Learning)

Bei dieser Art des Lernens erhält der Agent keine Antwort über die Richtigkeit einer Entscheidung, sondern einen Hinweis über die Güte der Entscheidung. Die Aufgabe des Agenten ist hier das Maximieren der Güte seiner Entscheidungen.

Im Rahmen dieser Arbeit führt der nächste Abschnitt in das Problem des Reinforcement Learning (RL) ein.

2.3 Reinforcement learning

2.3.1 Übersicht

Das Problem des RL erweitert das Agentenkonzept um einen Belohnungsfaktor, wobei die vom Agenten getroffenen Entscheidungen mittels einem Feedback bewertet werden. Diese Rückmeldung ist in der Literatur mit dem Begriff ``*Belohnung´´ (engl.: rewards)* versehen. Durch das Maximieren der *Gesamtbelohnung (engl.: Return)* während seines Lebenszyklus lernt der Agent, wie er die in seiner Umwelt durchzuführenden Aktionen festlegt. Das eigentliche Ziel eines RL Agenten ist somit, seine Gesamtbelohnung zu maximieren. Wie in Abb. 2.3 gezeigt, erhält der Agent zu jeder Situation $s_t \in S$, die nach der Ausführung einer Aktion beobachtet wird, eine Belohnung r_t. Rewards teilen dem Agenten mit, wie gut die Auswahl einer Aktion a_t in einer Situation s_t gewesen ist. Dabei ist S die Menge aller möglichen Situationen, und a_t eine Aktion aus der Menge $A(s_t)$ der verfügbaren Aktionen in der Situation s_t.

Für eine episodische Aufgabe errechnen sich die *Returns* aus der Summe der zu jedem Schritt der Episode erhaltenen *rewards*:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T, \tag{2.1}$$

mit T einem Zeitstempel, der das Ende eines Episoden kennzeichnet.

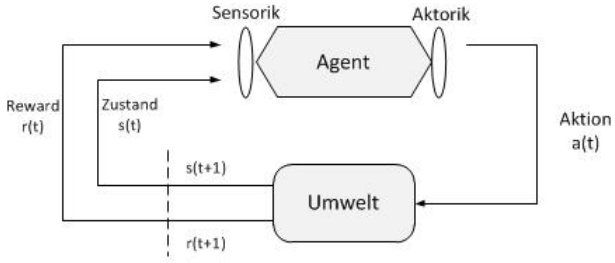

Abb. 2.4: Agent-Umwelt-Interaktion (Wolter, 2008, 3).

In einem kontinuierlichen Aktionsraum wächst T gegen unendlich ($T \rightsquigarrow \infty$). Die Einführung eines Diskontierungsparameters (γ) verhindert einen Überlauf der Returns und die Gesamtbelohnung R_t ist in diesem Fall:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.2}$$

mit γ ein Diskontierungsparameter $(0 \leq \gamma \leq 1)$.

2.3.2 Exploration vs. Exploitation

Eine Eigenschaft des RL Frameworks ist es, dass der Agent nur aus eigenen Erfahrungen lernt und daher keine Hilfe von Außen erhält. Das Lernen wird mit folgender Vorgehen realisiert: dem *Explorationsverfahren* und *dem Exploitationsverfahren*. Im ersten Verfahren lernt der Agent durch Ausprobieren von neuen Aktionen seine Umwelt *(bzw. die erlaubten Aktionen in seiner Umwelt)* kennen. Im zweiten Verfahrens nutzt der Agent das aus dem *Explorationsverfahren* erhaltene Wissen, um künftige Entscheidungen zu treffen. Das Verhältins zwischen *Exploration* und *Exploitation* (Exploitationsrate) definiert also die Wahrscheinlichkeit, mit der eine Aktion erfolgreich wird.

E-greedy

Ein Verfahren zur Ermittlung dieser Wahrscheinlichkeit ist unter ϵ-*greedy (epsilon-greedy)* bekannt, wobei $\epsilon \in [0,\ 1]$ einen Faktor definiert. Für $\epsilon > 0.5$ ist die Wahrscheinlichkeit, eine erfolgreiche Aktion auszuführen, größer als 50% und weist somit auf eine höhere Exploitationsrate hin.

Softmax Action Selection

Die Auswahl einer ϵ-*greedy* Aktion kann unter Umständen in einer RL Aufgabe ungünstig ausfallen. Die ϵ-*greedy* Methode gibt nur die Ausgangswahrscheinlichkeit einer Aktionswahl an. Dabei könnte allerdings die geschätzte Fehlerquote in Situationen ohne Fehlertoleranz zum

Problem führen. Um dies zu Lösen, werden *Softmax Action Selection Regeln* eingesetzt. Sie basieren auf einer graduierten Änderung der Aktionswahrscheinlichkeiten in die Abhängigkeit der geschätzten Bewertungen. Aktionen werden im $k - ten$ Schritt anhand der *Boltzmann-Distribution* mit der folgenden Wahrscheinlichkeit ausgewählt:

$$\frac{e^{Q_k(a)/\tau}}{\sum_{b=1}^{n} e^{Q_k(b)/\tau}}. \tag{2.3}$$

Der Parameter τ ist die *Boltzmann-Temperatur*. Bei hohen Temperaturen geschieht die Aktionswahl mit der gleichen Wahrscheinlichkeit. Für ein niedriges τ nähert sich das Verfahren einer ϵ-*greedy* Aktionswahl an und bewirkt wiederum das zuvor erläuterte Problem der ϵ-*greedy* Methoden.

Der folgende Abschnitt erläuert die mathematische Grundlagen des RL-Problems. Anschließend werden Algorithmen des verstärkenden Lernens kurz erläutert.

2.3.3 Markov-Entscheidungsprozesse

Ein Markov-Entscheidungsprozess (engl.: Markov Decision Processes (MDP)) ist ein Tupel $(S,\ A,\ P,\ \gamma,\ R)$, dass einen *stochastischen* Prozess beschreibt und indem einen Agent zu jeder Schrittzeit eine Aktion auswählt dafür eine Belohnung erhält. Dabei ist S eine *Menge von Zuständen*, A eine *Menge von Aktionen*, P die *Übergangswahrscheinlichkeit*, γ ein *Diskontierungsfaktor* und R die *Belohnungsfunktion*.

Ein Aktionsraum besitzt die Markov-Eigenschaften, falls bei jedem Zustandsübergang $P^a_{ss'}$ und $R^a_{ss'}$ bekannt sind (Sutton und Barto, 1998). Dies bedeutet, dass das Paar (s_t, r_t)[2] alle Informationen für die nächste Entscheidung bietet. $P^a_{ss'}$ ist dabei die Übergangswahrscheinlichkeit vom Zustand s zu Folgezustand s' (Gl. 2.4) und $R^a_{ss'}$ der prognostizierte *reward* nach der Auswahl einer Aktion a (Gl. 2.5).

$$P^a_{ss'} = Pr\left\{ s_{t+1} = s' \mid s_t = s,\ a_t = a \right\} \tag{2.4}$$

$$R^a_{ss'} = E\left\{ r_{t+1} \mid s_t = s,\ a_t = a,\ s_{t+1} = s' \right\} \tag{2.5}$$

RL-Probleme, in denen Zustände die Markov-Eigenschaften aufweisen, können somit wie MDP behandelt werden. Der Agent hat zur Folge die Fähigkeit, die nächste Belohnung anhand der aktuellen Situation und ausgewählten Aktion zu schätzen und diesbezüglich seine Gesamtbelohnung zu maximieren.

[2] aktueller Zustand und reward

2.3.4 Bewertungsfunktionen

In Abschnitt 2.3.1 wurde der Agent als ziel-basiert definiert, mit dem Ziel, seinen gesamten *reward* zu maximieren. Das Verfahren über die Vorhersage der *Returns* wird mit *Bewertungsfunktionen (engl.: Value functions)* realisiert.

Bewertungsfunktionen geben über einen Erwartungswert den Hinweis, wie gut es für den Agenten ist, sich in einer bestimmten Situation zu befinden. Da die Belohnung stark von der ausgewählten Aktion abhängt, erweist sich die *Strategie (engl.: policy)* zur Auswahl von Aktionen als wichtiger Parameter der Bewertungsfunktionen. Sie werden im Folgenden beschrieben und sind (Sutton und Barto, 1998) entnommen.

Die Strategie (π) ist als ein Abbild des Paares (s, a) auf der Wahrscheinlichkeit $\pi(s, a)$, dass eine Aktion a in einer Situation s ausgewählt wird, definiert. Die Bewertungsfunktion $V^\pi(s)$ einer Situation s, die *Zustand-Wert-Funktion (engl.: state-value function)* bei Verfolgung der Strategie (π) ist:

$$V^\pi(s) = E_\pi \left\{ R_t \mid s_t = s \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\}, \tag{2.6}$$

dabei ist E_π die Wahrscheinlichkeit, dass der Agent die Strategie (π) verfolgt und γ ein Diskontierungsparameter $(0 \leq \gamma \leq 1)$.

Weiterhin ist die Bewertungsfunktion der Aktion-Situation-Paare $Q^\pi(s, a)$ beim Nachlauf einer Strategie (π) ebenfalls definiert. Die *Aktion-Wert-Funktion (engl.: action-value function)* $Q^\pi(s, a)$ ist die erwartete Gesamtbelohnung, beginnend vom Zustand s mit der Aktion a und beim Verfolgen der Strategie (π), und wird mit folgender Gleichung beschrieben:

$$Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\} \tag{2.7}$$

Bei der Verfolgung der optimalen Strategie (π^*) wird eine optimale Bewertungsfunktionen erreicht. Die optimale Zustand-Wert-Funktion (V^*) und Aktion-Wert-Funktion (Q^*) sind wie folgt definiert:

$$V^*(s) = \max_\pi V^\pi(s), \forall\, s \in S \tag{2.8}$$

und

$$Q^*(s, a) = \max_\pi Q^\pi(s, a), \forall\, s \in S \text{ und } a \in A(s), \tag{2.9}$$

mit $A(s)$ die Menge der verfügbaren Aktionen in der Situation s.

Optimale Strategien führen zur Wahl von optimalen Aktionen. In (Wolter, 2008, 11) ist eine optimale Aktion bei Zustand-Wert-Funktionen diejenige, die die Summe aus der sofortigen

Belohnung und aus dem Zustand-Wert des Folgezustands s' maximiert und wird wie folgt beschrieben:

$$\pi^*(s) = arg\max_a[r + \gamma V^*(s')]. \tag{2.10}$$

Falls die Aktion-Wert-Funktion existiert, ist die optimale Aktion analog zu (2.10) definiert:

$$\pi^*(s) = arg\max_a Q^*(s, a). \tag{2.11}$$

Folgende Abschnitte erläutern einige Verfahren zur Optimierung der Bewertungsfunktionen.

2.3.5 Optimale Bewertungsfunktionen

Bewertungsfunktionen besitzen eine besondere *rekursive* Eigenschaft, die eine Konsistenz zwischen dem Wert einer Situation s und demjenigen ihres möglichen Nachfolgers sichert. Für eine beliebige Strategie (π) und eine Situation (s) kann V^π über eine Rekursion auf alle möglichen Folgesituationen ermittelt werden und ist durch die *Bellman*-Gleichung (2.12) gegeben.

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P^a_{ss'} \, [R^a_{ss'} + \gamma V^\pi(s')], \tag{2.12}$$

Die Bellman-Gleichung für die optimalen Bewertungsfunktionen spiegelt die Tatsache wider, dass der Wert eines Zustandes unter einer optimalen Strategie gleich der erwarteten Gesamtbelohnung bei der Ausführung der besten Aktion in der Situation ist. (Sutton und Barto, 1998, 76). Diese Gleichung ist für die Bewertungsfunktionen $V^*(s)$ und $Q^*(s, a)$ durch folgende Gleichungen beschrieben:

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'} \, [R^a_{ss'} + \gamma V^\pi(s')], \tag{2.13}$$

und

$$Q^*(s, a) = \sum_{s'} P^a_{ss'} \, [R^a_{ss'} + \gamma \max_a Q^*(s', a')]. \tag{2.14}$$

2.3.6 On- vs. Off-Policy

Durch das Gewährleisten der Eigenschaft, alle Aktionen unbegrenzt auswählen zu können, erhöht sich die Wahrscheinlichkeit einer kontinuierlichen Exploration. Dies wird mit den *On-* und *Off-Policy Methoden* realisiert. Auf der einen Seite versuchen On-Policy Methoden die Entscheidungspolitik in einer Funktion zu bewerten bzw. zu verbessern, während auf der anderen Seite Off-Policy Methoden die Verhaltens-Politik von der Bewertungspolitik unterscheiden, also im Gegensatz zu On-Policy Methoden die Optimierung beider Strategien in zwei separate Funktionen kapseln.

2.3.7 Temporal Difference Algorithmen

Das Temporal Difference (TD)-Lernen ist eine Art hybrides Verfahrens, das die Dynamische Programmierung *(engl.: Dynamic Programming)* und die *Monte Carlo Methoden* zusammenfasst.
Mit dem Begriff "Dynamische Programmierung" wird auf eine Kollektion von Algorithmen zum Berechnen von optimalen Strategien verwiesen:

- *(Iterative) Policy Evaluation*,
- *Policy Improvement*,
- *Policy Iteration* und
- *Value Iteration*.

Im Gegensatz zu der dynamischen Programmierung werden bei den Monte Carlo Methoden keine vollständigen Informationen über die Umwelt benötigt. Notwendig ist lediglich das Muster einer Abfolge von Zuständen, Aktionen und *rewards*. Das Verfahren basiert auf einer Mittelwertbildung der *returns*. Um die Existenz bzw. Verfügbarkeit der *Returns* zu gewährleisten, werden Monte Carlo Methoden nur bei episodischen Aufgaben eingesetzt.

- *Policy Evaluation*,
- *Action Values Estimation*,
- *On-Policy Monte Carlo Control* und
- *Off-Policy Monte Carlo Control*.

Aufbauend auf der dynamischen Programmierung nutzen TD-Methoden zusätzlich den Vorteil der Monte Carlo (MC) Methoden, keine Informationen über das *Modell* der Umgebung, der *Belohnung* und des *nächsten Zustands* zu benötigen.
Eine ausführliche Beschreibung der dynamischen Programmierung und der Monte Carlo (MC) Methoden kann in (Sutton und Barto, 1998) und (Boekhoven, 2011) nachgeschlagen werden.

Sarsa

Die Sarsa-Methode ist als ein *On-Policy TD Control* definiert. Die Methode erhält ihren Namen aus dem Quintupel *(**S**ituation, **A**ktion, **R**eward, **S**ituation, **A**ktion)*, das ihre Eigenschaft, Transitionen zwischen Aktion-Situation-Paaren zu berücksichtigen, ausdrückt. Zuerst wird mit der aktuellen Strategie und allen Situationen und Aktionen die Aktion-Wert-Funktion $Q^{\pi}(s, a)$ berechnet. Dann folgen Aktualisierung nach jeder Transition mit der folgenden Regel:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{2.15}$$

Für Endzustände ist $Q^{\pi}(s, a)$ als 0 definiert. In diesem Algorithmus (vgl. Lst. 2.1) hängt die Konvergenz der Aktion-Wert-Funktion $Q^{\pi}(s, a)$ von der Art der verfolgten Strategie ab.

```
Initialisiere Q(s,a) beliebig
Wiederhole für alle Episoden:
  Initialisiere s
  Wähle a von s mit einer von Q abgeleiteten Strategie
  Wiederhole für alle Schritte der Episode
    Nehme Aktion a, beobachte r, s'
    Wähle a' von s' mit einer von Q abgeleiteten Strategie
    Q(s,a) ← Q(s,a) + α [r + γQ(s',a') − Q(s,a)]
    s ← s';
    a ← a';
  bis s Endzustand ist.
```

Listing 2.1: Sarsa. (Sutton und Barto, 1998, 146)

Q-Learning

Die Q-Learning-Methode ist als ein *Off-Policy TD Control* definiert. Der eingesetzte Algorithmus (vgl. 2.2) führt eine direkte Approximation der optimalen Aktion-Wert-Funktionen $Q^{*}(s, a)$ bei der Berechnung der jeweiligen Aktion-Wert-Funktion $Q^{\pi}(s, a)$ durch. Die Konvergenz in diesem Fall wird durch eine kontinuierliche Aktualisierung der Aktion-Situation-Paare sichergestellt.

```
Initialisiere Q(s,a) beliebig
Wiederhole für alle Episoden:
  Initialisiere s
  Wähle a von s mit einer von Q abgeleiteten Strategie
  Wiederhole für alle Schritte der Episode
    Nehme Aktion a, beobachte r, s'
    Q(s,a) ← Q(s,a) + α [r + γ max_a' Q(s',a') − Q(s,a)]
    s ← s';
  bis s Endzustand ist.
```

Listing 2.2: Q-Learning. (Sutton und Barto, 1998, 149)

Aktor-Kritik-Methoden

Die *Aktor-Kritik-Methoden* definieren weitere On-Policy-TD-Methoden, die die Eigenschaft einer separaten Datenstruktur zur Darstellung der Strategie unabhängig von der Bewertungsfunktion besitzen. In der Architektur dieser Lernmethoden (vgl. Abb. 2.5) sind die Strategie mit *Actor* und die geschätzte Bewertungsfunktion mit *Critic* repräsentiert. Der Aktor trifft also Entscheidungen, die von der Kritik durch die Ermittlung der TD-Fehler δ_t auf Optimalität geprüft.

$$\delta_t = r_{t+1} + \gamma V(s_{t+1} - V(s_t)). \tag{2.16}$$

Die Gleichung 2.16 gibt den Rechenweg der TD-Fehler an. Bei positivem δ wird die Neigung zur ausgewählten Aktion in die Zukunft verstärkt, sonst wird sie geschwächt.

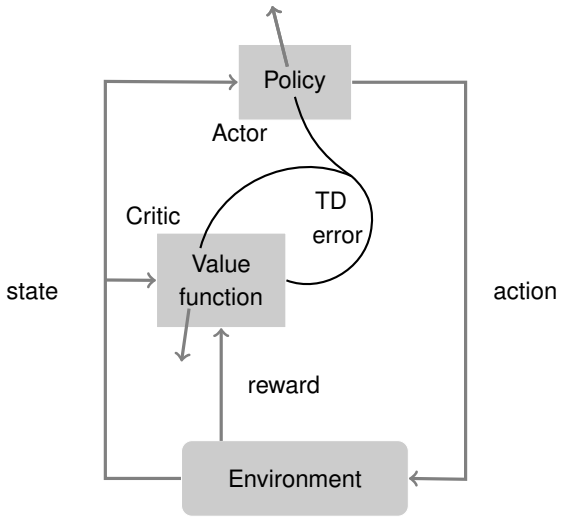

Abb. 2.5: Die Aktor-Kritik-Architektur (Sutton und Barto, 1998).

R-Learning

RL-Probleme in *kontinuierlichen* und nicht-diskontierten, *sequenziellen* Umgebungen werden mit der *R-Learning-Methode* angenähert (Sutton und Barto, 1998). Der Algorithmus ist ein *Off-Policy TD-Control* mit der Funktion, die maximale Belohnung pro Schrittzeit zu erhalten. Die Bewertungsfunktionen für eine Strategie π sind in diesem Fall relativ zum Durchschnittswert der erwarteten *rewards* ρ^π (Gl. 2.17) pro Schrittzeit definiert.

$$\rho^\pi = \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} E_\pi\{r_t\}, \tag{2.17}$$

In der Gleichung 2.18 bzw. 2.19 ist die *relative* Zustand-Wert-Funktion bzw. die *relative* Aktion-Wert-Funktion definiert. Diese Funktionen berechnen eine vorübergehende *Differenz* der *rewards*, wobei s einen Startzustand und a die ausgewählte Aktion beschreiben.

$$\tilde{V}^\pi(s) = \sum_{k=1}^{\infty} E_\pi\{r_{t+k} - \rho^\pi \mid s_t = s\}, \tag{2.18}$$

$$\tilde{Q}^\pi(s,a) = \sum_{k=1}^{\infty} E_\pi\{r_{t+k} - \rho^\pi \mid s_t = s, a_t = a\}, \tag{2.19}$$

Der *R-Learning-Algorithmus* setzt folgende Parameter ein: *eine Verhaltenspolitik*, *eine Bewertungspolitik*, *eine Aktion-Wert-Funktion* und *einen geschätzten Durchschnittswert der Belohnung*.

```
Initialisiere ρ und Q(s,a) für alle s, a, beliebig
Wiederhole unendlich:
  s ← aktuelle Situation
  Wähle a von s mit einer Strategie (z.B. ε − greedy)
    Nehme Aktion a, beobachte r, s'
    Q(s,a) ← Q(s,a) + α [r − ρ + max_a' Q(s',a') − Q(s,a)]
    Falls Q(s,a) = max_a Q(s,a), dann:
      ρ ← ρ + β [r − ρ + max_a' Q(s',a') − Q(s,a)]
```

Listing 2.3: R-Learning. (Sutton und Barto, 1998)

3 Systemspezifikation

In diesem Kapitel werden die Anforderungen an das Agentensystem definiert. Das System soll an der Verwaltung eines intelligenten Gebäudes teilnehmen. Zunächst hilft die Beschreibung der realen Umgebung dabei, klare Ziele für das System festzulegen.

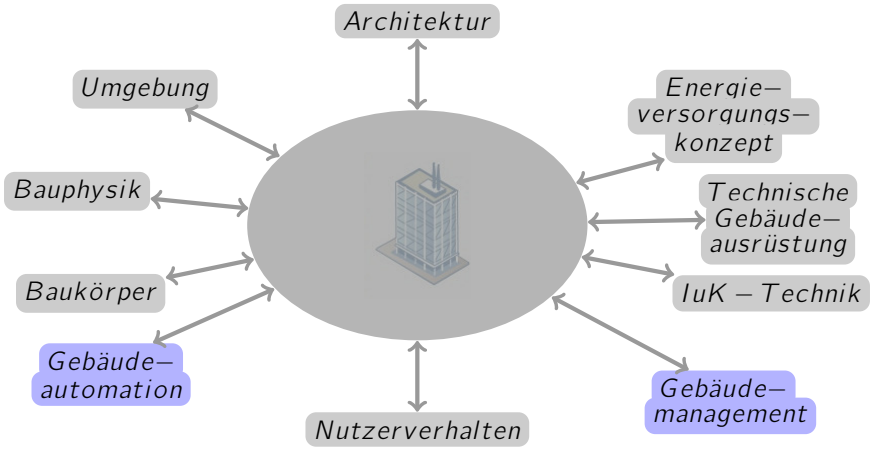

Abb. 3.1: Einflussgrößen auf den Gebäudebetrieb (VDI-Gesellschaft, 2003)

3.1 Analyse der realen Welt

Die reale Umwelt setzt sich, wie in Abb. 3.1 gezeigt, aus Gebäude und Einflussfaktoren zusammen. Für einen möglichen Einsatz des RL im Gebäudemanagement wird in diesem Abschnitt die Untersuchung eines intelligenten Gebäudes durchgeführt. In einem intelligenten Gebäude könnte das Management, aufgrund der vorhandenen Anlagen und Geräte, unterschiedliche Komplexität aufweisen. Je nach Gerätetyp, Anzahl oder seiner Funktion kann die Komplexität gering oder auch sehr groß sein. Im Rahmen dieser Arbeit ist zur Festellung der Komplexität einer Zone nur die Anzahl seiner Zustände relevant. Um die Komplexität des Problems zu reduzieren, wird es in diesem Abschnitt durch Analyse der realen Umgebung in mehrere Teilprobleme zerlegt.

3.1.1 Das Gebäude

In (Kretschmer, 2009) ist eine Zone *ein Teil- oder ein spezieller Nutzungsbereich eines Gebäudes. Dabei kann genau eine Zone einen Raum beschreiben oder es können auch mehrere Zonen in einem größeren Raum enthalten sein*. Weiterhin entspricht laut dem Autor die geografische Ausdehnung einer Zone dem Sensivitätsbereich eines Bewegungsmelders.

Basierend auf dieser Definition ist im Gebäudemanagement eine Zone durch den Raum bzw. die Fläche, die ein bzw. mehrere Sensor(en) erfassen kann, beschrieben. Die Komponenten des Gebäudes (Zimmer, Stockwerk, Flur) und das Gebäude selbst werden zur Folge als Zonen oder Hülle von Zonen betrachtet werden.

Innerhalb einer Zone können anhand der bestehenden technischen Infrastrukturen und Geräte verschiedene Umweltaspekte dargestellt werden. Eine Klassifikation der möglichen ausführbaren Aktionen in einer Zone erlaubt somit die Unterteilung nach verschiedenen Umwelteigenschaften *(siehe Tab. 3.10)*.

3.1.2 Einfluss der Natur

Die Steuerung eines intelligenten Gebäudes kann auf diverse Weise von der Natur beeinflusst werden. Die Natur stellt ein nicht beeinflussbares und seine Zustände ständig veränderndes System und somit einen Aktor dar. In diesem Teil der Umwelt ist die Ausführung von Aktionen nicht möglich.

Einerseits kann sich die Natur für die Erreichung der Systemziele als kontraproduktiv aufweisen, andererseits jedoch auch als Basis für innovative Konzepte im Gebäudemanagement. Bei niedrigen Temperaturen können beispielsweise die Heizkosten des Hauses steigen, während eine gute Ausnutzung des Tageslichtes zum Minimieren des Stromverbrauches beitragen könnte.

3.1.3 Nutzerverhalten

Ein weiterer nicht beeinflussbarer Teil der Umwelt ist das Nutzerverhalten. Der Unterschied zur Natur liegt darin, dass das Nutzerverhalten wichtige Informationen zum Bewerten der Funktionalität liefert. Die Zufriedenheit eines Nutzers wird meist durch sein Verhalten an das System mitgeteilt.

3.1.4 Versorgungsgüter

Neben den Baukosten eines intelligenten Gebäudes stellen die beim Gebäudebetrieb entstehenden Kosten einen wichtigen Faktor dar. Diesen Kosten werden z.B die Kosten der Versorgungsgüter zugerechnet.

Energien

Die Definition des intelligenten Gebäudes ist von der Art der Energiequelle unabhängig. Sie spielt heutzutage jedoch aufgrund umweltpolitischer und finanzieller Aspekte eine wichtige

Rolle bei der Planung des Gebäudes. Weiterhin verursacht der Energieverbrauch langfristige Kosten, die durch den Einsatz intelligenter Sparkonzepte gesenkt werden können. Je nach Art der Quelle und anhand der Verfügbarkeit können verschiedene Konzepte mit unterschiedlichen Einsparungspotentialen erzielt werden. Der Einsatz von Solaranlagen in Regionen mit einer hohen Rate an Sonnenstunden wirkt sich offensichtlich positiv auf die Stromkosten aus. Heutzutage wird beim Entwurf solcher Konzepte das Verhältnis zwischen *Preis*, *Leistung* und *Impakt auf die Umwelt* berücksichtigt. Ein optimales Sparkonzept könnte z.B. bei der Verfügbarkeit mehrerer Quellen und bei erläutertem Verhältnis bestimmen, welche Quelle zu welchem Zeitpunkt das Sparvorhaben maximiert.

Nahrungsmittel und Haushaltswaren

Auch Nahrungsmittel und weitere Haushaltswaren werden berücksichtigt. Das System könnte z.B. die Einkaufliste vorschlagen oder Vorräte planen und bestellen, Preise vergleichen und auf Mangel hinweisen. Es gibt eine Reihe von Haushaltsgeräten, deren Einsatz ein solches Management ermöglicht. Dabei könnte eine Einsparung des Haushaltsgeldes durch weniger Wegwerfen von Produkten erzielt und eine Zeitersparnis durch einen gut organisierten Einkaufsplan erreicht werden.

3.1.5 Technische Infrastrukturen und Anlagen

Die bestehende technische Infrastruktur ist ein wichtiger Parameter für eine optimale Wahrnehmung der realen Umwelt. Die Defintion eines intelligenten Gebäudes in Kapitel 1.1 betont ebenfalls die Relevanz der eingesetzten Technologie. Vorerst spielen die Infrastrukturen, die die Fähigkeit, interagieren zu können, anbieten bzw. ermöglichen, eine wichtige Rolle. Zu dieser Gruppe gehören z.B. *Sensoren*, das *Bussystem* und das *Kommunikationsnetzwerk*.

Sensoren

Die Sensivität der Sensoren beeinflusst die Definition ihres Wertbereichs. Daher sollte die Wahl der Sensoren das Verhältnis Empfindlichkeit und Problembereich[1] berücksichtigen. Allerdings können sehr empfindliche Sensoren einen großen Wertbereich aufweisen und somit aufgrund der resultierenden Anzahl der möglichen Zustände die Komplexität der Aufgabe erhöhen. Tabelle 3.10 listet einige Sensoren auf, die im Gebäudemanagement in Frage kommen.

Bussystem & Kommunikationsnetzwerk

Die Interaktionen zwischen den Geräten sowie ihre Koordination im Gebäudemanagement verursachen eine Vielzahl an Signalen und damit einen hohen Grad des Informationsflusses. Je nach eingesetztem Bussystem bzw. Netzwerkmodell kann die Herstellung der Kommunikation

[1] Der Problembereich wird hier durch den Wertbereich definiert, in dem die Sensorwerte liegen.

einen unterschiedlichen Aufwand in der Umsetzung erfordern. (Wenzel, 2011) erläutert dies bezüglich die bestehenden Abhängigkeiten. Weiteres kann dort nachgeschlagen werden.

Anlagen

Anlagen definieren je nach Typ unterschiedliche Teilprobleme innerhalb einer Zone und spielen bei der Definition des intelligenten Gebäudes (vgl. Abs. 1.1) eine wichtige Rolle. Auf der einen Seite bitten Geräte eine Grundlage für das alltägliche Leben in einem Gebäude, auf der anderen könnte ihre Komplexität das Problem der Steuerung erschweren. Die Komplexität der Geräten wächst mit der Anzahl ihrer Funktionen, besonders wenn diese eine Interaktion mit der Umgebung als Bedienform besitzen. Offensichtlich wurde sich die Anzahl der Versuche während einer gestuellen Interaktion mit einem Gerät durch das Auswerten ungenauer Eingangswerte vervielfachen. Es besteht daher für diesen Gerätetyp die Möglichkeit, die Eingangswerte zu verarbeiten um einen besseren Umgang mit dem Gerät zu gewinnen.

3.2 Systemziele

Bei der Beurteilung eines Produktes im Bereich des intelligenten Hauses (vgl. Abb. 3.2) setzt der Autor Merkmale, die zum Feststellen der Systemziele inspirieren bzw. dienen, ein. Diese Merkmale definieren bei der Verwaltung eines intelligenten Hauses und im Rahmen dieser Arbeit zusätzlich zum primären Ziel, die angedachte Funktionalität innerhalb des Systems, zwei weitere wesentliche Ziele, die verfolgt werden: die *Einsparung der Ressourcen* und die *Erhöhung der Lebensqualität*. Das Agentensystem soll die normale Funktionalität gewährleisten und durch Optimieren der Steuerung den Ressourcenverbrauch minimieren ohne den Wohlstand des Nutzers negativ zu beeinflussen.

3.2.1 Ressourcenverbrauch

Ressourcen in Bezug auf das Gebäudemanagement können in zwei Gruppen unterteilt werden: *materielle* und *immaterielle* Ressourcen.

Materielle Ressourcen

Diese Gruppe fasst alle Ressourcen mit einem in Kosten messbaren Verbrauch zusammen. Zu dieser Gruppe gehören z.B. alle Versorgungsgüter (Strom, Gas, Wasser, Internet, Nahrungsmittel, . . .).

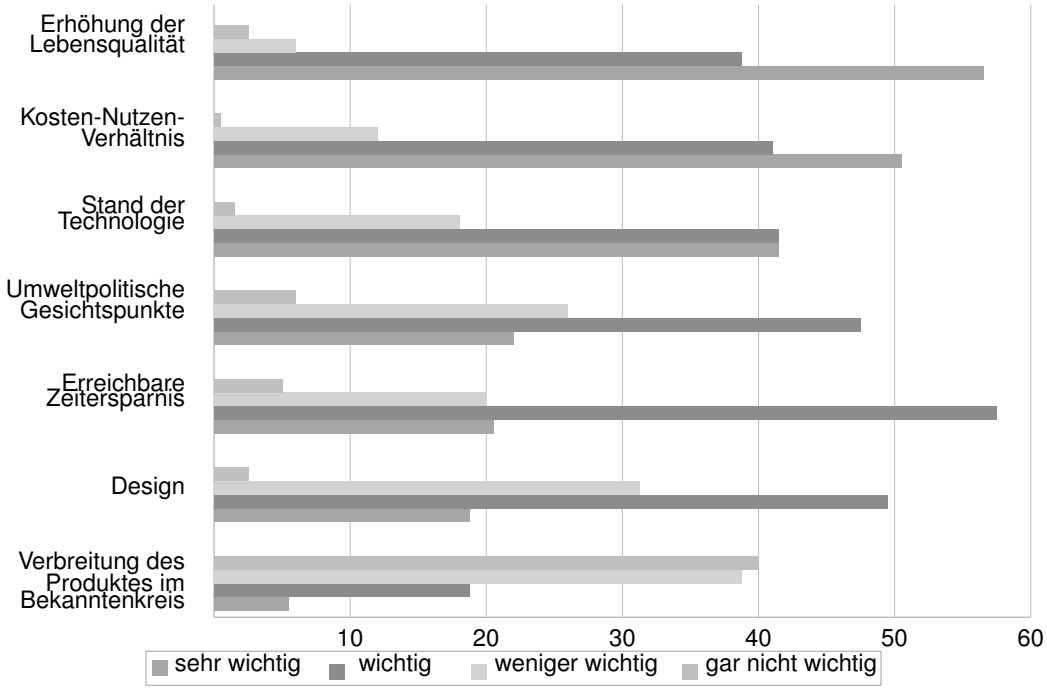

Abb. 3.2: Beurteilung von Produkten im Bereich des intelligenten Hauses (Angaben in %) (Heusinger, 2004, 165)

Immaterielle Ressourcen

In der zweiten Gruppe befinden sich Ressourcen, deren Messung relativ zum jeweiligen Nutzer bewertet werden. Die *Zeit* ist z.B. eine Ressource dieser Art. Die *Zeitersparnis* ist zwar direkt messbar, allerdings wird sie jedoch individuell vom Nutzer bewertet bzw. genutzt.

3.2.2 Erhöhung der Lebensqualität

Der Begriff Lebensqualität kann je nach Gesellschaft unter den gegebenen Umständen eine abweichende Bedeutung besitzen. Dies führt zu Eindeutigkeitsproblemen bei der Auswertung der Nutzerwünsche. Eine effiziente Bewertung des Systems sollte jedoch die Zufriendenheit der Nutzer berücksichtigen, die Einführung eines passenden Auswertungsmodelles erweist sich daher als nötig.

Das eingesetzte Auswertungsmodell im Rahmen dieser Arbeit wird bei der Ermittlung der Kosten im folgenden Abschnitt erläutert.

3.3 Kostenermittlung

Die Berechnung der Kosten stellt bei der Bewertung des Systems ein entscheidendes Kriterium dar. Während des Lernens spielen die Kosten eine wichtige Rolle, da sie als Basis für

die Belohnungspolitik dienen. Anhand der verfolgten Ziele setzen sich die Gesamtkosten aus der Summe der Verbrauchskosten und der aus den Unzufriedenheiten entstandenen Kosten zusammen.

3.3.1 Energiekosten

Die Ermittlung der Energieverbrauchskosten stellt im Grunde genommen kein Problem dar. Es könnte jedoch aufgrund der Aktualität der Messwerte zu einigen Unstimmigkeiten kommen. Für das System wäre es optimal, wenn bei Bedarf die aktuellsten Informationen vorliegen. Der Einsatz von *Smart-Meter* bringt durch flexible Messperioden einen Vorteil in dieser Richtung. Heutzutage sind Echtzeitmessungen mit *Smart-Meter* möglich und das Datum einer Messung wird nur noch bei nicht ausreichender technischer Infrastruktur relevant.

3.3.2 Nutzerbeschwerden

Im Fall einer Beleuchtung ist das Dämmern der Lichtquelle durch den Nutzer selbst offensichtlich unerwünscht, wenn es zu seinem Wohl vom System übernommen wird. Jegliche Interaktion des Nutzers für die genannte Aufgabe sollte auf einen Mangel hinweisen. In (Mozer, 2005) wurde für die Erfassung der Nutzerbeschwerden in ACHE jeder manuelle Eingriff des Nutzers in einer solchen Situation als Unannehmlichkeitskosten eingestuft und mit einer definierten Pauschale in Höhe von $\$\ 0,01$ angerechnet.

3.3.3 Gesamtkosten

Die Gesamtkosten ermöglichen es, die Nutzerfriedenheit mit dem System zu berücksichtigen. Die Gleichung 3.1 gibt die Formel zur Berechnung der Kosten an. Diese Formel wurde aus (Mozer, 2005) entnommen, sie wurde zur Berechnung der Kosten in ACHE eingesetzt.

$$J_0 \;=\; E\left[\lim_{k\to\infty}\ \frac{1}{k}\ \sum_{t=t_0+1}^{t_0+k} d(x_t)\ +\ e(u_t)\right] \tag{3.1}$$

Diese Kostenberechung fasst wie erwünscht die Verbrauchskosten (hier $e(u_t)$, die Energiekosten) und die Unannehmlichkeitskosten (hier $d(x_t)$, die von der Aktion x_t verursachten Unannehmlichkeitskosten) zusammen und könnte zur Festlegung der Belohnungspolitik eingeführt werden.

3.4 Optimierungsstrategien

Um das Hauptziel des Gebäudemanagementsystems zu erreichen, muss der Agent, nachdem er seine Umgebung gelernt hat, anhand bestimmter Strategien die Optimierungspotentiale erst

entdecken und sie dann bei der Aktionswahl berücksichtigen. Diese Strategien sind vom Entwickler zu definieren und den Eigenschaften der Umgebung hinzuzufügen.

3.4.1 Minimieren der Leistung

Die Senkung der Verbrauchskosten ist von bestehenden Sparpotentialen abhängig. Um diese ermitteln zu können, muss der Agent trotz einer gelernten optimalen Strategie ein Verhalten, das weiterhin die Auswahl von anderen Aktionen nicht ausschliesst, besitzen. Der Versuch, die Helligkeit bzw. die Temperatur in einer Zone zu senken bzw. zu erhöhen, könnte im Laufe der Zeit zur Erfassung der Sparpotentiale beitragen, stellt aber aufgrund eines möglichen negativen Einflusses auf die Nutzerfreundlichkeit ein Problem dar. Dies könnte das Maximieren der Gesamtbelohnung verlangsamen und somit den ganzen Lernprozess belasten. Eine Lösung wäre es, die Häufigkeit dieser Versuche anhand der Situationen auch durch einen Lernprozess ermitteln zu lassen.

3.4.2 Antizipatives Verhalten

Der Energieverbrauch könnte in manchen Fällen reduziert werden, wenn die Entscheidungen in optimaler Zeit getroffen worden sind. Die Startzeit eines Dämmerungsvorganges hat offensichtlich einen Einfluss auf die Dauer der Nutzung und zur Folge auch auf den Energieverbrauch. Ein Beispielszenario wäre der Fall eines Nutzers, der unter der Woche das Haus immer gegen 6 Uhr verlässt. Das Dämmern der Lichtquellen und der Heizungsanlagen könnte schon zu einer früheren Zeit, die vom Agenten zu ermitteln ist, beginnen.

3.4.3 Kooperation

Innerhalb des intelligenten Gebäudes können Aktivitäten in den verschiedenen Aufgabenfeldern sowohl *konkurrierend* als auch *kooperierend* anfallen. Offensichtlich steigern Situationen, in denen das Heizen und die Beleuchtung parallel ausgeführt werden, den Energieverbrauch, können aber auch durch eine Zusammenarbeit Einsparungen ermöglichen. Über eine gewisse Dauer trägt eine aktive Lampe zur Erwärmung der Zone bei, ein Fakt, der das Senken der Leistung des Heizsystems hervorruft. Je nach Anzahl und Dauer des Aufenthaltes können anwesende Nutzer auch zur Erwärmung des Ortes mitwirken.

3.4.4 Nutzung des erworbenen Wissen

Das bereits gewonnene Wissen könnte bei der Suche einer optimalen Strategie in verwandten Umgebungen eingesetzt werden. Der Agent sollte daher möglichst die Lerngegenstände *(Umwelt, Situation, Aktion)* unterscheiden bzw. vergleichen können.
Nach dem Finden einer optimalen Strategie in einem Aufgabenfeld (z.B. Beleuchtung) innerhalb einer Zone sollte der Agent für ein gleiches Aufgabenfeld in einer anderen Zone, falls

gewollt und die Bedingungen erfüllt sind, die Möglichkeit besitzen, das Wissen auf das neue Aufgabenfeld zu übertragen.
Einige Sicherheitskonzepte gegen Einbrüche bei Abwesenheit basieren auf dem Wiederholen eines Tagesablaufes bzw. einer Sequenz von Episoden. Der Agent sollte in diesem Fall die Möglichkeit haben, die erforderlichen Daten zu speichern.

Dieser Ansatz hängt hauptsächlich von der Art ab, wie der Agent sein Wissen verwaltet und das Wissen dargestellt ist. Er wird in Kapitel 5 bei der Implementierung des Agenten berücksichtigt.

3.5 Ermittlung der Lernparameter

In Abschnitt 2.3.3 wurde die Möglichkeit, das verstärkende Lernen als eine Form von MDP zu betrachten, erläutert. Fakt ist, dass im Rahmen des RL die Umgebung dem Agenten zum Beginn des Lernens unbekannt ist. Dies bedeutet, dass das Tupel $(S,\ A,\ P,\ \gamma,\ R)$, welches die Eigenschaften der Umgebung beschreibt, ebenfalls unbekannt ist. Dieser Abschnitt schlägt eine Beschreibung der Parameter des Lernens innerhalb einer Zone vor. Dies wird durch die Untersuchung zweier Aktionsräume mit unterschiedlichem Schwierigkeitsgrad realisiert.

3.5.1 Situationen

Situationen sind durch die Aufzeichnung der Veränderungen innerhalb einer Zone beschrieben. Um diese wahrzunehmen kommt je nach Funktionsbereich bzw. Aufgabenfeld eine unterschiedliche Kombination von Sensoren zum Einsatz. Für ein werden z.B. Situationen hauptsächlich mit dem Paar *(Helligkeit, Anwesenheit)* beschrieben und als Randbedingungen werden Messwerte von einem *Anwesenheits-* und *Helligkeitssensor* mit einbezogen.

Eigenschaften

Unter der Berücksichtigung der im Abschnitt 2.1.3 erläuterten Merkmale der Umgebung eines Agenten wurden für eine Zone folgende Eigenschaften festgestellt:

- Die von den eingesetzten Sensoren erhaltenen Informationen reichen dem Agenten für die Auswahl von Aktionen aus. Diese Umgebung ist daher *vollständig beobachtbar* unter der Annahme, dass die Sensoren eine akzeptable[2] Ausfall- bzw. Fehlerrate besitzen.
- Der Wertebereich der Messwerte ist endlich, und somit ist die Menge (S) aller Situationen in einer Zone ebenfalls endlich. Es handelt sich also in diesem Fall um eine *diskrete* Umgebung.

[2] Hängt vom Grad der Fault-Toleranz ab.

- Nutzerverhalten und die Natur stellen unberechenbare Faktoren dar, die während eines Entscheidungsprozesses den Zustand der Umgebung verändern können. Der nächste Zustand hängt somit nicht nur vom aktuellen Zustand und der ausgeführten Aktion ab. Dieser Teil der Welt ist zur Folge *stochastisch* und *dynamisch*.
- Die Aktivität in einem intelligenten Gebäude kann im Laufe der Zeit als eine Wiederholung von bestimmten Prozessen beschrieben werden. Diese Prozesse können je nach Art der Aufgabe *episodische* oder *sequentielle* Eigenschaften besitzen.

Da die Belohnungsfunktion diese letzte Eigenschaft berücksichtigt, wird an dieser Stelle ein kurzer Überblick auf die *episodische* bzw. die *sequentielle* Eigenschaft der Prozesse gegeben. Abbildung 3.3 schildert eine allgemeine Sicht auf die Aufgaben innerhalb einer Zone in Bezug auf ihre zeitliche Eigenschaft. Die Verwaltung einer Zone hat die Dauer des Lebenszyklus des Objektes und wird deshalb als eine *sequentielle* Aufgabe behandelt. Innerhalb der Zone selbst existieren Prozesse mit *episodischem* Zeitverhalten (vgl. Abb. 3.3).

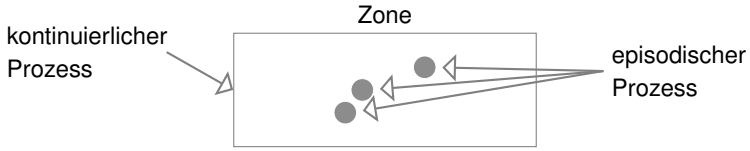

Abb. 3.3: Überblick über episodische und kontinuierliche Aufgaben

Das Management einer Zone wird zur Folge als eine *sequentielle* Ausführung von *episodischen* Aufgaben betrachtet.

Dimension

Die Dimension des Problems in einer Zone ist durch die Anzahl allen möglichen Situationen definiert. Sie hängt stark von der Anzahl der Sensoren, von deren Skalierung und von der Anzahl der Zustandsübergänge im Aktionsraum ab. Im Fall eines Beleuchtungssystems mit n Helligkeitsstufen und m Zustandstransitionen (m Dimmerstufen z.B.) ist die Dimension die Anzahl $|S|$ der Elemente von S:

$$|S| = n \times m \times k, \tag{3.2}$$

mit $k = 2$ die Kardinalität der boolschen Menge über der Anwesenheit eines Benutzers.

Für ein Lichtssystem mit zwei Zuständen (vgl. Abb. 3.4) wird der [3] im Laufe des Tages auf die Menge $\{hell, dunkel\}$ bzw. $\{0, 1\}$ abgebildet.

[3] Der Helligkeitswert bezieht sich auf den ursprünglichen Helligkeitswert einer Zone ohne die Zufuhr von künstlichen Lichtquellen.

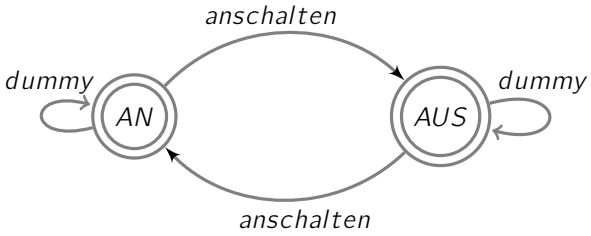

Abb. 3.4: Zustände

Die vom Anwesenheitssensor erfassten Werte werden ebenfalls auf der Menge $\{0, 1\}$ abgebildet. Zustandsübergänge können mit der Schalterposition (hier AN bzw. AUS) beschrieben werden. Tabelle 3.1 listet die Elemente von S im Beispielfall eines einfachen Lichtsystems. In

	Schalter	*Helligkeit*	*Anwesenheit*
s_1	*AUS*	0	0
s_2	*AUS*	0	1
s_3	*AUS*	1	0
s_4	*AUS*	1	1
s_5	*AN*	0	0
s_6	*AN*	0	1
s_7	*AN*	1	0
s_8	*AN*	1	1

Tab. 3.1: Liste der möglichen Situationen für ein einfaches Lichtsystem

einem Aktionsraum mit einer höheren Komplexität ist die Dimension deutlich größer. Das ist z.B. der Fall bei einem Lichtssystem mit Dimmerfunktion. Dieses Aufgabenfeld hat, wenn z.B. die Helligkeit auf zwölf Werte abgebildet ist und es zehn Dimmerstufen gibt, 240 Situationen.

3.5.2 Zustandsübergänge

Die Zustandsübergangsfunktion stellt aufgrund dieser nicht-deterministischen Eigenschaft der Umgebung ein stochastisches Problem dar. Die gleiche Aktion könnte in einer ähnlichen Situation, z.B. aufgrund externer Ereignisse, verschiedene Ausgänge haben. Die Wahrscheinlichkeit $P^a_{ss'}$, dass in einer Situation s die Wahl der Aktion a zur Situation s' führt, wird somit bei jedem Schritt neu ermittelt.

3.5.3 Aktionen

Eine Aktion im intelligenten Gebäudemanagement könnte sowohl die Steuerung eines Gerätes oder auch die Ausführung eines Dienstes bedeuten. Die Anzahl der Aktionen in einer

Zone hängt von der Art der Teilbereiche und von den verfolgten Zielen innerhalb der Zone ab. Diese kann grundsätzlich durch eine Multiplikation der Anzahl, der Aktuatoren und der Anzahl der Befehle pro Aktuator approximiert werden. Dazu können, falls definiert, weitere Aktionen mit einem Optimierungszweck gefügt werden (vgl. Gl. 3.4 und Gl. 3.5). Dies kann sich unter Umständen ungünstig erweisen, da die Dimension des Problemes sich für weitere Aktion vergrößert. Diese Aktionen sollten daher durch eine wirkliche Optimierungsabsicht eingeführt werden.

Die Ausführung einer Aktion kann je nach Art der Umgebung in beliebig vielen Schritten durchgeführt werden. Sie kann sich in manchen Fällen auf einen einfachen Ein-/Ausschalt-Vorgang begrenzen. In einem Lichtssystem aus zwei Zuständen (vgl. Abb. 3.4) z.B. führen Aktionen nur in einem Schritt zu einem Endzustand. In anderen Fällen soll eine Aktion das Einstellen eines Wertes unter vordefinierten Bedingungen bewirken. Diese Vorbedingungen können z.B. den Ausführungspfad einer Aktion über mehrere Transitionen erfordern. Bei Aufgaben mit Regelungsansprüchen z.B. (vgl. Abb. 3.8) hängt die Anzahl der Schritte von der aktuellen, der Zielposition des Aktuators und von der definierten Änderungsgeschwindigkeit ab. Die Änderungsgeschwindigkeit ist vom Hersteller vordefiniert und wird bei der Berechnung der Anzahl der Zwischenschritte berücksichtigt. Bei einem Dämmerungsvorgang wird z.B. auf die Nutzerfreundlichkeit geachtet. So sollte die Umstellung zwischen extremen Werten in wenigen Schritten vermieden werden.

Es werden zunächst die Aktionen eines einfachen Lichtssystems untersucht. In diesem Aktionsraum ist die ursprüngliche Menge (A) der Aktionen:

$$A = \{anschalten, ausschalten\} \tag{3.3}$$

Damit der Agent ständig die Möglichkeit hat, neue Informationen über seine Umgebung zu erhalten, benötigt er eine Aktion, die keine Änderung der Umwelt bewirkt *(eine leere Nachricht z.B.)*. Diese Art von Aktionen bietet dem Agenten eine *Strategie*, um Lernprozesse zu beginnen bzw. zu beenden. In der weiteren Verwendung wird diese Aktion mit dem Begriff *"dummy Aktion"* bezeichnet. Die Menge (A) wird deshalb um diese Aktion erweitert. Dabei ist (A):

$$A = \{anschalten, ausschalten, dummy\} \tag{3.4}$$

Bei der Beschreibung dieser Situationen (vgl. Tab. 3.2) weisen einige auf dem ersten Blick Anomalien auf. Sie scheinen, wenn z.B die Situationen s_5 und s_6 (vgl. Tabs. 3.1 bzw. 3.2) betrachtet werden, zuerst problematisch. Solche Situationen können jedoch auf technische Probleme und / oder auf Wartungsbedürfnisse (des Systems) hinweisen. Das ist z.B. der Fall bei einer beschädigten Lampe. Die Aktionsliste wird deshalb um eine weitere Aktion erweitert. Diese Aktion sollte den Wartungsdienst über die Notwendigkeit eines Einsatzes informieren.

Die Menge A wird nun wie folgt beschrieben:

$$A = \{anschalten, ausschalten, dummy, wartung\} \tag{3.5}$$

Die Untersuchung des einfachen Lichtsystems hat die Einführung von zwei weiteren Aktionen

S	Beschreibung
s_1	Anlage aus, Helligkeitswert null und keine Nutzer anwesend
s_2	Anlage aus, Helligkeitswert null und Nutzer anwesend
s_3	Anlage aus, es ist hell und keine Nutzer anwesend
s_4	Anlage aus, es ist hell und Nutzer anwesend
s_5	Anlage an, Helligkeitswert null und keine Nutzer anwesend
s_6	Anlage an, Helligkeitswert null und Nutzer anwesend
s_7	Anlage an, es ist hell und keine Nutzer anwesend
s_8	Anlage an, es ist hell und Nutzer anwesend

Tab. 3.2: Beschreibung der Situationen.

ergeben. Diese Aktionen bieten zusätzliche Optimierungsmöglichkeiten und können in jedem Aktionsraum eingesetzt werden. Dies impliziert die Definition von mindestens eines weiteren Zustands für die Wartungsfälle und die Erweiterung der Aktionsliste mit beiden Aktionen.

3.5.4 Verbotene Aktionen

In einer beliebigen Situation müssen nicht unbedingt alle Aktionen verfügbar sein. Wenn z.B. eine Situation vorliegt, in der ein Lichtsystem im Zustand ”AUS” ist, wird die Aktion “ausschalten” sinnlos und für manche Beleuchtungssysteme ein Risiko. Als Folge wird die Aktion in dieser Situation als *verboten* eingestuft. Weitere Aktionen können je nach Aufgabenfeld und anhand der Situation aus Sicherheitsgründen oder wegen der Nutzerfreundlichkeit z.B. ausgeschlossen werden. Zum Wohl des Nutzers könnte beispielsweise bei einem Dämmerungsvorgang (Heizung, Beleuchtung) die Anzahl der Schritte pro Aktion begrenzt werden. Alle nicht zu diesem Ansatz passenden Aktionen können als verbotene Aktionen definiert werden.

3.5.5 Belohnungsfunktion

Im Gebäudemanagement drücken die Belohnungsfunktionen die Zufriedenheit des Nutzers aus. Um seine Zufriedenheit zu erreichen, sind je nach Aufgabenfeld Ziele definiert, die die Belohnungspolitik festlegen. Die Definition der Belohnungsfunktionen *(engl. reward functions)* kann zu einigen Problemen führen, die als nächstes erläutert werden.

1. Problematik

Die Kosten einer Aktion, wie in Abschnitt 3.3 erläutert wurde, entstehen aus den Verbrauchskosten und den Nutzerbeschwerden bzw. Zustimmungen. Es wurde ebenfalls die Schwierigkeit bezüglich der Messbarkeit von Nutzerwünschen dargelegt. Unter dieser Umständen müssen zwangsläufig Kompromisse gefunden werden. Tabelle 3.5 spiegelt die Kompromisse wider, die für das Beispiel eines einfachen Lichtssystems gefunden wurden. Für die Situation $s_2 = (AUS,\ 0,\ 1)$ wurde kein Kompromiss gefunden, da sie noch nicht geklärt wurde. Diese Situation könnte die Nachtruhe beschreiben oder aber auch eine Fehlentscheidung des Systems. Ohne klare Regeln ist die Belohnung des Agenten in einer solchen Situation eine Herausforderung.

2. Problematik

Im Fall einer Kohabitation von mehreren Nutzern erschwert sich umso mehr das zuvor erläuterte Belohnungsproblem, da die Anzahl der Kompromisse mit derjenigen der Nutzer wächst. Die Nutzer können unter Umständen unterschiedliche, gegenseitig widersprüchliche Wünsche haben. Die Belohnungsfunktion sollte nichtsdestotrotz in der Lage sein, planmäßig zu belohnen.

3. Problematik

Weiterhin ist die Belohnungsfunktion in den meisten Fällen spezifisch für einen Funktionsbereich, es muss daher im schlimmsten Fall mit genau so vielen Belohnungsfunktionen wie der Anzahl der Aufgabenfelder gerechnet werden.

4. Problematik

Der Belohnungszeitpunkt spielt in manchen Aktionsräumen eine wichtige Rolle. Es muss überlegt werden, ob und wie Teilziele belohnt werden, da die Gefahr besteht, dass der Agent nie ans Ziel kommt, weil er schon belohnt wurde.

Beispiel einer Belohnungsfunktion

In einem einfachen Beleuchtungssystem könnte z.B. die Belohnungsfunktion über ein Abbild der Menge der *Aktion-Situation-Paare* auf die Mengen $R = \{0,\ 1\}$, $R' = \{-1,\ 1\}$ *bzw.* $R'' = \{-1,\ 0,\ 1\}$ realisiert werden. Anhand der Systemziele führt die folgende Analyse eine Überprüfung der vorgeschlagenen Mengen auf deren Brauchbarkeit durch.

- Mit R werden hauptsächlich ausgewählte Aktionen durch die Zuweisung von "0" bzw. "1" bewertet. Alle Folgezustände, die mit einem *reward* von "1" erreicht werden, können damit nicht unterschieden werden.

- Die Wahl von R' hat die gleiche Wirkung wie die von R. Der einzige Unterschied liegt in der Belohnung von Fehlentscheidungen. Das könnte sich auf die Gesamtbelohnung auswirken.
- Beim Einsatz von R'' können je nach Günstigkeit des aktuellen Zustandes unerwünschte Aktionen mit '"0" bzw. " -1" und gewünschte Aktionen mit "0" bzw. "1" bewertet werden.

Für die verfolgten Systemziele eignet sich am besten die Menge R''. Folgezustände, die das Einsparen des Energieverbrauches unterstützen, können dann von jenen, die durch eine zuvor optimale Entscheidung des Agenten entstanden sind aber den Verbauch erhöhen, unterschieden werden. Als Belohnungsfunktion in dem einfachen Beleuchtungssystem könnte somit die Funktion r (Gl. 3.6) eingesetzt werden.

$$r : S \times A(s) \rightarrow \{-1,\ 0,\ 1\}, \tag{3.6}$$

mit $A(s)$ die Menge der verfügbaren Aktionen in Situation $s \in S$.

Zustände können in jedem Aktionsraum, anhand der verfolgten Ziele, mit einer *Nutzwertfunktion* bewertet werden. D.h. für ein Heizsystem, wenn angenommen wird, dass günstige Situationen jene sind, die niedrige Temperaturen aufweisen, hat z.B. eine Situation mit $10°C$ einen höheren Nutzwert wie jede Situation mit einer höheren Temperatur. Diese Entscheidung wird jedoch immer unter Berücksichtigung der Nutzerfreundlichkeit getroffen. Bei niedrigen Temperaturen wird zwar weniger Energie verbraucht, diese Situation beeinflusst bei Anwesenheit eines Nutzers jedoch die Lebensqualität negativ.

Im Rahmen des Gebäudemanagements ist eine Situation günstiger als eine andere, wenn diese weniger Ressourcen verbraucht und die Lebensqualität des Nutzers gewährleistet ist. Das Verhältnis zwischen diesen beiden Zielen kann je nach Nutzer und Erwartungen unterschiedlich festgelegt werden. Bei Menschen mit Handikap z.B. stellt die Sicherung der Lebensqualität einen wichtigen Punkt dar und sollte deshalb vorrangig berücksichtigt werden.

Bei einem existierenden *Nutzwert* für alle Zustände eines Aktionsraumes und mit R'' als Menge der *rewards* könnte die Strategie der Belohnungserteilung wie folgt erläutert werden:

- Zustandsübergänge von günstigen zu weniger günstigen Situationen werden bei richtigen Entscheidungen am besten belohnt *(1 z.B.)* und bei Fehlentscheidungen am wenigsten bestraft *(0 z.B.)*.
- Transitionen von weniger günstigen zu *"guten"* Zuständen werden bei zutreffenden Entscheidungen am wenigsten belohnt *(mit 0 z.B)* und bei falscher Aktionswahl am höchsten bestraft *(-1 z.B.)*.

Diese Strategie spiegelt die Tatsache wider, dass Zustandsübergänge von günstigen Zuständen aus maximal belohnt werden. Der RL-Agent hat als Ziel das Maximieren seiner Gesamtbelohnung und kann während des Lernens die Situationen, die eine maximale Belohnung versprechen, herausfinden.

Abbildung 3.5 zeigt eine Simulation der Belohnungsfunktion. Zustände sind von links nach rechts anhand ihrer Günstigkeit mit den Markierungen auf der Achse (hier der Aktionsraum) dargestellt.

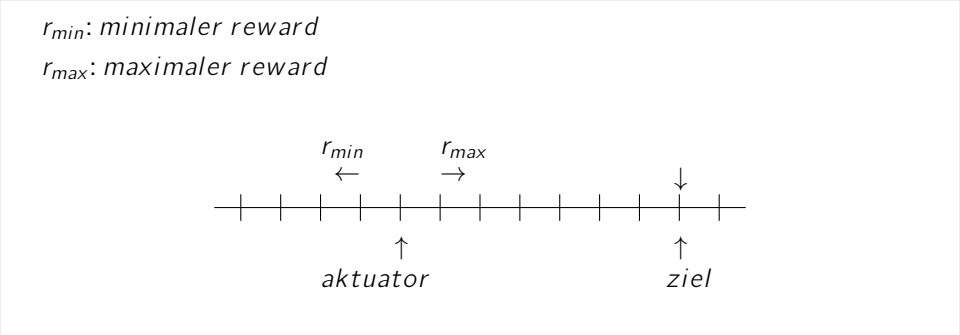

Abb. 3.5: Belohnungsstrategie.

Für ein einfaches Lichtsystem schlagen die Tabellen 3.4 und 3.3 für zwei verschiedene Situationen (vgl. Tab. 3.1) eine Belohnungsstrategie vor. Eine Änderung der Belohnungsfunktion

a	r
$anschalten$	-1
$dummy$	1
$wartung$	-1

Tab. 3.3: Belohnungsfunktion; mögliche rewards in Situationen $s_1 = (AUS, 0, 0)$.

a	r
$ausschalten$	-1
$dummy$	0
$wartung$	-1

Tab. 3.4: Belohnungsfunktion; mögliche rewards in Situationen $s_8 = (AN, 1, 1)$.

könnte neue Ziele definieren. Ein dynamischer Einsatz der Belohnungsfunktionen könnte z.B. dazu führen, dass energieverbrauchende Aktionen eher bei niedrigen Stromtarifzeiten durchgeführt werden.

3.5.6 Bewertungsfunktionen

Bewertungsfunktionen weisen den Agenten darauf hin, wie gut es für ihn ist, sich in einer bestimmten Situation zu befinden. Die verfolgten Systemziele schreiben dem Agenten jene Situationen vor, die für den Agenten günstig sind. Unter der Annahme, dass Situationen, die

einen minimalen Ressourcenverbrauch verursachen, am günstigsten sind, werden alle Situationen bewertet. In dieser Umgebung wird beispielsweise die Situation $s_1 = (AUS, 0, 0)$ als günstig betrachtet. Allerdings werden der Energieverbrauch und die Lebensqualität in Situation s_1 nicht belastet. Weiter bringt der Zustand s_9 aufgrund der möglichen verursachten Kosten keinen Vorteil. Auf der einen Seite könnten mehr Ressourcen in dieser Situation benötigt werden, auf der anderen werden die Entscheidungen des Agenten für die Dauer dieses Zustands keinen Einfluss auf die Umgebung haben. Tabelle 3.5 gibt für jede Situation den Grad *(hoch, mittel, niedrig)* der Last auf das System an. Durch eine optimale Belohnungsfunktion bzw. eine Belohnungsstrategie können Bewertungsfunktionen die Vorteile einer Situation widerspiegeln.

	Situation	*Last*
s_1	$(AUS, 0, 0)$	*niedrig*
s_2	$(AUS, 0, 1)$	–
s_3	$(AUS, 1, 0)$	*niedrig*
s_4	$(AUS, 1, 1)$	*niedrig*
s_5	$(AN, 0, 0)$	*hoch*
s_6	$(AN, 0, 1)$	*hoch*
s_7	$(AN, 1, 0)$	*hoch*
s_8	$(AN, 1, 1)$	*mittel*
s_9	*Wartung*	*hoch*

Tab. 3.5: Situationen und Last eines einfaches Lichtsystem

3.6 Eigenschaften des Lernens

3.6.1 Strategie

Zu Beginn der Anwendung gibt es noch keine Strategie *(engl. policy)* zur Aktionsauswahl. Die eigentliche Aufgabe des Agenten ist es, eine Strategie zu finden und diese zu optimieren. Je nach Problem könnte jedoch eine bestimmte Entscheidungspolitik, die die Aktionswahrscheinlichkeiten für die Anfangsphase definiert, angegeben werden. So wäre es aufgrund der Nutzerfreundlichkeit im Fall einer Beleuchtung sinnvoll, die Wahrscheinlichkeit der Aktionen, die den Bedarf an ausreichender Helligkeit decken, festzulegen. Dieser Ansatz könnte auch bei hohen Stromtarifzeiten eingesetzt werden, wobei die Anfangsstrategie anhand der Sparziele definiert wird. Die Aktionen werden ansonsten zum Beginn des Lernens im allgemeinen Fall (also bei Routineaufgaben) mit einer Zufallsstrategie ausgewählt.

3.6.2 Lernrate

Neben dem Erlernen einer optimalen Strategie zur Aktionsauswahl besteht eine weitere Aufgabe des RL darin, eine optimale Lernrate zu schätzen. Die Lernrate ist durch das Verhältnis Exploration/Exploitation beschrieben. Unter den vorgegebenen Anfangsbedingungen ist zum Beginn des Lernens eine höhere Lernrate wünschenswert. Diese Entscheidung beeinflusst jedoch die Gesamtbelohnung, wenn der Agent mit der Zeit eine größere Wissenbasis besitzt und muss deshalb im Laufe des Lernens angepasst werden.
In einer realen Anwendung hängt die Lernrate stark von der Frequenz der Ereignisse ab. Bei gleich bleibender Frequenz wächst die Anzahl der gelernten Zustände durch das Auftreten der eher wahrscheinlichen Situationen für eine bestimmte Zeit relativ schnell. Im Gebäudemanagement ist dies z.B. der Fall bei Gewohnheiten des Nutzers, da diese eine höhere Auftrittsrate besitzen. Dieser Zuwachs stagniert jedoch, wenn mit der Zeit die meisten Gewohnheiten aufgetreten sind. Falls die Informationen über seine Lernfortschritte vorliegen, hat der Agent anhand der Frequenz der Ereignisse und des Zuwachses seiner Wissenbasis pro Zeiteinheit die Möglichkeit, seine Lernrate verhältnismäßig zu schätzen. Abb. 3.6 zeigt eine Approximation des Zuwachses der Zustände pro Zeiteinheit und bei gleichbleibender Frequenz der Ereignisse.
Dieser Ansatz hängt wie erläutert vom Lernerfolg des Agenten ab und kann eingesetzt werden, wenn der Agent bereits für alle ihm bekannten Situationen die optimale Strategie berechnet hat.

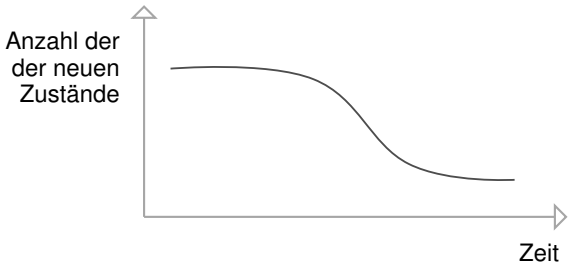

Abb. 3.6: Approximation der Anzahl der neuen Zustände pro Zeiteinheit.

3.6.3 Auswahl der Bewertungsfunktion(en)

Das Lernen beginnt in einer unbekannten Umgebung, die Q-Learning-Methode erweist sich für den Start des Lernprozesses als geeignet. Die Methode berechnet die Zustand-Aktion-Werte $Q(s,\ a)$ (vgl. Abschnitt 2.3.7) und ermöglicht den Beginn des Lernens von einer beliebigen Situation aus. Die Suche nach einer optimalen Strategie ist bei dieser Methode bereits während der Ermittlung der *Q-Werte* mit eingebunden.
Bei der Erreichung einer optimalen Strategie ist der Einsatz weiterer Methoden für eine Überprüfung möglich.

3.7 Umgebung des Agentensystems

Auf der Systemebene nimmt das Agentensystem seine Umwelt über Messwerte der Sensoren[4] wahr. Diese Werte werden anhand verschiedener Kriterien[5] an einen bzw. mehrere Hosts übermittelt. Daraus resultiert ein Netzwerk, dessen Architektur die Umwelt des Agentensystems charakterisiert. Die Umgebung des lernenden Agenten besteht aus drei wesentlichen Komponenten:

- ein Operator,
- ein Modell des intelligenten Gebäudes und
- die Schnittstelle(n) zur realen Welt.

Der Zugriff auf die reale Welt erfolgt im Rahmen dieser Arbeit über das WBEM. Es ist jedoch kein Bestandteil dieser Arbeit und wird deswegen in Abschnitt 3.7.3 nur kurz erläutert. In (Wenzel, 2011) erklärt der Autor die Grundlagen des WBEM und untersucht dabei seinen Einsatz im Energiemanagement. Die vorgeschlagene Lösung wird bei der Umsetzung auf den dort entnommenen Erkenntnissen basieren. Eine ausführliche Lektüre über das WBEM kann ebenfalls auf (Hobbs, 2004) nachgeschlagen werden.

3.7.1 Der Operator

Er ist die zentrale Einheit der Abstraktion der realen Welt. Der Operator leitet die Entscheidungen des RL-Agenten an die Schnittstelle(n) zur realen Welt weiter und bekommt über diese Informationen über den Zustand der Umgebung.

3.7.2 Modell des intelligenten Gebäudes

Die reale Welt wird hauptsächlich durch das Erkennen ihrer Veränderungen und beim Ausführen von Aktionen wahrgenommen. Das Modell des intelligenten Gebäudes wird daher zuerst mit dem Paar $(Aktionsliste, Sensorenliste)$ definiert. Für einen Einsatz des RL wird für jedes Aufgabenfeld das Modell um eine Belohnungsfunktion erweitert. Diese Belohnungsfunktion spezifiziert das im Aktionsraum zu erreichen Ziel. Zum Beginn des Lernprozesses gibt die Momentanaufnahme der Sensorenwerte den Startzustand.

Spätere Änderungen des Modells sind nicht ausgeschlossen. Diese Änderungen können alle Gegenstände des Modells betreffen. Dadurch könnten z.B beim Ändern der Belohnungsfunktion neue Ziele definiert werden (siehe Abs. 3.5.5). Die Nutzerfreundlichkeit könnte z.B. durch Erweitern der Aktionsliste zusätzliche Optimierungsmöglichkeiten gewinnen.

[4] Hier werden physische Sensoren, die im Abschnitt 3.1.5 aufgelistet wurden, betrachtet.

[5] Hier können mehrere Kriterien, die von ökonomischen und / oder Design-Aspekten abhängen, in Frage kommen.

3.7.3 Das Web-Based Enterprise Management (WBEM)

Das Web-Based Enterprise Management (WBEM) ist ein von der Distributed Management Task Force (DMTF) entwickeltes Werkzeug mit dem Zweck, Defizite im Bereich des Service- bzw. Device-Managements zu mindern. Die DMTF ist ein Gremium von mehreren Firmen und akademischen Institutionen, mit dem Ziel, die Entwicklung, die Einführung sowie die Interoperabilität von Standards im Bereich des Managements zu veranlassen (Ref. (dmtf, 2014)). Das WBEM besteht aus folgenden standardisierten Technologien: dem *Common Information Model (CIM)*, einem *WBEM-Server* und *WBEM-Schnittstelle(n)*, einer Spezifikation für die Kodierung der Informationen (xmlCIM) und einem HTTP-Zugriff für die Weiterleitung von Anfragen/Antworten über das Netzwerk.

Das Common Information Model (CIM)

In Verbindung mit dem WBEM bietet das WBEM/CIM einen hohen Grad der Abstraktion bei der Verwaltung von Diensten bzw. Geräten. Das CIM ist ebenfalls eine Entwicklung der DMTF, welches bei der Modellierung der zu verwaltenden Objekte für die Implementation einer WBEM-Anwendung eingesetzt wird. Es setzt sich aus einer Anzahl von Standard-Modellen (Schemas) für Systeme, Anwendungen, Netzwerke, Geräte und für weitere Komponenten zusammen. Dabei werden Modelle im Managed Object Format (MOF) erfasst. Das MOF wurde von der DMTF entwickelt und ist im Dokument DSP0004 (Ref. (dmtf, 2014)) spezifiziert.

Als Sprache zur Objektrepräsentation bietet die Syntax des MOF eine detaillierte Wiedergabe der Eigenschaften eines Objektes. Daneben können Objekte und ihre Eigenschaften textuell beschreiben werden und Programmierer unabhängig von der Plattform Programme kommunizieren lassen und somit eine der Anforderungen an verteilte Anwendungen erfüllen. Abschnitt 4.2 schlägt für einen Einsatz des WBEM im Gebäudemanagement und unter Verwendung des CIM-Schemas eine Modellierung des intelligenten Gebäudes und dessen Komponenten vor. Die Repräsentation der Modelle in MOF (Abs. 5.1) vermittelt anhand konkreter Beispiele eine Idee des Einsatzes der Sprache. Mehr über die Syntax kann in (Hobbs, 2004) bzw. auf (dmtf, 2014) nachgeschlagen werden.

WBEM-Server

Der WBEM-Server ist der Kern einer WBEM-Anwendung. Der Server spielt die Rolle eines Vermittlers zwischen den Operatoren, Listeners und den Providern. Er basiert auf dem CIM Object Manager (CIMOM), der anhand des im Speicher *(engl.: repository)* gehaltenen Modells eine direkte Interaktion zwischen Clienten, Listenern und Providern ermöglicht. Der CIMOM stellt die Schnittstellen zu folgenden Komponenten bereit:

- *Repository*: in dieser Komponente werden die Modelle der verwalteten Objekte gespeichert. Dies erfolgt durch das Hochladen der MOF-Datei der Modelle in das *repository*.

- *Provider*: die Komponente übernimmt die Rolle eines Treibers zwischen dem abstrakten Modell der Objekte und die realen Welt.
- *Client*: der WBEM-Client kann nach der Anforderung eines Operators und durch Interaktionen mit dem CIMOM den Status des Managed-Objektes ändern bzw. abfragen.
- *Listener*: die Komponente überwacht im Auftrag eines Clienten die Ereignisse des verwalteten Gerätes und leitet die entsprechende Information weiter. Die Nutzung dieses Dienstes setzt eine Anmeldung beim WBEM-Server voraus.
- *Security plug-in*: für die Authentifizierung von Nutzer bzw. Operatoren.

3.8 Java Agent Development Framework (JADE)

Nachdem die Umwelt analysiert wurde und die durch den Einsatz des RL im Gebäudemanagement bestehenden Fakten abgeleitet wurden, wird nun das Agentensystem beschrieben. Im Rahmen dieser Arbeit wurde für die Entwicklung des Agentensystems das Java Agent Development Framework (JADE) ausgewählt. Dieser Abschnitt leitet die Eigenschaften des Agentensystems ab und stellt kurz die Agenten-Plattform JADE vor. Dabei wird vorerst die Architektur des Frameworks erläutert. Anwendungsspezifische Aspekte dieses Agenten-Frameworks werden anhand kronkreter Beispiele in Kapitel 4 bzw. 5 veranschaulicht.

3.8.1 Eigenschaften

Das Java Agent Development Framework ist eine Middleware, das Werkzeuge zur Entwicklung von Agentensystemen auf der Java-Sprache bietet und wurde anhand der Foundation for Intelligent Physical Agents (FIPA)-Spezifikationen entwickelt. Die FIPA ist ein Verein von Entwicklern mit dem Ziel, Standards und Spezifikationen im Gebiet der Agententechnologie zu entwickeln. Die Schwerpunkte dieser Arbeitsgruppe sind: die *Agentenkommunikation*, die *Agentenarchitektur* und das *Agentenmanagement*. (Bellifemine u. a., 2007, 10) erläutert die Entwicklung der FIPA seit ihrer Entstehung 1996. Mehr über die FIPA kann auf (fipa, 2014) nachgeschlagen werden. Das Framework unterstützt Folgendes:

- die Entwicklung eines verteilten Systems von Agenten,
- eine leistungsfähige Sendung von asynchronen Nachrichten,
- die Mobilität der Agenten,
- einfache Verwaltung der Agenten,
- die Anwendung von Ontologien und *Content*-Sprachen.

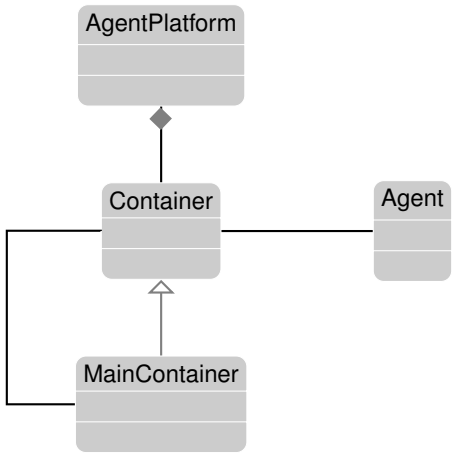

Abb. 3.7: Beziehungen zwischen Hauptelementen der Architektur. (Bellifemine u. a., 2007, 33)

3.8.2 Architektur

Die JADE-Plattform setzt sich aus Java-Prozessen, die ihre Laufzeit und alle nötigen Dienste zum Hosten der Agenten bereitstellen, zusammen. Diese Prozesse sind auf der Plattform mit dem Namen *Container* versehen. Dabei besitzt ein Container auf der selben Plattform spezielle Eigenschaften: der *Main-Container*. Dieser Prozess spielt als Startpunkt des Ladeprogramms der Plattform eine wichtige Rolle. Die Einführung von weiteren Containern auf der selben Plattform erfolgt über eine Registrierung beim *Main-Container*. Der Container wird daher als erstes gestartet und, wenn nicht spezifiziert, mit dem Default-Namen "*Main-Container*" auf der Plattform identifiziert. Abbildung 3.7 zeigt ein UML-Diagramm der Beziehungen zwischen den Hauptelementen der Architektur.

3.8.3 Agentenkommunikation

Kommunikation bietet den Agenten eine wichtige Fähigkeit, die auf der JADE-Plattform ebenfalls bereitgestellt wird. Sie basiert auf einem *asynchronen Message Passing*. Agenten besitzen eine Art Mailbox, um ihre Nachrichten zu empfangen. Sie werden beim Erhalt von Nachrichten avisiert. Dabei ist der Bearbeitungszeitpunkt dem Entwickler überlassen. Das Format der Nachrichten basiert auf dem Agent Communication Language (ACL) Format, eine Definition der FIPA. Das ACL setzt eine Struktur, die dem Agenten bzw. dem Programmierer die nötige Informationen zum Verarbeiten der Nachrichten bereitstellt, ein. Die Struktur enthält folgende Deklarationen:

- der Sender,
- die Liste der Empfänger,
- die *Performativ*,

- der Inhalt,
- die Sprache des Inhalts,
- die Ontologie und
- die Konversation-ID.

Die *Performativ* definiert einen Parameter, der die Absicht des Senders angibt.

3.8.4 Agentenmanagement

Weiterhin bietet JADE für das Management auf der Plattform ein Modell, das die Erzeugung, Registrierung, Lage, Kommunikation, Migration und Handhabung der Agenten einführt. Das Modell besteht aus folgenden Komponenten:

- *Agent Platform (AP)*: die Plattform stellt die physische Infrastruktur bereit, in der die Agenten sich aufhalten. Sie besteht aus Hosts, Betriebssystemen, Agenten und zusätzlichen Support-Anwendungen. Die interne Ausgestaltung der Plattform ist dem Entwickler frei überlassen.
- *Agent*: der Agent hält sich auf der Plattform auf und bietet seine Dienste an. Die Implementierung eines Dienstes hängt von der Aufgabe ab und ist daher dem Entwickler überlassen.
- *Directory Facilitator (DF)*: die Komponente bietet den Agenten einen *yellow page service*, in dem sie nach Diensten suchen und den eigenen veröffentlichen können.
- *Agent Management System (AMS)*: diese Komponente ist von der Agent-Platform mandatiert und übernimmt Aufgaben wie die Erzeugung bzw. Beendigung der Agenten und deren Migration zwischen Plattformen.
- *Message Transport Service (MTS)*: der *MTS* ermöglicht einen plattform-unabhängigen Austausch von FIPA-ACL Nachrichten zwischen den Agenten.

Eine ausführliche Lektüre kann auf (Bellifemine u. a., 2007) nachgeschlagen werden.

3.9 Beispielszenario eines Dimmers

Ein zuverlässiges unter Berücksichtigung des Tageslichtes *(siehe Abb. 3.8)* dient in diesem Szenario als Basis für die Beleuchtung eines Hauses. Beginnend mit dem Ausprobieren von Aktionen soll das Agentensystem bei der Suche einer optimalen Strategie lernen, wie die Systemziele erreicht werden.

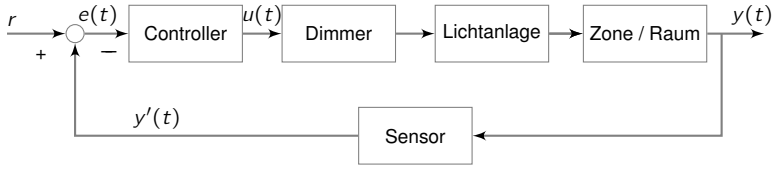

Abb. 3.8: Steuerung nach (Wang, 2010, 113).

3.9.1 Aufgabenstellung

Die Umgebung ist dem Agenten am Anfang unbekannt. Die Zustandsübertragungsfunktion und die Belohnungsfunktion sind daher auch nicht bekannt. Der Agent hat nun die Möglichkeit, durch das Exploration-Verfahren die Bewertungsfunktion $Q(s, a)$ zu lernen.

3.9.2 Annahmen

Für das Szenario wird angenommen, dass die Werte des Helligkeitssensors auf $drei$ natürlichen Zahlen $(0 \ldots 2)$[6] abgebildet sind. Für die Menge aller Aktionen wird die Menge $A = \{null, down, up\}$ eingesetzt. Der Agent hat somit in einer beliebigen Situation die Möglichkeit, das Dimmniveau zu ändern oder auch nicht. Diese Änderungen sind aufgrund der Nutzerfreundlichkeit höchstens um eine Stufe in einem Schritt möglich. Die Tabelle 3.6 zeigt die möglichen Aktionen pro Aktuatorstufe, die verbotenen Aktionen sind ebenfalls angegeben.

Aktuator	*Mögliche Aktionen*	*Verbotene Aktionen*
0	*null, up*	*down*
1	*null, up, down*	—
2	*null, down*	*up*

Tab. 3.6: Position des Aktuators und mögliche Aktionen.

Die Werte des Helligkeitssensors weisen zugleich auf den Bedarf an Licht hin, somit entstehen die Situationen aus dem Tupel *(Bedarf, Anwesenheit, Aktuatorstufe)*. Die Tabellen 3.7 und 3.8 listen alle möglichen Situationen unter der Berücksichtigung der eingesetzten Sensoren und des Aktuators auf. Alle Situationen $s_{i,j,k}$ fassen nun die Helligkeit, die Anwesenheit und die Position des Aktuators zusammen. Dabei weisen die Index-Teile i, j und k auf den Bedarf an Licht, auf die Anwesenheit des Nutzers und auf die Position des Aktuators hin.
Alle Situationen in diesem Szenario besitzen die Eigenschaften von Anfang- und Endzuständen. Die in den Tabellen 3.7 und 3.8 markierten Stellen zeigen die Situationen, die zusätzlich die Eigenschaften von Zwischenzuständen besitzen. Die Aufgabe des Agenten ist die momentan gewünschte Helligkeit unter Berücksichtigung der Nutzerfreundlichkeit in der betroffenen

[6] Im realen Fall ist dieser Wertbereich deutlich größer, das Szenario begrenzt sich jedoch aufgrund der Anzahl der resultierenden Zustände auf $drei$ Werte.

$s_{i,j,k}$	$A(s_{i,j,k})$
$s_{0,0,0}$	$\{null,\ up\}$
$s_{0,1,0}$	$\{null,\ up\}$
$s_{0,0,1}$	$\{null,\ down,\ up\}$
$s_{0,1,1}$	$\{null,\ down,\ up\}$
$s_{0,0,2}$	$\{null,\ down\}$
$s_{0,1,2}$	$\{null,\ down\}$
$s_{1,0,0}$	$\{null,\ up\}$
$s_{1,1,0}$	$\{null,\ up\}$
$s_{1,0,1}$	$\{null,\ down,\ up\}$

Tab. 3.7: Mögliche Situationen und Mengen der erlaubten Aktionen

$s_{i,j,k}$	$A(s_{i,j,k})$
$s_{1,1,1}$	$\{null,\ down,\ up\}$
$s_{1,0,2}$	$\{null,\ down\}$
$s_{1,1,2}$	$\{null,\ down\}$
$s_{2,0,0}$	$\{null,\ up\}$
$s_{2,1,0}$	$\{null,\ up\}$
$s_{2,0,1}$	$\{null,\ down,\ up\}$
$s_{2,1,1}$	$\{null,\ down,\ up\}$
$s_{2,0,2}$	$\{null,\ down\}$
$s_{2,1,2}$	$\{null,\ down\}$

Tab. 3.8: Mögliche Situationen und Mengen der erlaubten Aktionen (Folgerung aus Tab. 3.7).

Zone anzupassen. Dabei soll er selbständig die Aktionen lernen, die seine *Returns* maximieren.

3.9.3 Einsatz des Reinforcement Learning (RL)

Belohnungsfunktion

Die Belohnung wird wie folgt erteilt:

- 0: falls die Aktion verboten ist,
- $+1$: falls die Aktion in einem Zwischenzustand in die Zielrichtung führt,
- $+10$: falls durch die Aktion das Ziel erreicht wird und
- -1: für jede andere Aktion.

Die Lernrate α ist zum Beginn auf 0.3 gesetzt, der Diskontierungsfaktor γ auf 0.9. Die Berechnung der Q-Werte wurde nach der Formel aus der *Q-Learning-Methode* (vgl. Listing 2.2) realisiert.

Um die Aktion-Wert-Funktion $Q(s,\ a)$ zu berechnen, muss der Agent zu Beginn des Lernens von einer beliebigen Situation aus eine zufällige Aktion ausführen. Tatsache ist aber, dass der Agent noch kein Wissen über die Umgebung besitzt. Wir nehmen deshalb in diesem Szenario an, dass der Lernende mit der *abstrakten* Situation ($\tilde{s} = s_{x,x,x}$) und der *abstrakten* Aktion ($\tilde{a} = x$) voreingestellt ist. Dies ermöglicht das Starten der Lernphase. Mit den erhaltenen Informationen nach der Ausführung dieser Aktion bekommt der Agent eine "reale" Situation

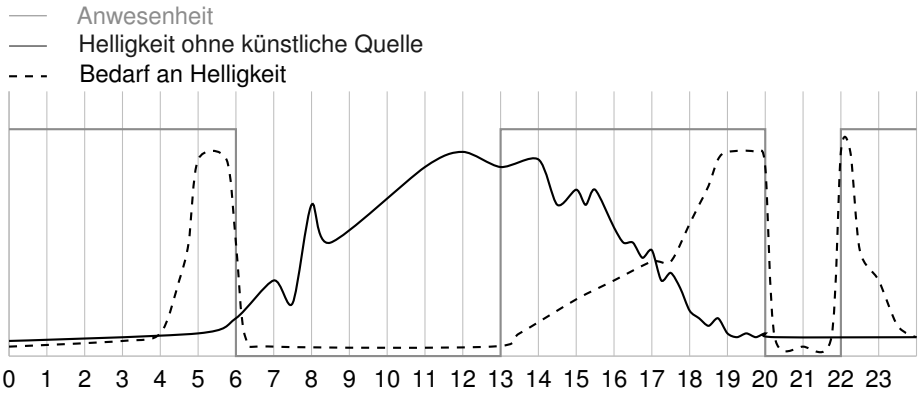

Abb. 3.9: Beispielszenario

samt ihrer Menge erlaubter Aktionen.

	$s_{i,j,k}$	$A(s_{i,j,k})$	a	r	$s'_{i,j,k}$	$Q(s,\ a)$
0	$s_{x,x,x}$	x	x	0	$s_{0,1,0}$	0.0
1	$s_{0,1,0}$	*null, up*	*up*	-1	$s_{0,1,0}$	-0.3
2	$s_{2,1,0}$	*null, up*	*null*	0	$s_{2,1,0}$	0.0
3	$s_{2,1,0}$	*null, up*	*up*	$+1$	$s_{2,1,1}$	$+0.3$
4	$s_{2,1,1}$	*null, down, up*	*null*	0	$s_{2,1,1}$	0.0
5	$s_{2,1,1}$	*null, down, up*	*up*	$+10$	$s_{2,1,2}$	3.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Tab. 3.9: Beispiel zur Berechnung der Q-Werte; $\alpha = 0.3$ und $\gamma = 0.9$

Tabelle 3.9 spiegelt den Beginn des Lernens für das Szenario wider. Als Anfangssituation wurde der Tagesanbruch und als Referenz des Tagesverlaufes das in der Abbildung 3.9 dargelegte Szenario gesetzt.

Schritt 0 - Das Lernen fängt an dieser Stelle an. Es ist z.B. kurz vor vier Uhr. Der Agent hat noch kein Wissen, er besitzt jedoch die abstrakte Situation $s_{x,x,x}$ und die abstrakte Aktion x. Nun führt der Agent die Aktion x, um das Lernen zu starten. Als Rückmeldung erhält er die Folgesituation $s_{0,1,0}$, die die Anwesenheit des Nutzers und keinen Bedarf an Helligkeit signalisiert. Die Belohnung und die Aktion-Wert-Funktion besitzen noch ihren initialen Wert.

Schritt 1 - In diesem Schritt behandelt der Agent zum erstmalig die Situation $s_{0,1,0}$. Die Wert-Funktion ist noch null, die Aktionswahl erfolgt deshalb zufällig. Die Aktion $a = up$ wird ausgewählt. Da der Zustand der Umwelt sich nicht geändert hat und daher noch kein

Bedarf an Licht besteht, wird die Nutzerfreundlichkeit ist von dieser Aktionswahl negativ beeinflußt. Der Agent erhält zur Folge eine Belohnung von -1 und die Situation $s_{0,1,0}$ als Antwort. Die Aktion-Wert-Funktion liefert $Q(s_{0,1,0},\ up) = -0.3$ zurück. Bei einem Wiederauftauchen der Situation $s_{0,1,0}$ führt die aktuelle optimale Strategie zur Auswahl der Aktion $a = null$. Der Grund dafür ist, dass diese Aktion noch nicht ausgeführt wurde. Die Wert-Funktion $Q(s_{0,1,0},\ null)$ besitzt noch den initialen Wert von 0.0, welcher größer als -0.3 ist.

Schritt 2 - Der Zustand der Umwelt hat sich geändert, die Umgebung befindet sich in der Situation $s_{2,1,0}$. Es besteht daher ein Bedarf an Licht mit der Stärke 2. Diese Information ist dem Agenten aber nicht bekannt, er handelt deswegen mit seiner aktuellen Kenntnis über seine Umwelt. Er wählt anhand seiner aktuellen Strategie für $s_{0,1,0}$ die Aktion $a = null$. Die Aktion hat keine Änderungen bewirkt, der Agent ist mit 0 belohnt worden und befindet sich weiterhin im Zustand $s_{2,1,0}$. Dieser Zustand tritt zum ersten Mal auf, die Wert-Funktion beträgt 0.0. Die nächste Aktionswahl erfolgt deshalb zufällig.

Schritt 3 - Der Agent hat nun die Situation $s_{2,1,0}$ und wählt zufällig die Aktion $a = up$, die zum Zustand $s_{2,1,1}$ führt. Dies bewirkt die Erfüllung eines Teilzieles, der Agent erhält dementsprechend eine Belohnung von $+1$, $Q(s_{2,1,0},\ up)$ ist gleich +0.3. Die aktuelle Strategie im Fall des Wiedereintretens von $s_{2,1,0}$ sollte wieder zur Auswahl der Aktion $a = up$ führen, da dieses Aktion-Situation-Paar momentan den größten Q-Wert besitzt.

Schritt 4 - Die Situation $s_{2,1,1}$ ist eine neue Erfahrung, der Agent handelt also mit einer Zufallsstrategie und wählt die Aktion $a = null$ aus. Es folgt $r = 0$ und $Q(s_{2,1,1},\ null) = 0.0$. Die Strategie bleibt in der selben Situation zufällig.

Schritt 5 - In der Situation $s_{2,1,1}$ wählt nun der Agent anhand seiner aktuellen (Zufall-) Strategie die Aktion $a = up$. Das Ziel ist erreicht, der Agent erhält die maximale Belohnung von $+10$.

Das Szenario hat gezeigt, wie die Q-Learning-Methode zur Steuerung eines Dimmers nach den vorgenommenen Annahmen eingesetzt wird. In den Schritten $0 - 5$ (Tab. 3.9) baut der Lernende seine Entscheidungspolitik auf und lernt einen Pfad zum aktuellen Ziel. Sollte die Umwelt sich in der selben Situationen befinden, würde der Agent anhand seiner Strategien handeln.
Um das Lernen zu starten, wurde im Rahmen dieses Szenario dem Agenten eine abstrakte Situation sowie eine abstrakte Aktion vorgegeben. Der Agent hat dadurch die Möglichkeit, selbst ohne Vorkenntnisse mindestens eine abstrakte Aktion auszuführen und somit einen Lernprozess zu starten.

3.10 Zusammenfassung

Tabelle 3.10 bietet einen Überblick auf die Komponenten des intelligenten Gebäude. Diese wurden nach Eigenschaften Klassifiziert.

Umgebung	Sensoren	Aktuator	Eigenschaften	Leistungsmessung
Zone	• Anwesenheitssensor • Helligkeitssensor ⋮	•Terminal •Dimmer •Fernbedinung	nicht-deterministisch, kontinuierlich, dynamisch	Ressourcenverbrauch minimieren und Lebensqualität in der Zone erhöhen
Beleuchtung	• Helligkeitssensor • Anwesenheitssensor	Dimmer	episodisch, nicht-deterministisch	Optimale Beleuchtung je nach Situation bei niedrigem Stromverbrauch
Klima	• Feuchtigkeitssensor • Temperatursensor • Anwesenheitssensor	Regler	episodisch, nicht-deterministisch	Optimales Raumklima je nach Situation bei niedrigem Energieverbrauch
Sicherheit	• Videokamera • Rauchmelder	Sensor	kontinuierlich, dynamisch, nicht-deterministisch	Sicherheit gewähren

Tab. 3.10: Unterteilung einer Zone nach Umwelteigenschaften

4 Entwurf

Gewünscht ist ein Modell, welches durch Reaktion auf Zustandsänderungen seiner Umgebung lernt, um seine Zielvorgaben zu erreichen. In Kapitel 2 wurde das Problem des *RL* erläutert. Nachdem der Gegenstand der Untersuchung in Kapitel 3 spezifiziert wurde, schlägt dieses Kapitel ein Modell des intelligenten Gebäudes sowie eines Modells seiner Steuerung auf der Basis des Reinforcement Learning (RL) vor.

4.1 Systemkomponenten

Der Typ eines Gebäudes bzw. eines Zimmers ist anhand konzeptionsbedingter Faktoren *(Architektur oder eingesetzte Baumaterialien)* vordefiniert und wird erst durch seine Nutzungsart neu definiert bzw. bestätigt. Die Funktionalität und die Baukriterien bestimmen, ob ein Gebäude Geschäfts- und / oder Wohnräume bietet und ob ein beliebiger Raum Wohn- und / oder Schlafzimmer ist.

In Kapitel 3 hat die Untersuchung der Umwelt des Agentensystems eines intelligenten Gebäudes die Einflussfaktoren ermittelt. Diese Faktoren lassen sich in zwei Gruppen unterteilen: Randbedingungen und Nutzungsart. In Abbildung 4.1 wird versucht, aus Systemsicht die Interaktionen zwischen dem Agentensystem und seiner Umwelt darzustellen.

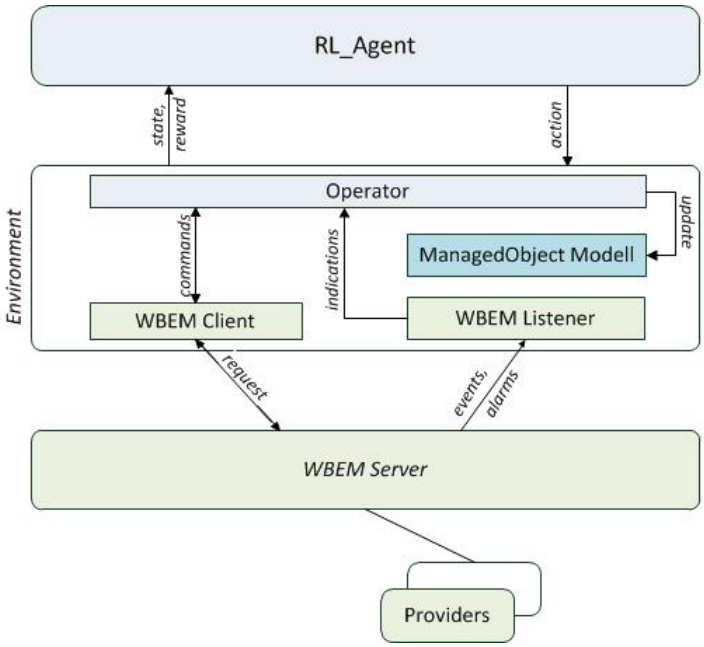

Abb. 4.1: Toplevel des Frameworks.

4.2 Modellierung der Umgebung

4.2.1 Modell des intelligenten Gebäudes

Aus der Analyse der realen Welt[1] in Kapitel 3 leitet die UML-Modellierung in Abb. 4.2 die Beziehungen zwischen den beschriebenen Objekten her. Diese Modellierung, die das *Composite Pattern* verwendet, bietet eine gewisse Hierarchie innerhalb eines Gebäudes und kann sowohl für das ganze bzw. einen Teil des Gebäudes eingesetzt werden. Die Verwaltung eines Hauses setzt sich also aus der Verwaltung seiner Komponenten zusammen. Weiterhin besteht die Möglichkeit, ein dynamisches Framework zu implementieren. Es ermöglicht ein einfaches und detailliertes Abbild von unterschiedlichen intelligenten Gebäudearten durch die Übergabe ihrer Eigenschaften an das Framework.

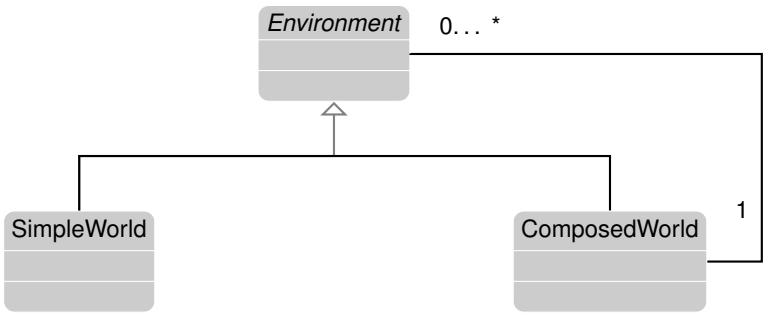

Abb. 4.2: Modell des Gebäudes.

Die Komponenten eines intelligenten Gebäudes können in die zwei folgenden Gruppen unterteilt werden:

- **Einfache Komponenten** (*SimpleWorld*)
  Alle Teile der realen Welt, die die Eigenschaften einer Zone nicht besitzen, werden mit der Klasse $SimpleWorld$ modelliert. Kandidaten für diese Klasse sind zum Bsp. Geräte. Komplexe Geräte, die aus mehreren Komponenten bestehen, können mit einer Ausnahme durch die Klasse $ComposedWorld$ abgebildet werden.

- **Zusammengesetzte Komponenten** (*ComposedWorld*)
  Diese Klasse kapselt alle Teile des intelligenten Gebäudes, die die Eigenschaften einer Zone besitzen. Das ist das Gebäude selbst sowie Teile wie Räume und Stockwerke. Eine Instanz der Klasse $ComposedWorld$ kann beliebig viele Instanzen der selben Klasse und der Klasse $SimpleWorld$ enthalten.

[1] Die reale Welt ist im Rahmen dieser Arbeit ein intelligentes Gebäude.

Die Vererbung der Eigenschaften der Klasse *CIM_ManagedSystemElement* (vgl. Abb. 4.3) an das Modell des Gebäudes ermöglicht eine dynamische Verwaltung seiner Komponenten. Dies bewirkt aufgrund des benutzten Patterns die Vererbung dieser Eigenschaften an alle konkreten Teile der realen Welt. Das intelligente Gebäude sowie alle seine Komponenten bieten somit unabhängig voneinander die Möglichkeit, sie zu verwalten. Die Klasse *CIM_ManagedSystemElement* vererbt ihre Eigenschaften an die Klassen *CIM_LogicalElement* und *CIM_PhysicalElement*. Diese Klassen ermöglichen die Spezialisierung von nötigen Klassen, die für Management von Soft- und Hardware eingesetzt werden.

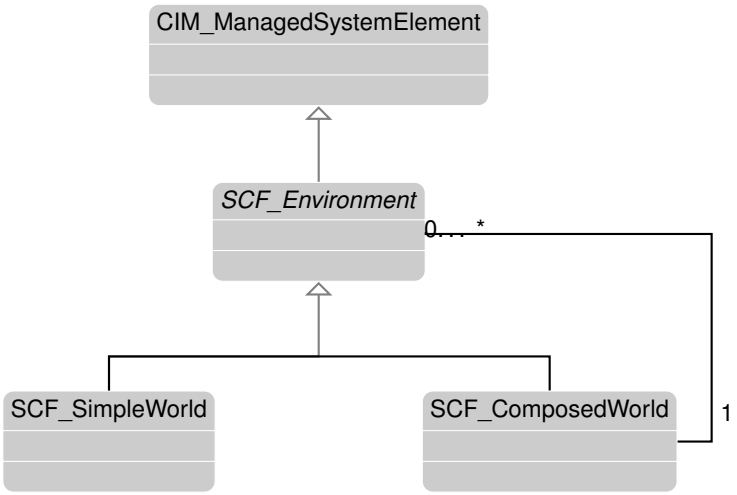

Abb. 4.3: Modell des intelligenten Gebäudes; Vererbung der Eigenschaften der Klasse CIM_ManagedSystemElement.

Abbildung 4.4 zeigt das Gesamtmodell mit bestehenden Assoziationen zwischen den Klassen. Diese Assoziationen spiegeln die Tatsache wider, dass eine Zone *(z.B. das Wohnzimmer)* beliebig viele Aktionsräume enthalten kann. Eine Vererbung der Eigenschaften der Klasse *CIM_SystemComponent* an Instanzen dieser Assoziation sichert deren Handhabbarkeit.

4.3 Lernmodell

In diesem Abschnitt wird anhand der in Kap. 3 durchgeführten Analysen ein Lernmodell zur Optimierung der Steuerung eines intelligenten Gebäudes vorgestellt.

Das Lernen im Rahmen dieser Arbeit basiert auf dem im Abschnitt 2.2.4 vorgestellten Modell des lernenden Agenten. Dieser Ansatz wird in diesem Framework mit einem Agentensystem, bestehend aus zwei Arten von Agenten, umgesetzt: einem *Operator-Agent*, der die Rolle des *Leistungselements* übernimmt und einem *RL-Agent*[2], der die restlichen Komponenten kapselt.

[2]Reinforcement Learning Agent bzw. lernender Agent.

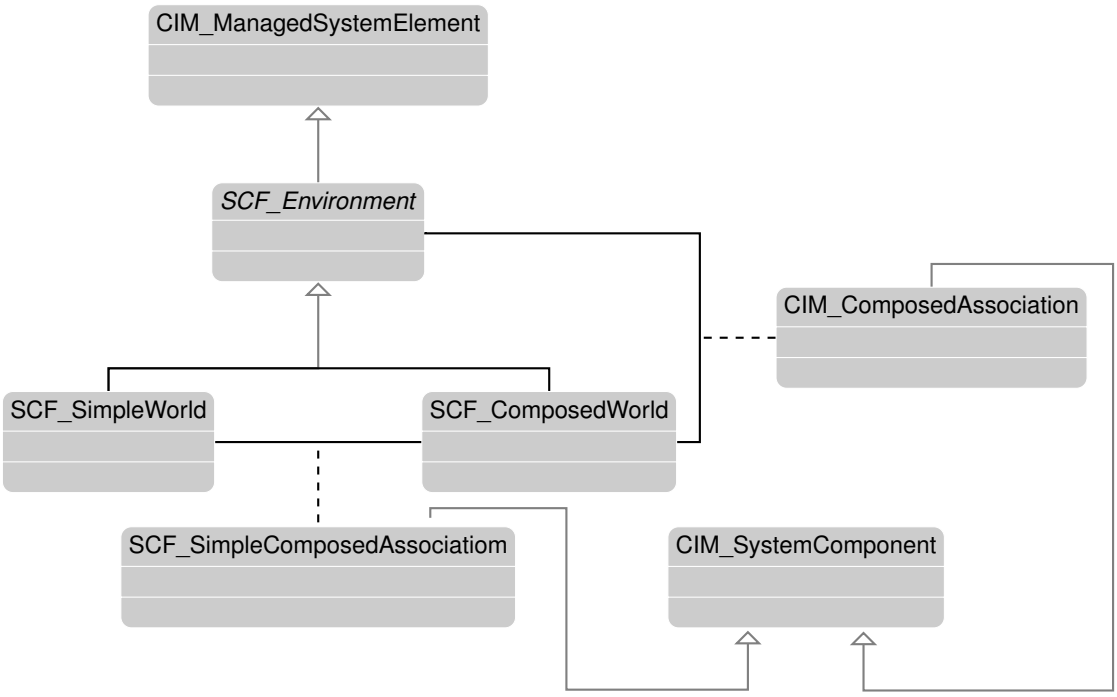

Abb. 4.4: CIM-Schema.

Diese Trennung bietet mehr Flexibilität im Design bzw. bei der Implementation, indem sie je nach Komplexität des Problems den Einsatz von einer variablen Anzahl an Leistungselementen unterstützt.

4.3.1 Der Operator

Als *2nd-level* Komponente des lernenden Agentensystems spielt der Operator die Rolle eines "*Vollstreckers*". Der Auftraggeber *(der lernende Agent)* trifft alle Entscheidungen, die eine Veränderung der Umwelt bewirken sollen. Für jeden übernommenen Auftrag teilt der Operator dem lernenden Agenten mit, wie gut die Entscheidung gewesen ist sowie den aktuellen Zustand der Umwelt. Er ist also für die Beobachtungen der Umgebung und die Ausführung von Aktionen zuständig. Die Interaktionen zwischen Operator und lernendem Agenten kapseln daher einen wichtigen Aspekt des RL.

Für die Steuerung eines intelligenten Gebäudes kann die Anzahl der Operatoren je nach Größe[3] des Hauses variieren. Das Verhältnis Ressourcenverbrauch/Leistungsfähigkeit wird daher bei wachsender Anzahl der verwalteten Objekte berücksichtigt. Dies impliziert einen dynamischen Einsatz der Operatoren. Das im Abschnitt 4.2.1 vorgestellte Modell ermöglicht je nach Situation den Einsatz von einem bzw. mehreren Operator(en).

[3] Die Größe bezieht sich hier auf die Anzahl der verwalteten Objekte.

Es wird bei der Implementation dieses Frameworks in Kap. 5 angenommen, dass pro Zone ständig ein Operator zur Verfügung steht. Dieser Ansatz wird vom Modell des Hauses und von der Agentenplattform unterstützt.

Auf der JADE-Plattform leben Agenten in Containern. Diese Eigenschaft bietet die Möglichkeit, Zonen mit Containern zu repräsentieren, was zu einer Modularisierung des Gebäudes für eine spätere Implementation führt. Abb. 4.5 zeigt das Beispiel des Abbildes eines Gebäudes bestehend aus vier Einheiten *(Bad, Küche, Wohn- und Schlafzimmer)*. Wie zu sehen ist, wurde das Haus in mehrere Teile zerlegt, die mittels JADE-Containern gekapselt sind. Die Container hosten jeweils einen Operator, der für eine bestimmte Zone zuständig ist.

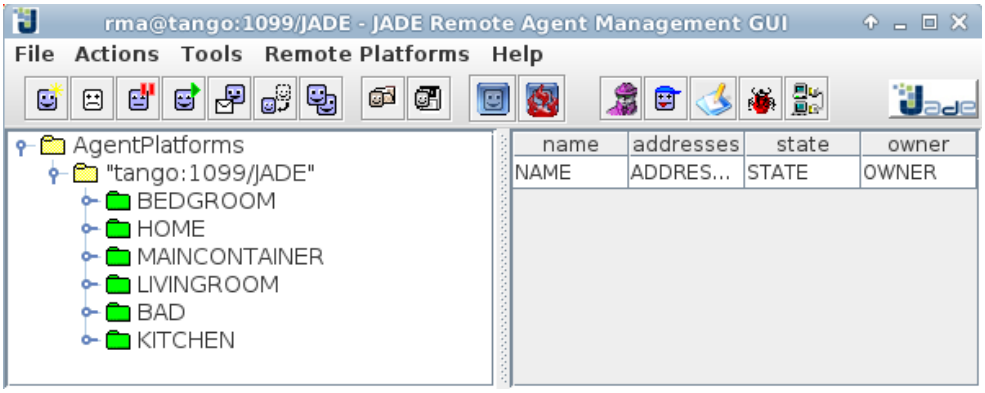

Abb. 4.5: RMA-GUI; Beispiel einer Implementation eines Gebäudes auf der JADE-Plattform.

4.3.2 Lernender Agent

Bei der Einführung des Problems des RL in Abschnitt 2 wurde auf die Eigenschaft des selbständigen Lernens hingewiesen. Diese Tatsache ruft eine Änderung der zweiten Komponente des lernenden Agentensystems hervor, da die *Kritik* im Rahmen des RL durch das Erteilen der Belohnungen von der Umwelt übernommen wird. Der lernende Agent besteht also aus einem *Lernelement* und einem *Problemgenerator*.

Lernelement

Das Lernelement repräsentiert die Entscheidungs- und Optimierungseinheit. Für die bestehende Aufgabe wurde aus der Spezifikation des Lernens die Tatsache, dass am Beginn des Prozesses die Umgebung dem Agenten unbekannt ist, erläutert. Das Lernelement wird deshalb mit der *Q-Learning-Methode* voreingestellt.

Problemgenerator

Das Lernen im Rahmen dieser Arbeit wurde im Bezug zum intelligenten Gebäude spezifiziert, eine Anpassung des Lernmodells erweist sich deshalb an dieser Stelle nötig.
Bei der Analyse der Zone wurde ihre Verwaltung als eine sequentielle Ausführung von episodischen Prozessen beschrieben. Die Bewertung des sequentiellen Prozesses, die zugleich die Nutzerzufriedenheit innerhalb der Zone ausdrückt, hängt von der Erfolgsquote der episodischen Tasks ab. Diese werden am Ende jeder Episode belohnt, die Belohnung der sequentiellen Aufgabe wird in einer vom Nutzer bzw. Entwickler definierten Zeit (z.B. Abrechnungszeitraum) erteilt. Durch diese Belohnung wird der Lerneinheit mitgeteilt, wie zufrieden der Nutzer mit dem Ressourcenverbrauch und seiner Lebensqualität in einer Periode innerhalb einer Zone ist. Folgende Annahmen werden daher getroffen:

1 - Das Management einer Zone (also der sequentielle Prozess) ist über einen Zeitraum günstig, wenn in allen Aufgabenfeldern die Systemziele erreicht wurden, also einen Lernerfolg aufweisen.

2 - Der Lernerfolg in einem Aufgabenfeld hängt von der eingestellten Explorationsrate ab.

Zur Folge werden innerhalb einer Zone zwei Lektionen definiert:

- Das Erlernen einer optimalen Explorationsrate für jeden in der Zone ausgeführten episodischen Prozess. Diese Aufgabe wird anhand der erläuterten Abhängigkeiten mit dem sequentiellen Prozess umgesetzt. Das Management der Zone ist somit auf der Lernebene für das Bereitstellen von optimalen Explorationsraten zuständig. Das Lernen wird in diesem Fall so organisiert, dass in jedem Abrechnungszeitraum anhand der erhaltenen Belohnung die Explorationsrate optimiert wird. Der Erhalt einer positiven Belohnung weist auf eine aktuell gute Explorationsrate hin und wird deshalb weiter eingesetzt. Bei einer negativen Belohnung wird die Explorationsrate auf der Basis des RL optimiert. Die Umgebung ist in diesem Fall durch eine Liste von Eigenschaften (z.B. die Gesamtbelohnung) abgebildet, die den jeweilligen Lernfortschritt der episodischen Tasks angeben. Die Zustände der Umwelt sind also die Einträge in der Liste.

- Das Erlernen einer optimalen Strategie in den jeweilligen Aufgabenfeldern.

Das resultierende Framework besteht wie in der Abbildung 4.6 gezeigt aus einem Haupt-RL-Problem (die sequentielle Verwaltung der Zone), innerhalb dessen beliebig viele RL-Probleme mit episodischen Eigenschaften ausgeführt werden können.

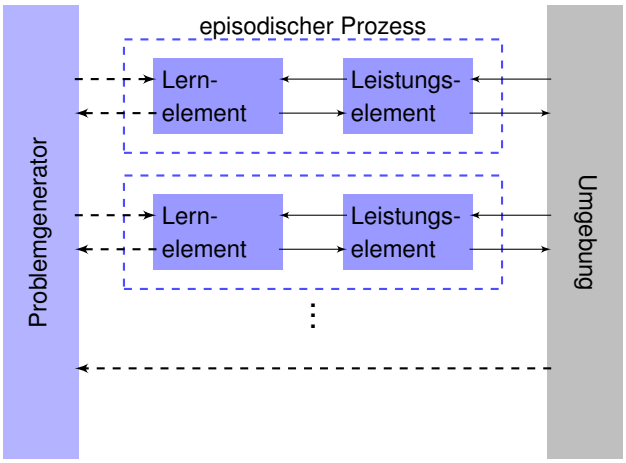

Abb. 4.6: Lernmodell innerhalb einer Zone

5 Implementation

Dieses Kapitel schlägt eine Umsetzung des in Kap. 4 entworfenen Modells vor.

5.1 Umgebung

5.1.1 Gebäudemodell

Das im Abschnitt 4.2.1 vorgeschlagene Gebäudemodell (vgl. Abb. 4.4) wird nun in MOF beschrieben. Abb. 5.1 zeigt eine MOF-Definition der abstrakten Komponente des *Composite Pattern*, hier die Klasse *SCF_Environment*.

```
...
[ Version ("1.0.0"),
  Abstract,
  Description( "Provide an Interface for the SCF_ComplexWorld"
               "and the SCF_SimpleWorld." ) ]
  class SCF_Environment : CIM_ManagedSystemElement
  {
      string environmentId;
  };
...
```

Listing 5.1: Umgebung-Klasse in MOF.

Eine Zone *(Gebäude bzw. Raum)* wurde mit der Klasse *ComposedWorld* modelliert. Abb. 5.2 zeigt eine MOF-Repräsentation einer Zone.

```
...
[ Description ( "A class derived from SCF_Environment." ) ]
  class SCF_ComplexWorld : SCF_Environment
  {};
...
```

Listing 5.2: Definition einer Zone in MOF.

Assoziationen

Listing 5.3 gibt das Beispiel der Umsetzung der in der Abbildung 4.4 skizzierten Klassenassoziationen an.

```
...
[ Association,
  Description ( "Asscociation between SCF_ComplexWorld"
                "and SCF_SimpleWorld.") ]
```

```
  class SCF_ComplexSimpleAssociation : CIM_SystemComponent
  {
   [ Description( "Reference to SCF_ComplexWorld." ) ]
   SCF_ComplexWorld REF antecedent;
   [ Description( "Reference to SCF_SimpleWorld." ) ]
   SCF_SimpleWorld REF dependent;
  };
...
```

Listing 5.3: Assoziation der Klassen SCF_SimpleWorld und SCF_ComplexWorld.

5.1.2 Umsetzung in Java

```
...
public abstract class Environment
                      extends
                      CIM_ManagedSystemElement {}
...
```

Listing 5.4: Vererbung der Eigenschaften der Klasse CIM_ManagedSystemElement

Der Konstruktor der Klasse *ComposedWorld* erwartet fünf Parameter *(siehe Abb. 5.5)*. Vier davon *(host, port, localport und containername)* werden bei der Erzeugung von Containern auf der JADE-Plattform benötigt. Der letzte Parameter *(model)* ist ein Array von Objekten, der die Eigenschaften der realen Welt im Rahmen des *RL* im Modell der Umgebung abbildet. Das Modell wird als Argument während der Initializierung an den Operator-Agent übergeben. Es besteht aus folgenden Objekten:

- eine Liste von Aktionen,
- eine Liste von Sensoren,
- die Belohnungsfunktion.

```
...
public ComposedWorld ( String host,
                       String port,
                       String localport,
                       String containername,
                       Object[] model );
...
```

Listing 5.5: Konstruktor der Klasse ComposedWorld (Zone)

Diese Klassen definieren im Rahmen eine WBEM-Anwendung zugleich die Provider. Listing 5.6 zeigt das Beispiel der Registrierung der Provider der Klassen *SCF_ComplexWorld* und *SCF_SimpleWorld*.

```
# Classname Namespace ProviderName ProviderModule ProviderTypes ...
 SCF_ComplexWorld root/cimv2 OSBase_ComplexWorldProvider cmpiOSBase_ComplexWorldProvider
   class
```

```
SCF_SimpleWorld root/cimv2 OSBase_SimpleWorldProvider cmpiOSBase_SimpleWorldProvider class
```

Listing 5.6: Registrierung der Provider

5.1.3 Der WBEM-Client

Die Entwicklung von WBEM-Client-Anwendungen in Java erfolgt unter der Java Specification Request (JSR) 48 API *(Ref. (jcp, 2013))*.

5.2 Agentensystem auf der Basis von JADE

Dieser Abschnitt behandelt die Umsetzung des Agentensystems auf der JADE-Plattform.

5.2.1 Agentenkommunikation & Protokolle

In Abschnitt wurde das ACL als das von JADE verwendete Nachrichtenformat erläutert. Die Klasse *ACLMessage* ermöglicht das Versenden bzw. das Empfangen von Nachrichten. Folgende Methoden werden eingesetzt:

- send(ACLMessage),
- receive(),
- blockingReceive(),
- createReply(),
- block().

5.2.2 Agententask

Auf der JADE-Plattform wird die Ausführung einer Aktion wie ein Verhalten (engl.: behaviour) interpretiert. Die Plattform bietet zudem die abstrakte Klasse *Behaviour* aus dem Paket *jade.core.behaviours*. Diese Klasse ermöglicht die Spezialisierung weiterer Klassen zur Umsetzung beliebigen Agentenverhaltens. Die folgenden Klassen wurden verwendet:

- *OneShotBehaviour*,
- *CyclicBehaviour*,
- *TickerBehaviour*.

Eine Erläuterung der Eigenschaften dieser Klassen erfolgt anhand eines Beispieleinsatzes im Abschnitt 5.2.3.

5.2.3 Operator

Auf der Basis der Definition eines Agenten in Kap. 3 könnte der Operator als ein einfacher Agent ohne besondere Intelligenz implementiert werden. Der Operator wird beim Initializieren der Umgebung erzeugt und gestartet. Er bekommt als Argument das in Abschnitt 5.1.2 beschriebene Modell der Umwelt.

```
1  @Override
2  public void init() {
3    try {
4      System.out.println( "Creating an operator for " + this.iD );
5      // create and start the operator
6      AgentController ac = container.createNewAgent(iD + "_OPERATOR",
                            "agentsystem.environment.Operator", model);
7      ac.start();
8    } catch (StaleProxyException e) {
9      e.printStackTrace();
10   }
11 }
```

Listing 5.7: Initializierung der Umgebung; der Operator wird dabei gestartet.

Der Ablauf des Operator-Programms könnte wie in Abbildung 5.1 gezeigt in drei Phasen ausgeführt werden.

1. Registrierung

In der ersten Phase veröffentlicht der Operator zuerst seine Dienste in dem Directory Facilitator (DF). Durch diese Veröffentlichung wird die Lernkomponente auf die Existenz eines neuen Operators hingewiesen. Diese Phase wird in der inneren Klasse *ServiceReleaseBehaviour* implementiert. Die Phase 1 wird genau einmal beim Start eines Operators ausgeführt und wird auf JADE als ein einmaliges Verhalten des Operators betrachtet. Die Klasse *ServiceReleaseBehaviour* erbt deshalb als einmaliges Verhalten eines Agenten die Eigenschaften der Klasse *OneShotBehaviour*. Abb. 5.8 zeigt das Beispiel einer Registrierung der Dienste eines Agenten.

```
...
  DFAgentDescription dfd = new DFAgentDescription();
  dfd.setName(getAID());
  ServiceDescription sd = new ServiceDescription();
  ...
  dfd.addServices(sd);
  try {
    DFService.register(myAgent, dfd);
}
...
```

Listing 5.8: Veröffentlichung der Operator-Dienste.

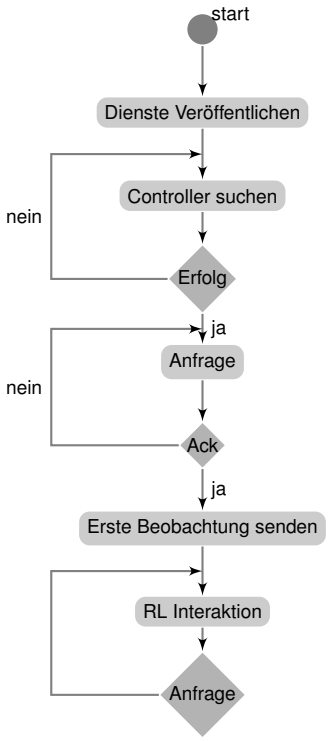

Abb. 5.1: Flussdiagramm des Operator-Programms.

2. Suche eines Controllers

Die 2. Phase beginnt mit der Suche nach den Diensten eines Controllers. Die Rolle des Controllers ist, wie in Abschnitt 4.3.1 erläutert wurde, von einem RL-Agent übernommen. Nach einer erfolgreichen Suche des Controllers sendet der Operator eine Anfrage zur Übergabe der Steuerung der vertretenen Umgebung an den Controller und wartet auf eine Bestätigung. Bei einer bestätigten Anfrage sendet der Operator den initialen Stand der Umgebung an seinen Controller. Diese Phase definiert ebenfalls ein einmaliges Verhalten des Operators und wird in der inneren Klasse *SearchControllerBehaviour* implementiert.

3. Steuerung

In Phase 3 wird in einem kontinuierlichen Prozess die eigentliche Steuerung des verwalteten Objektes ausgeführt. Die Steuerung ist im Rahmen des RL eine wiederholte Ausführung eines 3-Phasen-Zyklus (vgl. Abschnitt 2.1.1).
Die Phase 3 wird in einer inneren Klasse mit der Vererbung der Eigenschaften der Klasse *CyclicBehaviour* implementiert. Jeder Zyklus wird durch den Erhalt einer Anforderung (die durchzuführende Aktion) des Lernenden ausgelöst.

5.3 Die Lernkomponente

Basierend auf der Analyse zum Entwurf des Lermodells in Abschnitt 4.3.2 sollte das im Rahmen eines Gebäudemanagements geplante Lermodell die Möglichkeit, weitere Lektionen zur Optimierung der Lerneigenschaften zu starten, besitzen. Diese Lektionen beschreiben ein RL-Problem einer anderen Art, da ihre Umgebungen Unterschiede aufweisen. Dazu wird dem Framework die Möglichkeit gegeben, unabhängig der Art der Umgebung ein Lernverfahren starten zu können. Dies impliziert die Realisierung eines Frameworks, das eine *generische* Erzeugung von Objekten unterstützt.

5.3.1 Abstrakte Situation / Situation

Die Eigenschaften eines Zustandes der Umwelt wurden in der Klasse *State* umgesetzt.

- *iD*: eine eindeutige Kennung. Um die Eindeutigkeit zu sichern, nimmt das Attribut den Wert einer Kombination der Sensorwerte an.
- *qvalue*: vom Typ *double*, das Attribut definiert den Nutzwert einer Situation. Es wird im Laufe des Lernens nach jedem Schritt aktualisiert.
- *typ*: das Attribut gibt die Art *(Start, Ende, Zwischen)* des Zustandes an. Durch diese Angabe kann der Status eines Task berücksichtigt werden.
- *actionSet*: eine Liste von Aktionen von Typ *ArrayList<ActionSet>*, die die Menge $A(s)$ (vgl. Abs. 2.3.1) der möglichen Aktionen in der Situation abbildet.

5.3.2 Abstrakte Aktion / Aktion

Die Aktionen wurden mit der Klasse *Action* umgesetzt. Die Klasse *Action* ist eine Spezialisierung einer abstrakten Aktion.

5.3.3 Aktion_Situation_Paare

Das Aktion-Situation-Paar wurde in der Klasse *ActionStatePair* implementiert. Diese klasse muss für eine gute Umsetzung der Algorithmen die Interface *java.lang.Comparable* implementieren. In der Methode *compareTo()* reicht die Gleichheit der Aktion und Situation aus (damit zwei Aktion-Situation-Paare gleich). Grund dafür ist die Aktualisierung der Q-Werte, die sich auf das Attribut *value* auswirken soll.

- *action*: die Aktion,
- *state*: die Situation,
- *value*: der Aktion-Wert $Q(s, a)$ vom Typ *double.*

5.3.4 Das Problem

Die Klasse *Problem* ermöglicht es dem Agenten, die Steuerung einer Zone als ein RL-Problem zusammenzufassen.

- *operator*: das Attribut ist vom Typ *String* und repräsentiert den Namen eines Operators auf der Plattform.
- *memory*: eine Instanz der inneren Klasse *Memory*, die das Lernen in der betroffenen Zone realisiert.

Für jeden Operator legt der Agent ein Objekt der Klasse *Problem* an. Dies bewirkt die Erzeugung einer Instanz der Klasse *Memory*, die das RL-Problem beschreibt (vgl. Lst. 5.9).

```
...
  public Problem ( String operator ) {
    this.operator = operator;
    this.memory = new Memory();
  }
...
```

Listing 5.9: Konstruktor der Klasse Problem.

Die Klasse *Memory* kapselt den Lernprozess und bietet zugleich eine Übersicht über die momentane Beobachtung (aktuelle Aktion, Situation und Belohnung) und über den Verlauf des Lernprozesses an. Die Klasse wurde mit folgenden Attributen definiert:

- *action*: eine Instanz der Klasse *Action*, die die aktuelle Aktion referenziert.
- *state*: der aktuelle Zustand der Umgebung, Instanz der Klasse *State*.
- *reward*: die aktuelle Belohnung vom Typ *Integer*.
- *elements*: vom Typ *ArrayList<Object>*, das Attribut speichert alle benötigten Objekte für den Einsatz der RL-Algorithmen. Diese Objekte wurden aus folgenden Typen erzeugt:
  - *ArrayList<Action>*: enthält alle in der Umgebung existierenden Aktionen.
  - *ArrayList<State>*: die Liste umfasst alle Situationen der Umgebung.
  - *Algorithm*: die im Rahmen des RL eingesetzte Methode zur Berechnung der Nutzwerte.
  - *ArrayList<ActionStatePair>*: die Liste enthält alle besuchten Aktion-Situation-Paare.
  - *ArrayList<StateActionSet>*: enthält die Liste der Menge $A(s)$ aller Situationen.

  Die Größe der Listen ist im Rahmen des RL dem Agenten zum Beginn des Lernens nicht bekannt gegeben.

5.4 Realisierung des Frameworks

Um die Anwendung zu starten, wurde eine einfache TCP/IP Client-Server Anwendung eingesetzt. Der Client ermöglicht das Hochladen des Gebäudemodells auf den Server, der für die Bearbeitung der Eingabedatei zuständig ist.

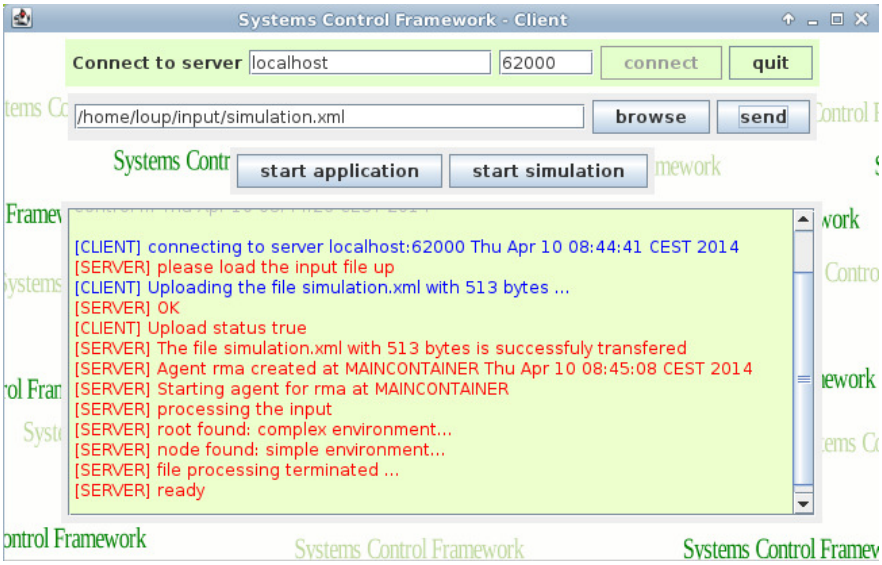

Abb. 5.2: Client-GUI;

Nach einem erfolgreichen Hochladen des Modells startet der Server die JADE-Plattform und erzeugt anhand des erhaltenen Modells die nötigen Container zum Kapseln der definierten Komponenten des Gebäudes. Dabei werden schon die Operatoren (auch Leistungselemente) erzeugt und gestartet (vgl. Abb. 5.3).

Das Framework kann nun gestartet werden in Hinsicht auf eine direkten Anwendung oder als eine Simulation. Beim drücken der Taste *start simulation* (vgl. Abb. 5.2), startet der Simulator und gibt die Möglichkeit, einen Operator auszuwählen sowie die nötigen Lernparametern (Lernrate, Diskontierungsparameter) einzustellen (vgl. Abb. 5.4).

Der Start des Simulators bewirkt das Starten des Lernelementes und ein Lernprozess kann dann beginnen.

Für die Übergabe des Gebäudemodells an das Framework wurde das XML-Format verwendet. Listing 5.10 zeigt das Modell einer Zone, das in XML umgesetzt wurde. Die Struktur des Gebäudemodells ist wiederzuerkennen.

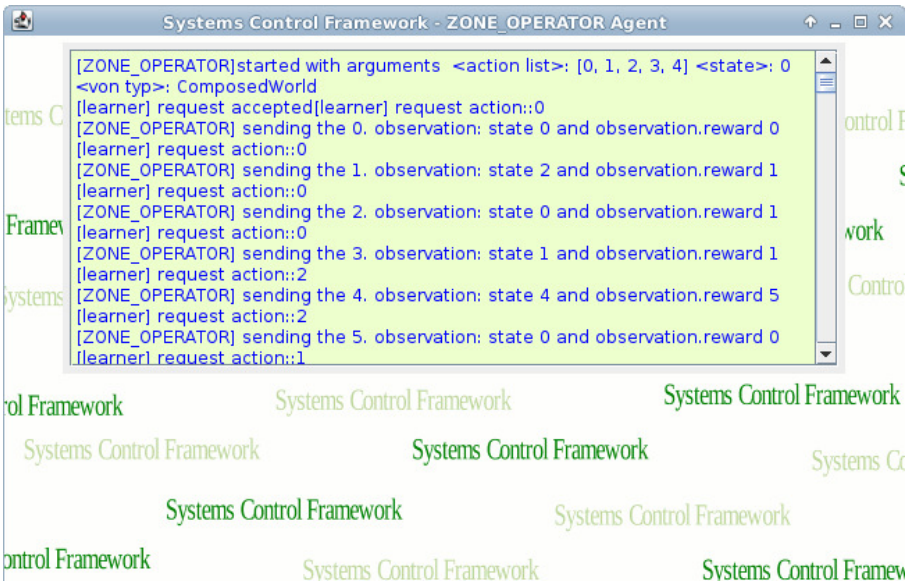

Abb. 5.3: Operator-GUI

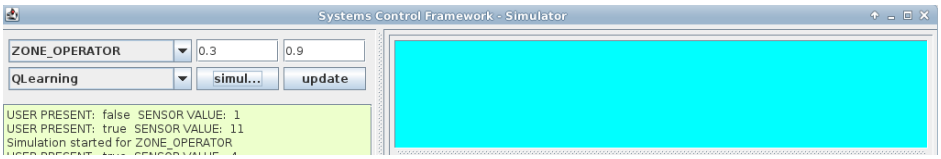

Abb. 5.4: Simulator-GUI

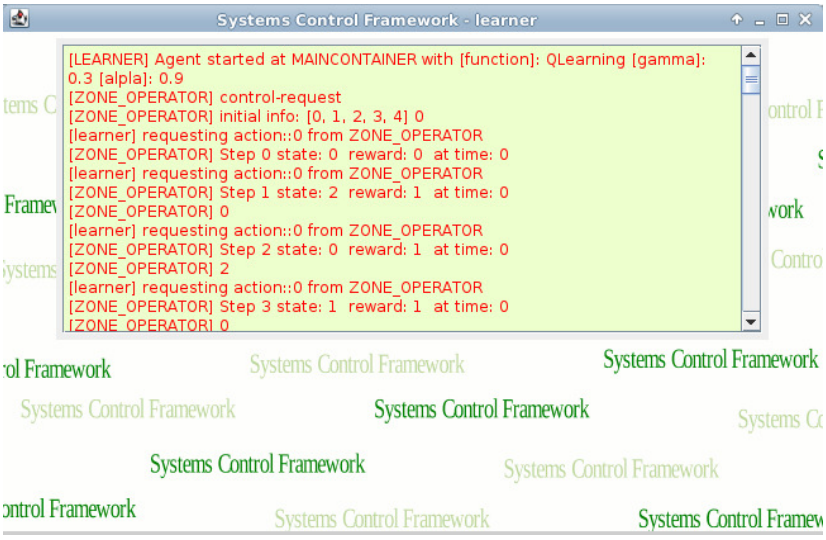

Abb. 5.5: GUI des Lernelementes.

```
<xml version="1.0" encoding="UTF-8" >
<complex> <!-- Teil der realen Welt: die Zone -->
       <iD id='zone'/>
       <address host ='localhost' port = '1099' localport = '62020'/>
       <description />
       <simple> <!-- Aufgabenfeld innerhalb eines Zimmers: Beleuchtung -->
              <iD id='light'/>
              <address host ='localhost' port = '1099' localport = '62025'/>
              <description />
              <action type ="array">
                     <value>null</value>
                     <value>down</value>
                     <value>up</value>
              </action>
       </simple>
</complex>
```

Listing 5.10: Eingabedatei eines Gebäudemodells; das Modell wird in XML-Format eingegeben

5.5 Simulation

Auf der Basis der Erkenntnisse aus den vorherigen Kapiteln bietet dieser Abschnitt eine Simulation der Steuerung eines intelligenten Gebäudes. Im Rahmen dieser Arbeit wurde ein Agenten-Framework mit *verstärkenden Lernfähigkeiten* auf JADE implementiert. Dieser Abschnitt schlägt ein Konzept der Simulation des verstärkenden Lernens in einem intelligenten Gebäude vor.

5.5.1 Vorhaben

Als Umgebung wird das Beispiel einer Zone mit einer Dämmerungsaufgabe betrachtet. Es wird weiterhin angenommen, dass in diesem Funktionsbereich bis zu n Zustände beobachtet werden. Die Anzahl der Aktionen ist hier gleich der Anzahl der Zustände. Es wird angenommen, dass alle Situationen anhand der Systemziele eindeutige Nutzwerte besitzen. Da der Nutzwert der Zustände von der Tageszeit abhängig sein kann, könnte somit die Nutzwertfunktion (engl.: utility function) eines Zustandes im Laufe des Tages unterschiedlich ausfallen.

Situationen	s_0	s_1	s_2	s_3	...	...	s_{n-2}	s_{n-1}	s_n
Nutzwert	0	1	2	3	...	...	$n-2$	$n-1$	n

Tab. 5.1: Situationen

Die Versuche basieren auf dem Q-Learning-Algorithmus. Dabei werden die Einflüsse des Dis-

kontierungsparameters (γ), der Lernrate (α) und der Dimension[1] des Problems auf das Lernen untersucht.

5.5.2 Simulator

Die Ereignisse innerhalb der Zone werden durch einen Simulator, der das Testen einer beliebigen Zone ermöglicht, erzeugt. Der Simulator wurde als ein Agent, der über seine Benutzeroberfläche das Auswählen des Operators der zu testenden Zone bietet, implementiert. Der Simulator-Agent wurde mit mehreren Verhalten realisiert, die die Zustände der Umgebung simulieren. Eine Anforderung an den Simulator ist das Widerspiegeln der in Abschnitt 3.5.1 erläuterten Eigenschaften der Zone bei der Entstehung der Ereignisse.

Tag

Die Simulation der Stunden eines Tages wurde mit der inneren Klasse *DaySimulator* realisiert. Um die Stunden mit einem gleichen Takt abbilden zu können, implementiert die Klasse *DaySimulator* das Interface *TickerBehaviour*. Die Periode bildet hier eine reale Stunde ab und wurde auf $12500ms$ gesetzt. Somit beträgt die Simulation eines 24 Stunden Tages 5 Minuten.

Anwesenheit

Die Anwesenheit eines Nutzers wurde in der Klasse *PresenceSimulator* umgesetzt. Da die Information über die Anwesenheit einmal bei jeder Beobachtung vom Operator nachgefragt wird, implementiert die Klasse *PresenceSimulator* das Interface *OneShotBehaviour*. Es wurde aufgrund der Anforderungen kein typischer bzw. fester Tagesablauf nachgebildet. Ein typischer Tagesablauf würde z.B. den Tagesverlauf eines Arbeiters, eines Studenten, eines Rentners oder einer Hausfrau abbilden und dadurch quasi deterministische Eigenschaften aufweisen. Für die Simulation des Anwesenheitssensors wurde die Änderung der Temperatur eines Rechners in einer alternierenden Folge von 0 und 1 abgebildet. Für die Zeit der Simulation ändert sich die Temperatur des Rechners relativ langsam, der Zeitpunkt der Änderungen bleibt jedoch nicht-deterministisch. Dieser Ansatz simuliert den Tagesablauf eines allgemeinen Nutzers und erfüllt die *dynamische u. nicht-deterministische* Anforderung.

Sensor

Die Klasse *StateSimulator* implementiert die Zustandsübertragungsfunktion. Um die *nicht-deterministische* Eigenschaft zu gewährleisten, wurde die Abfolge der Zustände nach einem Zufallsprinzip realisiert.

[1] Die Dimension bezieht sich auf die Anzahl der Zustände

Aktuator

Die Simulation der Position des Aktuators wurde in der Klasse *ActuatorSimulator* erfasst. Durch dieses Verhalten aktualisiert der Simulator die Position des Aktuators nach jeder Änderung. Die Veränderung der Position ist von einer Interaktion des Nutzers oder von der zuletzt durchgeführten Aktion abhängig. Aus nutzerfeundlichen Gründen soll der Aktuator höchstens zwei Stufen in einem Schritt bewegt werden.

5.6 Entwicklungsumgebung

5.6.1 JADE-Plattform

Der Einsatz der JADE-Plattform setzt die Installation der folgenden Bibliotheken voraus:

- *jade.jar*
- *jadeTools.jar*
- *iiop.jar*
- *http.jar*
- *commons-codec-1.3.jar*

Die JADE-Plattform unterstützt die Entwicklung von verteilten Applikationen, vorausgesetzt wird eine Installation der erläuterten Bibliotheken auf alle beteilligten Hosts.

5.6.2 CIM-Server

Die Anwendung wurde unter einem Linux-Betriebsystem entwickelt. Die Suche nach einem CIM-Server wurde vereinfacht, da die verwendete Distribution des Betriebsystems das Small Footprint CIM Broker (SFCB) bietet, welches von Standards Based Linux Instrumentation for Manageability (SBLIM) entwickelt wurde.

6 Ergebnisse

In Kapitel 2 wurde das Ziel dieser Arbeit erläutert. Nach der Spezifikation der Aufgabe wurden im 4. Kapitel Modelle entworfen und ihre Implementation in Kapitel 5 vorgeschlagen. Dieses Kapitel bietet eine Auswertung des entworfenen Frameworks bestehend aus dem Gebäudemodell und dem Lernmodell.

6.1 Auswertung der Modelle

6.1.1 Gebäudemodell

Bei der Analyse der realen Welt wurde die Notwendigkeit einer Zerlegung des Problemes erläutert. Diese Vorgehensweise hat die Modularisierung des Gebäudes und dessen Komponenten ermöglicht und die Wahl eines Entwurfmusters vereinfacht. Basierend auf dem *Composite Pattern* ist es gelungen, ein dynamisches Modell des intelligenten Gebäudes zu entwerfen, welches die Vorgaben erfüllt. Während der Umsetzung des Modells hat sich gezeigt, dass es sich ohne großen Aufwand ändern und erweitern lässt.

6.1.2 Lernmodell

Die Steuerung eines intelligenten Gebäudes wurde in der Spezifikation als Reinforcement Learning Problem umgesetzt. Die nötigen Lernparameter wurden im Rahmen eines Lernverfahrens in gegebener Umgebung erfasst und abgeleitet. Ein Teil der Komplexität in Hinsicht auf das Lernen wurde schon im Gebäudemodell abgenommen. In einer modularisierten Umgebung wurde das Lernen zonenweise organisiert und damit die Komplexität des Problemes der Steuerung eines Gebäudes auf eine Zone reduziert.

Um den Einsatz des Modells eines allgemeinen lernenden Agenten in Hinsicht auf das RL zu ermöglichen, wurde die Kritik-Komponente entkoppelt und ihre Eigenschaften in die Umwelt verschoben. Dieser Schritt ermöglicht erst den Einsatz des Modells im RL. Das Modell wurde anschließend für einen Einsatz des verstärkenden Lernens als Optimierungslösung im Rahmen eines Gebäudemanagement erneut angepasst. Diese Notwendigkeit ergibt sich beim Festlegen der Eigenschaften des intelligenten Gebäudes als Umgebung eines Agenten. Die Untersuchung des zeitlichen Verhaltens der Prozessabläufe innerhalb des Gebäudes hat für eine Zone die Aufteilung in zwei Lernprozesse ergeben. Eine Hierarchie sowie die Abhängigkeit wurde zwischen diesen Prozessen abgeleitet. Hieraus ergab sich das Management einer Zone als Haupt-Prozess, innerhalb dessen der zweite Prozess abläuft und von dem seine Effizienz abhängt. Das Lernmodell wurde entsprechend erweitert und besteht nun aus einem Haupt-RL-Prozess, der eine optimale Lernrate für alle anderen RL-Prozesse, die unter ihm ablaufen, erlernt. Die Abstraktion der Steuerung einer Zone wurde somit mit dem resultierenden

Lernmodell realisiert.

Das Dilemma beim Start des Lernens in unbekannten Umgebungen wurde durch die Definition einer abstrakten Situation und einer abstrakten Aktion angenähert. Es ist damit gelungen, einen Agenten, der selbst ohne Wissen einen Lernprozess beginnen kann, zu modellieren. Dem Lernmodell wurde die Fähigkeit gegeben, für sich selbst zu Lernen. Dies wurde durch die Umsetzung des Lernelementes mit generischen Eigenschaften erreicht. Durch diesen Ansatz ist es gelungen, ein Lernmodell als Ausgangsbasis für das verstärkende Lernen zu entwerfen.

Die Effizienz des Modells kann momentan nur theoretisch evaluiert werden, da es nicht gelungen ist, das Lernmodell zu testen. Die Gründe sind:

- Das zuerst eingesetzte Lernmodell hat sich bei der Klärung der Belohnungsfrage nicht als ausreichend genug erwiesen. Die Hierarchie innerhalb der Gebäudekomponenten sowie die bestehenden Abhängigkeiten wurden von dem Modell nicht berücksichtigt. Durch die wiederholte Analyse einer Zone wurden neue Erkenntnisse gewonnen, die zur Optimierung des ersten Modells geführt haben. Die Teste konnten aufgrund des späten Reenginering des Modells aus zeitlichen Gründen nicht mehr durchgeführt werden.
- Das vorbereitete Simulationswerkzeug war für den ersten Entwurf realisiert und muss entsprechend angepasst werden, um Teste auf dem Modell durchzuführen. Einige Eigenschaften der Umwelt konnten aus zeitlichen Gründen ebenso nicht mehr abgebildet werden, damit eine konsequente Bewertung des Modells erfolgen kann.

6.2 Einsatz von JADE im Gebäudemanagement

Auf der JADE-Plattform wurde das Gebäudemodell erfolgreich eingeführt. Die Zonen innerhalb des Gebäudes und das Gebäude selbst wurden jeweils einem Operator zugeteilt und in den zuständigen Containern untergebracht und realisieren somit ein Netzwerk von Agenten auf der Plattform. Die in Abschnitt 1.1 ausgewählte Definiton des intelligenten Gebäudes als *ein vernetztes und attentives Gebäude mit intelligenten und kommunizierenden Objekten mit der Fähigkeit zu Lernen* wurde durch die Kombination des auf der Basis des *Composite Pattern* entworfenen Gebäudemodells und der verteilten Eigenschaften der JADE-Plattform gewährt.

Durch den Einsatz des von JADE verwendeten Nachrichtenformates wurde ein einfaches Protokoll genutzt, das einen kontinuierlichen Ablauf des 3-Phasen-Zyklus ermöglicht. Die Fähigkeit, miteinander interagieren zu können, wurde dem Lernelement und dem Leistungselement (der Operator) gegeben und somit die Interaktion Agent-Umwelt für die Übernahme der Optimierungen realisiert.

JADE unterstützt die Entwicklung von fault-toleranten Agentensystemen, eine Eigenschaft,

die für die Steuerung des Gebäudes einen Vorteil darstellt, da mit einer geringen Ausfallrate zu rechnen ist. Die generische Eigenschaft des Lernelementes ist ebenfalls in Hinsicht auf die Programmleistung ein Vorteil, da JADE eine hohe Effizienz der Laufzeit unterstützt.

6.3 Reinforcement Learning im Gebäudemanagement

Es wurde im Rahmen dieser Arbeit ein Framework realisiert, welches auf JADE basiert. Die Grundlagen für eine Interaktion Agent-Umgebung wurden ebenfalls geschaffen. Auf diesem Stand könnte das Framework auf der Basis einer voreingestellten Handlungspolitik problemlos die Steuerung des Gebäudes übernehmen. Dieser Ansatz schreibt schon einen nicht vernachlässigbaren Aufwand vor, um Optimierungen zu ermöglichen. Zum einen könnte die Erstellung von Handlungspolitiken für inviduelle Haushalte einen hohen Zeit- und Programmieraufwand verursachen. Zum anderen müssten dann regelmäßige Aktualisierungen erfolgen, damit das System alle neuen Potentiale berücksichtigt.

Die ausgewählte Optimierungslösung (Reinforcement Learning (RL)) im Rahmen dieser Arbeit nutzt die Selbstanpassungsfähigkeit, die auf dem Lernen aus eigenen Erfahrungen basiert. Diese Lösung ist vielversprechend, jedoch sollte ihr Einsatz folgendes berücksichtigen:

- Das RL basiert auf einer Belohnungsphilosophie und erfordert in Bezug auf Menschen Kompromisse einzugehen.
- Der Einsatz des RL bringt ein Problem der Lernrate mit sich, das auf das Verhältnis zwischen Exploitation und Exploration zurückzuführen ist.
- Der Lernfortschritt könnte aufgrund seiner geringen Geschwindigkeit zu Effizienzproblemen führen.
- Der Aufwand bei der Konzeption der Belohnungsfunktionen könnte je nach der Anzahl und Art der gesteuerten Funktionsbereiche sehr groß ausfallen. Zusätzlich könnte die Verwendung von nicht geeigneten Belohnungsfunktionen zum Verlangsamen der Lernfortschritte führen oder die Motivation zu lernen verringern.

Eine Lösung wäre es, das Leistungselement mit einer voreingestellten Handlungsstrategie zu realisieren und das Lernelement samt seines Problemgenerators als eine nebenläufige Simulation umzusetzen. Das Leistungselement hätte somit die Möglichkeit, beim Ausführen einer Aktion zu prüfen, ob eine bessere Strategie angeboten wird. Der Simulationsprozess könnte weiterhin auch belohnt bzw. bestraft werden, je nachdem, ob seine Vorschläge an der Erfüllung der Systemziele mitwirken. Diese Lösung reduziert die durch den Einsatz des RL möglichen Risiken sowie den Aufwand beim Entwerfen einer nutzerabhängigen Belohnungsstrategie. Allerdings würde die Belohnung in diesem Fall von der Zufriedenheit des Leistungselementes mit dem Simulationsprozess abhängen und nicht mehr direkt von der Umwelt.

7 Zusammenfassung

Ziel dieser Arbeit war die Realisierung eines Frameworks, welches die Untersuchung der Verwendung von Agentensystemen zur Optimierung der Steuerung im Rahmen des Gebäudemanagements ermöglicht.

7.1 Fazit

Durch die Analyse eines intelligenten Gebäudes in Kapitel 3 ist es gelungen, die Systemziele im Rahmen des Gebäudemanagements zu spezifizieren. Unter der Berücksichtigung dieser Ziele wurde ein Modell des Hauses entworfen, welches eine Flexibilität in der Verwaltung des intelligenten Gebäudes bietet. Es ist weiterhin gelungen, die Steuerung des intelligenten Gebäudes als RL-Problem zu beschreiben und zu modellieren. Die Möglichkeit, das entworfene intelligente Gebäudemodell in einer realen Anwendung zu integrieren, wurde mit dem Beispiel seiner Kopplung an das CIM-Schema angesprochen, ein Vorschlag, der den Einsatz des WBEM ermöglicht und somit eine zusätzliche Flexibilität bei der Verwaltung von Geräten und Informationen bietet. Es ist gelungen, ein Lernmodell in Hinsicht auf die Optimierung der Steuerung im Gebäudemanagement zu entwerfen, das in der Kombination mit JADE und dem intelligenten Gebäudemodell Optimierungsmöglichkeiten nicht ausschließt.

Im Rahmen des intelligenten Gebäudemanagements wurden mögliche Einflussfaktoren, die sich auf den Betrieb des intelligenten Gebäudes auswirken können, erläutert. Diese Faktoren definieren zugleich die Art der Feststellung und der Ausnutzung von Optimierungspotentialen. Weiterhin wurde das Gewährleisten der Nutzerzufriedenheit als zentraler Aspekt zur Erfüllung der Systemziele vorausgesetzt. Unter der Berücksichtigung dieser Voraussetzung könnte sich ein direkter (realer) Einsatz des Reinforcement Learning zur Steuerung eines intelligenten Gebäudes auf die Nutzerzufriedenheit negativ auswirken. Allerdings benachteiligt das Lernen durch das Ausprobieren von Aktionen die Nutzerfreundlichkeit und / oder erhöht die Verbrauchkosten. Eine Strategie zur Behandlung von Kostenintensiven Aktionen könnte als eine Alternative betrachtet werden, sie löst damit jedoch einen Teil der Lernaufgabe im Vorwege mit dem Risiko, unerlaubt das Wissen mit dem Lernenden zu teilen. Dieser Ansatz begrenzt offensichtlich entweder den Einsatz des Reinforcement Learning oder verletzt die Definiton des Problems des Reinforcement Learning.

7.2 Ausblick

Die Komplexität bzw. die Dimension des Problems und mehr noch die Unsicherheiten bei der Ermittlung einer optimalen Exploitationsrate haben beim Einsatz des Reinforcement Learning zur Optimierung der Steuerung im Gebäudemanagement Grenzen aufgezeigt. Bei sehr komplexen Umgebungen ist es eine Überlegung wert, ein Verfahren einzusetzen, das die Dimension des Problems in so kleinem Umfang wie möglich, ohne Änderung der Systemziele selbst, reduziert. Hierzu existieren bekannte wissenschaftliche Verfahren, so wäre z.B der Einsatz der *Tree Decomposition Methods* denkbar.

Um eine optimale Exploitationsrate zu ermitteln, könnten mehrere Lernelemente mit der gleichen Aufgabe betreut werden und deren Ergebnisse in Abhängigkeit ihrer unterschiedlichen Exploitationsrate verglichen werden. Dabei bleibt zu untersuchen, in welcher Art und Weise dieser Vergleich vorgenommen wird.

Eine weitere Überlegung wäre es, ein Framework zur Generierung von Belohnungsstrategien auf der Basis von Nutzerverhaltenmodellen oder -präferenzen zu implementieren, das von einer, falls nötig, Weiterentwicklung des entworfenen Lernmodells eingesetzt wird.

Anhang

Literaturverzeichnis

[siemens 2008] : *Normung der Gebäudeautomation*. 2008. – URL https://www.cee.siemens.com/web/austria/de/industry/bt/systeme/gebaeudeautomation/systemintegration/offene/Pages/normunggebaeudeautomation.aspx. – Zugriffsdatum: 20-01-2014

[iba 2013] : *Hybrid House*. 2013. – URL http://www.iba-hamburg.de/projekte/bauausstellung-in-der-bauausstellung/hybrid-houses/hybrid-house/projekt/hybrid-house.html. – Zugriffsdatum: 18.12.2013

[jcp 2013] : *WBEM Services Specification*. 2013. – URL https://jcp.org/en/jsr/detail?id=48. – Zugriffsdatum: 18.12.2013

[dmtf 2014] : *Distributed Management Task Force, Inc.* 2014. – URL http://dmtf.org/. – Zugriffsdatum: 15.01.2014

[fipa 2014] : *The Foundation for Intelligent Physical Agents*. 2014. – URL http://www.fipa.org/. – Zugriffsdatum: 15.01.2014

[Bellifemine u. a. 2007] BELLIFEMINE, Fabio ; CAIRE, Giovanni ; GREENWOOD, Dominic: *Developing multi-agent systems with JADE*. 2. Aufl. Liverpool : Wiley, 2007. – ISBN 978-0-470-05747-6

[Boekhoven 2011] BOEKHOVEN, Patrick: *Entwicklung eines Reinforcement Learning Frameworks auf Basis eines Agentensystems*. Departement Informatik, Hochschule für angewandte Wissenschaften Hamburg, Bachelor Thesis, 2011. – URL http://edoc.sub.uni-hamburg.de/haw/volltexte/2011/1269/. – Zugriffsdatum: 20.01.2014

[Bull und Kovacse 2005] BULL, Larry ; KOVACSE, Tim: *Foundation of Learning Classifier Systems*. 1. Aufl. Berlin · Heidelberg · New York : Springer, 2005. – ISBN 3-540-25073-5

[Forbrig 2007] FORBRIG, Peter: *Objektorientierte Softwareentwicklung mit UML*. 3. Aufl. München : Hanser, 2007. – ISBN 978-3-446-40572-1

[Fraden 2010] FRADEN, Jacob: *Handbook of Modern Sensors*. 4. Aufl. Dordrecht · Heidelberg · London · New York : Springer, 2010. – ISBN 978-1-4419-6465-6

[Gilbert 2008] GILBERT, Nigel: *Agent-based Models*. 1. Aufl. California : Sage Publications, Inc., 2008. – ISBN 978-1-4129-4964-4

[Harper 2003] HARPER, Richard: *Inside the Smart Home*. 1. Aufl. London · Berlin · Heidelberg · New York · Hong Kong · Milan · Paris · Tokyo : Springer, 2003. – ISBN 1-85233-688-9

[Heusinger 2004] HEUSINGER, Winfried: *Das Intelligente Haus Entwicklung und Bedeutung für die Lebensqualität*. 1. Aufl. frankfurt am Main · Berlin · Bern · Bruxelles · New York · Oxford · Wien : Peter Lang, 2004. – ISBN 3-631-53616-X

[Hobbs 2004] HOBBS, Chris: *A Practical Approach to WBEM/CIM Management*. 1. Aufl. Boca Raton · London · New York · Washington, D.C. : Auerbach Publications, 2004. – ISBN 0-8493-2306-1

[Kretschmer 2009] KRETSCHMER, Henri: *Ein Modellbasierter autarker Anwesenheitsschätzer für die benutzeradaptive Gebäudeautomatisierung im Heimbereich*. Fakultät IV - Elektrotechnik und Informatik, Technische Universität Berlin, PhD Thesis, 2009

[Mozer 1999] MOZER, Michael C.: An intelligent environment must be adaptive. In: *IEEE Intelligent Systems and their Applications* (1999), S. 11–13. – URL http://www.cs.colorado.edu/~mozer/Research/Selected%20Publications/ieee.html. – Zugriffsdatum: 20-01-2014

[Mozer 2005] MOZER, Michael C.: Lessons from an Adaptive house. In: *D. Cook & R. Das (Eds.), Smart environments: Technologies, protocol and applications* (2005), S. 273–294. – URL http://www.cs.colorado.edu/~mozer/Research/Selected%20Publications/reprints/smart_environments.pdf. – Zugriffsdatum: 20-01-2014

[Padgham und Winikoff 2004] PADGHAM, Lin ; WINIKOFF, Michael: *Developing intelligent agent systems*. 1. Aufl. West Sussex : Wiley, 2004. – ISBN 0-470-86120-7

[Russell und Norvig 2003] RUSSELL, Stuart J. ; NORVIG, Peter: *Artificial Intelligence, A Modern Approach*. 2. Aufl. New Jersey : Prentice Hall, 2003. – ISBN 0-13-080302-2

[Sutton und Barto 1998] SUTTON, Richard S. ; BARTO, Andrew G.: *Reinforcement Learning: An Introduction*. Cambridge · London : The MIT Press, 1998. – ISBN 0-262-19398-1

[VDI-Gesellschaft 2003] VDI-GESELLSCHAFT: *Gebäudeautomation¯Voraussetzung für das Gebäudemanagement*. 1. Aufl. Düsseldorf : VDI Verlag, 2003. – ISBN 3-18-091740-7

[Wang 2010] WANG, Shengwei: *Intelligent Building and Building Automation*. 1. Aufl. London · New York : Spon Press, 2010. – ISBN 978-0-415-47571-6

[Wenzel 2011] WENZEL, Markus: *Aspekte des Web-Based Enterprise Management (WBEM) im Energiemanagement*. Departement Informatik, Hochschule für angewandte Wissenschaften Hamburg, Bachelor Thesis, 2011. – URL http://edoc.sub.uni-hamburg.de/haw/volltexte/2011/1197/. – Zugriffsdatum: 20.01.2014

[Wolter 2008] WOLTER, Anne: *Reinforcement Learning in der Roboter-Navigation*. 1. Aufl. Saarbrücken : VDM, 2008. – ISBN 978-3-639-04702-8

[Wooldridge 2002] WOOLDRIDGE, Michael: *An Introduction to MultiAgent Systems*. 1. Aufl. West Sussex : Wiley, 2002. – ISBN 978-0-471-49691-5

Listings

2.1 Sarsa. (Sutton und Barto, 1998, 146) . . . 15
2.2 Q-Learning. (Sutton und Barto, 1998, 149) . . . 15
2.3 R-Learning. (Sutton und Barto, 1998) . . . 17

5.1 Umgebung-Klasse in MOF. . . . 52
5.2 Definition einer Zone in MOF. . . . 52
5.3 Assoziation der Klassen SCF_SimpleWorld und SCF_ComplexWorld. . . . 52
5.4 Vererbung der Eigenschaften der Klasse CIM_ManagedSystemElement . . . 53
5.5 Konstruktor der Klasse ComposedWorld (Zone) . . . 53
5.6 Registrierung der Provider . . . 53
5.7 Initializierung der Umgebung; der Operator wird dabei gestartet. . . . 55
5.8 Veröffentlichung der Operator-Dienste. . . . 55
5.9 Konstruktor der Klasse Problem. . . . 58
5.10 Eingabedatei eines Gebäudemodells; das Modell wird in XML-Format eingegeben 61

Stichwortverzeichnis

Agent, 4
Agentenkonzept, 4
Agentensystem, 18
Autonomie, 2

Belohnung, 9
Belohnungsfunktion, 40

Container, 53

Explorationsrate, 50

Gebäudemanagement, 19

Lernelement, 50
Lernfortschritt, 50
Linux, 63

Middleware, 37
Modell, 47

Rechnersystem, 4

Sensor, 19
    Anwesenheitssensor, 25
    Helligkeitssensor, 25
Spezialisierung, 47

Umwelt, 4

Zone, 19
Zustandsübertragungsfunktion, 40

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 10. April 2014	
Ort, Datum	Unterschrift