Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Jan Jennrich

Interpretation sensorbasierter Daten in einer Smart-Home-Umgebung

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jan Jennrich

Interpretation sensorbasierter Daten in einer Smart-Home-Umgebung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Gunter Klemke

Eingereicht am: 29. Juli 2014

Jan Jennrich

Thema der Arbeit
Interpretation sensorbasierter Daten in einer Smart-Home-Umgebung

Stichworte
Smart Home, Smart Environment, Sensordaten, Interpretation, Plattform, Living Place Hamburg, Disappearing Computing

Kurzzusammenfassung
In dieser Arbeit wird eine Plattform zur Interpretation von sensorbasierten Daten entworfen. Als Beispiel für eine Smart-Home-Umgebung dient dabei der Living Place Hamburg. Es werden Verfahren zur Visualisierung und Verarbeitung von Messdaten vorgestellt, Anforderungen definiert und in einem erweiterbaren Systemdesign umgesetzt.

Jan Jennrich

Title of the paper
Interpretation of sensor-based data in a smart home environment

Keywords
smart home, smart environment, sensor data, interpretation, platform, Living Place Hamburg, disappearing computing

Abstract
In this paper, a platform for the interpretation of sensor-based data is designed. The Living Place Hamburg serves as an example of a smart home environment. Methods for visualising and processing measurements are presented, requirements are defined and implemented in an extensible system design.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Smartphones, Smartcards, Smart Meters, Smart Homes – Dinge als *Smart* zu bezeichnen ist ein Trend, der sich anhaltender Beliebtheit erfreut. Ein großer IT-Dienstleister spricht gar vom Smarter Planet [IBM (2008)].

Smartcards gehören zu den frühen Vertretern dieser Kategorie und sind Plastikkarten mit eingebetteten Mikroprozessoren. Auch wenn diese Karten, für sich allein genommen, zunächst wenig interessant erscheinen mögen, besitzen sie das entscheidende Merkmal, das für Dinge mit diesem Namenszusatz symptomatisch ist: einen integrierten Computer.

Mag dessen Anwesenheit bei einem *Smartphone* offensichtlich sein, sieht man einem *Smart Meter*, also etwa einem „intelligenten“ Strom- oder Wasserzähler, diese Eigenschaft nicht unbedingt an.

Smart Homes, oder allgemeiner *Smart Environments*, besitzen nicht bloß einen Computer, sondern eine Vielzahl von Sensoren und Aktoren sowie ein Netzwerk, das all die einzelnen Bestandteile zu einem Gesamtsystem verbindet.

Smart bedeutet schließlich, auf dieser Grundlage die Umgebung wahrzunehmen, zu interpretieren und anhand der gewonnenen Erkenntnisse adäquat zu reagieren. An der Stelle setzt diese Arbeit an.

1.2 Gliederung der Arbeit

In der *Analyse* (Kapitel 2) wird zunächst über den etwas allgemeineren Begriff des *Disappearing Computing* (2.1) an das grundlegende Themengebiet dieser Arbeit, *Smart Environments*, herangeführt. Weiter geht es über den *Kontext*-Begriff (2.2) zu einer Beschreibung des *Living Place Hamburg* (2.3), der als Inspiration und dessen Architektur als Grundlage für diese Arbeit dient.

Im Unterkapitel *Von Low-Level zu Higher-Level-Sensoren* (2.4) wird gezeigt, wie aus *quantitativen* Daten *qualitative Aussagen* generiert werden können. Es folgen einige *Anwendungsfälle* (2.5) und das *Fazit* (2.6) der Analyse.

In Kapitel (3) geht es um *Verfahren*, die zur *Verarbeitung von Messdaten* (3.2) und *Visualisierung* (3.3) eingesetzt werden.

Im Kapitel *Systemdesign* (4) werden zunächst *Anforderungen* (4.1) an eine Plattform zur Interpretation von Sensordaten definiert. Nach Details zu *Architektur* (4.2) und *Persistenz* (4.3) wird im Unterkapitel *Plattform* (4.4) auf das Konzept eines *virtuellen Sensors* und die *Verarbeitung von Interpretationsaufträgen* eingegangen.

Schließlich werden im letzten Kapitel (5) die Ergebnisse der Arbeit noch einmal zusammengefasst und ein Ausblick formuliert.

2 Analyse

2.1 Disappearing Computing und Smart Environments

Die Idee, Wohnräume mit intelligenter Technik auszustatten, ist nicht neu. Eine der wegweisenden Veröffentlichungen in dem Bereich, „The Computer for the 21st Century“ [Weiser (1991)], wurde bereits im Jahr 1991 von Mark Weiser im Rahmen seiner Arbeit am Xerox Palo Alto Research Center veröffentlicht.

Erfolgreiche Technologien werden, wenn sie sich durchgesetzt haben, oft nicht mehr als solche wahrgenommen, weil sie ein Teil des alltäglichen Lebens und damit praktisch unsichtbar geworden sind. Als Beispiel dafür nennt Weiser die Schrift, die, obwohl sie einen wesentliche Rolle in der Entwicklung der modernen Welt gespielt hat, nicht mehr als Technologie wahrgenommen wird:

> „Consider writing, perhaps the first information technology. [...] Today this technology is ubiquitous in industrialized countries.“ [Weiser (1991)]

Im Gegensatz dazu sieht er die Informationstechnik, die zu schwer zugänglich sei, um sich erfolgreich in den Alltag integrieren zu können:

> „Silicon-based information technology, in contrast, is far from having become part of the environment. [...] It is approachable only through complex jargon that has nothing to do with the tasks for which which people use computers.“ [Weiser (1991)]

Dabei sollte allerdings berücksichtigt werden, dass der zitierte Artikel bereits im Jahr 1991 erschienen ist. Doch obwohl die Verwendung von Computern inzwischen tatsächlich allgegenwärtig geworden ist, etwa dank der immer größeren Verbreitung von informationstechnischen Geräten wie Smartphones und Tablets, steht der Durchbruch im Wohnbereich noch aus.

Um das Konzept von Smart Living bzw. Smart Homes in der Breite etablieren zu können, müssen die eingesetzten Technologien also zunächst so weit entwickelt sein, dass sie nicht

mehr als solche auffallen, sondern sich harmonisch in das Gesamtbild einfügen und so scheinbar „verschwinden“.

> „[...] only when things disappear in this way are we freed to use them without thinking [...]“ [Weiser (1991)]

Als Beispiel für das Verschwinden von Technik nennt er die zahlreichen Elektromotoren in Autos, deren Vorhandensein der Fahrer sich zwar bewusst machen kann, aber nicht muss, um ihre Funktionen nutzen zu können:

> „[Motors and solenoids] start the engine, clean the windshield, lock and unlock the doors, and so on. By paying careful attention the driver might be able to discern whenever he or she activated a motor, but there would be no point to it.“ [Weiser (1991)]

Schließlich sagt er bereits eine Entwicklng voraus, die eine Grundlage für Smart Environments sein wird, nämlich die Allgegenwärtigkeit von Computern und die Vernetzung derselben untereinander:

> „Already computers in light switches, thermostats, stereos and ovens help to activate the world. These machines and more will be interconnected in a ubiquitous network.“ [Weiser (1991)]

2.2 Kontext

Im Zusammenhang mit Smart Environments wird gerne der Begriff *Kontext* verwendet. Anind K. Dey und Gregory D. Abowd definieren diesen Begriff in ihrem Artikel „Towards a Better Understanding of Context and Context-Awareness“ wie folgt:

> „Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.“ [Abowd u. a. (1999)]

Als Kontext wird hier die Menge der Informationen bezeichnet, die den Zustand der Umgebung beschreiben, in der sich der Nutzer eines Systems aufhält und mit der er interagiert.

Angewendet auf das Thema dieser Arbeit kann also in Smart Homes mithilfe von Sensoren, die Auskunft über den Zustand der Wohnung und des Benutzers geben, zusammen mit Wissen über den Aufbau und die Gegebenheiten der Wohnung, der Kontext ermittelt werden.

2.3 Living Place Hamburg

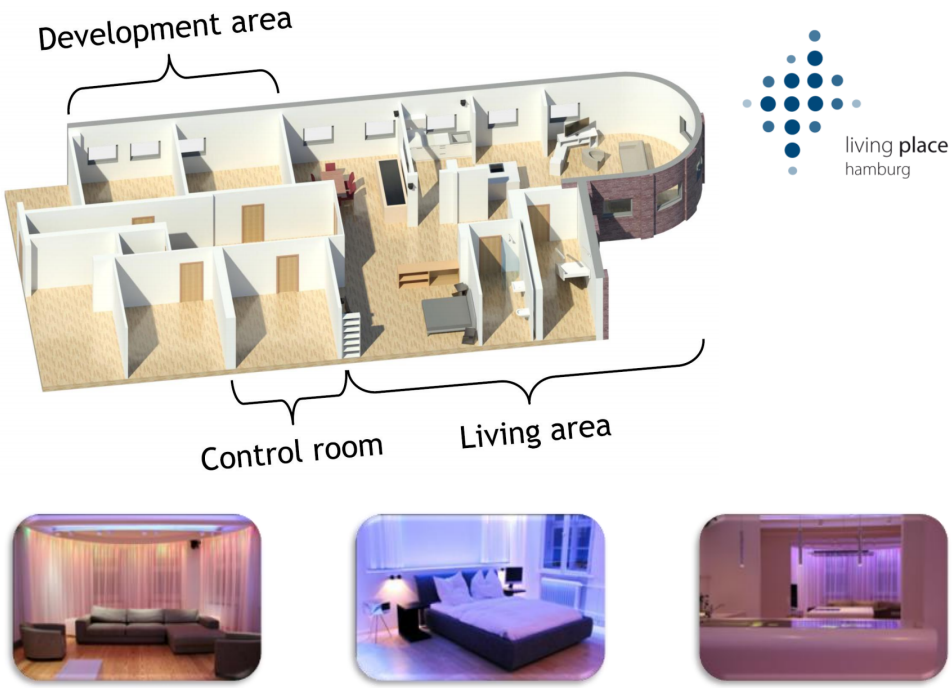

Abbildung 2.1: Schematische Darstellung des Living Place inklusive Wohnbereich sowie Räumlichkeiten für Entwicklung und Kontrollraum [Voskuhl (2011)]

Eines der Kernthemen dieser Arbeit sind Smart Homes. Dabei orientiert sie sich an einem konkreten Projekt, dem Living Place Hamburg.

Der Living Place ist ein Labor der Hochschule für Angewandte Wissenschaften Hamburg, das eingerichtet ist wie eine Einpersonenwohnung und sich zum Ziel gesetzt hat, das Wohnen der Zukunft zu erforschen. Es bietet Raum für Projekte aller Art im Bereich des intelligenten Wohnens, die in Zusammenarbeit unterschiedlicher Disziplinen wie Architektur, Lichtdesign, Interaktionsdesign und Informatik [Ellenberg u. a. (2011)] entstehen.

Ziel ist es, den Nutzer, also den fiktiven Bewohner der Wohnung, bei seinem Tagesablauf so gut wie möglich zu unterstützen. Einige der Komponenten arbeiten im Hintergrund, ohne dass er sich um ihre Aktivierung oder Steuerung Gedanken zu machen braucht, allerdings sollten diese sich zurückhaltend genug verhalten, um ihm nicht das Gefühl zu geben, bevormundet zu werden. Andere können bewusst von ihm genutzt oder in ihrer Funktion beeinflusst werden. Dabei sollten sie in ihrer Bedienung ansprechend und einfach sein, um den Bewohner zur Nutzung zu motivieren, ohne ihn zu überfordern. Eine der größten Herausforderungen ist es, die richtige Balance zwischen Automatisierung und Bedienung durch den Nutzer zu finden.

Einer der Schwerpunkte der Forschung am Living Place ist es, herauszufinden, von welchen Erweiterungen Menschen in ihren Wohnungen profitieren könnten, und diese zu entwickeln sowie die Interaktion des Nutzers mit der Wohnung zu untersuchen. Dazu dient auch der Kontrollraum, von dem aus die Wohnung gesteuert und überwacht werden kann.

2.3.1 Infrastruktur

Abbildung 2.2: Laborsensorik im Wohnbereich des Living Place Hamburg: [1] Positionserkennung [2] Kameras [3] Kabelloses Sensornetzwerk [4] Multitouch-Tresen SiBar [5] Mikrofone [6] Couchsensorik [7] Intelligentes Bett [8]Sturzerkennung [9] Multitouch Tisch [Otto (2013)]

Die Wohnung ist ausgestattet mit einer Vielzahl von Sensoren (vgl. Otto (2013) und Pautz (2012) sowie Abbildung 2.2):

- Schwenkbare sowie 360-Grad-Fischaugen- *Kameras*
- *Richtmikrofone*
- Sensoren zur Erkennung der *Position* spezieller Tags innerhalb der Wohnung
- *Schlafphasenerkennung* mittels im Bett eingebauter Dehnungsmessstreifen
- *Temperatursensoren*
- *Helligkeitssensoren*

- Sensoren zur Ermittlung des *Kippstands der Fenster*
- Sensoren zur *Sitzerkennung* in der Couch
- Sensoren zur *Sturzerkennung* im Teppich

Die Sensoren für Temperatur, Helligkeit und Kippstand der Fenster sind Teil eines Sensornetzwerks, das Alexander Pautz im Rahmen seiner Masterarbeit entwickelt hat: „In jedes der fünf Zimmer wird ein Sensorknoten mit Temperatur- und Helligkeitssensoren installiert. Zusätzlich werden vier Fenster der Wohnung überwacht, ob diese geöffnet oder geschlossen sind." [Pautz (2012)].

2.3.2 Architektur

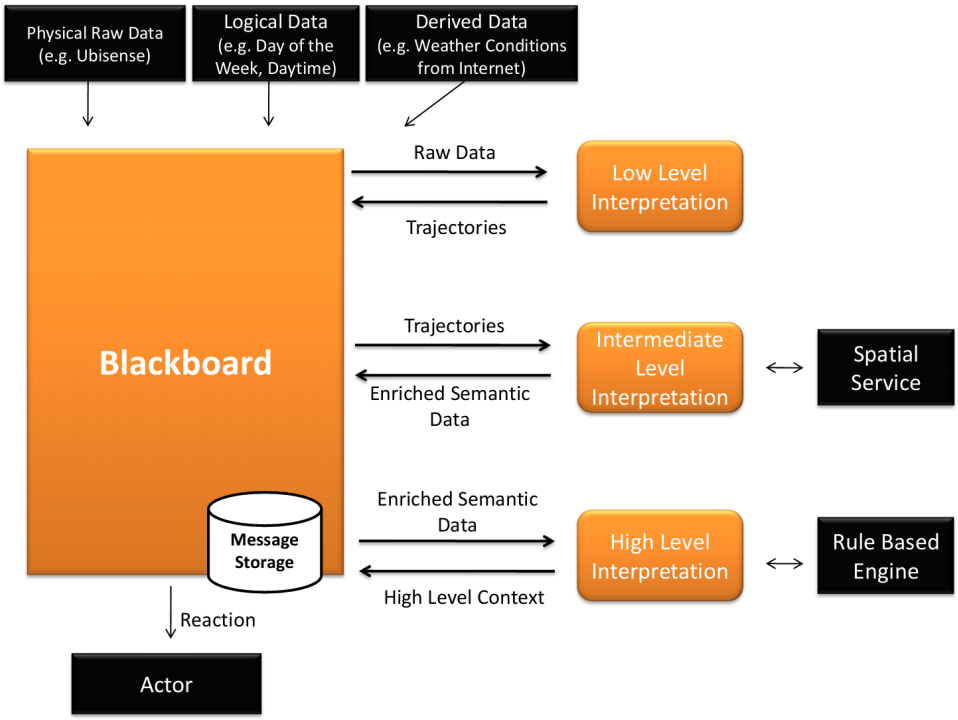

Abbildung 2.3: Blackboard-Architektur [Voskuhl (2011)]

Den Kern der im Living Place vorgefundenen Architektur bildet ein *Blackboard-System* (Abb. 2.3), das durch den Message-Broker ActiveMQ implementiert wird, über den ein Großteil der

Kommunikation läuft. Die angebrachten Sensoren und Aktoren senden und empfangen ihre Daten in der Regel über den ActiveMQ. Eine Ausnahme stellen Audio- und Videodatenströme von Mikrofonen und Kameras dar, da die von ihnen erzeugten Datenmengen zu groß sind und den Message-Broker unnötig stark belasten würden.

Die Nutzung eines *Message-Brokers* als zentralen Knotenpunkt hat den Vorteil, dass die anderen Blackboard-Teilnehmer sich untereinander nicht kennen beziehungsweise finden können müssen. Um diese Art der Kommunikation zu realisieren, werden zwei Mechanismen benutzt, das Producer-Consumer- und das Publish-Subscribe-Pattern.

Producer-Consumer (Abb. 2.4) wird vom ActiveMQ durch *Queues* realisiert, über die mittelbar eine Punkt-zu-Punkt-Verbindung (1:1) zwischen Sender (Produzent) und Empfänger (Konsument) besteht.

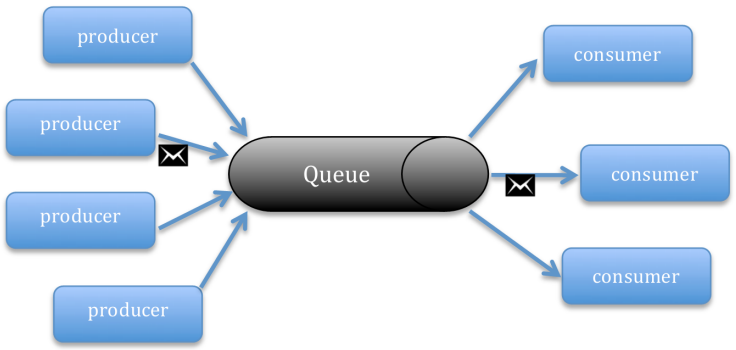

Abbildung 2.4: Producer-Consumer-Pattern [Bornemann (2013)]

Bei *Publish-Subscribe* (Abb. 2.5) gibt es ebenfalls einen Sender (Publisher), aber mehrere Empfänger (Subscriber), die sich auf ein vom ActiveMQ verwaltetes *Topic* registrieren können, so dass jeder Empfänger alle Nachrichten erhält, die an dieses Topic gesendet werden. Somit besteht eine 1:n-Verbindung zwischen einem Publisher und mehreren Subscribern.

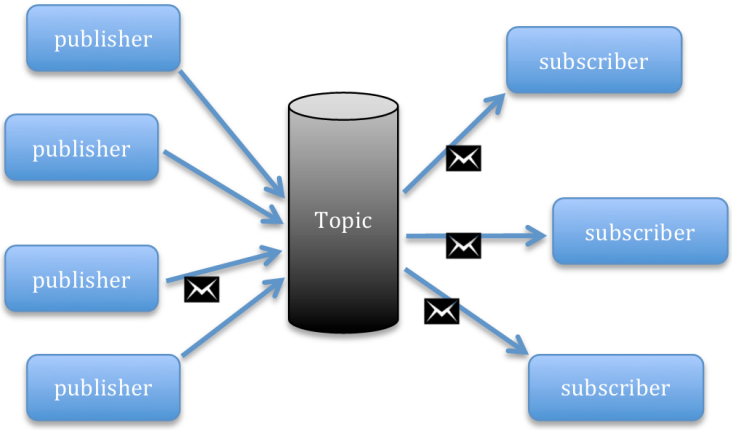

Abbildung 2.5: Publish-Subscribe-Pattern [Bornemann (2013)]

Als externe Persistenz wird im Living Place neben dem ActiveMQ die dokumentenorientierte Datenbank MongoDB eingesetzt. Für die Nutzung von ActiveMQ und MongoDB kann ein Living-Place-spezifischer Wrapper verwendet werden, der parallel auf beide Systeme schreibt.

Das verwendete Format für Nachrichten ist die JavaScript Object Notation (JSON). Der wesentliche Vorteil von JSON ist die breite Unterstützung, die das Format erfährt, da für viele Programmiersprachen entsprechende Bibliotheken zur Verfügung stehen. (Vgl. [Otto (2013)])

2.4 Von Low-Level zu Higher-Level-Sensoren

2.4.1 Von Quantität zu Qualität

Herkömmliche Sensoren, wie sie auch in Smart Homes eingesetzt werden, erfassen zunächst sehr einfache (grundlegende) Daten. So können beispielsweise gemessene Temperatur- oder Helligkeitswerte direkt in Grad Celsius oder Lumen wiedergegeben werden. Die Aussagen, die unmittelbar mittels solcher Daten getroffen werden, sind quantitative Aussagen.

Beispiel: „*Die Temperatur beträgt 21 Grad Celsius.*“

Quantitative Aussagen sind dazu geeignet, sehr präzise Auskunft über bestimmte Parameter, die die Situation ausmachen, zu geben. Soll jedoch die Situation an sich beurteilt werden,

können Qualitäten aussagekräftiger sein. Qualitative Aussagen, die auf Grundlage solcher Daten getroffen werden könnten, sind etwa

„*Es ist warm.*“,

was eine Einteilung der Temperaturskala in kalt / mittel / warm usw. voraussetzen würde.

Wird ein Reihe von mehreren Messwerten über die Zeit abgeleitet, ergeben sich neue Quantitäten, in diesem Fall der Verlauf der Temperaturkurve. Qualitative Aussagen, die daraufhin getroffen werden können, wären etwa

„*Es wird wärmer.*“ oder

„*Die Raumtemperatur fällt stark ab.*“.

2.4.2 Unterschiedliche Arten von Messwerten und Sensoren

Viele der Parameter, die eine Situation ausmachen, sind messbar. Dabei können die gemessenen Werte von unterschiedlicher Art sein. Ebenso ist die Art des Rückgabewerts ein Merkmal, nach dem sich verschiedene Arten von Sensoren unterscheiden lassen.

Dabei werden nicht nur herkömmliche Sensoren, wie etwa Temperaturfühler, sondern alle Bauteile, die eine Auskunft über einen Zustand geben, als Sensor verstanden. So schließt die hier verwendete Definition auch etwa Lichtschalter mit ein. Arten von Messwerten, wie sie in Smart Homes vorkommen, können zum Beispiel sein:

Art	Vorkommen
Binär (An/Aus, Ja/Nein)	• Lichtschalter • Tür auf / zu
Abgestuft	• Regler, der ganzzahlige Werte von 1 bis 5 annehmen kann
Stufenlos	• Temperatur • Helligkeit
Mehrdimensional	• Positionsdaten von Personen oder Gegenständen
Komplex	• Kamerabild

2.4.3 Sensorinterpretation

Durch das Aneinanderreihen von (eindimensionalen) Messwerten ergeben sich Kurven, über die sich verschiedene generelle Aussagen treffen lassen.

- Steigt / fällt / stagniert + Grad der Steigung
- Werte ober- / inner- / unterhalb eines definierten Bereichs
- im Normbereich
- Oszillation
- Bsp: Heizung heizt, System stellt zu hohe Temperatur fest, Klimaanlage wird angeschaltet (zu stark), Temperatur fällt, System stellt zu niedrige Temperatur fest, Klimaanlage wird heruntergefahren, Temperatur steigt wieder ⇒ Regel müsste Temperatur auf festen Wert einstellen
- Entsprechend kann eine Oszillation im kleineren Rahmen auftreten, wenn etwa das vom Hardware-Sensor erzeugte Signal verrauscht ist.

Beispiele

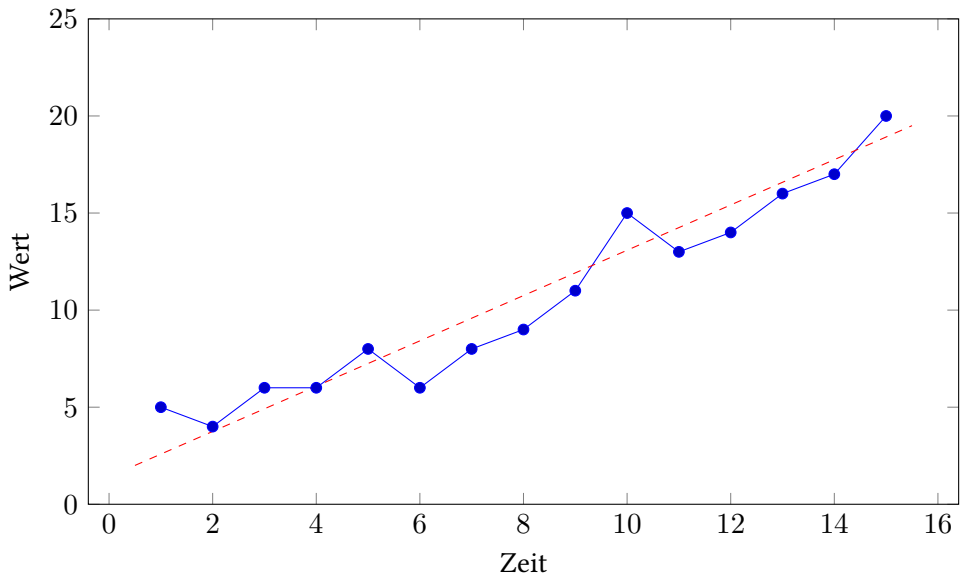

Abbildung 2.6: Steigung

Abbildung 2.6 zeigt einen Verlauf, der insgesamt als steigend bezeichnet werden kann.

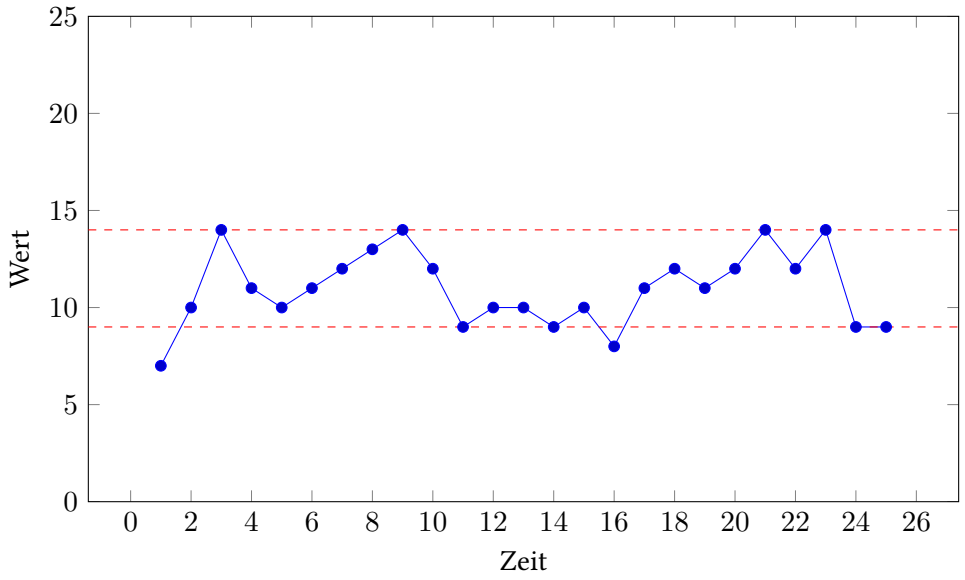

Abbildung 2.7: Korridor

In Abbildung 2.7 befinden sich die Werte hauptsächlich innerhalb eines *Korridors*.

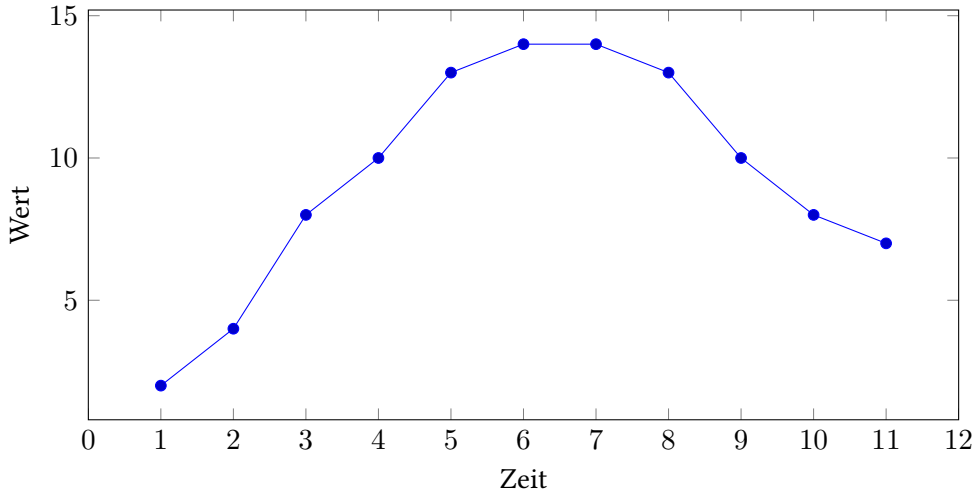

Abbildung 2.8: Beispielhafter Verlauf

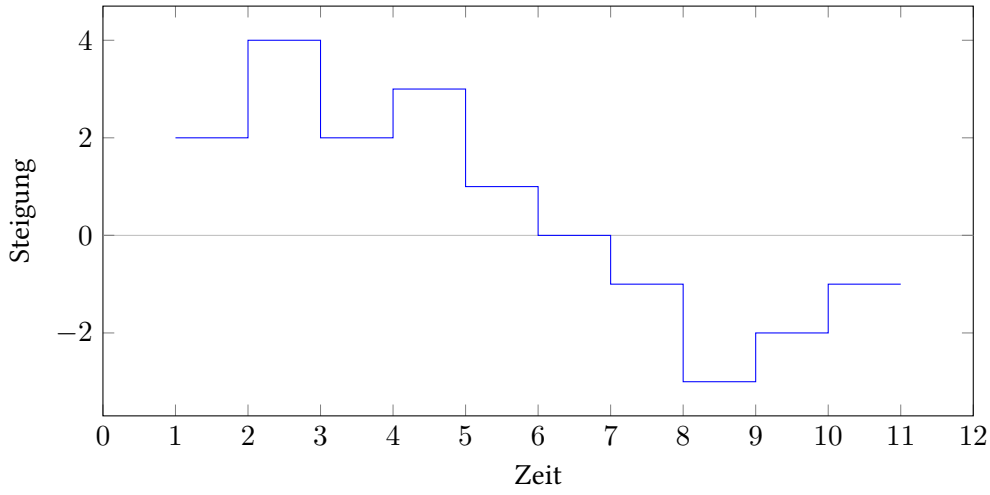

Abbildung 2.9: Steigung zwischen Werten aus Abbildung 2.8

Abbildung 2.8 zeigt einen Verlauf, bei dem der Wert erst langsam, dann schneller und wieder langsamer ansteigt, um nach Erreichen eines Scheitelpunktes wieder abzufallen. Um eine entsprechende Aussage maschinell zu treffen, können anstatt der tatsächlichen Werte die jeweiligen Steigungen betrachtet werden, wie Abbildung 2.9 veranschaulicht. Für den Verlauf der Steigung wurde eine rechteckige Form gewählt, da zwischen zwei Messwerten keine Aussage über den Verlauf getätigt werden kann.

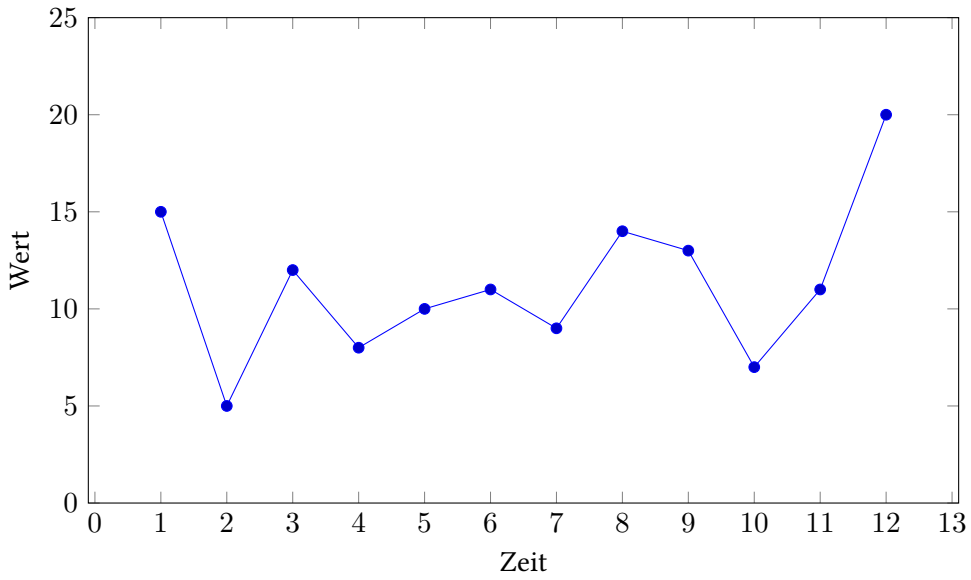

Abbildung 2.10: Oszillation

Die Werte in Abbildung 2.10 *oszillieren* relativ stark, je nach Anwendung können Aussagen über Varianz, Mittelwert und Breite des Korridors von Interesse sein.

2.5 Anwendungsfälle

2.5.1 Temperatur, Beleuchtung, Position

In Kapitel 2.4 wurde beschrieben, welche Aussagen sich grundsätzlich anhand von Sensordaten treffen lassen. Darauf aufbauend lassen sich einige beispielhafte Anwendungsfälle definieren:

Raumtemperatur

- Innentemperatur steigt über akzeptablen Bereich, Außentemperatur ist niedriger
  ⇒ Fenster öffnen, Heizungsventil schließen
- Innentemperatur steigt über akzeptablen Bereich, Außentemperatur ist geringer
  ⇒ Fenster schließen, Heizungsventil schließen, eventuell Klimaanlage einschalten

- Innentemperatur sinkt unter akzeptablen Bereich
  ⇒ Fenster schließen, Heizungsventile öffnen

Beleuchtung (wenn Bewohner anwesend)

- In der Wohnung ist es zu dunkel, außen ebenfalls
  ⇒ Licht an
- In der Wohnung ist es zu hell, außen ebenfalls
  ⇒ Fenster abdunkeln
- In der Wohnung ist es zu hell, außen dunkel
  ⇒ Innenbeleuchtung reduzieren

Indoor Positioning

- Person verlässst Raum A und betritt Raum B
  ⇒ Beleuchtung in Raum A herunterregeln, in Raum B erhöhen
- Person sitzt auf der linken oder rechten Seite des Sofas, es gibt einen Lesemodus für die Beleuchtung
  ⇒ Deckenbeleuchtung links oder rechts ausrichten, gegebenenfalls Leselampe zuschalten

2.5.2 Andere Regelsysteme

Ein weiterer Anwendungsfall sind Regelsysteme wie in [Kantak (2013)] beschrieben, die bereits qualitative Aussagen voraussetzen. So werden dort in einer Simulation Sensoren verwendet, die Werte zurückgeben, die nur durch eine vorhergehende Interpretation qualitativer Sensordaten bestimmt werden können, wie beispielsweise der simulierte Sensor „location“:

> „Der Sensor location liefert den Raum, in welchem sich der Bewohner aktuell befindet. Dies könnte zum Beispiel durch eine Kombination aus [Indoor-Positioning-System] und der Einteilung der Wohnung in semantische Bereiche, welche über die Positionskoordinate determiniert werden können, erreicht werden.“ [Kantak (2013)]

2.6 Fazit

Um die in diesem Kapitel vorgestellten Möglichkeiten als Entwickler einfach und komfortabel nutzen zu können, ist eine einheitliche Schnittstelle erforderlich. Eine Plattform, die das bietet, wird im weiteren Verlauf dieser Arbeit entwickelt und vorgestellt werden. Das ein System dieser Art notwendig ist, wurde bereits in [Otto (2013)] treffend formuliert:

> „Was bei der Kommunikation fehlt, ist eine zentrale Stelle, an der die Aktionen und Informationen gebündelt zur Verfügung stehen. Es gibt keine Möglichkeit, ein Gesamtbild der Informationen zu interpretieren, um anhand der dadurch gewonnen Erkenntnisse eine Aktion auszuführen. Gerade bei der Kommunikation auf niedrigen Abstraktionsleveln, zum Beispiel bei Sensordaten oder Befehlen für Aktoren, ist es wichtig zu wissen, ob eine Aktion schon ausgeführt wird oder ob ein anderes Projekt um eine exklusive Ressource konkurriert. Die Rohdaten werden zwar alle über den ActiveMQ vermittelt, aber es gibt keine Möglichkeit diese interpretieren zu lassen, um im Zusammenhang Schlüsse zu ziehen.“ [Otto (2013)]

3 Verfahren

3.1 Allgemein

Generell soll die Plattform, die in dieser Arbeit entwickelt wird, es für andere Entwickler vereinfachen, eigene Interpretationsplugins zu erstellen (siehe dazu auch 4.2.1). In diesem Kapitel wird darauf eingegangen, welche Verfahren für die Entwicklung eines beispielhaftes Plugins benötigt bzw. angewendet werden.

3.2 Verarbeitung von Messdaten

Messdaten liegen ursprünglich in Form von einzelnen Punkten vor. Um Aussagen über die Daten treffen zu können, ist es hilfreich, zunächst eine Funktion zu bestimmen, die den Verlauf beschreibt. Die einzelnen Messpunkte werden auch bei gleichmäßigem Verlauf selten auf einer exakten Geraden liegen, da Sensoren und zu messende Größen Schwankungen unterliegen können. Die Funktion, anhand der eine Aussage getroffen werden kann, muss also zunächst mithilfe mathematischer Verfahren approximiert werden.

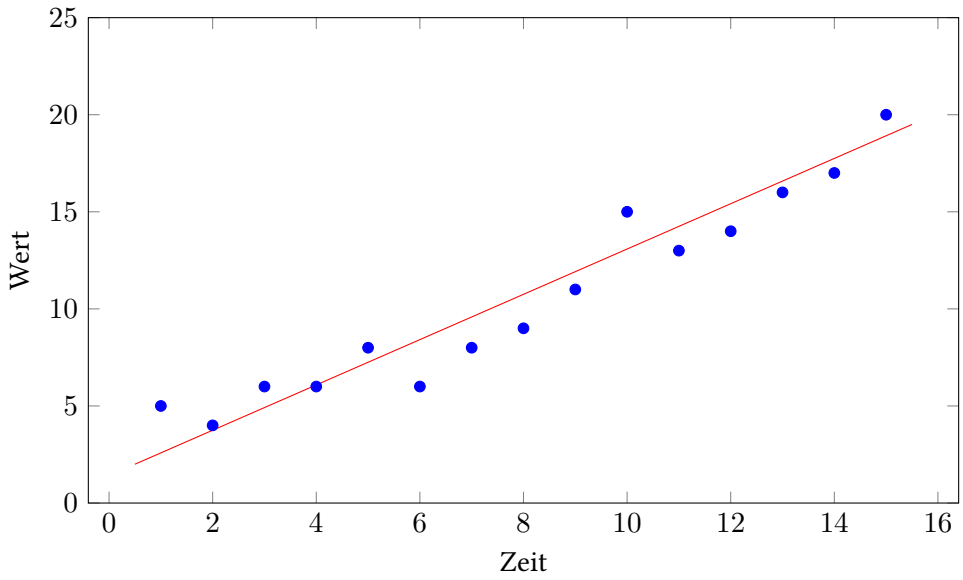

Abbildung 3.1: Punkte zu Gerade

3.2.1 Lineare Regression

Es existieren verschiedene Herangehensweisen, um aus einem *Punkthaufen* (auch *Punktwolke*: „eine Menge von Punkten eines Vektorraums [...], die eine unorganisierte räumliche Struktur ("Wolke") aufweist“ [Wikipedia (2014)]) eine Gerade zu bestimmen. Ein für diese Anwendung etabliertes Verfahren ist die lineare Regression:

> „Die lineare Regression ist eine der wichtigsten Methoden der Datenanalyse. Sie dient der Bestimmung von Modellparametern, der Modellanpassung, der Überprüfung der Stärke von Einflussfaktoren und der Prognose in allen Bereichen der Human-, Natur- und Wirtschaftswissenschaften.“ [Oberguggenberger und Ostermann (2009)]

> „Die Standardaufgabe der (einfachen oder univariaten) linearen Regression ist es, ein *lineares Modell* $y = \beta_0 + \beta_1 x$ an die Messdaten anzupassen, also eine *beste Gerade* oder *Regressionsgerade* durch die Punktwolke zu legen.“ [Oberguggenberger und Ostermann (2009)]

Da hier eine Bibliothek verwendet wird, in der dieses Verfahren bereits implementiert wurde, wird auf die genaue Funktionsweise nicht näher eingegangen. Details dazu lassen sich etwa in Oberguggenberger und Ostermann (2009) oder Lerch (2012) finden.

3.2.2 Verwendete Software

Apache Commons Math

Für die Bereitstellung der mathemathischen Funktionen bietet sich die Math-Bibliothek aus dem Apache-Commons-Projekt an, da sie etwa die lineare Regression bereits gebrauchsfertig implementiert.

Seine Leitprinzipien definiert das Projekt wie folgt: (Vgl. [The Apache Software Foundation (2014)]):

- Fokus auf Funktionen, für die es tatsächliche Anwendungsfälle in der realen Welt gibt
- Kleine, ohne aufwändige Konfiguration einbindbare Komponenten
- Verwendete Algorithmen sind vollständig dokumentiert.
- Verschiedene Implementierungen für das selbe Problem sind untereinander austauschbar.
- Keine aufwändigen Abhängigkeiten

3.3 Visualisierung

Um getroffene Aussagen und die Grundlagen, auf deren Basis diese Aussagen zustande gekommen sind, zu veranschaulichen, kann es sinnvoll sein, diese Daten zu visualisieren. Eine Visualisierung kann einerseits der Nachvollziehbarkeit dienen, aber auch zur Überprüfung der Korrektheit der Daten und der eingesetzten Algorithmen hilfreich sein. Dabei kann unterschieden werden zwischen Live- und statischer Visualisierung, auf die im Folgenden eingegangen wird.

3.3.1 Live-Visualisierung

Um die aktuell verwendeten Daten betrachten zu können und diese auch animiert darstellen zu können, wird das Java-Framework JFreeChart verwendet. Ein Beispiel für die Verwendung ist in Abbildung 3.2 zu sehen.

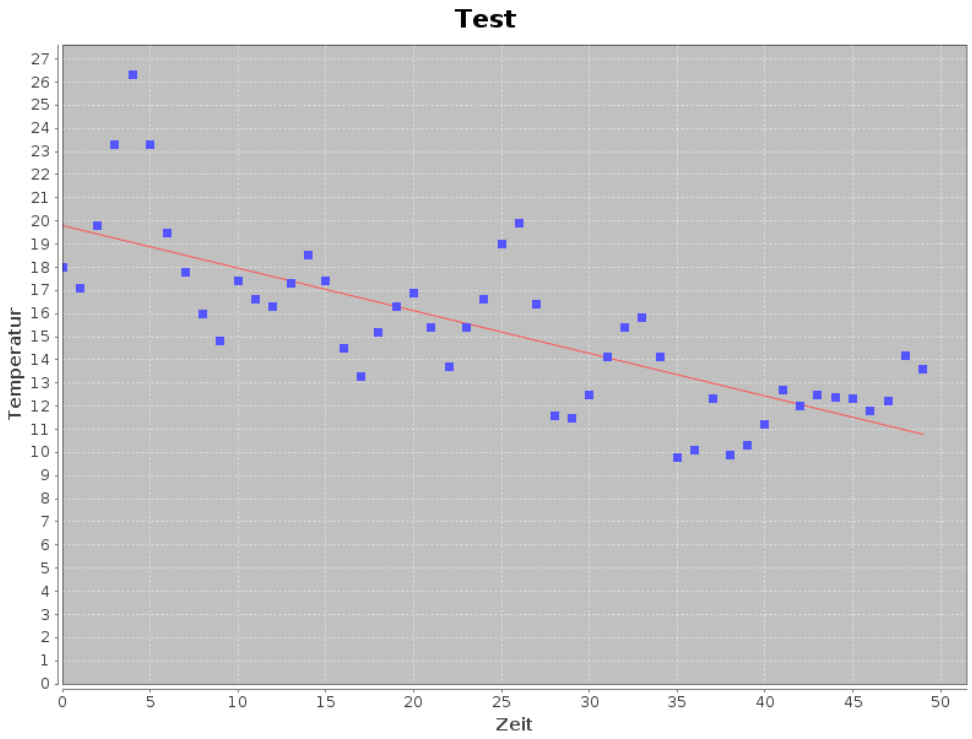

Abbildung 3.2: Screenshot WeatherTrend

3.3.2 Statische Berichte

Neben der Live-Visualisierung kann es bei bestimmten Gelegenheiten erforderlich sein, statische Berichte zu erstellen. Ein gängiges Format für portable Dokumente ist das *Portable Document Format* (PDF), das 1993 von Adobe veröffentlicht wurde und seit 2008 ISO-Standard ist [Adobe Systems Incorporated (2014)].

PDFs eignen sich zum einfachen Versenden, da sie auf vielen Systemen darstellbar sind. Sie lassen sich zudem ausdrucken, um etwa in Meetings als Handreichung zu dienen.

PGFPlots

PGFPlots ist ein LaTeX-Paket, das es ermöglicht, aus zwei- oder dreidimensionalen Daten Diagramme zu erstellen. Es basiert seinerseits auf PGF/TikZ, einer Kombination aus dem Makropaket PGF und dem zugehörigen Frontend TikZ, die gemeinsam der Erstellung von Vektorgrafiken in LaTeX dienen.

Zur Nutzung wird eine installierte LaTex-Distribution mit den entsprechenden Paketen vorausgesetzt. LaTeX-Code wird zunächst als Plaintext-Datei erstellt und lässt sich so von einem Software-System automatisiert generieren. Ein weiterer Vorteil bei der Verwendung von Paketen wie PGF/TikZ ist, dass damit PDF-native Vektorgrafiken erzeugt werden, die wenig Speicher belegen und beliebig vergrößerbar sind, ohne dass die Darstellungsqualität leidet.

Um beispielsweise Abbildung 3.1 in diesem Dokument zu erzeugen, wurde folgender Code verwendet:

```
\begin{tikzpicture}
        \begin{axis}[
                width=0.9\linewidth,
                height=8cm,
                ymin=0,
                ymax=25,
                xlabel={Zeit},
                ylabel={Wert},
                ]
                \addplot[blue, only marks] table {
                        1       5
                        2       4
                        3       6
                        4       6
                        5       8
                        6       6
                        7       8
                        8       9
                        9       11
                        10      15
                        11      13
                        12      14
                        13      16
                        14      17
```

```
                        15      20
                };
                \draw[red] (axis cs:0.5,2) -- (axis cs:15.5,19.5);
        \end{axis}
\end{tikzpicture}
```

4 Systemdesign

4.1 Anforderungen

Das Ergebnis der Arbeit soll eine Plattform sein, die eine Interpretation beliebiger Daten ermöglicht und es außerdem für Dritte einfacher macht, eigene Interpretationen umzusetzen. Sie baut auf die vorhandene Infrastruktur des Living Place auf, es wird also ein Blackboard-System mit Message Broker, hier ein ActiveMQ, und einer Datenbank, hier MongoDB, vorausgesetzt, auf dem bereits die Roh-(Sensor-)daten anfallen, auf Grundlage derer mittels Interpretation eine Aussage getroffen werden soll.

Die Anforderungen an die Plattform sind wie folgt:

4.1.1 Daten vom Restsystem / den Sensoren abholen

Die Sensoren, deren Messdaten als Grundlage für eine weitere Verarbeitung dienen sollen, können von unterschiedlicher Art sein. Es wird davon ausgegangen, dass „Low-Level"-Sensoren, die im konkreten Anwendungsfall genutzt werden, bereits im Labor installiert und an das Blackboard-System angebunden sind. Entsprechendes gilt für Sensoren aus externen Quellen, etwa Temperaturwerte von Wetterdiensten. Der Interpretationsdienst muss über das Blackboard auf diese Daten zugreifen können, also über einen entsprechenden Adapter verfügen.

Zu Sensoren siehe auch 2.4.

4.1.2 Interpretationen durchführen und Ergebnisse zur Verfügung stellen

Es sollen verschiedene Arten von Interpretationen durchgeführt werden können. Die Ergebnisse sollen dem Restsystem transparent zur Verfügung gestellt werden, also für Drittanwendungen wie Daten aus anderen Quellen, zum Beispiel Sensoren, über das Blackboard abrufbar

sein. Einzelne Instanzen des Interpretationsdienstes werden sich also wie virtuelle Sensoren verhalten.

4.1.3 Erweiterbarkeit durch Interpretationsplugins

Der Interpretationsdienst soll als Plattform dienen, die es anderen Nutzern ermöglicht, mit möglichst wenig Aufwand neue Interpretationsarten zu implementieren. Daher werden Interpretationsarten über Plugins realisiert, für die die Plattform eine Schnittstelle anbietet.

4.1.4 Lose Kopplung des Systems

Zusätzlich zur Erweiterbarkeit durch Plugins soll eine lose Kopplung des Gesamtsystems dafür sorgen, dass bei Bedarf auch die anderen Komponenten möglichst einfach an neue Gegebenheiten angepasst werden können.

4.1.5 Interpretationsaufträge erstellen

Interpretationen werden nur auf Anforderung getätigt. Nutzer des Interpretationsdienstes, also andere in der Umgebung laufende Dienste, sollen mit den zur Verfügung stehenden Interpretationsarten frei Interpretationsaufträge erstellen können, von denen dann in definierbaren Intervallen Ergebnisse auf das Blackboard geschrieben werden. Zum Erstellen, Verwalten und Entfernen dieser Aufträge werden Nachrichten in einem definierten Format auf das Blackboard geschrieben, die der Interpretationsdienst entgegennehmen und bearbeiten können muss.

4.1.6 Identifizierbarkeit der Aufträge

Damit mehrere Interpretationsaufträge verwaltet werden können, muss jeder generierte Auftrag eindeutig identifizierbar sein. So wird auch das gezielte Löschen oder Beenden nicht mehr benötigter Aufträge ermöglicht.

4.1.7 Logging

Um die Funktion des Interpretationsdienstes nachvollziehen und gegebenenfalls fehlerhafte Ausgaben untersuchen zu können (Debugging), sollte dieser seine Aktivitäten, wie etwa Erstellung und Durchführung von Aufträgen, möglichst umfassend protokollieren können.

4.2 Architektur

4.2.1 Plugin-Architektur

Die eigentliche Interpretationslogik wird in Form von Bibliotheken realisiert, die in die Plattform eingebunden werden. Das hat den Vorteil, dass im Bedarfsfall auf einfache Weise neue Arten von Interpretationen implementiert werden können.

Bei der Umsetzung der Plugin-Architektur gibt es grundsätzlich zwei Möglichkeiten, wie mit dem Caching, also dem Vorhalten der Daten, verfahren werden kann:

Worker hält Liste mit Punkten Die eine Möglichkeit ist, den Worker die Daten, also etwa die Liste mit Einzelpunkten, verwalten zu lassen. Für dieses Verfahren spricht, dass so der Implementierungsaufwand für weitere Interpretationsbibliotheks-Plugins gering gehalten werden kann, da die Bibliothek nur eine nach außen, also für den Worker sichtbare Methode bereitstellen muss, die eine Liste mit Datenpunkten entgegennimmt, die Berechnung durchführt und das Ergebnis zurückgibt.

Interpretationsbibliothek hält Liste mit Punkten Die andere Möglichkeit ist, die Interpretationsbibliothek selbst die Daten verwalten zu lassen. Für dieses Verfahren spricht, dass es weniger Overhead durch wiederholte Übertragung von Daten vom Worker an die Bibliothek gibt, allerdings mit dem Nachteil, dass der Implementierungsaufwand für jede Bibliothek größer ist. Diesem Problem kann begegnet werden, indem eine abstrakte Klasse vorgegeben wird, die den Puffer verwaltet.

4.2.2 Plugins als Java Archive

Für die Umsetzung von Plugin-Architekturen gibt es spezialisierte Frameworks, für Java-basierte Projekte konnte sich etwa OSGi („*Open Services Gateway initiative*“) etablieren. Für

den vorliegenden Fall ist ein solches Framework allerdings überdimensioniert. Stattdessen bietet sich die Möglichkeit an, die zur Berechnung der Interpretation benötigten Klassen in JAR-Dateien („*Java Archive*“) zusammenzufassen, die sich in Java-Anwendungen nativ einbinden lassen.

Als Beispiel soll ein Plugin dienen, das wie in Kapitel 3 beschrieben, Aussagen über den Verlauf von eindimensionalen Daten, wie etwa die Steigung des Temperaturverlaufs, trifft. Weiterhin würde es sich anbieten, ein Plugin zu entwickeln, das Aussagen über zweidimensionale Daten, etwa eine Reihe von Positionsdaten, trifft. Inhalt der Aussage kann dann die Bewegung einer Person oder eines Gegenstandes von einem in einen anderen Bereich der Wohnung oder der Aufenthalt in einem bestimmten Bereich sein.

4.3 Persistenz

4.3.1 Aussagen über Zeiträume

Aussagen über den derzeitigen Zustand der Wohnung können allein anhand der Live-Daten vom ActiveMQ getätigt werden. Wenn die Vergangenheit mit einbezogen werden soll, wird eine Persistenz benötigt. Für „Live-Aussagen“ werden zwar auch Daten aus der Vergangenheit benötigt, diese können aber lokal vorgehalten werden. Wenn beispielsweise jede Sekunde ein neuer Temperaturwert von den Sensoren über den ActiveMQ geliefert wird und eine Aussage über den Verlauf innerhalb der letzten 30 Minuten getroffen werden soll, muss der Dienst diese 30*60 (= 1800) Werte vorhalten. Diese Werte werden in einem Ringpuffer (Abbildung 4.1) lokal gespeichert, so dass ständig die letzten n Werte vorliegen. Ab einer halben Stunde Laufzeit kann also komplett auf gesammelten Live-Daten gearbeitet werden. Innerhalb der ersten 30 Minuten muss auf Daten aus einer externen Persistenz (hier MongoDB) zurückgegriffen werden. Liegen solche Daten nicht vor, muss die Aussage über einen kleineren Zeitraum getroffen werden, falls die gewählte Interpretationsart dies zulässt.

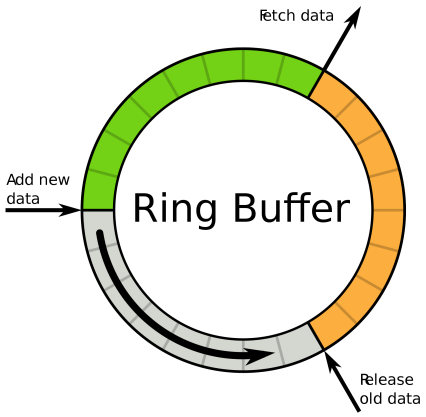

Abbildung 4.1: Ringpuffer [User:Sven, Wikimedia Commons (2010)]

4.4 Plattform

4.4.1 Virtueller Sensor

Der Interpretationsdienst verhält sich nach außen wie eine Art virtueller Sensor in dem Sinne, dass er wie ein normaler Sensor in regelmäßigen Abständen Daten auf das Blackboard schreibt. Von diesen „virtuellen Sensoren“ sollen mehrere Instanzen gestartet werden können, die dann selbstständig über einen bestimmten Zeitraum oder bis auf Abruf laufen. Dazu bedarf es einer Verwaltungsinstanz, die Aufträge entgegennimmt und Instanzen einzelner Interpretationen („*Interpretationsauftrag*“) starten und beenden kann.

Um so einen Interpretationsauftrag zu starten, wird eine JSON-Nachricht auf eine bestimmte Queue des ActiveMQ geschrieben, die folgende Daten enthält:

- *SensorID* des zu verwendenden Sensors oder der Gruppe von Sensoren
- *ClientID* der Anwendung, die diesen Auftrag erteilt hat
- *Art der Interpretation*
- *Laufzeit*
- *Intervall*, nach dem neue Ergebnisse produziert werden sollen

4.4.2 Verarbeitung von Interpretationsaufträgen

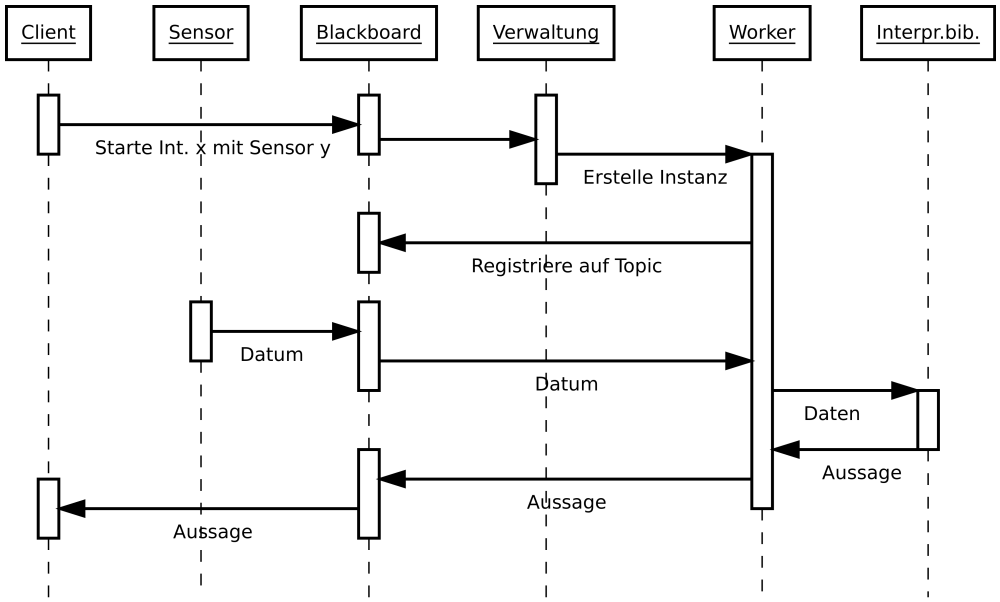

Abbildung 4.2: Verarbeitung eines Interpretationsauftrags

Der generelle Ablauf bei der Verarbeitung von Interpretationsaufträgen ist wie folgt:

Die Verwaltungskomponente beobachtet eine Queue auf dem Blackboard, hier „Interpretation.Request“, auf der Anfragen für neue Interpretationsaufträge eingehen. Ein Client, der eine bestimmte Interpretation verwenden möchte, schreibt eine Nachricht auf diese Queue, in der er angibt, welcher Sensor genutzt werden soll und welche Art von Aussage anhand der Daten dieses Sensors getroffen werden soll. Die Verwaltung erstellt daraufhin eine Worker-Instanz, die von nun an mit der Bearbeitung dieses konkreten Interpretationsauftrags beschäftigt ist. Der Worker registriert sich auf dem Topic, auf das der Sensor seine Messwerte liefert. Von den erhaltenen Messwerten hält er genau so viele, wie für die Erstellung einer Aussage nötig sind, in einer lokalen Liste vorrätig. Diese Liste gibt er an die Interpretationsbibliothek, die darüber eine Aussage trifft und diese an den Worker zurückgibt. Der Worker schreibt diese Nachricht auf ein Topic, das ein oder mehrere Clients abonniert haben können.

4.5 Komponenten Interpretationsdienst

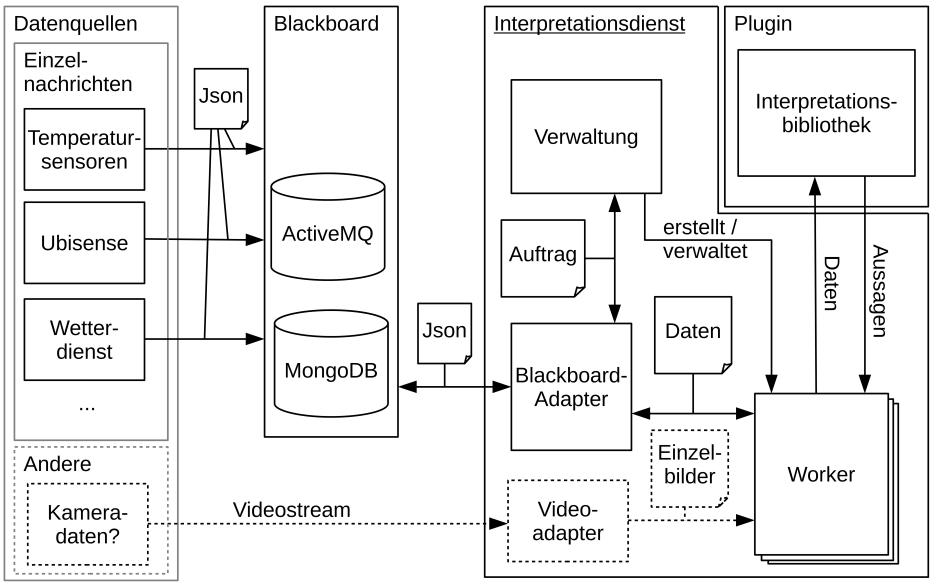

Abbildung 4.3: Außenansicht Gesamtsystem

4.5.1 Interpretationsauftrag

Ein Interpretationsauftrag ist ein Datentyp, in dem definiert wird, welche Art der Interpretation auf welche Weise durchgeführt werden soll, er enthält also Konfigurationsdaten. Nutzdaten sind hier nicht enthalten, sondern werden entsprechend der Konfiguration durch den Worker vom Blackboard bezogen.

JSON auf „Interpretation.Request“ :

```
{"addInterpretation": {
        "arguments": {
                "ClientID": "...",
                "SensorID": "...",
                "TypeOfQuality": "...",
                "TimeRange": "..."
        }
```

```
8 }}
```

4.5.2 Blackboard-Adapter

Der Blackboard-Adapter übernimmt die Kommunikation zwischen dem Blackboard und den Komponenten des Interpretationsdienstes. Da in diesem Fall das Blackboard durch ActiveMQ und MongoDB realisiert wird, ist der Adapter genau darauf zugeschnitten. Die Kapselung des Blackboard-Adapters als eigenständige Komponente ermöglicht eine einfache Austauschbarkeit für den Fall, dass etwa der ActiveMQ durch einen anderen Message Broker ersetzt werden würde, und trägt so zu einer losen Kopplung des Interpretationsdienstes und des Gesamtsystems bei.

So wie das Blackboard selbst implementiert auch der Blackboard-Adapter Publish-Subscribe, also das Observer Pattern. Eine Komponente, die ihn lesend nutzt, registriert sich bei dem Adapter, der sich wiederum bei dem Message Broker des Blackboards, also dem ActiveMQ registriert. Entsprechend benachrichtigt bei einer eingegangenen Nachricht der ActiveMQ den Adapter, der daraufhin die registrierte Zielkomponente benachrichtigt.

4.5.3 Verwaltung

Die Verwaltung erstellt und verwaltet Worker. Sie ist der zentrale Thread innerhalb des Interpretationsdienstes. Eine Instanz des Interpretationsdienstes hat genau eine Verwaltung, die durchgehend auf Aufträge wartet.

Die Verwaltung nutzt den Blackboard-Adapter, um die Queue „Interpretation.Request“ auf dem Blackboard zu registrieren, auf der Anfragen für Interpretationsaufträge eingehen. Die aktiven Aufträge werden in einer Liste verwaltet. Um eine Identifizierbarkeit der einzelnen Aufträge zu gewährleisten, wird jedem angenommenen Auftrag eine eindeutige ID zugewiesen, die dem Auftragsteller über das Blackboard bekannt gemacht wird.

Wenn die Verwaltung eine Nachricht erhält, aus der ein Interpretationsauftrag hervorgeht (siehe 4.5.1, „AddInterpretation“), wird dieser Auftrag in die Liste geschrieben und eine Instanz des Workers gestartet, die diesen Auftrag bearbeiten soll. Wird ein bestimmter Interpretationsauftrag nicht mehr benötigt, kann er entsprechend mit „removeInterpretation“ und der eindeutigen ID wieder entfernt werden.

4.5.4 Worker

Eine Worker-Instanz bearbeitet genau einen Interpretationsauftrag. Es kann mehrere Instanzen geben, die gleichzeitig laufen. Bei der Erstellung wird festgelegt, welche Interpretation durchgeführt, also welche Interpretationsbibliothek genutzt werden soll und auf welche Topics sich der Worker dazu registrieren muss, um die Sensordaten zu erhalten, die als Grundlage für die Berechnung dienen. Außerdem werden für die Interpretation notwendige Parameter gesetzt. Die Dauer der Ausführung ist ebenfalls konfigurierbar, so kann ein Worker sich nach einer bestimmte Anzahl von Durchläufen selbstständig beenden oder bis auf Abruf laufen.

4.5.5 Interpretationsbibliothek

In einer Interpretationsbibliothek findet die eigentliche Berechnung statt. Die Bibliothek wird als Plugin (4.2.1) in den Interpretationsdienst eingebunden. Auf diese Weise können je nach Bedarf passende Plugins entwickelt werden, die ausschließlich die jeweilige Interpretationslogik implementieren müssen.

5 Schluss

5.1 Zusammenfassung

In dieser Arbeit wurde ein System entworfen, das eine Plattform für die Interpretation von Sensordaten bietet. Als Beispiel für eine Smart-Home-Umgebung diente dabei der Living Place der Hochschule für Angewandte Wissenschaften Hamburg, dessen Architektur untersucht und im Analysekapitel (2) beschrieben wurde.

Weiterhin wurden Verfahren (3) zur Visualisierung und Verarbeitung von Messdaten ausgewählt und vorgestellt, die bei der Entwicklung eines solchen Systems genutzt werden können. Auf Grundlage zuvor aufgeführter Anwendungsfälle wurden Anforderungen an die Plattform definiert und im Systemdesign (4) umgesetzt.

5.2 Ausblick

Wie gut das hier entworfene System sich in den Living Place integrieren lässt und die Anforderungen anderer Entwickler, die es in Zukunft nutzen könnten, erfüllt, muss sich noch zeigen.

Die Plattform bietet eine einfache Erweiterbarkeit, die durch eine lose Kopplung und die Plugin-Fähigkeit des Systems erreicht wurde. Das ist von Vorteil, da, obwohl nicht vorausgesehen werden kann, welche Interpretationsarten später einmal benötigt werden, diese so einfach ergänzt werden können.

Die Arbeit orientiert sich an einer konkreten Smart-Home-Umgebung und deren Architektur. Während sich die erarbeiteten Konzepte auf andere Smart Environments übertragen lassen, wird noch überprüft werden müssen, inwieweit das auch für die entwickelten Komponenten gilt.

Abschließend kann gesagt werden, dass Smart Homes, aber auch Smart Environments im Allgemeinen, ohne eine semantische Interpretation von Sensordaten nicht denkbar sind.

Literaturverzeichnis

[Abowd u. a. 1999] Abowd, Gregory D. ; Dey, Anind K. ; Brown, Peter J. ; Davies, Nigel ; Smith, Mark ; Steggles, Pete: Towards a better understanding of context and context-awareness. In: *Handheld and ubiquitous computing*, 1999, S. 304–307

[Adobe Systems Incorporated 2014] Adobe Systems Incorporated: *PDF Reference and Adobe Extensions to the PDF Specification*. 2014. – URL http://www.adobe.com/devnet/pdf/pdf_reference.html. – [Online; Stand 23. Juli 2014]

[Bornemann 2013] Bornemann, Sven B.: *Entwicklung eines kontextsensitiven Berechtigungssystems für Smart Homes*, HAW Hamburg, Masterarbeit, 2013

[Ellenberg u. a. 2011] Ellenberg, Jens ; Karstaedt, Bastian ; Voskuhl, Sören ; Luck, Kai von ; Wendholt, Birgit: An environment for context-aware applications in smart homes. In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN), Guimaraes, Portugal*, 2011

[IBM 2008] IBM: *Smarter Planet*. 2008. – URL http://www.ibm.com/smarterplanet. – [Online; Stand 28. Juli 2014]

[Kantak 2013] Kantak, Malte: *Kontextgestützte semi-automatische Erreichbarkeitsermittlung von Bewohnern in intelligenten Umgebungen*, HAW Hamburg, Masterarbeit, 2013

[Lerch 2012] Lerch, Reinhard: Regression, lineare Korrelation und Hypothesen-Testverfahren. In: *Elektrische Messtechnik*. Springer Berlin Heidelberg, 2012 (Springer-Lehrbuch), S. 473–496. – URL http://dx.doi.org/10.1007/978-3-642-22609-0_14. – ISBN 978-3-642-22608-3

[Oberguggenberger und Ostermann 2009] Oberguggenberger, Michael ; Ostermann, Alexander: Lineare Regression. In: *Analysis für Informatiker*. Springer Berlin Heidelberg, 2009 (eXamen.press), S. 223–240. – URL http://dx.doi.org/10.1007/978-3-540-89823-8_18. – ISBN 978-3-540-89822-1

[Otto 2013] Otto, Kjell: *Aktuelle Entwicklungskonzepte zur Projektintegration in einem Smart Home anhand von Maven, OSGi und Drools Fusion*, HAW Hamburg, Masterarbeit, 2013

[Pautz 2012] Pautz, Alexander: *Kabellose, stromsparende Sensornetzwerke im Bereich Ambient Intelligence*, HAW Hamburg, Masterarbeit, 2012

[Schilit u. a. 1994] Schilit, B. ; Adams, N. ; Want, R.: Context-Aware Computing Applications. In: *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, 1994, S. 85–90

[The Apache Software Foundation 2014] The Apache Software Foundation: *Commons Math: The Apache Commons Mathematics Library*. 2014. – URL http://commons.apache.org/proper/commons-math/. – [Online; Stand 21. Juli 2014]

[User:Sven, Wikimedia Commons 2010] User:Sven, Wikimedia Commons: *File:Ring buffer.svg*. 2010. – URL http://commons.wikimedia.org/w/index.php?title=File:Ring_buffer.svg&oldid=55301635. – [Online; Stand 3. Juli 2014]

[Voskuhl 2011] Voskuhl, Sören: *Modellunabhängige Kontextinterpretation in einer Smart Home Umgebung*, HAW Hamburg, Masterarbeit, 2011

[Weiser 1991] Weiser, Mark: The computer for the 21st century. In: *Scientific american* 265 (1991), Nr. 3, S. 94–104

[Wikipedia 2014] Wikipedia: *Punktwolke — Wikipedia, Die freie Enzyklopädie*. 2014. – URL http://de.wikipedia.org/w/index.php?title=Punktwolke&oldid=132054865. – [Online; Stand 21. Juli 2014]

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 29. Juli 2014 Jan Jennrich