Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Christian Kirchner

Entwicklung eines kontextsensitiven Agenten zur Verwaltung und Verteilung von Informationen im Living Place

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Christian Kirchner

Entwicklung eines kontextsensitiven Agenten zur Verwaltung und Verteilung von Informationen im Living Place

Eingereicht am: 25.8.2014

Inhaltsverzeichnis

1. Einführung

In dieser Projektarbeit sollen die ersten praktischen Grundlagen für die Entwicklung einer Komponente zur zentralen Verwaltung von Inhalten und entsprechenden Anzeigegeräten im Living Place (LP) erläutert werden. Das LP ist eine, für Forschungszwecke in den Bereichen Smart-Home-Enviroment und Urban-Living umgebaute, Loft-Wohnung mitten auf dem Campus der HAW Hamburg.

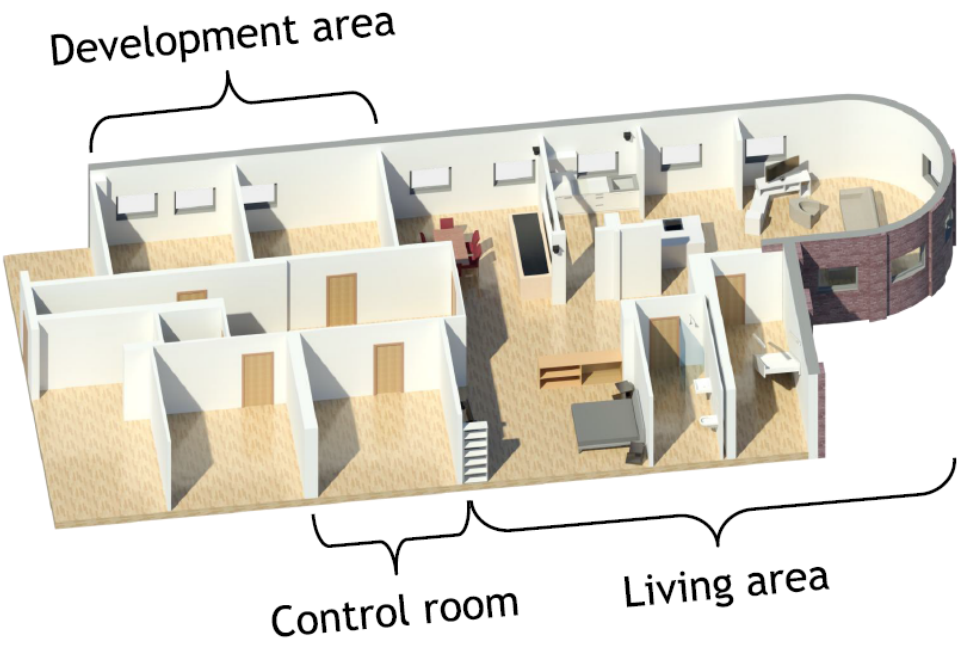

Abbildung 1.1.: Das Living Place Hamburg Karstaedt (2012)

Durch die Vielzahl von Hardware- und Softwaresystemen in diesem Wohnbereich entstehen viele verschiedene Arten von Informationen, von Sensordaten zur Position der Fenster oder dem Status der Lichtanlagen, bis hin zu Twitter Feeds zum aktuellen Fernsehprogramm. Um diese Inhalte den Personen in der Wohnung präsentieren zu können, müssen passende Anzeigegeräte identifiziert werden, welche in der Lage sind die Informationen zu verarbeiten und entsprechend darzustellen. Das zu entwickelnde System soll dabei helfen diesen Prozess zu vereinfachen, indem es eine zentrale Schnittstelle für diese Objekte bereitstellt und die Kontextinformationen aus der Wohnung nutzt, um die verschiedenen Inhalte möglichst intelligent auf den jeweils optimalen Anzeigegeräten zu platzieren.

Die Einleitung (Kapitel 1) gibt einen Überblick über die Gliederung dieses Berichts und wird abgeschlossen mit der Zielsetzung dieses Projekts. Im darauf folgenden Kapitel 2 werden die wichtigsten Komponenten der Architektur des LP und speziell die neu entwickelte Middleware vorgestellt. Das Kapitel 3 beschreibt das bevorstehende Projekt genauer, indem der Grundge-

danke aufgegriffen wird und die wesentlichen Anforderungen an das System zusammengefasst werden. Im Kapitel 4 wird ein Überblick über die Architektur des Systems gegeben und die wesentlichen Komponenten beschrieben. Das Kapitel 5 befasst sich mit der Evaluation des Projektes anhand eines Beispielszenarios und beschreibt die erzielten Ergebnisse. In der Schlussbetrachtung (Kapitel 6) wird das Projekt kurz zusammengefasst und abschließend ein Ausblick über das weitere Vorgehen in folgenden Projektarbeiten gegeben.

1.1. Zielsetzung

Das übergeordnete Ziel des Projektes ist es ein System zu entwickeln, was neuen Entwicklern das Anzeigen von Inhalten im LP erleichtert, indem es einen Überblick über die vorhandenen Anzeigegeräte und dort angezeigten Inhalten ermöglicht. Als Ergebnis soll ein Framework entstehen, welches anderen Entwicklern dabei hilft neue Geräte und Inhalte im LP zu installieren. Das Framework stellt dafür Agenten mit gewissen Basisfunktionen zur Verfügung, auf denen neue Inhalts- und Geräteagenten aufgebaut werden können.

Die Aufgabe für die erste Projektarbeit ist es zunächst die Anforderungen an ein solches System zu identifizieren und anschließend eine passende Architektur zu erstellen. Anhand eines Beispielszenarios soll eine erste Version des Systems im LP installiert werden, durch das die Anforderungen verifiziert und validiert werden können. Dadurch entsteht ein erster Einblick in die neuen Möglichkeiten der intelligenten Inhaltspräsentation.

2. Living Place

In diesem Kapitel wird ein Überblick über die im LP vorhandene Software-Architektur und die in dem Projekt verwendete Hardware gegeben. Als grundlegende Komponente wird zunächst die neu entworfene Middleware vorgestellt, welche als zentrale Kommunikationseinheit für neu entwickelten Softwarelösungen im LP dienen soll. Abschließend wird erläutert, wie die Ubisense Komponente zur Positionsbestimmung in das gesamt System des LP's integriert ist und welche Art Informationen es liefert.

2.1. Middleware

Die Middleware (vgl. Eichler (2013) und Eichler (2014)) die im LP zum Einsatz kommt besteht aus einer Ansammlung von Agenten, welche in Gruppen zusammenarbeiten und über eine Gruppenkommunikation Informationen oder Funktionen bereitstellen. Realisiert ist die Middleware in der Programmiersprache Scala unter Zuhilfenahme der Aktorbibliothek Akka.

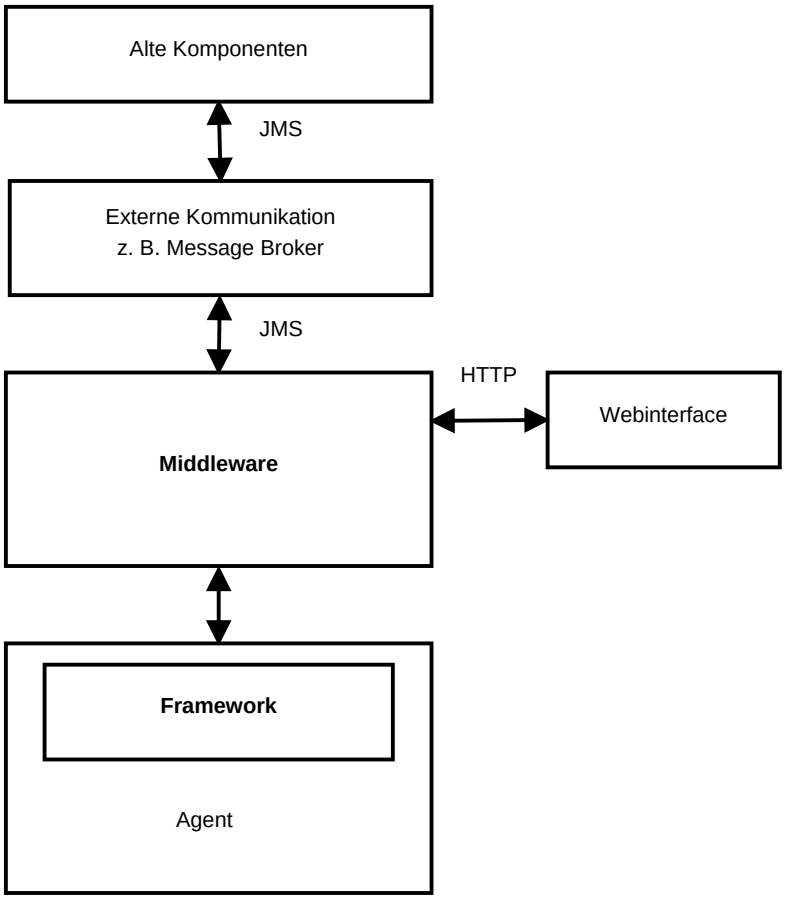

Abbildung 2.1.: Übersicht des Interaktionsverlaufes der Middleware Eichler (2014)

Ein eigens zur Entwicklung neuer Komponenten für das LP entwickelte Framework (vgl Abbildung 2.1), unterstützt die Entwickler bei der Umsetzung neuer Agenten. Es kapselt die Interaktion mit der Middleware und übernimmt Aufgaben wie Serialisierung von Nachrichten oder das Abonieren von Nachrichten aus ausgewählten Gruppen. Zur Serialisierung der Nachrichten wird das JSON Format verwendet. Außerdem bietet es verschiedene Kommunikationspatterns an, mit denen Nachrichten an Gruppen geschickt oder Informationen daraus abgefragt werden können. Zusätzlich ermöglicht das Framework die Implementation von Unit Tests für einzelne Komponenten, mit denen mögliche Fehlfunktionen der Agenten überprüft werden können.

2.2. Ubisense

Im LP wird für das Indoor Positioning ein System der Firma Ubisense[1] eingesetzt, welches auf der Basis von UWB Signalen Location Tags lokalisiert (vgl. Voskuhl (2011)). Bei einer Positionsänderungen der Tags werden JSON Nachricht in einem Topic, eines im LP installierten ActiveMQ Servers, publiziert. Die Nachricht enthält die ID des Tags und seine Koordinaten, zusammen mit der dazugehörigen Maßeinheit. Der ActiveMQ Server ist an die Middleware angebunden und erstellt für jedes seiner Topics eine Gruppe in der die Nachrichten veröffentlicht werden.

[1] http://www.ubisense.net/en/

3. Projektbeschreibung

Dieses Kapitel wird die grundliegende Idee und das Ziel dieses Projektes erläutern und mit Hilfe eines Beispielszenarios verdeutlichen.

3.1. Idee

Durch die stetig wachsende Präsenz von Themen wie Smart-Home-Enviroment, Urban-Living und Second Screen steigt auch die Anzahl an verschiedenen Informationen, mit denen man im Alltag konfrontiert wird. Dies wird begünstigt durch die vermehrte elektronische Bereitstellung von relevanten Daten und den Anstieg von Anzeigegeräten wie Smartphones, Tablets oder Smart TV's. Speziell im LP ist die Anzahl und das zukünftige Potential für solche Inhalte besonders hoch. Unter einem Inhalt (Content) versteht man jegliche Art von Informationen oder medialen Inhalten, die Bewohnern oder Besuchern des LP's präsentiert werden können. Dabei kann es sich um Sensordaten der Heizkörper oder Lichtanlagen handeln, aber auch Videodateien, Fernsehbilder oder Musik. Möchte man nun jedoch diese Informationen als Inhalte auf entsprechenden Geräten (Devices) platzieren, entstehen viele Fragen, die beantwortet werden müssen:

- Welche Inhalte und Geräte stehen zur Verfügung?
- Welche Eigenschaften haben die Inhalte und auf welchen Geräten können sie verarbeitet werden?
- Welche Eigenschaften haben die Geräte und befinden sie sich in dem richtigen Zustand um Inhalte angemessen zu präsentieren?
- Für welche Benutzer sind entsprechende Inhalte zugänglich?
- Welcher Benutzer hat Zugriff auf welche Geräte?
- Welcher Inhalt wird momentan auf welchen Geräten verarbeitet?
- Existieren überhaupt Geräte die bestimmten Inhalt verarbeiten können oder muss der Inhalt transformiert werden um dargestellt zu werden.

Da alle diese Fragen zu beantworten sind, unabhängig davon um welchen Inhalt es sich handelt, befasst sich dieses Projekt mit der Entwicklung eines kontextsensitiven Agenten zur Verwaltung dieser Informationen und der intelligenten Verteilung der Inhalte im LP. Durch die Bereitstellung einer öffentlichen API soll er das Anzeigen von Inhalten im LP vereinheitlichen und dadurch Entwicklern im LP helfen, geeignete Anzeigegeräte für ihre Inhalte zu identifizieren.

Gespeist durch Kontextiformationen aus dem LP stellt der Agent verschiedene Strategien bereit, mit deren Hilfe entsprechende Inhalte jeweils auf den geeignetsten Geräten wiedergegeben werden. Der Grundgedanke hinter diesem Projekt ist es, dass sich zukünftige Entwickler keine

Gedanken mehr machen müssen, wie oder wo sie ihre Inhalte platzieren. Die Inhalte müssen nur noch spezifiziert und über eine ausgewählte Strategie auf den passenden Geräten verabreitet werden. Am Ende des gesamten Projektes soll dadurch eine intelligente Verteilung der Inhalte unter Berücksichtigung aller zur Verfügung stehenden Kontextinformationen des LP möglich sein.

3.2. Anforderungen

Damit das geplante System in das LP integriert werden kann und auch für die stetige Weiterentwicklung im LP geeignet ist, wurden zunächst die wichtigsten Anforderungen an die Architektur identifiziert:

- **Eine lose Kopplung zwischen den Komponenten**
  Sie ermöglicht es Komponenten ohne großen Aufwand auszutauschen und soll gewährleisten, dass das System flexibel bleibt. Dafür müssen die Schnittstellen möglichst allgemein gehalten werden, um die Erweiterbarkeit zu verbessern und möglichst eine Unabhängigkeit von konkreten Programmiersprachen oder spezieller Software herzustellen.
- **Hohe Wiederverwendbarkeit der Komponenten für andere Agenten**
  Die einzelnen Komponenten sollten so konzipiert sein, dass auch andere Agenten die Funktionalitäten und Informationen von einzelnen Komponenten weiter verwenden können.
- **Hohe Fehlertoleranz und starke Robustheit gegen Ausfälle einzelner Komponenten**
  Auf Grund der vielen Abhängigkeiten zwischen verschiedenen Komponenten in diesem Projekt muss sichergestellt werden, dass nicht der Ausfall einer Teilkomponente zu dem Versagen des Gesamtsystems führt. Außerdem muss auf Grund der späteren Erweiterungen und der losen Kopplung eine gewisse Fehlertoleranz gewährleistet sein, welche eventuelle Fehlfunktionen der Erweiterungen korrigiert.
- **Ein hohes Maß an Erweiterbarkeit**
  Da stetig neue Inhalte und Geräte im LP installiert werden, muss das System soweit erweiterbar sein, dass diese möglichst einfach in das bestehende System integriert werden können.

4. Architektur

In dem folgenden Kapitel werden die wichtigsten Design Entscheidungen für den Entwurf des Systems beschrieben. Die Unterteilung in Komponenten und deren Aufgaben werden erläutert. Um sich nahtlos in die bestehende Architektur des LP's einzufügen, wurden alle Komponenten mit Hilfe des Middleware Frameworks konzipiert. Die Kommunikation zwischen ihnen findet ausschließlich über die von der Middleware bereitgestellte Gruppenkommunikation statt. Dadurch können die Informations- und Kontrollnachrichten der einzelnen Agenten über verschiedene Gruppen abboniert und von anderen Agenten weiter verwendet werden.

Als Basis für das System und die interne Kommunikation der Komponenten wird ebenfalls das Aktor-Framework Akka verwendet, welches auch in der Middleware zum Einsatz kommt. Das Framework enthält unter anderem Funktionen zur transparenten Kommunikation über das Netzwerk und Fehlerbehandlungsstrategien für Aktoren (vgl. Kapitel 2.1 von Eichler (2014)). Durch die Verwendung von Aktoren und der Kommunikation über Nachrichten, lässt sich die Architektur sehr gut erweitern. Dafür müssen lediglich neue Agenten entwickelt werden, die zusätzliche Nachrichten verarbeiten und mit Hilfe eines Routers die alten Nachrichten an die ursprünglichen Agenten weiterleiten.

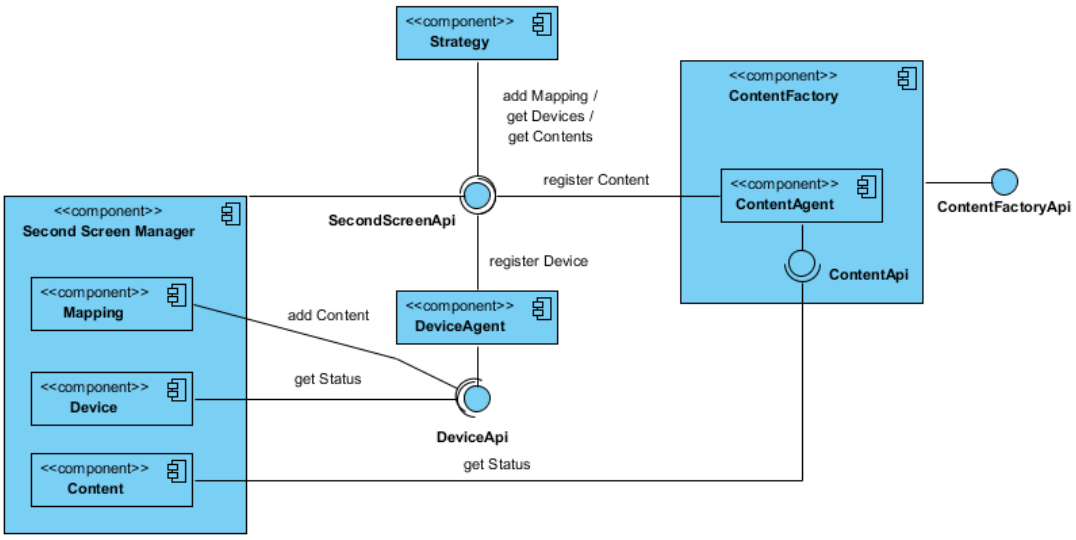

Abbildung 4.1.: Überblick über den Zusammenhang der einzelnen Komponenten und deren Kommunikationswege

4.1. Inhalt

Als Inhalt (Content) wird jegliche Art von Information aufgefasst, die auf einem Gerät verarbeitet werden kann. Über *Factories* werden die Inhalte als eigenständige Agenten spezifiziert und erstellt. Die Inhaltsagenten (Content Agent) registrieren sich daraufhin selbständig beim *DAC-Manager* (vgl. Abschnitt 4.3). In dem Beispielszenario werden über die API einer *Video Content Factory* verschiedene *Video Contents* erstellt. Es kann sich allerdings auch um andere Arten von Inhalten handeln, wie Wetter Informationen, Twitter Feeds oder Kontrollelemente wie die virtuelle Fernbedienung für das LP.

Inhaltsagenten sind definiert als eigenständige Agenten im LP, die alle durch Eigenschaften und einen Zustand spezifiziert sind und deren Eigenschaften und Zustände über eine allgemeine API abgefragt werden können. Zur Individualisierung der unterschiedlichen Inhaltstypen kann jeder Inhaltsagent zusätzlich noch eine speziell an den Inhalt angepasste API implementieren, welche die Interaktion mit dem Inhalt erlaubt. Am Beispiel eines Videos könnte dies das Festlegen des Abspielzustandes (playing, paused) sein oder das Setzen der aktuellen Abspielposition des Videos.

4.2. Geräte

Geräte (Devices) sind Komponenten, welche in der Lage sind die spezifizierten Inhalte zu verarbeiten. Dabei kann es sich um konventionelle Geräte wie Ferseher, Tablets oder Smartphones handeln, welche Video, Audio oder Textinhalte wiedergeben, allerdings auch um abstraktere Geräte, wie Lampen, die entsprechend zum Takt der Musik oder zum Ambiente eines Films ihr Licht verändern. Genau wie bei Inhalten existiert auch für jedes Gerät ein eigener Agent mit Eigenschaften und einem Status. In ihren Eigenschaften definieren sie, welche Art von Inhalt sie verarbeiten können.

4.3. DAC (Device and Content) Manager

Die zentrale Komponente der gesamten Architektur ist der *DAC-Manager*. Bei ihm werden alle Inhalte und Geräte registriert, wodurch eine Übersicht für andere Agenten im LP entsteht. Mit Hilfe eines Heartbeats wird sichergestellt, dass registrierte Inhalte und Geräte tatsächlich verfügbar sind. Der *DAC-Manager* stellt über eine öffentliche API Funktionen zur Verfügung, über die alle Geräte oder Inhalte abgefragt werden können, welche bestimmte Bedingungen erfüllen. Für diese Abfragen werden die spezifizierten Eigenschaften und aktuellen Stati (vgl. Abschnitt 4.3.1) mit Mustern (vgl. Abschnitt 4.3.2) verglichen. Durch diese Abfragen können beispielsweise alle Inhalte zurückgegeben werden, die auf bestimmten Geräten angezeigt werden

oder umgekehrt alle Geräte auf denen ein bestimmter Inhalt angezeigt wird. Über eine bei der Middleware registrierte Gruppe werden Updates verschickt, sobald ein Gerät oder Inhalt registriert wird oder ein Inhalt auf einem Gerät angezeigt oder entfernt wird.

4.3.1. Eigenschaften & Zustände

Sowohl Inhalte als auch Geräte werden durch ihre Eigenschaften (Specifications) und Stati beschrieben. Der wesentliche Unterschied dieser beiden Attribute ist, dass die Eigenschaften bei der Initialisierung festgelegt werden und sich anschließend über die Lebensdauer des Inhaltes oder des Gerätes nicht ändern. Ein Beispiel dafür wäre der Codec eines Videos und die entsprechende Auflösung. Stati hingegen können sich mit der Zeit ändern, wie beispielsweise die aktuelle Abspielposition des Videos oder seine Lautstärke. Entscheidend ist diese Unterteilung für die Funktionalität der Zuweisungen (vgl. Abschnitt 4.3.3). Die Kombinantion aus Eigenschaften und Stati gibt zu jedem Zeitpunkt den aktuellen Zustand des Objektes wieder.

Da sowohl Eigenschaften als auch Stati sehr stark von dem entsprechenden Inhalt oder Gerät abhängen, müssen deren Schnittstellen flexibel und die verwendeten Datentypen erweiterbar sein. Bei der Spezifizierung und Entwicklung weiterer Inhalte und Geräte können stetig neue Attribute entstehen, die sich wenn möglich ohne Anpassungen in das System integrieren lassen. Deshalb werden die Eigenschaften und Stati jeweils in einer verschachtelten Map von Attributen im JSON Format an die API übergeben. Bei der Registrierung über die API des *DAC-Managers* wird die Map mit Eigenschaften überprüft und sicher gestellt, dass alle benötigten Attribute vorhanden sind und das JSON Format eine valide Struktur aufweist.

Die Geräte und Inhalte aus dem Beispielzenario werden wie im Anhang A und B spezifiziert. Dabei müssen die Attributnamen auf ihrer Verschachtelungstiefe eindeutig sein, ansonsten lässt sich das Objekt nicht registrieren. Bis jetzt sind die vorausgesetzten Attribute die eindeutige ID eines jeden Inhalts oder Gerätes und die Namen der Gruppen über die sie kommunizieren. Im Verlauf der weiteren Projekte werden weitere allgemeine Eigenschaften identifiziert und mit in die Überprüfung aufgenommen.

4.3.2. Muster

Damit über die API des *DAC-Managers* Geräte und Inhalte mit bestimmten Eigenschaften oder Zuständen gesucht werden können, müssen Muster (Patterns) definiert werden, mit denen die Eigenschaften und Stati der Inhalte und Geräte verglichen werden. Patterns sind ähnlich aufgebaut wie die Eigenschaften und Stati, allerdings enthalten sie für die unterschiedlichen Attributtypen verschiedene Operatoren wie „gleich“, „ungleich“, „größer als“ und „kleiner als“. In der Zukunft wären auch weitere Operatoren wie „und“ oder „oder“, aber auch Operatoren für Listen wie „contains“ denkbar. Durch diese Möglichkeiten lassen sich gezielt Geräte oder Inhalte herraussuchen, die gerade einen bestimmten Status oder spezielle Eigenschaften haben. Ein Beispiel für ein Pattern zur Identifikation aller Geräte mit einem Seitenverhältnis von 16:9, einer Auflösung die größer ist als 1024 Pixel in der Breite und der Möglichkeit Video Content abzuspielen gibt es im Anhang C.

4.3.3. Zuweisungen

Um Inhalte auf ausgewählten Geräten zu verarbeiten, gibt es Zuweisungen (Mappings), welche über die *DAC-Manager* API hinzugefügt werden können. Ein Mapping beinhaltet ein *Content Pattern*, ein *Device Pattern* und zusätzlichen Optionen. Wird ein Mapping über die API hinzugefügt, werden alle Inhalte die dem *Content Pattern* entsprechen auf allen Geräten die dem *Device Pattern* entsprechen verarbeitet. In den Optionen können zusätzliche Informationen zum Verarbeiten der Inhalte hinterlegt werden. Beispielsweise kann festgelegt werden, ob sich der Inhalt über einen bereits bestehenden Inhalt auf dem Device legen soll, im Hintergrund wartet bis der aktive Inhalt entfernt wurde oder parallel verarbeitet werden soll. Alle Mappings werden beim Hinzufügen von neuen Inhalten und Geräten und bei Veränderungen der Stati erneut überprüft. Im Falle einer Übereinstimmung werden die passenden Inhalte auf den Geräten verarbeitet. Dadurch kann ein Mapping nicht nur zum aktuellen Zeitpunkt dafür verwendet werden einen bestimmten Inhalt auf einem bestimmten Gerät wiederzugeben, sondern sie können präventiv eingesetzt werden um jegliche Art von Inhalt, der in der Zukunft einmal registriert wird und einem bestimmten Muster entspricht, auf den passenden Geräten zu verarbeiten.

4.4. Strategien & Filter

Strategien sind eigenständige Agenten und haben die Aufgabe Inhalte, durch das Erstellen und Entfernen von Mappings beim *DAC-Managers* auf geeigneten Geräten zu verarbeiten. Dafür können sie Informationen aus dem Kontext des LP's in ihre Entscheidungen mit einbeziehen. Für das Beispielszenario wurde ein *Following Strategy* implementiert, welche dafür zuständig ist, einen beliebigen Inhalt einer Person im LP folgen zu lassen. Die Kontextinformation ist hierbei die Position der jeweiligen Person, welche mit Hilfe des installierten Ubisense Systems ermittelt wird. Damit die Inhalte auch möglichst nur auf Geräten angezeigt werden, die von der Position der Person auch einsehbar sind, werden zusätzliche Function Spaces eingeführt. Für das Beispielszenario wurden diese zunächst sehr grob abgesteckt, sodass sich für das LP die in Abbildung Abbildung 4.4 eingezeichneten Bereiche ergeben, in denen sich eine Person aufhalten kann. Jedes Gerät verfügt über ein Statusattribut „Position", welches durch eine Koordinate aus Ubisense Bezugssystem repräsentiert wird. Dadurch lassen sich die registrierten Geräte eindeutig einem Function Space zuordnen.

Die *Following Strategy* berechnet bei jeder Positionsänderung der Person, das geografisch gesehen dichtesten Gerät im gleichen Functions Space, welches in der Lage ist den entsprechenden Inhalt zu verarbeiten. Anschließend fügt es ein Mapping für den Inhalt und das ermittelte Gerät über den *DAC-Manager* hinzu und entfernt gegebenfalls das bereits bestehende Mapping für das Gerät auf dem der Inhalt zuvor angezeigt wurde. Die Informationen für den Entscheidungsprozess der Strategien können beliebig komplex sein und lassen sich auf Grund der modularen Architektur sehr gut durch verkettete Filter realisieren. Ein Filter nimmt eine Liste von Inhalten und Geräten und grenzt diese anhand von Kontextinformationen aus dem LP ein. In dem zuvor genannten Beispiel könnte ein Filter dafür zuständig sein, zunächst aus allen registrierten Geräten nur die für den Inhalt geeigneten Geräte zu identifizieren. Der nächste Filter bestimmt aus den restlichen Geräten diejenigen, die sich im selben Function Space wie der Benutzer befinden. Der letzte Filter würde abschließend das geografisch dichteste Gerät berechnen. Nun muss die Strategie nur noch ein Mapping mit den entsprechenden Parametern hinzufügen und beim Hinzufügen und Entfernen von neuen Geräten beim *DAC-Manager* erneut überprüfen. Durch dieses Prinzip kann das System sehr gut um weitere Strategien und Filter erweitert werden und die Wiederverwendbarkeit von bereits vorhandenen Komponenten ist gewährleistet.

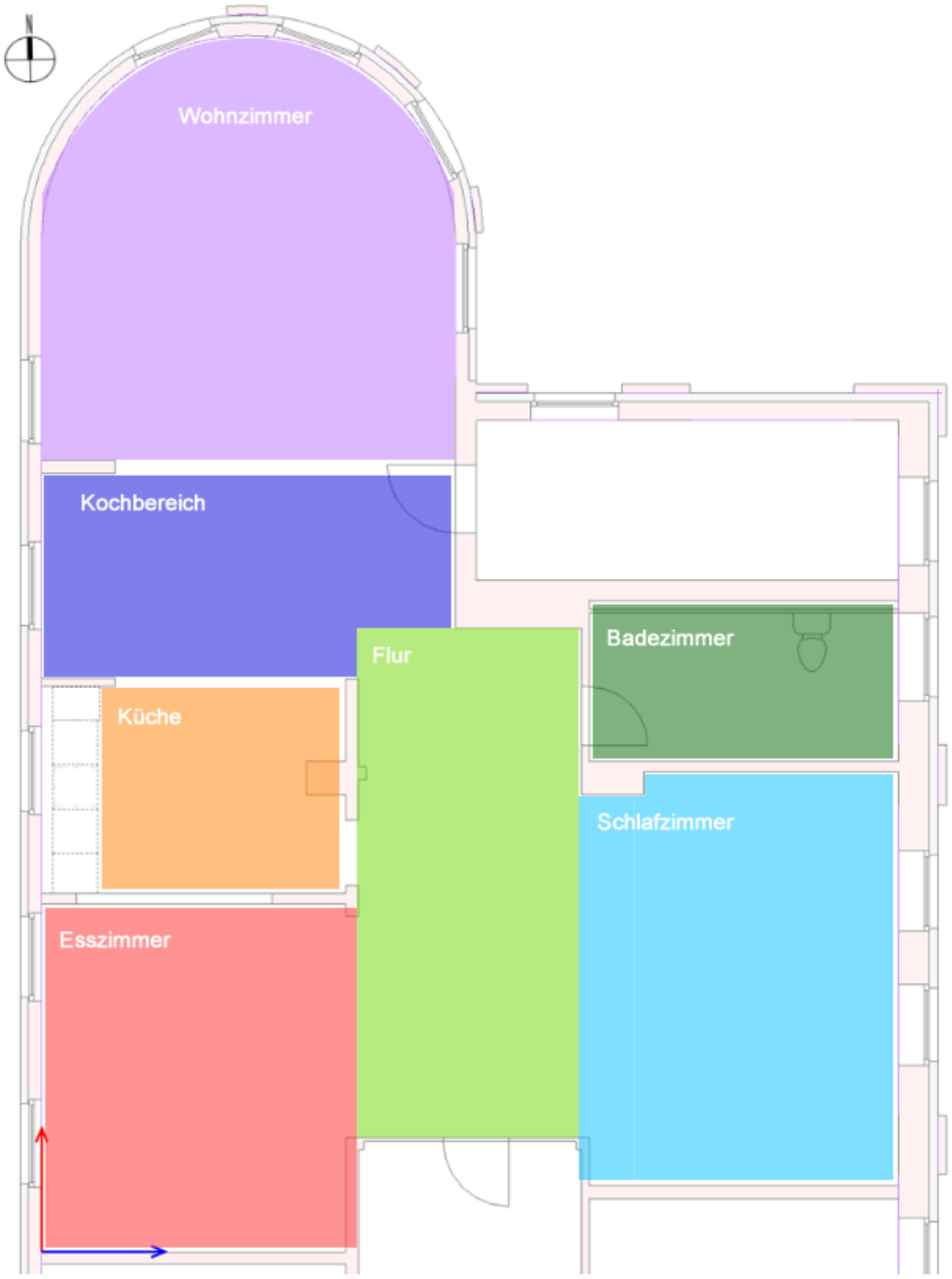

Abbildung 4.2.: Verschiedenen Function Spaces im Living Place

5. Evaluation

In diesem Kapitel wird ein im LP installiertes Beispielszenario beschrieben, mit dessen Hilfe die Ergebnisse des Projektes evaluiert werden. In dem Szenario folgt ein Video seinem Betrachter durch die Wohnung, indem es jeweils auf dem geeignetsten Anzeigegerät platziert wird. Sollte sich kein Gerät in der unmittelbaren Umgebung befinden, wird das Video pausiert und erst dann fortgesetzt, sobald ein passendes Anzeigegerät für den Betrachter verfügbar ist, welches in der Lage ist den Inhalt zu verarbeiten.

Für jeden im LP befindlichen Fernseher wurde ein Geräteagent implementiert, über den mit Hilfe eines Webbrowsers Videodateien wiedergegeben werden können. Außerdem wurde ein Media Streaming Server aufgesetzt, der RTMP Streams für verschiedene Videoformate ausliefern kann. Durch eine bereitgestellte API einer *Video Content Factory* können verschiedene Video Content Agenten gestartet werden, welche diese Streams als Inhalte im LP zur Verfügung stellen. Alle diese Agenten registrieren sich anschließend bei einer über die Middleware bereitgestellten Schnittstelle des DAC-Managers. Der *Following Strategy Agent* sorgt dafür, dass die erstellten Videoinhalte auf dem Fernseher angezeigt werden, welcher einem ausgewählten Ubisense Tag am nächsten gelegen ist. Dafür empfängt er von einem *Position Tracking Service* im LP JSON Nachrichten, sobald sich die Position des entsprechenden Tags ändert. Über den DAC-Manager lässt sich die Spezifikation der Anzeigegeräte abrufen und deren Position bestimmen. Mit diesen Informationen wird das nächstgelegene Anzeigegerät ermittelt und über ein *Mapping* der anzuzeigende Inhalt auf dem ausgewählten Gerät abgespielt. Das folgende Sequenzdiagramm (vgl. Abbildung 5) gibt noch einmal einen groben Überblick über den Verlauf des Beispielszenarios:

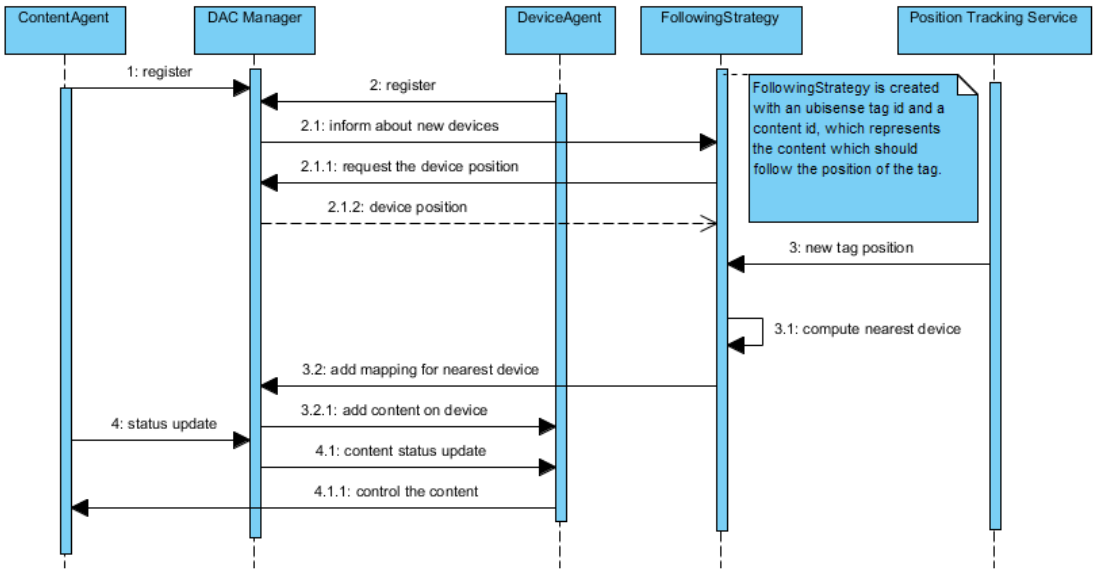

Abbildung 5.1.: Überblick über den Ablauf des Beispielszenarios

Dadurch, dass jeder Inhalt einen eigenen Zustand besitzt, ist es auch möglich ihn gleichzeitig auf mehreren Geräten abzuspielen. Das Video kann also auf beiden Fernseher gleichzeitig abgespielt werden, wobei Steuerungsbefehle zum pausieren oder vor spulen simultan auf allen Anzeigegeräten gleichermaßen ausgeführt werden.

Mit Hilfe des Beispielszenarios konnte die Funktionalität der Architektur nachgewiesen und die Umsetzung der funktionalen Anforderungen überprüft werden. Der *DAC-Manager* ermöglicht einen Überblick über die im LP vorhandenen Inhalte und Geräte und es wurde eine zentrale Schnittstelle zur Präsentation von Video Inhalten im LP geschaffen. Da die Architektur auf der neuen Middleware basiert, integriert sich das neue System besonders gut in das bereits bestehende Netzwerk von Agenten. Auf Grund der Trennung in unterschiedliche Komponenten und die Konzeption der Middleware (vgl. Kapitel 2.1) führt dies automatisch zu einer losen Kopplung der Agenten und erhöht deren Wiederverwendbarkeit.

6. Schlussbetrachtung

Abschließend wird der Projektbericht kurz zusammengefasst und ein Ausblick über das weitere Vorgehen in diesem Projekt gegeben.

6.1. Zusammenfassung

Das Ziel des Projektes ist es eine zentrale und lose gekoppelte Schnittstelle zur intelligenten Verwaltung von Inhalten und Anzeigegeräten im LP zu etablieren. In dem Bericht wurden die dafür verwendeten und entwickelten Komponenten vorgestellt und die Anforderungen an das Gesamtsystem aufgezeigt. Durch die Architektur der Komponenten als einzelne Agenten und deren Kommunikation über die Middleware des LP's konnte ein hohes Maß an Wiederverwertbarkeit und Erweiterbarkeit erreicht werden. Die Funktionalität wurde durch die Implementierung eines ersten Beispielszenarios sichergestellt,

6.2. Ausblick

Durch das Beispielszenario konnten, weitere hilfreiche Funktionen für das System identifiziert werden, welche in der Implementierung der ersten Version nicht berücksichtigt wurden. Zum Beispiel die Unterstützung einer Zuordnung von Inhalten und Geräten zu bestimmten Benutzern oder die Umwandlung von bestimmten Inhalten zur Verarbeitung auf anderen Gerätetypen. Außerdem soll die Entwicklung eines Framework und die Einführung eines Plugin-Systems dazu beitragen die Entwicklung neuer Geräte und Inhalte für andere Entwickler komfortabler zu gestalten.

In der Zukunft sollen genauere Tests durchgeführt werden, um das Verhalten des Systems im LP besser zu überwachen. Damit auch die nicht funktionalen Anforderungen getestet werden können, müssen weitere Inhalte und Geräte erstellt werden, um die Last des Systems und deren Latenzen zu testen. Durch die Einführung neuer Inhalte und Geräte lässt sich dann auch die Erweiterbarkeit des Systems überprüfen. Ein für das kommende Projekt geplanter Inhalt ist die virtuelle Fernbedienung des LP's, welche angezeigt werden soll.

Außerdem könnte die Anbindung des Ubisense Systems an die Middleware optimiert werden, indem der Umweg über den ActiveMQ Server entfernt wird und die Nachrichten direkt in einer Gruppe der Middleware publiziert werden. Die neue Komponente könnte allgemeine Funktionen bereitstellen, wie beispielsweise die Möglichkeit gezielt Positionen von Ubisense Tags abzufragen, da diese Informationen momentan nur bei einer Positionsänderung verschickt werden. Die Funktionalität Positionsänderungen zu abonnieren, sobald sie einen individuell definierbaren Wert überschreiten, gäbe den Agenten die Möglichkeit, die Granularität in der sie über Änderungen benachrichtigt werden selbst zu bestimmen.

A. Beispiel zur Spezifikation von Geräten

```
{
 "type":"AddDevice",
 "spec":{
  "attributes":{
   "id":{
    "type":"StringValue",
    "value":"fernseher"},
   "video-information":{
    "type":"MapValue",
    "value":{
     "colorful":{
      "type":"BooleanValue",
      "value":true},
     "resolution":{
      "type":"MapValue",
      "value":{
       "width":{
        "type":"IntValue",
        "value":1920},
       "height":{
        "type":"IntValue",
        "value":1080}}},
     "refreshrate":{
      "type":"IntValue",
      "value":400}}},
   "audio-information":{
    "type":"MapValue",
    "value":{
     "speakertype":{
      "type":"StringValue",
      "value":"7.1"}
    }
   }
  }
 }
}
```

B. Beispiel zur Spezifikation von Inhalten

```
{
 "type":"AddContent",
 "spec":{
  "attributes":{
   "id":{
    "type":"StringValue",
    "value":"video1"},
   "provider":{
    "type":"StringValue",
    "value":"Youtube Content Provider"},
   "video-information":{
    "type":"MapValue",
    "value":{
     "codec":{
      "type":"StringValue",
      "value":"MPEG-4 (XVID)"},
     "resolution":{
      "type":"MapValue",
      "value":{
       "width":{
        "type":"IntValue",
        "value":720},
       "height":{
        "type":"IntValue",
        "value":296}}},
     "framerate":{
      "type":"IntValue",
      "value":24
     }
    }
   }
  }
 }
}
```

C. Beispiel für die Definition eines Patterns

```
{
 "type":"AddMapping",
 "pattern":{
  "type":"MappingPattern",
  "silent":false,
  "device":{
   "type":"Pattern",
   "spec":{
    "video-information":{
     "type":"MapPattern",
     "value":{
      "resolution":{
       "type":"MapPattern",
       "value":{
        "aspectratio":{
         "type":"Equals",
         "value":{
          "type":"StringValue",
          "value":"16:9"}},
        "width":{
         "type":"GreaterEquals",
         "value":{
          "type":"IntValue",
          "value":1024
         }
        }
       }
      }
     }
    }
   }
  }
 }
}
```

Literaturverzeichnis

[Eichler 2013] Eichler, Tobias: Entwicklung einer Middleware für das Living Place (Projekt 1). (2013)

[Eichler 2014] Eichler, Tobias: Entwicklung einer Middleware für das Living Place (Projekt 2). (2014). – URL https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2014-proj/eichler.pdf

[Karstaedt 2012] Karstaedt, Bastian: *Kontextinterpretation in Smart Homes auf Basis semantischer 3D Gebäudemodelle*, Diplomarbeit, 2012. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/karstaedt.pdf

[Voskuhl 2011] Voskuhl, Sören: Modellunabhängige Kontextinterpretation in einer Smart Home Umgebung. (2011). – URL https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/voskuhl.pdf