Survey: Service Discovery in der Smart City

Tristan Damm, Ellen Landschoof, Lukas Obermann, Louis Richter, Tjorben Wade

26. Juli 2022

Inhaltsverzeichnis

Abstract

Die Service Discovery (SD) in der Smart City beschreibt ein Verfahren, womit IoT-Systeme in der Stadt öffentlich dem Bürger zugänglich gemacht werden. Um dieses Verfahren zu ermöglichen, gibt es verschiedene SD-Ansätze, wovon drei (Semantisch, Kontextabhängig und Quality-of-Service-basiert) genauer betrachtet werden. Diese Verfahren beschreiben aber eine abstrakte Definition und keinen vordefinierten Standard. Somit sind diese eher ungeeignet oder mit sehr viel Aufwand verbunden, um eine SD in der Smart City zu realisieren. Webstandards, wie das von der W3C definierte Web of Things (WoT), beschreibt bereits eine genaue Architektur für eine SD. Mithilfe von Thing Descriptions (TDs) werden Metadaten mit Things verknüpft, welche dann von einem Service zur Verfügung gestellt werden. Die Punkte Sicherheit und Datenschutz spielen dabei auch eine Rolle, werden aber nicht genauer betrachtet, da sie für eine öffentliche SD nur teilweise relevant sind. Das WoT liefert, auch wenn die Spezifikation noch nicht vollständig ist, ein genaues Verfahren, womit eine SD in der Smart City realisiert werden kann. Es bestehen aber immer noch Probleme bei der Definition von Ontologien und von Quality-of-Service-Aspekten (QoS) für die SD.

1 Einleitung

Über die letzten Jahre haben viele Städte damit begonnen, unterschiedliche Sensoren zu verbauen. Diese werden unter anderem dazu genutzt, um zu prüfen, ob ein Parkplatz frei ist, wie gut oder schlecht die Luftqualität ist oder wie warm es aktuell ist [5]. Darüber hinaus können viele weitere Werte gemessen werden.

Bei den so entstandenen Netzen handelt es sich zumeist um sogenannte Low Power Wide Area Networks (LPWAN). Diese können auf Basis unterschiedlicher Technologien aufgebaut sein und funktionieren daher nicht alle einheitlich. Das hat zur Folge, dass in den meisten Fällen nur der Ersteller des Netzes Zugriff auf die Daten hat. Eine Interaktion zwischen mehreren Netzen, um genauere Daten zu erzielen oder Daten besser verteilen zu können, wird deutlich erschwert.

Aus der Open-Data-Strategie der Bundesregierung geht hervor, dass die meisten der so gesammelten Daten von den Städten öffentlich zur Verfügung gestellt werden müssen [4]. Wie dies genau geschieht, ist den Städten selbst überlassen. Viele Daten werden unter `https://www.govdata.de/` zur Verfügung gestellt, aber es gibt auch andere Quellen. Zumeist müssen die Daten dabei manuell heruntergeladen werden.

Das Format der Daten unterscheidet sich ebenfalls. Während sie meisten als CSV-Datei veröffentlicht werden, sind auch PDF-, XML- und HTML-Dateien sehr häufig vertreten. Für einen großen Teil der Daten auf GovData ist zudem kein Dateityp angeben.

All das macht es schwierig, die Daten in Anwendungen einzubinden und so für den Alltag nutzbar zu machen.

Dabei helfen könnte unter anderem eine Schnittstelle, die die Daten von unterschiedlichen Sensoren für Geräte auffindbar macht. Die größten Probleme, die sich dabei stellen, sind die Service Discovery und die verwendete Sprache, um die Sensoren zu beschreiben oder zu finden. Dabei können Begriffe, die von Menschen leicht verknüpft werden können, von Maschinen nicht in Verbindung gebracht werden, was dazu führt, dass Services gegebenenfalls nicht gefunden werden. Dieses zweite Problem bezieht sich daher auf die Ontologie.

In diesem Bericht werden einige Rechercheergebnisse beschrieben, die für die Arbeit in diesem Bereich von Bedeutung sind. Dazu werden in Abschnitt 2 zunächst verwandte Arbeiten beschrieben. Anschließend werden in Abschnitt 3 einige Arten der Service Discovery dargestellt und in Abschnitt 4 auf das Web of Things eingegangen, bevor in Abschnitt 5 die Ergebnisse diskutiert werden. Abschließend wird in Abschnitt 6 ein Fazit gebildet.

2 Related Works

Es gibt zahlreiche Literatur im Bereich der SD, in denen die verschiedene Methoden entweder allgemein oder speziell für einen bestimmten Anwendungsbereich beschrieben werden.

Die semantische SD bietet Vorteile in der Verwendung von Ontologien [17] und findet auch im Bereich von WoT ihren Verwendungszweck [21]. Es gibt aber auch einen Ansatz der in den Bereichen der Zuverlässigkeit und Sicherheit ihren Nutzen findet [15]. Diese wird QoS-basierte SD [14] genannt und wurde unter anderem von Kosunalp und Demir behandelt. Die Kontextabhängige SD [22] arbeitet stattdessen mit den Informationen aller relevanten Entities.

Die SD ist ein wichtiger Punkt in den Internet of Things (IoT)-Systemen und einen guten Einblick in dieses umfangreiche Themenfeld bietet die Arbeit [2] von Achir u. a.

Für das WoT existiert ein sehr aktuelles und umfassendes Paper von Sciullo u. a., welches auf alle Ansätze der letzten Jahrzehnte eingeht und in diversen Kategorien systematisch analysiert und vergleicht [20]. Auf der Webseite des World Wide Web Consortium (W3C) finden sich verschiedene Spezifikationen und Richtlinien zum Thema WoT [11, 10, 6, 18].

In dieser Arbeit werden die für uns relevantesten Verfahren kurz beschrieben, was als eine Recherchearbeit für ein Projekt im Bereich der IoT-Discovery dient.

3 Service Discovery

Das IoT hat heutzutage einen großen Einfluss auf unseren Alltag. Über eine Vielzahl von Sensoren werden dabei eine Menge an Daten aufgezeichnet, die anschließend in den unterschiedlichsten Services zur Verfügung gestellt werden. Dabei ist das Auffinden und Auswählen des richtigen Services nicht immer ganz einfach. Es haben sich daher unter-

schiedliche Arten der SD entwickelt, die jeweils für unterschiedliche Bereiche besonders gut geeignet sind [1].

Im folgenden Abschnitt werden drei Arten der SD genauer beschrieben, die am bedeutendsten sind, wenn es darum geht, Daten aus LPWAN von Städten, wie in der Einleitung beschrieben, über eine Schnittstelle für Maschinen zugänglich zu machen. Dabei handelt es sich um die semantische SD, die Quality-of-Service-basierte SD und die Kontextabhängige SD.

3.1 Semantisch

Das Ziel der semantischen SD ist es, den Service, der am besten zu den gegebenen Ansprüchen passt, mithilfe einer Ontologie zu finden. Eine Ontologie stellt dabei die Zusammenhänge zwischen verschiedenen Begriffen oder Eigenschaften dar, die in der Suche genutzt werden können. Die Begriffe selbst werden ebenso definiert und beschrieben, um für das jeweilige Themengebiet ein einheitliches Vokabular nutzen zu können. Auf diese Weise ist es möglich, ein gemeinsames Verständnis über ein bestimmtes Gebiet zu schaffen. Dabei können im Gegensatz zu den meisten anderen Systemen Services nicht nur funktionale, sondern auch nicht funktionale Eigenschaften – wie Performance, Kosten oder Zuverlässigkeit – beachtet werden.

Zum Erstellen einer semantischen Service Discovery gibt es keine festgelegte Standardisierung, die verfolgt werden muss. Bei bisherigen Arbeiten wurde daher jeweils ein eigener Ansatz entworfen und umgesetzt. Diese Ansätze können mehr oder weniger stark voneinander abweichen.

In der Arbeit von Suraci, Mignanti und Aiuto [23] wird eine kontextbewusste Architektur vorgestellt, die für eine Service Discovery genutzt werden kann. Die Architektur selbst ist dabei nicht abhängig von der verwendeten Technologie und kann mit vielen vorhandenen Service-Discovery-Protokollen genutzt werden.

Im Kontrast dazu handelt die Arbeit von Iqbal u. a. [7] davon, wie Services, spezifisch mit SAWSDL und SPARQL, beschrieben werden können. Auch Ben Mokhtar u. a. [3] bauen bei ihrer Arbeit auf spezifische Sprachen auf. Sie nutzen Amigo-S für semantische Spezifikationen und S-Ariadne, eine Erweiterung von Ariadne, als Discovery-Protokoll. Ihr Ziel war es, einen effizienten Weg zum Finden von Services zu entwickeln.

In weiteren Arbeiten geht es häufig darum, wie und auf Basis welcher Grundlagen passende Services für die Nutzer ausgewählt werden können. Majithia u. a. [16] haben so beispielsweise ein Framework entwickelt, dass Services basierend auf ihrem Ruf findet. Zu dem Ruf eines Services gehört dabei unter anderen dessen Zuverlässigkeit und Erreichbarkeit. Jia, Li und Zhou [8] schlagen hingegen ein zentralisiertes, mehrstufiges semantisches Service-Matching vor, das in vier Schichten abläuft, und Zhao u. a. [24] legen ihren Fokus auf die Beschreibung von IoT-Services und deren Ähnlichkeit zueinander. Dabei werden die Ähnlichkeiten gemessen und ein Cluster erstellt, um ähnliche Services zu sammeln.

Ein Vorteil der semantischen Service Discovery ist, dass Ontologien mit in Betracht gezogen werden können. Aufgrund der fehlenden Standardisierung müssen Arbeiten jedoch von Grund auf eigenständig entwickelt werden. Das kann Vorteile mit sich bringen, aber ist gleichzeitig mit einer großen Arbeitslast verbunden und nicht für alle Projekte geeignet.

3.2 Quality-of-Service-basiert

Bei der Quality-of-Service-basierten (QoS) SD stehen nichtfunktionale Bewertungskriterien im Vordergrund. Diese werden auf die Sicht des Anwendenden bezogen und können als Güte für deren Anforderungen verstanden werden.

Die verschiedenen QoS-Ansätze werden für die SD in verschiedene Unterkategorien sortiert [1]. Je nach Optimierungsziel des Ansatzes wird dieser folgenden Kategorien zugeordnet:

- Sicherheitsbasiert
- Energieverbrauchsbasiert
- Netzinfrastrukturbasiert

Des Weiteren wird eine Hybrid-Kategorie gebildet, um Ansätze, die mehrere Faktoren abdecken, aufzunehmen. Eine QoS-basierte SD verfolgt verschiedene Ziele. Die Geschwindigkeit bei der Suche soll z. B. möglichst schnell ablaufen. Algorithmen, die sich auf dieses Ziel konzentrieren, werden daher in die Netzinfrastruktur-Kategorie eingeordnet. Andere Algorithmen, die den Prozess hinsichtlich der Sicherheit der Suche z. B. mithilfe von Authentifizierungen verbessern, können für diese Arbeit vernachlässigt werden, da es sich um einen Open-Data-Standard handeln soll. Die Kategorie für den Energieverbrauch hat, zusätzlich zur Netzinfrastruktur, eine hohe Bedeutung für den Bereich dieser Arbeit. Es muss darauf geachtet werden, dass die Services möglichst wenige und kurze Berechnungen zur SD durchführen müssen.

Die QoS-basierte SD nutzt unterschiedliche Verarbeitungsinformationen, basiert jedoch im Grundsatz auf vorhandenen SD-Ansätzen, wie z. B. dem semantischen Ansatz. Das Finden der Dienste basiert häufig auf den vorhandenen Ansätzen. Nachdem ein Dienst gefunden wurde, fungiert die QoS-basierte SD als eine Art QoS-Filter [14]. Dadurch werden Dienste, die nicht verfügbar oder nach vorhandenen Information nicht geeignet sind, eliminiert.

SARL [14] ist ein Konzept, das die QoS-basierte SD nutzt. Es basiert auf einem P2P-Konzept, das noch keine QoS-Faktoren abdeckt. Die einzelnen Services kennen bei diesem Ansatz nur die Art der Daten, die sie selbst bereitstellen, und Wege zu Nachbarknoten. Es gibt verschiedene Ansätze, den Weg zum Zielknoten (Zieldienst) zu finden.

Ein nicht QoS-basierter Ansatz ist, die Suchanfrage ohne zusätzliche Informationen zu nutzen, von Knoten zu Knoten zu schicken, bis sie beim Ziel ankommt. Dadurch entsteht

ein Nachrichtenüberfluss, der die Geschwindigkeit verlangsamt und somit für Nutzende im Hinblick auf die Zuverlässigkeit wahrnehmbar ist.

Bei einem QoS-basierten Ansatz werden zur Weiterleitung zwischen den Knoten (Diensten) verschiedene Informationen herangezogen. Es wird ein QoS-Ranking gebildet, das die Rückgabe eines Dienstes bestimmt. Dieses basiert auf Informationen von vorherigen Anfragen oder selbstständig gesammelten Informationen. Damit verbessert der SARL-Ansatz die QoS hinsichtlich der Netzinfrastruktur.

Die Schwierigkeit bei Verbesserungen hinsichtlich der QoS besteht darin, Mittelwege zwischen aktuellen Informationen über andere Knoten und veralteten Daten zu finden. Die Netzwerklast kann beispielsweise nicht nur von zu vielen unstrukturierten Abfragen, sondern auch von zu vielen Abfragen, die den Status einzelner Knoten erfassen, erhöht werden. Dies hat eine Verletzung der QoS-Faktoren zur Folge.

3.3 Kontextabhängig

Dieses Kapitel beschäftigt sich mit dem Thema Kontextabhängige Service Discovery und erklärt die verschiedenen Zusammenhänge, die damit verbunden sind. Zunächst wird jedoch der einzelne Begriff Kontext beschrieben. Dieser wird in vielen Literaturen definiert, aber in dieser Arbeit wird der englische Begriff Kontext in dem Zusammenhang mit Kontext-Awareness verwendet und sollte nicht mit direkt mit dem deutschen Begriff Kontext verwechselt werden.

Eine sehr gute und allgemeine Definition des Begriffes Kontext lautet: „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.“ [22]

Bezogen auf unsere Arbeit bedeutet dies, dass der Kontext jede Information ist, die zur Beschreibung einer relevanten Entität verwendet werden kann. Außerdem muss der Kontext für die SD so strukturiert sein, dass alle beteiligten Entitäten darin zusammengefasst und verwaltet werden können.

Hier nutzt die SD einen Kontext, welcher für jeden einzelnen Anwendungsbereich neu modelliert wird. Dafür muss im Vorfeld recherchiert werden, welche Informationen die einzelnen Entitäten liefern können und für die Interaktion wichtig sind. Ist zum Beispiel ein Sensor in der Interaktion beteiligt, so muss die Frage geklärt werden, ob der Messwert als Information ausreicht, oder ob zusätzliche Informationen wie Hersteller, Standort, oder Größe eine wichtige Rolle spielen.

Ein weiterer wichtiger Punkt ist die Art und Weise, wie die SD alle erforderlichen Informationen abfragen kann. Häufig besitzen die Entitäten dafür eine vordefinierte Schnittstelle, welche die bei einer Abfrage die gewünschten Werte zurückliefert. Eine weitere Möglichkeit wäre das manuelle Eintragen von Werten. Das sollte allerdings vermieden werden, damit das System autonom bleibt. Eine mögliche Ausnahme können statische Werte sein, die sich ohnehin nicht ändern.

Haben wir ein System, das solch einen Kontext berücksichtigt, dann wird von Context-Awareness gesprochen. Es nutzt bei Anfragen eines Benutzers die vorhandenen Informationen, um eine bestmögliche Antwort zu liefern. In dem Bereich der SD werden so gezielt Services lokalisiert und genutzt.

4 Web of Things

Ein Problem bei der Service Discovery im IoT sind immer die Interoperabilität und Kompatibilität zwischen verschiedenen Systemen. Ein Bereich, in dem diese Aspekte von großer Relevanz sind, ist das Internet. Webtechnologien müssen für eine Vielzahl von Geräten und Anwendungen funktionieren. So wurde das Internet auch als Inspirationsquelle für die Lösung solcher und anderer Probleme im IoT genutzt. Diese Kombination wird im Allgemeinen als WoT bezeichnet.

Das WoT ist eine Erweiterung des IoT, die grundsätzlich die Things (oder deren Intermediaries) um eine Web-Server-Komponente erweitert, mit der Kommunikation von außen über einen Uniform Resource Identifier (URI) oder Internationalized Resource Identifier (IRI) initialisiert und durchgeführt werden kann.

Im Laufe der Jahre haben sich dabei die WoT-Spezifikationen und -Empfehlungen des W3Cs, welches auch die Spezifikationen für andere Webtechnologien herausbringt, zu den meistgenutzten und vollständigsten entwickelt [vgl. 20, S. 47573], weshalb diese in den nachfolgenden Unterkapiteln näher betrachtet werden.

Es gibt verschiedene Spezifikations- und Richtliniendokumente, die das W3C bereitstellt.

- Die *Architecture* [11] gibt einen Überblick über das WoT, welches vom W3C standardisiert bzw. empfohlen wird.
- Die *Thing Description* [10] ist die Spezifikation für das Format, in dem einzelne Web Things beschrieben werden.
- Die *Binding Templates* [13] sind Richtlinien, wie Netzwerkschnittstellen definiert werden sollen.
- Die *Scripting API* [12] beschreibt eine (optionale) JavaScript-API, die für eine Anwendung genutzt werden kann, die auf das WoT zugreifen können soll.
- Die *Discovery* [6] beschreibt, wie sich sowohl Things nach außen bekannt machen können, als auch Anwendungen nach existierenden Things erkundigen können. Zum Zeitpunkt der Erstellung dieses Berichts ist dieses Dokument jedoch noch nicht in einem fertigen Zustand gewesen.
- Die *Security und Privacy Guidelines* [18] enthalten Richtlinien zur sicheren Implementierung und Konfiguration von Things sowie möglicherweise auftretende Probleme und deren Behebung.

Es gibt noch weitere Dokumente, die jedoch nicht relevant für den Anwendungszweck der Service Discovery sind und daher hier nicht angeführt werden. Sie können unter `https://www.w3.org/TR/?tag=wot` eingesehen werden.

4.1 Architekturüberblick

Für das WoT gibt es verschiedenste Anwendungsfälle. All diese haben Gemeinsamkeiten und Unterschiede, aus denen sich die Anforderungen an eine WoT-Architektur [vgl. 11] ergeben.

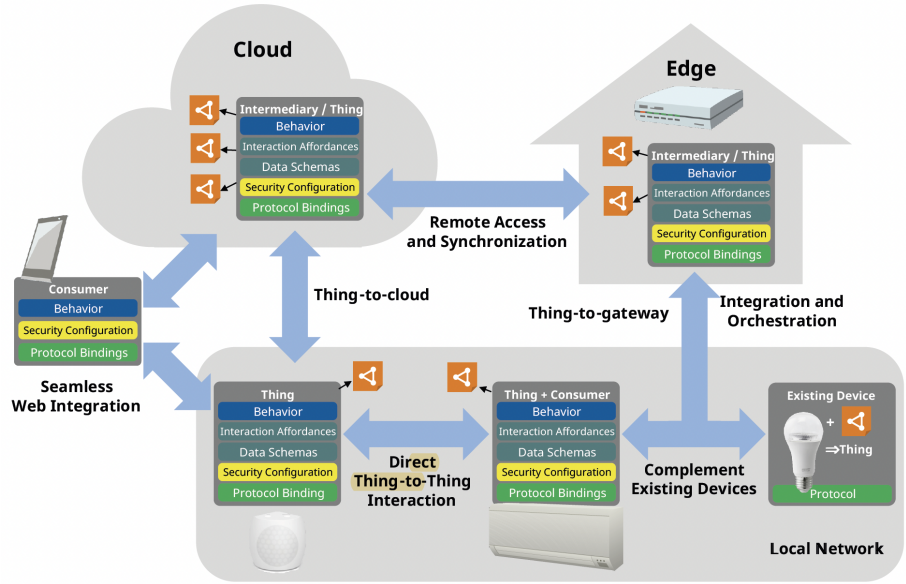

Abbildung 1: WoT-Architektur

Durch die starke Heterogenität der Anwendungen und Geräte, aber auch der bereits existierenden IoT-Systeme, entsteht eine Vielzahl von Anforderungen. Diese lassen sich häufig in zwei Kategorien einteilen: Positive Anforderungen, die in allen betroffenen Entitäten umgesetzt werden können, oder negative Anforderungen, die nicht in allen betroffenen Entitäten umgesetzt werden können und wodurch die Anforderung aus einer Unabhängigkeit von diesem Teilaspekt besteht. Im Zentrum stehen vier Kernelemente: Flexibilität, Kompatibilität, Skalierbarkeit und Interoperabilität.

Die Architektur muss demnach hardwareunabhängig sein. Auch bestimmte Technologien können nicht vorausgesetzt werden, denn es muss Kompatibilität zu allen vorhandenen Geräten und Systemumgebungen sichergestellt werden. Dadurch fokussiert sich das WoT des W3Cs auf höhere Systemebenen und somit vor allem die Kommunikationsebene.

Auf dieser Basis wird die Architektur eines Web Things in verschiedene Bereiche unterteilt. Neben dem eigentlichen *Verhalten* sind das *Interaktionsaffordanzen*, die dazugehö-

rigen *Datenschemata*, die *Sicherheitskonfiguration* und *Protokollbindungen*. Dabei sind die Protokollbindungen die Definition, wie konkret Interaktionen durchgeführt werden können, während die Affordanzen nur beschreiben, was möglich ist, unabhängig von Protokollen oder Datenkodierung. Das *Interaktionsmodell* beschreibt dabei neben Navigation drei weitere Typen von Interaktionsaffordanzen:

- *Eigenschaften* bilden den Zustand eines Things ab.
- Mittels *Aktionen* kann der Zustand eines Things oder eine Funktion auf diesem geändert bzw. aufgerufen werden.
- *Ereignisse* finden bei Zustandsänderungen statt und können Daten asynchron an Consumer senden.

Dabei werden die einzelnen Interaktionsschnittstellen durch Verknüpfungen in Form von URIs und optionale Zusatzbeschreibungen maschinenlesbar definiert.

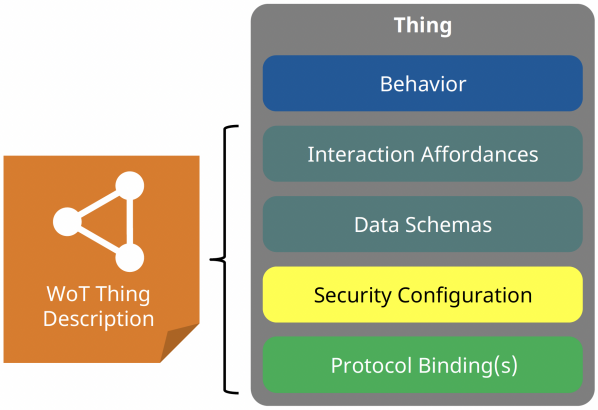

Abbildung 2: Thing-Architektur

Jedes Thing (oder dessen Intermediary) muss diese Funktionalitäten und Metadaten über sich bereitstellen können. Dazu gehören die bereits erwähnten Interaktionsaffordanzen, Datenschemata, die Sicherheitskonfiguration und die Protokollbindungen, neben dem Namen oder der Version des Things aber nicht zuletzt auch verwandte Things. Diese Eigenschaften sollen sowohl menschenlesbar als auch maschinenlesbar sein, semantisch annotiert werden sowie, falls zutreffend, Internationalisierung unterstützen können. Die Beschreibung dieser Eigenschaften wird TD genannt. Die TD wird als JavaScript Object Notation (JSON) übermittelt und dient als Einstiegspunkt für ein Thing.

Im Idealfall wird eine TD direkt auf dem jeweiligen Thing erstellt und bereitgestellt. Domänenspezifisches Vokabular wird dabei zwar nicht durch die WoT-Spezifikationen des W3C abgedeckt, da die TD aber JSON for Linked Data (JSON-LD) implementiert, kann beliebiges Vokabular als Kontext eingebunden werden. Eine genauere Beschreibung der Thing Description ist in Unterabschnitt 4.2 zu finden.

Web Things müssen verschiedene Protokolle unterstützen können. Dazu zählen nicht nur Internetprotokolle, sondern auch Protokolle, die im Local Area Network (LAN) genutzt werden. Außerdem sollten auch Kombinationen von Protokollen genutzt werden können. So soll grundsätzlich mit REST-APIs gearbeitet werden, jedoch soll auch der Datenaustausch mit dem PubSub-Muster möglich sein.

Damit mit einem Web Thing – also der Repräsentation eines Things im Web – interagiert werden kann, muss ein Consumer aber von diesem wissen. Dafür muss nach dem Web Thing gesucht werden können. Diese Suche muss syntaktisch (Attribute, ...) und semantisch (Funktionalitäten auf Basis eines einheitlichen Vokabulars) über verschiedene Formate möglich sein. Es kann hilfreich sein, für die Erkundung ein Verzeichnis zu nutzen, bei dem sich die einzelnen Web Things automatisiert oder manuell registrieren können und welches darauf basierende Suchanfragen bearbeiten kann. (Siehe dazu auch Unterabschnitt 4.3.)

Die TD umfasst auch Angaben zur Sicherheit nach außen. Dazu gehören z. B. Authentifizierungsmethoden wie Basic oder OAuth2.0. Grundsätzlich sollten (laut der Spezifikation) TDs nur autorisierten Personen zugänglich gemacht werden, da die enthaltenen Metadaten potenziell sensible Informationen beinhalten können. Trotzdem – oder gerade deswegen – sollten die Personen, die TDs erstellen oder verwalten, darauf achten, dass nur öffentliche Sicherheitsinformationen – wie die jeweilige Authentifizierungsmethode – in einer TD vorkommen.

Barrierefreiheit ist im Rahmen der Spezifikationen kein Thema, da es um die Kommunikation zwischen Anwendungen und Geräten geht und Personen nicht direkt auf das WoT zugreifen. Allerdings gilt es zu bedenken, dass mit entsprechenden Annotationen möglicherweise Code oder ganze Interfaces generiert werden könnten, sodass die Integration von Barrierefreiheit durchaus auch Auswirkungen auf andere Bereiche hätte.

Weitere Anforderungen, die jedoch nicht zur Thematik der Service Discovery beitragen, können in *Web of Things (WoT) Architecture* [11] nachgelesen werden.

4.2 Thing Description

Die Thing Description (TD) [10] hat, wie in Unterabschnitt 4.1 bereits oberflächlich beschrieben, vier Hauptbestandteile:

- Beschreibende Metadaten über das Thing;
- Interaktionsaffordanzen, die beschreiben, wie man das Thing nutzen kann;
- Schemata, die beschreiben, wie die Daten aussehen (müssen), die entweder an das oder von dem Thing geschickt werden;
- Verweise auf verwandte Things oder Dokumente.

Eine TD wird in JSON verfasst und folgt dem JSON-LD-Standard. Dabei existieren verschiedene Vokabulare für die unterschiedlichen Bereiche des Informationsmodells, die als Kontext verwendet werden können.

Jedes Vokabular enthält im Grunde Definitionen von Datenstrukturen, die im objektorientierten Kontext als Objekte verstanden werden können. Das Kernvokabular definiert dabei den grundsätzlichen Aufbau einer TD, während es dabei die anderen spezifischeren Vokabulare für ihre jeweiligen Teilbereiche referenziert.

4.2.1 Kernvokabular

Das Kernvokabular deckt das Interaktionsmodell aus Eigenschafts-, Aktions- und Ereignisinteraktionsaffordanzen ab.

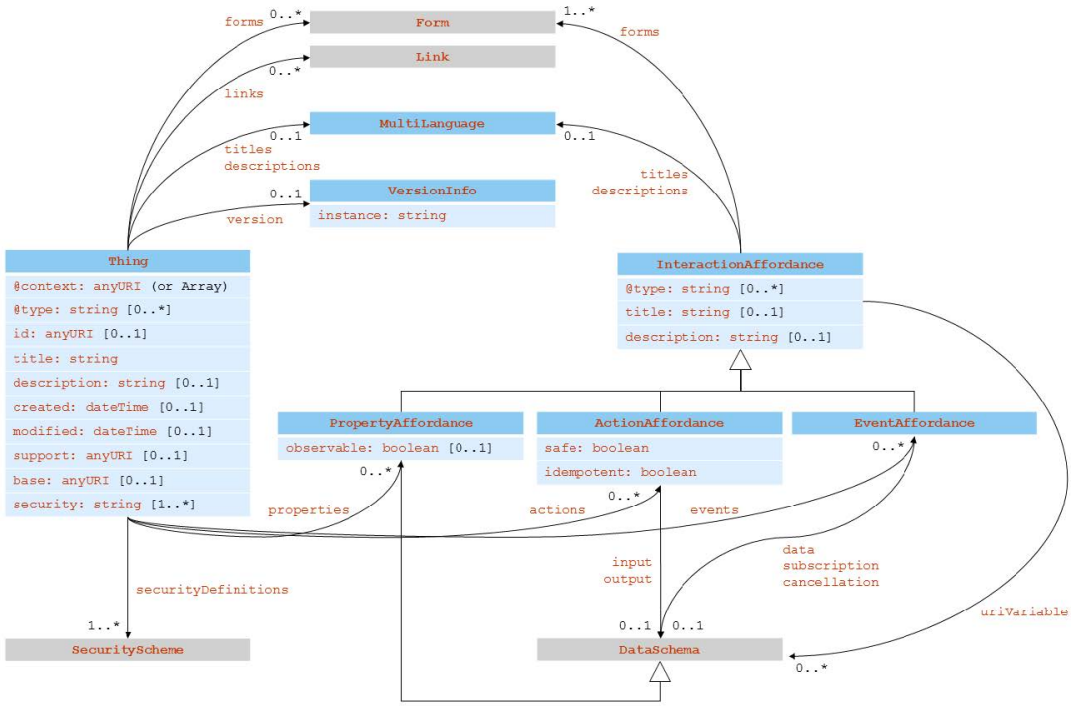

Abbildung 3: TD Kernvokabular

Das `Thing` ist dabei das Grundobjekt, welches die Wurzel eines TD-Dokuments definiert. Dazu gehören der URI des Things, JSON-LD-Annotationen, Beschriftungen, Metadaten, vor allem aber die einzelnen Bereiche des Interaktionsmodells, Links zu anderen Things, Hypermediakontrolldefinitionen sowie Sicherheitsmechanismen des Things.

Es gibt drei Arten von `InteractionAffordance`s:

- `PropertyAffordance`s
- `ActionAffordance`s

- `EventAffordances`

Allen ist gemein, dass sie Beschriftungen haben, Hypermediakontrollen, die beschreiben, wie mit ihnen interagiert werden kann, sowie Schemata für Werte, die übergeben werden müssen. Für eine `PropertyAffordance` kann zusätzlich definiert werden, ob diese überwacht werden kann. Bei einer `ActionAffordance` kann noch definiert werden, wie Eingabe- und Ausgabedaten aussehen sollen und wie sich die Aktion verhält. Bei einer `EventAffordance` können stattdessen die Daten definiert werden, die bei der Abonnierung und Deabonnierung benötigt werden sowie die, die dann erhalten werden.

4.2.2 Datenschemavokabular

Das Datenschemavokabular leitet sich aus der JSON-Schema-Spezifikation [9] ab.

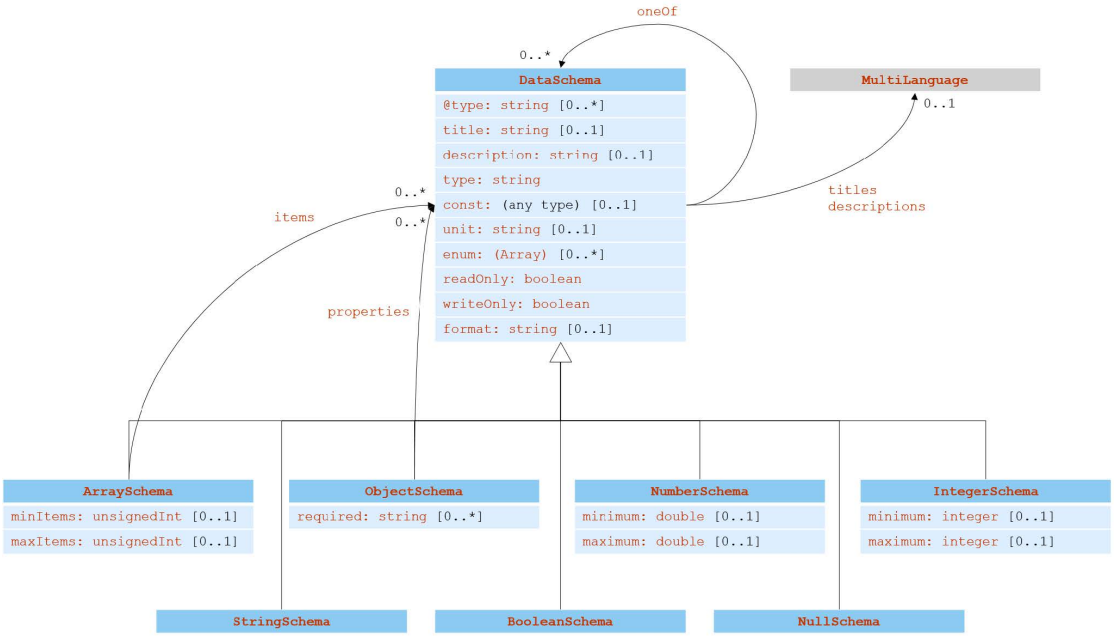

Abbildung 4: TD Datenschemavokabular

4.2.3 Sicherheitsvokabular

Das Sicherheitsvokabular definiert Sicherheitsmechanismen und dessen Konfigurationen.

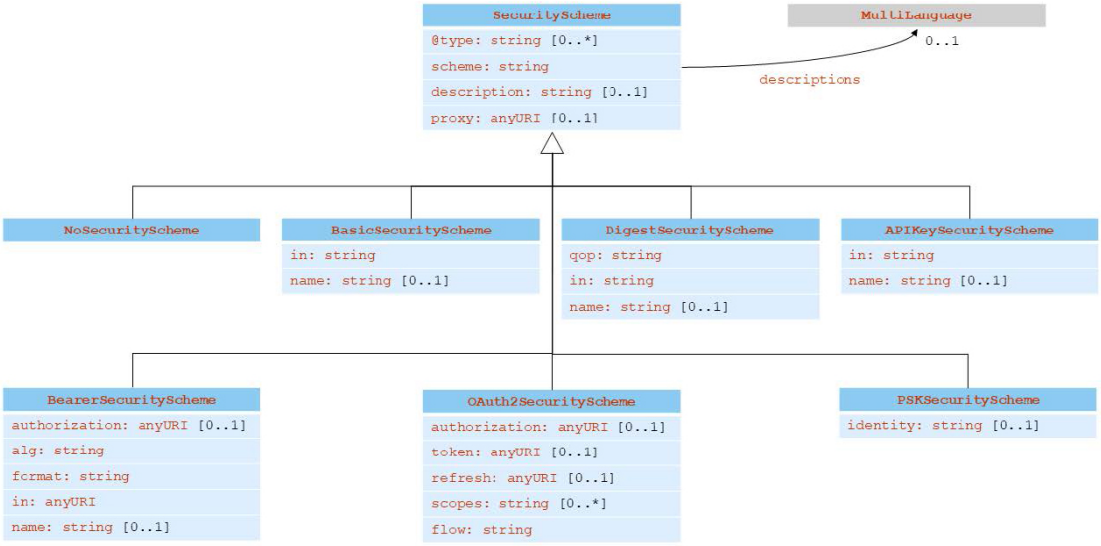

Abbildung 5: TD Sicherheitsvokabular

Ein `SecurityScheme` beinhaltet dabei grundsätzlich einen Bezeichner für das jeweilige Sicherheitsschema und möglicherweise Beschriftungen oder einen URI, auf die das Sicherheitsschema einen möglichen Zugriff definiert – wenn es nicht um den aktuellen URI geht. Der Schemabezeichner wird dabei als Diskriminator für die einzelnen Sicherheitsschemata verwendet. Dabei sind je nach Sicherheitsschema weitere Objektattribute erforderlich oder möglich.

4.2.4 Hypermediakontrollvokabular

Das Hypermediakontrollvokabular definiert die Hauptbestandteile von Kommunikation über Representational State Transfer (REST) mit Verweisen und Formularen.

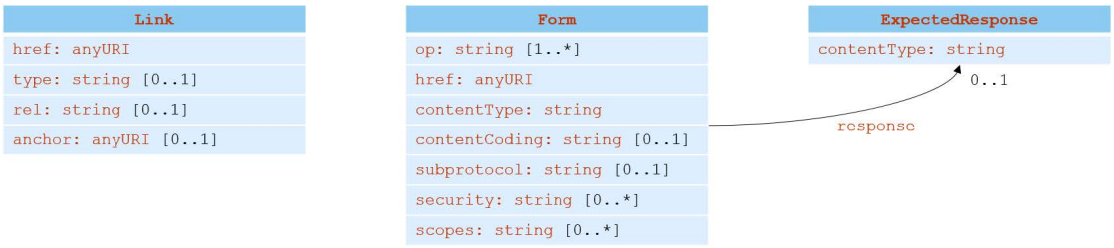

Abbildung 6: TD Hypermediakontrollvokabular

`Links` funktionieren dabei sehr ähnlich zu Ankerelementen in HyperText Markup Language (HTML). Sie bestehen hauptsächlich aus einem IRI und können durch den Medientyp des Ergebnisses, eine Relationsbeschreibung und einen Linkkontext erweitert werden.

`Forms` werden an vielen Stellen verwendet. Sie bestehen hauptsächlich aus einem Bezeichner für die Operation(en), die durchgeführt werden soll(en), sowie aus einem IRI, der beschreibt, wohin das Formular geschickt werden soll. Es können noch weitere Werte ergänzt werden, wie die Art und Codierung des Inhalts, das verwendete Subprotokoll, Sicherheitsaspekte oder die Art der Antwort.

4.3 Discovery

Die -Discovery beschreibt einen Prozess, mit dem eine Thing Description (TD) gefunden werden kann. Dabei soll die Verbreitung in einer Vielzahl von Anwendungsfällen unterstützen werden, wie Beispielsweise in lokalen und öffentlichen Netzwerken. Der Prozess muss dabei mit bestehenden Discovery-Mechanismen funktionieren, sicher sein und private Informationen schützen. Es muss ebenfalls in der Lage sein, Aktualisierungen von TDs effizient zu handhaben.

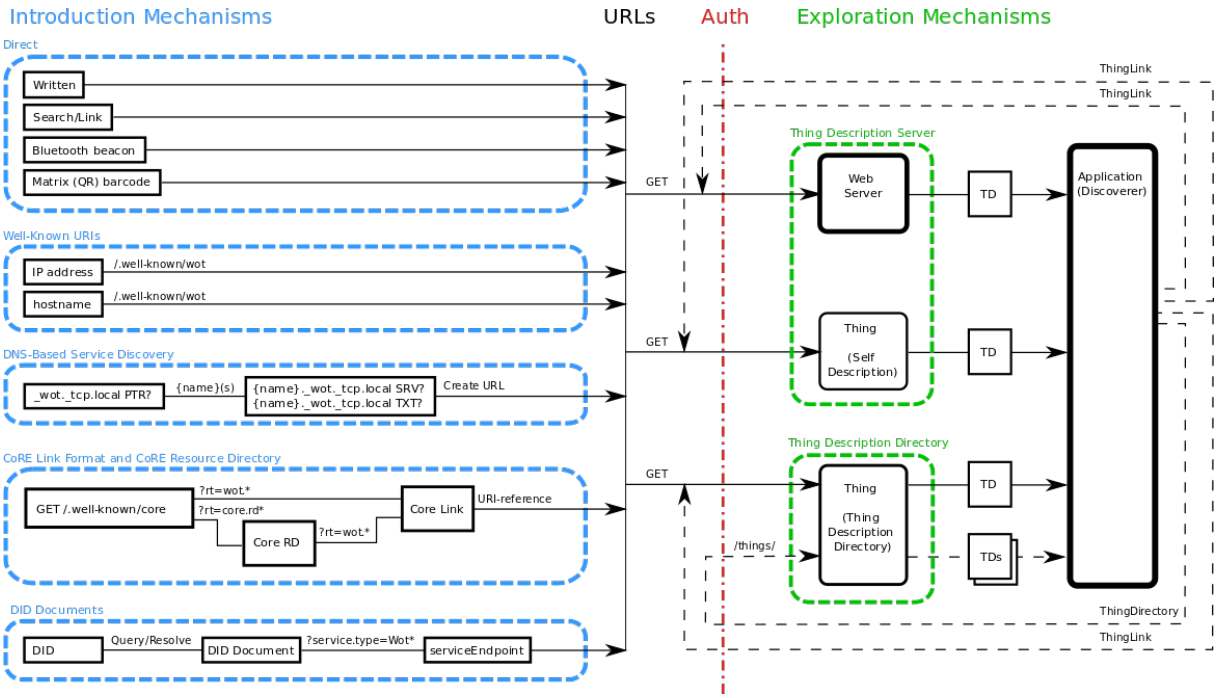

Abbildung 7: Discovery-Architektur

Der SD Prozess ist dabei in zwei Phasen unterteilt, die Einführungs- und die Erkundungsphase. Die Einführungsphase nutzt bestehende Erkundungsmechanismen, um ein oder mehrere Uniform Resource Locators (URLs) zu erzeugen. Diese URLs enthalten selbst keine Metadaten und verweisen auf Erkundungsdienste. Erst diese Erkundungsdienste stellen dann Metadaten in Form von TDs und Thing Description Directories (TDDs) bereit. Dabei gibt es zwei Arten von Erkundungsdiensten:

- Ein Dienst, der sich um die Verteilung einer einzelnen TD kümmert, wie z. B. ein einfacher Webservice.
- Ein Dienst für das Durchsuchen eines TDDs, worin sich mehrere TDs befinden.

Der Discovery-Prozess verwendet eine zweistufige Architektur. Im ersten Schritt generieren die Einführungsmechanismen URLs, die dann in der Erkundungsphase referenziert und mit Metadaten verknüpft werden. Die Einführung kann durch jeden Mechanismus durchgeführt werden, der eine URL liefern kann. Der Einführungsmechanismus kann dabei auch mehrere URLs liefern, die in der Erkundungsphase in mehreren TDs resultiert. Eine URL, die von einem Einführungsmechanismus zur Verfügung gestellt wird, zeigt dabei immer auf einen Endpunkt eines Erkundungsmechanismus, der eine TD liefert. Dies ist im einfachsten fall eine gewöhnliche Ressource, die von einem Webserver bereitgestellt wird und eine TD zurückgibt. Im Sonderfall gibt es noch eine selbst beschreibende URL für ein Thing, was seine eigene TD liefert.

Jeder Client, der eine einzelne TD mit einer einzelnen URL abrufen kann, unterstützt WoT-Discovery. Sie muss dabei aber gewisse Anforderungen erfüllen, wie z. B. dass mindestens ein Einführungsmechanismus vorhanden sein muss oder das mehrere Aufrufe desselben Einführungsmechanismus möglich sind. Diese Anforderungen sorgen dafür, dass genau definiert ist, wie sich die TD in verschiedenen Szenarien verhalten soll. Es können verschiedene Einführungsmechanismen verwendet werden, wie z. B. die direkte Variante über eine URL, eine Variante mit DNS-basierter Service-Discovery, oder eine Variante mithilfe von Constrained RESTful Environment (CoRE). Das Ergebnis ist aber immer eine URL eines Erkundungsmechanismus, welcher die Metadaten einer TD beinhaltet.

Im Erkundungsprozess werden TDs mithilfe von zwei Mechanismen über einen Server zur Verfügung gestellt. Jeder Webdienst, der über eine URL referenziert werden kann und eine TD zurückliefert, kann als Erkundungsmechanismus verwendet werden und wird als TD-Server bezeichnet. Ein TD-Server muss dabei nicht zwangsläufig ein Thing sein, sondern kann auch über ein einfachen Webserver implementiert werden. TD-Server können auch zur Selbstbeschreibung verwendet werden. Für die Selbstbeschreibung hostet ein Thing seine eigene TD und stellt sie zur Verfügung. Neben TD-Servern gib es noch die bereits erwähnten TDDs, worin sich keine oder mehrere TDs befinden und verwaltet werden können. Für jede TD enthält das TDD zusätzliche Metadaten für Buchführungs- und Suchzwecke.

4.4 Sicherheit & Datenschutz

Die Sicherheit eines WoT-Systems kann sich auf die Sicherheit der TD selbst, oder auf die Sicherheit des Things beziehen. Das WoT führt keine neuen Sicherheitsmechanismen ein. Es wird lediglich garantiert, dass bestehende Funktionalität und Sicherheit der Systeme beibehalten wird. Hierbei werden verschiedene Anforderungen gestellt. Bei der Offenlegung einer TD sollte es z. B. möglich sein, bewährte Verfahren für die Sicherheit und Integrität anzuwenden. Eine TD sollte zudem den tatsächlichen Sicherheitsstatus des von ihr beschriebenen Geräts genau wiedergeben. Das WoT kann potenziell auf viele

Protokolle angewendet werden. Für die Begrenzung des Umfangs wird sich aber hauptsächlich auf HTTP(S), CoAP(S) und MQTT(S) konzentriert. Das WoT stellt zudem ein Framework bereit, welches eine Reihe möglicher Bedrohungen und Sicherheitsziele auflistet, die ein Anwender berücksichtigen sollten. Des Weiteren werden Sicherheitspraktiken für die Gestaltung einer TD beschrieben, sowie ein Beispiel für eine sichere Konfiguration vorgeführt.

Mehrere Bedrohungen können die Privatsphäre von WoT-Betreibern beeinträchtigen. Ein Betreiber kann hier z. B. eine Person sein, die ein Smart-Home besitzt, oder eine Firma, die mehrere WoT-Things in einer Fabrik betreibt. Eine TD kann z. B. datenschutzrelevante Informationen preisgeben, wie die detaillierte Konfiguration eines einzelnen Things. Es können ebenfalls datenschutzrelevante Informationen durch Beobachtung der Kommunikation zwischen einem WoT-Endpunkt (Client, Server oder Gerät) und einer TDD ermittelt werden. Abhängig von der Netzwerktopologie können WoT-Systeme Betreiberdaten zwischen dem WoT-Konsumenten und dem WoT-Thing über viele Zwischenknoten übertragen werden. Wenn diese Knoten auf die Betreiberdaten zugreifen oder sie verarbeiten, kann es dazu kommen, dass ungewollt Informationen preisgeben werden. Zusätzlich zu den oben beschriebenen Maßnahmen ist es wichtig, die Betreiber über die gesammelten Daten zu informieren und ihnen die Möglichkeit zu geben, den Grad dieser Offenlegung zu kontrollieren.

5 Diskussion

Aus der gesammelten Recherche geht hervor, dass sich die vorgestellten SDs nicht optimal eignen, um als SD einer Schnittstelle zu dienen, über die Städte gesammelte Daten öffentlich zugänglich machen können. Bei allen Varianten besteht das Hauptproblem dabei darin, dass keinerlei Standardisierung vorhanden ist. Das bedeutet nicht, dass man diese Ansätze nicht dennoch weiter verfolgen könnte, aber es empfiehlt sich, stattdessen mit den Standards des WoT zu arbeiten.

Dabei lässt sich als Erstes festhalten, dass auch die Standards des W3C [6, 10–13, 18] für das WoT noch nicht vollständig erstellt wurden. Das hat zur Folge, dass einige Aspekte zum aktuellen Zeitpunkt noch unklar sind und sich Hintergründe der Spezifikationen nicht immer nachvollziehen lassen. Obwohl es nicht immer ratsam ist, sich auf unfertige Arbeiten zu beziehen, kann davon ausgegangen werden, dass diese Arbeiten später die Grundlage für noch fehlende Standards bilden werden und daher als Grundlage genutzt werden können.

Die Spezifikationen decken zu diesem Zeitpunkt noch keine QoS-basierte SD ab. Unter anderem Informationen zur Verfügbarkeit und Aktualisierungsrate von Services können daher nicht direkt mit in Betracht genommen werden. An dieser Stelle haben wir weiterhin nach Möglichkeiten recherchiert, QoS-Aspekte zu beachten. Unter anderem wird in „Bringing Deterministic Industrial Networking to the W3C Web of Things with TSN and OPC UA“ [19] ein erweitertes Vokabular für TDs vorgestellt, mit dem QoS-Aspekte definiert werden können.

Die Sicherheitsaspekte, die vom W3C für das WoT beschrieben werden, müssen bei der Entwicklung der bereits erwähnten Schnittstelle nicht weiter in Betracht gezogen werden. Bei Daten, die von Städten zur Verfügung gestellt werden, handelt es sich meist um Open-Data, wo die Daten ohnehin frei zugänglich sind. Da es sich zudem nicht um personenbezogene Daten handelt, muss auch der Datenschutz nicht weiter beachtet werden.

Ein Problem, das teilweise bestehen bleibt, ist die Ontologie bei der Suche nach Services. Mit den beschriebenen Ansätzen ist es möglich, diese bei der SD in Betracht zu ziehen, aber es gibt keine allgemeingültigen Ontologien, die dafür verwendet werden können. Es ist daher nötig, eine eigene Ontologie zu erstellen.

6 Fazit

Die Recherche im Rahmen eines Ansatzes für eine Open-Data-Schnittstelle in Smart Cities hat gezeigt, dass die WoT vom W3C einen guten Ansatz bieten, um nach diesem Schema eine entsprechende Schnittstelle zu implementieren. Die Standards des W3C bieten Standardisierungen, die z. B. das Beschreiben von Sensoren als auch die SD abdecken. Zudem handelt es sich im Vergleich zu den untersuchten SD-Aufbauten um aktuelle und sehr umfangreiche Beschreibungen des Standards. An einigen Stellen ist der Standard jedoch noch nicht vollständig ausgearbeitet.

Die Nutzung der diskutierten SD aus den entsprechenden Papern hat sich als nicht sinnvoll herausgestellt, da diese oft für spezifische Anwendungsfälle entwickelt worden sind und somit keine Standardisierung vorhanden ist. Daher wird empfohlen, sich bei der Implementierung einer Schnittstelle an dem WoT-Standard zu orientieren, der zusätzlich den Vorteil mitbringt, dass er nicht nur die SD beschreibt.

Abbildungsverzeichnis

Glossar

API Application Programming Interface

Consumer Eine Entität, die Thing Descriptions verstehen und mit Things über diese interagieren kann.

CoRE Constrained RESTful Environment

HTML HyperText Markup Language

Intermediary Eine Entität zwischen Consumern und Things, die Things erweitern, zusammenstellen oder vertreten kann und eine Thing Description bereitstellt, die die Zugriffe auf Things kapselt. Ein Intermediary könnte daher von außen nicht von einem Thing unterschieden werden.

IoT Internet of Things

IRI Internationalized Resource Identifier

JSON JavaScript Object Notation

JSON-LD JSON for Linked Data

LAN Local Area Network

LPWAN Low Power Wide Area Network

PubSub Ein Nachrichtenübermittlungsmuster, bei dem Sender von Nachrichten, auch ‚Publisher' genannt, nicht direkt an bestimmte Empfänger, auch ‚Subscriber' genannt, sendern, sondern veröffentlichte Nachrichten in Kategorien eingeteilt werden, während die konkreten ‚Subscriber' unbekannt sind, wenn es überhaupt welche gibt. Auf ähnliche Art und Weise abonnieren ‚Subscriber' bestimmte Kategorien, ohne zu wissen, ob überhaupt ‚Publisher' zu dieser Kategorie beitragen und wenn ja, welche ‚Publisher' das sind.

QoS Quality-of-Service

REST Representational State Transfer

SD Service Discovery

TD Thing Description

TDD Thing Description Directory

Thing Hardware/Software, die kommunizieren kann, programmierbar ist und entweder Sensoren besitzt oder Elemente in Bewegung versetzen kann.

URI Uniform Resource Identifier

URL Uniform Resource Locator

W3C World Wide Web Consortium

Web Thing Digitale Repräsentation eines Things und dessen Fähigkeiten mittels Webtechnologien. (Je nach Implementierung im Speziellen mit einigen ggf. wichtigen Unterschieden, aber das ist allgemeiner Konsens.)

WoT Web of Things

Literatur

[1] Meriem Achir, Abdelkrim Abdelli und Lynda Mokdad. „A taxonomy of service discovery approaches in IoT“. In: *2020 8th International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 2020 8th International Conference on Wireless Networks and Mobile Communications (WINCOM) (Reims, France). IEEE, 2020, S. 1–6. ISBN: 978-1-7281-9015-0. DOI: 10.1109/WINCOM50532.2020.9272512.

[2] Meriem Achir u. a. „Service discovery and selection in IoT: A survey and a taxonomy“. In: *Journal of Network and Computer Applications* 200 (2022). PII: S1084804521003167, S. 103331. ISSN: 10848045. DOI: 10.1016/j.jnca.2021.103331.

[3] Sonia Ben Mokhtar u. a. „Efficient Semantic Service Discovery in Pervasive Computing Environments“. en. In: ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing. Springer, Berlin, Heidelberg, 2006, S. 240–259. DOI: 10.1007/11925071_13. URL: https://link.springer.com/chapter/10.1007/11925071_13.

[4] BMI. „Open-Data-Strategie der Bundesregierung“. In: (). URL: https://www.bundesregierung.de/resource/blob/992814/1940386/1d269a2ad1b6346fcf60663bdea9c9f8/2021-07-07-open-data-strategie-data.pdf?download=1.

[5] Bharat S. Chaudhari, Marco Zennaro und Suresh Borkar. „LPWAN Technologies: Emerging Application Characteristics, Requirements, and Design Considerations“. In: *Future Internet* 12.3 (2020). PII: fi12030046, S. 46. ISSN: 1999-5903. DOI: 10.3390/fi12030046.

[6] Andrea Cimmino u. a., Hrsg. *Web of Things (WoT) Discovery*. URL: https://www.w3.org/TR/wot-discovery/ (besucht am 14. 06. 2022).

[7] Kashif Iqbal u. a. „Semantic Service Discovery using SAWSDL and SPARQL“. In: *2008 Fourth International Conference on Semantics, Knowledge and Grid*. 2008 Fourth International Conference on Semantics, Knowledge and Grid (SKG) (Beijing, China). IEEE, 12/3/2008 - 12/5/2008, S. 205–212. DOI: 10.1109/SKG.2008.87.

[8] Bing Jia, Wuyungerile Li und Tao Zhou. „A Centralized Service Discovery Algorithm via Multi-Stage Semantic Service Matching in Internet of Things“. In: *22017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. IEEE, 2017, S. 422–427. ISBN: 978-1-5386-3220-8. DOI: 10.1109/CSE-EUC.2017.82.

[9] *JSON Schema*. URL: https://json-schema.org (besucht am 05. 07. 2022).

[10] Sebastian Kaebisch u. a., Hrsg. *Web of Things (WoT) Thing Description*. World Wide Web Consortium. URL: https://www.w3.org/TR/wot-thing-description/ (besucht am 13. 06. 2022).

[11] Zoltan Kis u. a., Hrsg. *Web of Things (WoT) Architecture*. World Wide Web Consortium. URL: https://www.w3.org/TR/wot-architecture/ (besucht am 27. 06. 2022).

[12] Zoltan Kis u. a., Hrsg. *Web of Things (WoT) Scripting API*. World Wide Web Consortium. URL: https://www.w3.org/TR/wot-scripting-api/ (besucht am 27. 06. 2022).

[13] Michael Koster und Ege Korkan, Hrsg. *Web of Things (WoT) Binding Templates*. 30. Jan. 2020. URL: https://www.w3.org/TR/wot-binding-templates/ (besucht am 10. 07. 2022).

[14] Selahattin Kosunalp und Kubilay Demir. „SARL: A reinforcement learning based QoS-aware IoT service discovery model“. In: *Journal of Electrical Engineering* 71.6 (2020), S. 368–378. DOI: 10.2478/jee-2020-0051.

[15] Juan Li u. a. „A Decentralized Trustworthy Context and QoS-Aware Service Discovery Framework for the Internet of Things“. In: 2017 (), S. 19154–19166. DOI: 10.1109/ACCESS.2017.2756446. URL: https://ieeexplore.ieee.org/document/8049281 (besucht am 09. 06. 2022).

[16] S. Majithia u. a. „Reputation-based semantic service discovery“. In: *Proceedings of the thirteenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. June 14-16, 2004, University of Modena and Reggio Emilia, Italy*. Thirteenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (Modena, Italy). Los Alamitos, California: IEEE Computer Society, 2004, S. 297–302. ISBN: 0-7695-2183-5. DOI: 10.1109/ENABL.2004.52.

[17] Oscar Novo und Mario Di Francesco. „Semantic Interoperability in the IoT“. EN. In: *ACM Transactions on Internet of Things* 1.1 (2020), S. 1–25. ISSN: 2691-1914. DOI: 10.1145/3375838.

[18] Elena Reshetova und Michael McCool, Hrsg. *Web of Things (WoT) Security and Privacy Guidelines*. 6. Nov. 2019. URL: https://www.w3.org/TR/wot-security/ (besucht am 10. 07. 2022).

[19] Luca Sciullo, Sushmit Bhattacharjee und Matthias Kovatsch. „Bringing Deterministic Industrial Networking to the W3C Web of Things with TSN and OPC UA“. In: *Proceedings of the 10th International Conference on the Internet of Things*. IoT '20. Malmö, Sweden: Association for Computing Machinery, 2020. ISBN: 9781450387583. DOI: 10.1145/3410992.3410997. URL: https://doi.org/10.1145/3410992.3410997.

[20] Luca Sciullo u. a. „A Survey on the Web of Things“. In: *IEEE Access* 10 (2022), S. 47570–47596. DOI: 10.1109/ACCESS.2022.3171575.

[21] Fernando Serena, María Poveda-Villalón und Raúl García-Castro. „Semantic Discovery in the Web of Things“. en. In: International Conference on Web Engineering. https://vicinity2020.eu/vicinity/system/files/publications/enwot_2017_paper_8.pdf. Springer, Cham, 2018, S. 19–31. DOI: 10.1007/978-3-319-74433-9_2. URL: https://link.springer.com/chapter/10.1007/978-3-319-74433-9_2.

[22] Sagar Sukode, Shilpa Gite und Himanshu Agrawal. „CONTEXT AWARE FRAMEWORK IN IOT: A SURVEY“. In: 1-9 (2015). URL: https://www.researchgate.net/publication/273456830_CONTEXT_AWARE_FRAMEWORK_IN_IOT_A_SURVEY (besucht am 09. 06. 2022).

[23] Vincenzo Suraci, Silvano Mignanti und Anna Aiuto. „Context-aware Semantic Service Discovery“. In: *2007 proceedings of the 16th ist mobile and wireless communications*. 2007 16th IST Mobile and Wireless Communications Summit (Budapest, Hungary). Alfred Music Publishing Co., Inc. [Place of publication not identified]: John Wiley, 2008, S. 1–5. ISBN: 1-4244-1662-0. DOI: 10.1109/ISTMWC.2007.4299110.

[24] Shuai Zhao u. a. „IoT Service Clustering for Dynamic Service Matchmaking“. In: *Sensors* 17.8 (2017). ISSN: 1424-8220. DOI: 10.3390/s17081727.