Kernkonzepte und Architektur des W3C-Web-of-Things

Jannik Bergmann

[1] HAW Hamburg, Berliner Tor 5, 20099 Hamburg, Germany

jannik.bergmann@haw-hamburg.de

Abstract. Already today, IoT devices are omnipresent in our lives. Whether it be in the use of smart home appliances or in agriculture. A major challenge in the development of IoT systems is the considerable fragmentation of the market. Many manufacturers develop closed solutions and use proprietary standards. This makes the maintenance and development of comprehensive applications significantly more difficult, which in turn holds back the actual potential of the IoT. One proposed solution is to create an application layer for the IoT that can be used to access things using standard web technologies - the W3C Web of Things (WoT). By using the WoT, the commercial potential of the IoT can be released, which until now has been held back by its strong fragmentation. The main task of this paper is to examine the motivation behind the WoT and the associated challenges of the IoT landscape. Furthermore, the concrete approach of the WoT and the system architecture is examined and evaluated.

Keywords: Web-of-Things, IoT, W3C-WoT-Architecture.

1 Einführung

In klassischen Projekten, welche die Entwicklung von Systemen bzw. Applikationen für das Internet-of-Things (IoT) betreffen, werden Entwickler mit einer heterogenen Technologielandschaft konfrontiert. Es werden unterschiedliche Datenmodelle, Sicherheitsanforderungen und teils proprietäre Kommunikationsprotokolle verwendet. Die dadurch ausgelöste Fragmentierung des IoT führt oft dazu, dass IoT-Projekte unter hohem Aufwand und für sehr spezifische Use-Cases entwickelt werden, was die Wartung sowie Erweiterung von Applikationen deutlich erschwert [1]. Allerdings wird grade die Interoperabilität zwischen verschiedenen IoT-Systemen laut [2] ein entscheidender Faktor für den wirtschaftlichen Erfolg von IoT-Services und -Anwendungen sein. Ein umfassender Ansatz, der eben dieser Fragmentierung entgegenwirken soll, ist das vom World-Wide-Web-Consortium (W3C) definierte W3C-Web-of-Things (WoT). Das W3C-WoT soll die Kommunikation zwischen verschiedenen IoT-Geräten ermöglichen bzw. erleichtern, unabhängig von verwendeten Kommunikationsprotokollen und Implementationsdetails.

Diese Arbeit beschäftigt sich mit der Motivation hinter der Konzeption des W3C-WoT sowie mit dessen Architektur und Kernkonzepten. Der nachfolgende Teil der Arbeit ist wie folgt aufgebaut. In Kapitel 2 wird die Motivation hinter dem WoT und

verwandte Literatur betrachtet. Kapitel 3 beschäftigt sich mit den Kernkonzepten und der generellen Architektur. In Kapitel 4 die One Data Model Initiative vorgestellt und in den Kontext der WoT-Architektur eingeordnet. In Kaptiel 5 wird eine Evaluation über das W3C-WoT in Hinsicht auf allgemeine WoT-Kernkonzepte durchgeführt. In Kapitel 6 wird ein abschließendes Fazit gegeben.

2 Motivation und verwandte Arbeiten

Die heutige IoT-Landschaft ist ein stark wachsendes Feld von großem wirtschaftlichen Interesse. Allein die Ausgaben für gewerblich genutzte IoT-Geräte wuchsen laut [3] im Jahre 2021 von 129 Milliarden US-Dollar um 22,4% auf 157 Milliarden US-Dollar. Obwohl das IoT demzufolge wirtschaftlich betrachtet ein zukunftsträchtiger Sektor ist, unterliegt dieser zurzeit einer starken Fragmentierung. In [2] werden die Interoperabilitäts- und Integrationsprobleme detailliert behandelt. Ein Lösungsansatz, der als Antwort auf dieses Fragmentierungsproblem folgte, war die Entwicklung einer Applikationsschicht für das IoT. Seit 2007, als das Konzept des Web of Things (WoT) erstmals definiert wurde [4], haben mehrere Forschungsstudien untersucht, wie IoT-Geräte mithilfe von Standard-Webtechnologien miteinander verbunden werden können. Dies hat zu einer Vielzahl von WoT-Frameworks und -Architekturen geführt, die sich in Bezug auf die unterstützten WoT-basierten Interaktionsmuster und die verfolgten funktionalen Ziele stark unterscheiden. In [5] werden dazu 26 verschiedene WoT-basierte Plattformen betrachtet und evaluiert, bezüglich 12 empirisch hergeleiteter WoT-Kernbestandteile. Die laut [5] als am wichtigsten zu betrachtenden Kernbestandteile sind dabei die Interaktion mit der Plattform nach dem REST-Architekturstil [6] und die Nutzung von weitverbreiteten Web-Datenformaten. Nur die Hälfte der evaluierten Plattformen erfüllte zum Zeitpunkt der Evaluation diese Kriterien. Die Verwendung der REST-Architektur scheint laut [5] ein wesentlicher Kernbestandteil der WoT-Architektur zu sein, dennoch wurden Alternativen zum HTTP-Protokoll für die Implementierung in Betracht gezogen: In [7] wird beispielsweise eine CoAP-basierte Architektur vorgeschlagen.

Das W3C hat es sich seit 2017 zur Aufgabe gemacht, einen Standard für das WoT zu definieren [8], welche die wichtigsten Aspekte des WoT vereint. Die Architektur soll bestehende IoT-Standards und -Lösungen komplementieren, Interoperabilität zwischen Systemen gewährleisten und damit das tatsächliche kommerzielle Potenzial des IoT erschließen [1]. Die W3C-WoT-Architektur als Lösungskonzept gegen die Fragmentierung des IoT wird in den folgenden Kapiteln genauer betrachtet.

3 Architektur des W3C-Web-of-Things

Das grundlegende Ziel des W3C-WoT ist es, die Interoperabilität von IoT-Systemen, unabhängig von Implementationsdetails und Netzwerkprotokollen herzustellen. Dabei sollen IoT-Geräten als Ressourcen mit einer zugehörigen Beschreibung ihrer Fähigkeiten verfügbar gemacht werden. Der Zugriff auf die Ressourcen soll dann wie der Zugriff auf eine Internetseite im traditionellen Web erfolgen. Dabei ist es wichtig zu

definieren, dass das WoT kein neues Kommunikationsprotkoll darstellt, sondern bestehende Protokolle nutzt, um die Kommunikation von Geräten entsprechend ihrer Fähigkeiten zu ermöglichen.

Ein einzelnes IoT-Gerät wird in diesem Kontext als Web-Thing, oder einfach nur als Thing, bezeichnet und beschreibt die Abstraktion einer physischen oder virtuellen Entität, die semantisch beschrieben werden kann [8]. Übergeordnete funktionelle Anforderungen, die sich das W3C bei der Erstellung des Standards gesetzt hat sind die folgenden [8]:

1. *Flexibilität*: Es gibt eine Vielzahl von physischen Gerätekonfigurationen für WoT-Implementierungen. Die abstrakte WoT-Architektur sollte in der Lage sein, alle Varianten abzubilden und abzudecken.
2. *Kompatibilität*: Es gibt bereits viele bestehende IoT-Lösungen und laufende IoT-Standardisierungsaktivitäten in vielen Geschäftsbereichen. Das WoT sollte eine Brücke zwischen diesen bestehenden und sich entwickelnden IoT-Lösungen und den auf Webtechnologien basierenden WoT-Konzepten bilden. Das WoT sollte mit bestehenden IoT-Lösungen und aktuellen Standards aufwärtskompatibel sein.
3. *Skalierbarkeit*: Das WoT muss für IoT-Lösungen, die Tausende bis Millionen von Geräten umfassen, skalierbar sein.
4. *Interoperabilität*: Das WoT muss Interoperabilität zwischen Geräte- und Cloud-Herstellern bieten. Es muss möglich sein, ein WoT-fähiges Gerät mit einem Cloud-Dienst von verschiedenen Herstellern zu verbinden, und zwar *out-of-the-box*.

Um die funktionellen Anforderungen erfüllen und somit die Prinzipien des WoT anwenden zu können, definiert das W3C-WoT die folgenden vier Bausteine, die sogenannten Building-Blocks des WoT [8]:

1. *WoT Thing-Description:* Repräsentiert die strukturelle Beschaffenheit eines Things
2. *WoT Binding-Templates*: Metadaten um die Kommunikationsstrategien zu beschreiben, die ein Thing implementieren kann
3. *WoT Scripting-API*: WoT-Schnittstelle, um Operationen wie z. B. das Anbieten von Ressourcen auf einem Thing durch Skripts ermöglichen zu können
4. *WoT Security-and-Privacy-Guidelines*: Leitfaden zur Gewährleistung von Sicherheit- und Privatsphärestandards im WoT

Die Building-Blocks des W3C-WoT sind essenziell für die Implementation von Systemen, die konform für die abstrakte Architektur sind. Eine Übersicht über die Building-Blocks gibt **Fig. 1**, wobei die Building Blocks durch linierte und gestrichelte schwarze Umrandungen hervorgehoben wurden.

In der Architekturbeschreibung des WoT wird der Begriff des Servients eingeführt. Einen Servient nennt man einen Software-Stack, der die Building-Blocks des WoT implementiert, also die laufende Instanz eines Web-Things [9]. Servients können Things hosten und *exposen*, also freigeben, oder als Host für *Consumer* fungieren. Consumer sind Entitäten, die TD verarbeiten und mit Things interagieren können. Servients können somit sowohl eine Server- als auch eine Klienten-Rolle für die

Things im WoT einnehmen. Die Umgebung, in der ein Servient die Building-Blocks zur Laufzeit ausführt, bezeichnet man als WoT-Runtime.

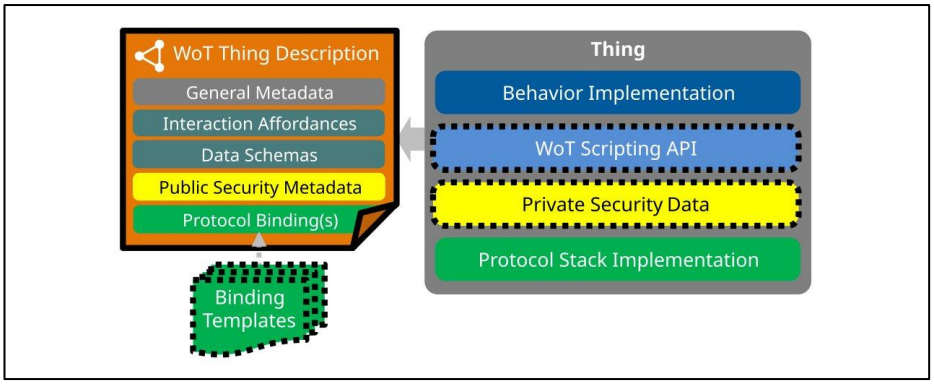

Fig. 1. Übersicht der Building-Blocks des W3C-WoT [8]

3.1 WoT-Thing Description

Die Thing-Description (TD) ist ein zentraler Baustein im WoT, vergleichbar mit der index.html einer Webseite in der traditionellen Web-Entwicklung [1]. Eine Übersicht über eine TD eines abstrakten Things ist in **Fig. 1** zu sehen, speziell der Teil der Abbildung mit der Bezeichnung WoT Thing Description. Wie der Abbildung zu entnehmen ist, hat die TD eines Things vier Hauptbestandteile [10]: Generelle und Security-Metadaten, Interaction-Affordances, Datenschemata und Protocol Bindings. Die TD dokumentiert die vier Hauptbestandteile in einer JSON-basierten Syntax, der JSON-for-Linked-Data (JSON-LD). Durch die Verwendung des JSON-LD-Formats ist die TD menschenlesbar und kann zeitgleich von Maschinen verstanden werden. Für Discovery-Zwecke können TDs von IoT-Geräten selbst oder von externen Hosts zur Verfügung gestellt werden. In **Fig. 2** kann man die Schlüsselworte *properties*, *actions* und *events* sehen. Diese definieren die sogenannten *Interaction-Affordances* des Things, welche eine dem WoT zugrundeliegende zentrale Interaktionsabstraktion beschreiben.

Interaction Affordances. Ein Hauptaspekt des W3C-WoT ist die Bereitstellung von Meta-Daten, die von Maschinen verstanden werden können [8]. Um diese Metadaten für andere Geräte verfügbar machen zu machen, werden in der Thing Description sogenannte Interaction-Affordances definiert, die in der Lage sein sollen, jedes IoT-Netzwerk-Interface zu beschreiben. Durch die Verwendung dieser Abstraktion haben Applikationen einen gemeinsamen Anker, um Metadaten auszutauschen und darüber informiert zu werden, wie Funktionen ausgeführt und Daten manipuliert werden können. Dabei sind die Hauptbestandteile aufgeteilt in *properties*, *actions* und *events*. Properties beschreiben die möglichen Status eines Things, sowie die verfügbaren Zugriffsmöglichkeiten auf diese. Actions beschreiben, wie Funktionen eines Things

ausgelöst werden können, welche wiederum Status manipulieren oder die Ausführung eines Prozesses im Thing anstoßen können. Events beschreiben den Zugriff auf Event-Quellen des Things, aus denen asynchron Daten zu Konsumenten gepusht werden können.

EXAMPLE 1: Thing Description Sample

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-WoTLamp-1234",
    "title": "MyLampThing",
    "securityDefinitions": {
        "basic_sc": {"scheme": "basic", "in":"header"}
    },
    "security": ["basic_sc"],
    "properties": {
        "status" : {
            "type": "string",
            "forms": [{"href": "https://mylamp.example.com/status"}]
        }
    },
    "actions": {
        "toggle" : {
            "forms": [{"href": "https://mylamp.example.com/toggle"}]
        }
    },
    "events":{
        "overheating":{
            "data": {"type": "string"},
            "forms": [{
                "href": "https://mylamp.example.com/oh",
                "subprotocol": "longpoll"
            }]
        }
    }
}
```

Fig. 2. Thing-Description in JSON-LD [10]

3.2 WoT-Binding-Templates

Eine Herausforderung des WoT ist die Vermittlung zwischen der großen Menge an IoT-Plattformen und der verwendeten Protokolle. Zur Erreichung dieses Ziels schlägt das W3C-WoT die Verwendung sogenannter Binding-Templates vor. Binding-Templates bestehen aus wiederverwendbarem Vokabular und Erweiterungen zum TD-Format, welches einem Klienten die Interaktion mit einem IoT-Gerät unter Verwendung eines konsistenten Interaktionsmodells ermöglichen soll. Binding-Templates befähigen Konsumenten dazu, sich den unterliegenden Protokollen eines Things anzupassen. Binding-Templates informieren im speziell über die folgenden Implementationsdetails [11]:

Protokollmethoden und Optionen. Die meisten Protokolle haben eine kleine Menge an Methoden, welche die Intention der Nachricht bestimmen. Binding-Templates beschreiben, wie diese existierenden Methoden und das dazugehörige Vokabular in der TD definiert werden können. Genauer funktioniert dies durch das Mapping von

Protokollmethoden zu sogenannten Interaction-Affordances-Ausdrücken, z. B. *readproperty*, *writeproperty*, *subscribeevent* oder unsubscribeevent.

Media-Types. Grundlegend werden im W3C-WoT IANA-Registered-Media-Types verwendet, u. a. um Applikationen von Things zu entkoppeln. Um die Media-Types zu nutzen sind oft Umwandlungen von proprietären Datenformaten zu Web-Standards wie JSON oder XML nötig. Durch Angabe der Media-Types wird die ordnungsgemäße Verarbeitung serialisierter Dokumente sichergestellt. Dadurch können Dokumente in jedem Format ausgetauscht werden und den höheren Schichten einer Anwendung wird die Anpassung an verschiedene Datenformate ermöglicht [11].

Payload-Struktur. Die zu einem standardisierten IANA-Media-Type serialisierten Daten bleiben in einer Struktur, die für das Datenmodell der jeweiligen IoT-Plattform spezifisch ist und von Konsumenten verstanden werden muss. Die Datendefinitionssprache der *DataSchema*-Elemente ermöglicht die Beschreibung beliebiger Strukturen durch Verschachtelung von Arrays und Objekten [11].

Datentypen und Wertebeschränkungen. Zur Anpassung an die zugrundeliegenden Datentypen stehen zusätzliche Formen von Constraints zur Verfügung. Eine plattformspezifische vorzeichenlose Ganzzahl kann z. B. als Integer mit einem Minimum von 0 und einem Maximum von 255 definiert werden [11].

3.3 WoT-Scripting-API

Die WoT-Scripting-API ist ein optionaler *Convenience*-Building Block, der die Entwicklung von WoT-Anwendungen vereinfachen soll [12]. Anwendungsgebiete für die Scripting-API sind üblicherweise Gateways oder Browser, die in der Lage sind eine WoT-Runtime auszuführen. Die Scripting-API basiert auf dem ECMAScript.

Traditionell wird IoT-Gerätelogik in der Firmware der Geräte implementiert, was zu Produktivitätseinschränkungen durch komplizierte Update-Prozesse führen kann. Mithilfe der WoT-Scripting-API kann Gerätelogik direkt durch wiederverwendbare Skripts implementiert werden, die in der WoT-Runtime ausgeführt werden, und somit keiner weiteren Aktualisierung von Firmware benötigen. Die standardisierte API erhöht die Portabilität von Anwendungsmodulen, um z.B. Logik mit hohem Rechenleistungsbedarf zu einem lokalen Gateway auszulagern, oder um zeitkritische Logik von der Cloud zu einem Gateway oder Randknoten zu verlagern

3.4 WoT-Security-and-Privacy-Guidelines

Das WoT führt keine neuen Sicherheitsmechanismen ein, sondern soll die Funktionalität und Sicherheit bestehender Mechanismen erhalten. Ein WoT-System besitzt allerdings auch eigene Komponenten, wie die Scripting-API oder die TD, deren Sicherheit zusätzlich gewährleistet werden muss. Hierfür ist der vierte Building-Block des

WoT zuständig. Dazu definiert das W3C aufgrund der großen Vielfalt an Geräten, Anwendungsfällen, Einsatzszenarien und Anforderungen ein Threat-Model [13], welches als allgemeiner Leitfaden für die Behandlung potenzieller Bedrohungen verstanden werden kann. Nutzer der WoT-Architektur können auf Grundlage ihrer konkreten Sicherheitsanforderungen Anpassungen an diesem Modell vornehmen. Dabei gibt das Threat-Model eine Reihe möglicher Bedrohungen vor, die bei Implementierung der Architektur in Betracht gezogen werden sollten, wie z. B. der Umgang mit datenschutzsensiblen Daten und Hochverfügbarkeitsanforderungen. Die Spezifikationen jedes Building-Blocks enthalten zudem Empfehlungen für die Gewährleistung der spezifischen Sicherheits- und Privatsphäreaspekte.

4 One Data Model im Kontext des W3C-WoT

Für die verschiedenen IoT-Domänen gibt es eine Vielzahl an Datenmodellen, welche sich bezüglich ihrer Anwendungsgebiete überschneiden und teilweise in Konkurrenz zueinanderstehen. Organisationen wie das IETF, aber auch unabhängige Firmen stellen Datenmodelle zur Verfügung z. B. SenML [14] oder IPSO Smart Objects [15]. Dieser Mangel an gemeinsamen Datenmodellen hat die One Data Model Initative (OneDM) hervorgebracht. Das Ziel von OneDM ist es, einen gemeinsamen Satz von Daten- und Interaktionsmodellen zur Beschreibung von IoT-Geräten zu entwickeln, welche im Semantic-Definition Format (SDF) definiert werden. Dadurch soll Anwendungen ermöglicht werden, mit IoT-Geräten aus verschiedenen Ökosystemen zu arbeiten, ohne dass Daten und Interaktionen aus dem Modell einer Organisation in das der anderen konvertiert werden müssen. Im Idealfall gibt es für jede Klasse von IoT-Geräten nur ein einziges, von den beteiligten Organisationen ausgewähltes Modell, das alle verwenden können [16]. Als Grundlage wird die SDF verwendet, die als Format für die Erstellung und Erhaltung von domänenspezifischen Daten- und Interaktionsmodellen dient. In SDF werden die allgemeinen Modelle definiert, letztlich werden sie in JSON, CBOR oder andere gebräuchliche Formate übertragen.

Das W3C bietet durch das verfügbare Vokabular der TD einen Baukasten für die Nutzung verschiedenster Datenmodelle. Wichtig dabei zu definieren ist jedoch, dass das W3C sich mit der Standardisierung der WoT-Architektur nicht auch die Standardisierung der Datenmodelle zur Aufgabe gemacht hat. Lediglich die Verwendung von im Web gebräuchlichen Datenformaten wie JSON oder XML wird vom Standard vorgegeben. Die Definition der Datenmodelle wird von unabhängigen Organisationen und Initiativen verwaltet, wie eben der OneDM-Initiative.

5 Evaluation

Die vorgestellte W3C-WoT-Architektur soll einen definitiven Standard für das WoT darstellen, welche alle Kernaspekte des WoT unterstützt. Zur Überprüfung ob das W3C diese Ziele mit seiner Architektur erreicht hat, werden die in [5] vorgestellten Kernbestandteile des WoT hinzugezogen und im Folgenden evaluiert, inwiefern das W3C-WoT diese Kriterien erfüllt. Der Autor listet in seiner Evaluation 12 Kernbe-

standteile mit der Nummerierung von E1 bis E12 auf. Als essenziell für die Umsetzung einer umfassenden WoT-Architektur bezeichnet der Autor in [5] die Evaluationskriterien E4-E6, und als wichtig betrachtet er die Kriterien E2, E3 und E7. Im Folgenden werden diese besonders wichtigen Evaluationskriterien herausgenommen und ihre Erfüllung in Bezug auf die WoT-Architektur des W3C überprüft. Die Kernbestandteile des WoT sind laut [5] die folgenden:

- *E2*: Integration von Things in das Internet, direkt oder indirekt über Gateways
- *E3*: RESTful Interaktion mit den Dinge mit den vier CRUD-Hauptoperationen
- *E4* RESTful Interaktion mit der Plattform, von Clients und Endnutzern.
- *E5* Datenformate, die im Internet bekannt, akzeptiert und weit verbreitet sind.
- *E6* Multiple Darstellungen der Antworten, die von Things geliefert werden, in verschiedenen Datenformaten.
- *E7* Sicherheit in Bezug auf Autorisierung auf der Plattform, Nachrichtenverschlüsselung und Integration.

Kriterien	**Abdeckung durch die W3C-WoT-Architektur**	**Umsetzung**
E2	Die Integration von Things im W3C-WoT geschieht durch die Definition der TD, welche, je nach Leistung der jeweiligen Geräte, von den Things selbst oder von externen Hosts zur Verfügung gestellt werden kann.	✓
E3	Eine Unterstützung der REST-Architektur wird durch Definition der Interaction Affordances und der dazugehörigen Ausdrücke sichergestellt. Dabei ist die konkrete Implementierung des Thing unabhängig in seiner Protokollwahl, denn durch die Binding-Templates wird eine Kommunikation nach dem REST-Prinzip sichergestellt. Ressourcen und Funktionen auf Things werden durch URIs verfügbar gemacht.	✓
E4	Das W3C-WoT unterstützt diese Feature durch die Anfragen der TD über HTTPS vom Thing direkt, oder über sogenannte TD-Directorys, die eine REST-API zur Verfügung stellen, um auf TD zuzugreifen zu können.	✓
E5	Das W3C-WoT setzt die Verwendung von IANA-Registererd-Media-Types und somit gebräuchliche Datenformaten vorraus.	✓
E6	Payload-Strukturen in beliebiger Form mithilfe des *DataSchema*-Vokabulars in der TD definiert werden.	✓
E7	Das WoT unterstützt alle gängigen Sicherheitsmechanismen des Webs und stellt mit dem Building-Block Security einen Leitfaden für die Anwendung von Sicherheitspraktiken zur Verfügung	✓

Table 1. Evaluation der W3C-WoT-Architektur

Wie man in **Table 1** sehen kann, erfüllt die W3C-WoT-Architektur alle wichtigen Anforderungen an eine WoT-Plattform. Die durch das W3C definierten Building-Blocks erlauben die Erstellung flexibler WoT-Systeme. Insbesondere die Einführung der Interaction-Affordances fördern die Entwicklung eines offenen WoT, da durch die Interaktionsabstraktion eine Vielzahl an IoT-Netzwerk-Interfaces beschrieben werden können. Demnach kann die W3C-WoT-Architektur dabei helfen, die Fragmentierung der IoT-Landschaft zu verringern.

6 Fazit

In der vorliegenden Arbeit wurde die Motivation für die Entstehung des W3C-WoT sowie dessen Architektur erläutert. Durch die Entwicklung der W3C-WoT-Architektur, die auf weitverbreiteten Web-Standards basiert, wurde ein konkreter Lösungsansatz entgegen der weiteren Fragmentierung des IoT geschaffen Das W3C schlägt dabei die Verwendung von WoT-Building-Blocks vor, welche die Basis der WoT-Architektur bilden. Dabei ist der erste Building-Block die Thing-Description, in der Metadaten der einzelnen IoT-Geräte beschrieben werden. Zusätzlich werden in der Thing-Description Interaction-Affordances eingeführt, welche in der Lage sein sollen, alle Interaktionsschnittstellen von IoT-Geräten beschreiben zu können. Als zweiter Building-Block der WoT-Architektur werden die Binding-Templates definiert, welche die Kommunikation verschiedener WoT-Geräte mit unterschiedlichen Kommunikationsprotokollen ermöglichen, durch die Beschreibung von angebotenen Interaktionsschnittstellen und deren Verwendung. Die Scripting-API stellt den dritten Building-Block des W3C-WoT dar und ermöglicht die Ausführung von Skripten auf WoT-Geräten in der WoT-Runtime sowie das Hinzufügen von zusätzlichen Thing-Funktionalitäten ohne die Ausführung aufwändiger Update-Prozesse. Der vierte Building-Block betrifft die Security und Privacy-Aspekte und beschreibt einen Leitfaden für die Bedrohungsvermeidung bei Implementation der WoT-Architektur.

Zusätzlich zur Betrachtung der W3C-WoT-Architektur wurde die One Data Model Initiative vorgestellt und in den Kontext der Architektur eingeordnet. Die Initative will eine Menge an Datenmodellen für einzelne IoT-Domänen zusammenfassen und jeweils ein definitives Datenmodell einführen, welches von allen Organisationen verwendet werden kann. Dabei tragen die gemeinsamen Datenmodelle zur Interoperabilität des WoT bei, sie sind aber kein Teil der W3C-WoT-Architektur. Abschließend wurde die W3C-WoT-Architektur im Hinblick auf ihr übergeordnetes Ziel der Definition eines umfassenden Standards für das WoT hin evaluiert. Es wurden sechs Kernbestandteile aus der Literatur zur Hilfe genommen und mit der Architektur verglichen. Das Ergebnis war eine starke Überdeckung mit den Kernkonzepten des allgemeinen WoT und dem W3C-Standard.

Zusammenfassend kann die Architektur des WoT als wichtiger Schritt in die Richtung eines offenen und interoperablen IoT bezeichnet werden. Durch die Verwendung von allgemeinen Building-Blocks und speziell der Einführung der Thing-Description, die eine weitreichendes Spektrum an Datenformaten und Anwendungsfällen beschreiben kann, wird eine Basis für die Kommunikation diverser Geräte der IoT-

Landschaft geschaffen. Obwohl der Ansatz erfolgsversprechend erscheint, bleibt abzuwarten wie die tatsächliche Adaption des Standards verlaufen wird und ob das W3C-Web-of-Things in Zukunft einen realen Mehrwert liefern kann.

7 References

[1] W3C, *Documentation - Web of Things (WoT).* [Online]. Available: https://www.w3.org/WoT/documentation/ (accessed: Jul. 6 2022).

[2] M. Aly, F. Khomh, Y.-G. Gueheneuc, H. Washizaki, and S. Yacout, "Is Fragmentation a Threat to the Success of the Internet of Things?," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 472–487, 2019, doi: 10.1109/jiot.2018.2863180.

[3] P. Wegner, *Global IoT market size grew 22% in 2021 — these 16 factors affect the growth trajectory to 2027.* [Online]. Available: https://iot-analytics.com/iot-market-size/ (accessed: Jul. 11 2022).

[4] E. Wilde, "Putting things to REST," *UCB iSchool Report 2007-015*, 2007.

[5] A. Kamilaris and M. I. Ali, "Do "Web of Things platforms" truly follow the Web of Things?," in *2016 IEEE 3rd World Forum on Internet of Things ((WF-IoT): 12-14 Dec. 2016*, Reston, VA, USA, 2016, pp. 496–501.

[6] R. T. Fielding, "Architectural styles and the design of network-based software architectures," University of California, Irvine.

[7] L. Mainetti, V. Mighali, and L. Patrono, "A Software Architecture Enabling the Web of Things," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 445–454, 2015, doi: 10.1109/jiot.2015.2477467.

[8] *Web of Things (WoT) Architecture.* [Online]. Available: https://www.w3.org/TR/2020/REC-wot-architecture-20200409/ (accessed: Jul. 6 2022).

[9] I. Zyrianoff, L. Gigli, F. Montori, C. Aguzzi, S. Kaebisch, and M. Di Felice, "Seamless Integration of RESTful Web Services with the Web of Things," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, Pisa, Italy, Mar. 2022 - Mar. 2022, pp. 427–432.

[10] *Web of Things (WoT) Thing Description.* [Online]. Available: https://www.w3.org/TR/wot-thing-description/ (accessed: Jul. 11 2022).

[11] *Web of Things (WoT) Binding Templates.* [Online]. Available: https://www.w3.org/TR/wot-binding-templates/ (accessed: Jul. 13 2022).

[12] *Web of Things (WoT) Scripting API.* [Online]. Available: https://www.w3.org/TR/wot-scripting-api/ (accessed: Jul. 13 2022).

[13] *Web of Things (WoT) Security and Privacy Guidelines.* [Online]. Available: https://www.w3.org/TR/wot-security/ (accessed: Jul. 16 2022).

[14] *RFC 8428 - Sensor Measurement Lists (SenML).* [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8428 (accessed: Jul. 24 2022).

[15] J. Jimenez, M. Koster, and H. Tschofenig, "IPSO Smart Objects," in *Position paper for the IOT Semantic Interoperability Workshop*.

[16] One Data Model, *Overview.* [Online]. Available: https://onedm.org/overview/ (accessed: Jul. 24 2022).