BOSCH
Technik fürs Leben

DHBW
Duale Hochschule
Baden-Württemberg

Entwicklung einer Schnittstelle zur Verwaltung von Fahrer- und Beifahrersitzfunktionalität unter Zuhilfenahme Digitaler Zwillinge

Bachelorarbeit

des Studiengangs Informatik
an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Nico Makowe

17. Juli 2022

Bearbeitungszeitraum:	12 Wochen
Matrikelnummer, Kurs:	9275184, STG-TINF19ITA
Dualer Partner:	Robert Bosch GmbH
Betreuer des Dualen Partners:	Georg Schmidt-Dumont
Gutachter der Dualen Hochschule:	Dr. Jamal Krini

Kurzfassung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1. Einleitung

1.1 Problemstellung

1.2 Vorgehensweise der Arbeit

2. Theoretische Grundlagen

2.1 Digitale Zwillinge

Ein digitaler Zwilling ist ein virtuelles Abbild eines Objekts oder eines Prozesses der realten Welt. Die abgebildeten Objekte und Prozesse werden auch als „Asset“ bezeichnet. Beispiele für Assets sind:

- Ein Maschinentyp, beispielsweise ein bestimmtes Bohrmaschinenmodell eines Herstellers.
- Eine Maschineninsantz, beispielsweies eine konkrete Bohrmaschine.
- Ein Software-System
- Ein Fahrzeug
- Eine bestimmte Komponente im Fahrzeug, beispielsweise ein einzelner Sitz.

Die Einsatzgebiete digitaler Zwillinge sind vielseitig. Ein Typischer Anwendungsfall ist das Überwachen und Auswerten von Daten. Ein Computerprogramm könnte über einen digitalen Zwilling auf die Laufzeitinformationen einer Maschine zugreifen und somit die Auslastung errechnen. Die gesammelten Daten können als Prognose für die Zukunft genutzt werden, um die Produktion zu optimieren.

Ein weiteres Einsatzgebiet ist das zentrale Abspeichern von Ereignissen einer Maschine, wie Wartungsarbeiten, Reparaturen oder Fehlermeldungen. Ziel dabei ist, den gesamten Lebenszyklus eines Produkts zu überwachen, um beispielsweise die Zuverlässigkeit zu verbessern.

Es ist möglich eine Echt-Zeit-Kommunikation zwischen einem Digitalem Zwilling und seinem realen Objekt zu erlauben. Somit kann ein Computerprogramm zum Beispiel eine Maschine steuern, indem es die Maschine über den digitalen Zwilling zu einer Aktion auffordert.

2.2 Digital Twin System

Die Implementierungen digitaler Zwillinge können sehr unterschiedlich sein und es gibt keinen allgemein verbreiteten Standard. Ein firmeninterner Ansatz zur Vereinheitlichung digitaler Zwillinge ist das Digital Twin System (DTS). Der Bosch Smantic Stack, eine Abteilung von Bosch, ist für die Entwicklung und Bereitstellung des Systems verantwortlich. Ein Teil des Digital Twin Systems ist nur kommerziell verfügbar. Ein weiterer Teil wird als Open-Source-

Projekt in der Open Manufacturing Platform (OMP) veröffentlicht. Dies ist eine Vereinigung vieler Unternehmen (z.B. BMW, Bosch, Microsoft, ZF), die das Ziel hat, die Fertigung von Produkten voranzutreiben. [vgl. OMP20]

Das Grundkonzept des Digital Twin Systems ist, dass sich ein Digital Zwilling aus Aspekten zusammensetzt. Ein Aspekt ist eine bestimmte Sichtweise auf das repräsentierte Objekt. Entwickelt man einen digitalen Zwilling einer Maschine, könnten man beispielsweise folgende Aspekte definieren.

- Einen Aspekt für den aktuellen Maschinenzustand, der Informationen über aktuelle Eigenschaften und Sensorwerten darstellt.
- Einen Aspekt für die Historie der Maschine, der den Wartungsverlauf zeigt.
- Einen Aspekt, der alle Produkte darstellt, die von der Maschine gefertigt werden können.

Die konkrete Implementierung eines Aspekts ist eine API, also eine Schnittstelle, auf die andere Programme zugreifen können, um Daten anzufragen. Diese Programme werden als Lösung (englisch: Solution) bezeichnet und setzen die genannten Anwendungsfälle (Überwachen, Auswerten, Steuern usw.) um.

Der Datenaustausch zwischen Aspekt und der Software Lösung findet auf Basis eines festgelegten Protokolls statt. Im Digital Twin System stehen aktuell die beiden Nachrichtenprotokolle HTTP und MQTT zur Verfügung.

HTTP steht für „Hyptertext Transfer Protokol“ wird zur Übertragung von Text beliebiger Länge eingesetzt. Ursprünglich war es für das Versenden von HTML-Nachrichten konzipiert [BL91]. Heutzutage wird es für alle möglichen Nachrichtenarten eingesetzt. In Webanwendungen werden zusätzlich zur HTML-Nachricht beispielsweise Bilddateien, Stylesheets (z.B. CSS) oder JavaScript-Dateien versendet. Zum reinen Informationsaustausch nutzt man häufig das JSON-Format. Beim Datenaustausch gibt es immer zwei Teilnehmer: einen Server und einen Client. Der Client kann eine Anfrage in Form einer Nachricht an den Server stellen. Der Server antwortet mit einer zweiten Nachricht. Die Beziehung zwischen den beiden Kommunikationspartnern ist asymmetrisch, da der Server keine Anfrage an den Client stellen kann. Des weiteren ist die Kommunikation zustandslos, das heißt auf eine Anfrage folgt genau eine Antwort. HTTP ist zustandslos, das heißt der Server behandelt jede Anfrage isoliert und unabhängig davon, welche Anfragen davor gesendet wurden. [vgl. Soc99]

MQTT ist kurz für „Message Queuing Telemetry Transport“. Es ist ein einfaches Nachrichten-Transport-Protokoll, das auf das Prinzip publish/subscribe basiert. Ein Teilnehmer kann Ereignisse abonnieren (subscribe) und wird benachrichtigt, wenn ein anderer Teilnehmer das Eintreten des Ereignisses veröffentlicht (publish). [Ban+19]

Um eine problemlose Kommunikation zwischen Aspekt und Solution zu ermöglichen, sollte zusätzlich zum Übertragungsprotokoll ein Datenmodell vorliegen, das die Struktur und den Inhalt der Daten spezifiziert. Es gibt bereits etablierten Metamodelle, die das Spezifizieren von Datenmodellen erlauben, zum Beispiel JSON Schema [vgl. Wri+22]. Für das Digital Twin System bei Bosch wird ein eigenes Meta Modell eingesetzt: das BAMM Aspect Meta Model. Es wurde für den Einsatz mit digitalen Zwillingen entwickelt und wird als Open-Source-Projekt in der Open Manufacturing Platform (OMP) veröffentlich.

2.3 Resource Description Framework (RDF)

Das Resource Description Framework (RDF) ist ein Modell, das zur Beschreibung von Daten eingesetzt wird [vgl. Gro14]. Die Datenstruktur formt einen gerichteten Graphen, der sich aus Knoten und Verbindungen (Kanten) zwischen der Knoten zusammensetzt. Ein Ziel von RDF ist, einen Standard schaffen, der emöglicht, beliebige Informationen in maschinenlesbarer Form darzustellen [vgl. SR14, Sektion 2].

Zu RDF gehören mehrere Spezifikationen, die beispielsweise das Datenmodell oder eine Syntax zur Beschreibung von Daten festlegen. Das World Wide Web Consortium (W3O) veröffentlicht diese Spezifikationen und viele weitere unter dem Begriff „Semantic Web“. Dieser Ausdruck beschreibt die Vision, dass beliebige Daten im Internet bereitgestellt, ausgetauscht und verarbeitet werden können. [vgl. W3O15]

2.3.1 Datenmodell

Ein RDF-Graph setzt sich aus einer beliebig großen Menge sogenannter Triples zusammen. [vgl. CWL14, Sektion 3.1] Ein einzelnes Triple ist formal gesehen ein 3-Tupel mit folgenden Elementen:

1. Das erste Element bezeichnet man als Subjekt. Es repräsentiert den Startknoten, von dem eine Verbindung ausgeht.
2. Das zweite Element nennt man Prädikat. Es stellt die konkrete Verbindung dar.

Subjekt	Prädikat	Objekt
<urn:relation#Anton>	<urn:relation#name>	"Anton"
<urn:relation#Anton>	<urn:relation#hatKind>	<urn:relation#Berta>
<urn:relation#Berta>	<urn:relation#hatVater>	<urn:relation#Anton>
<urn:relation#Berta>	<urn:relation#geboren>	_:Geburtstag
_:Geburtstag	<urn:relation#ort>	"Stuttgart"
_:Geburtstag	<urn:relation#datum>	"1980-01-01"

Tabelle 2.1: Beispiele für Triples

3. Das dritte Element bezeichnet man als Objekt. Dies ist der Endknoten, auf den die Verbindung zeigt.

Somit sind Subjekt und Objekt immer ein Knoten, das Prädikat ist immer eine Kante.

Es gibt drei Arten von Knoten:

1. Als Resource bezeichnet man einen Knoten, der sowohl Ein- als auch Ausgangsknoten besitzen kann und durch einen URI identifiziert. URI sind innerhalb eines Graphen einzigartig und können von Computerprogrammen eingesetzt werden, um einen bestimmten Knoten zu suchen.
2. Blank Nodes sind Knoten, die Ein- und Ausgangsknoten besitzen können, aber nicht durch einen URI identifiziert sind. Ein Programm kann nicht nach einem Blank Node suchen.
3. Literale sind Werte wie Ganzzahlen oder Strings und haben keine ausgehenden Kanten.

Kanten werden immer durch einen URI identifiziert.

In Tabelle 2.1 sind Beispiele für Triples zu sehen, die zusammen einen Graphen formen. Die beiden URIs `<urn:relation\#Anton>` und `<urn:relation\#Berta>` sind Resourcen. `_:Geburtstag` ist ein Blank Node und die Werte `"Anton"`, `"Stuttgart"` und `"1980-01-01"` sind Literale.

2.3.2 RDF Syntax

N-Triples ist eine Syntax, bei der beliebig viele Triples hintereinander aufgelistet werden [vgl. Bec14]. Die Grammatik ist in Form von regulären Ausdrücken definiert. Ein Ausschnitt davon ist in den Formeln 2.1 zu sehen. Jedes n-Triples-Dokument setzt sich demnach aus abwechselnden Triples und Zeilenumbrüchen (EOL, End-Of-Line) zusammen. Ein Triple besteht aus Subjekt, Prädikat und Objekt sowie einem Punkt am Ende. N-Triples-Dateien werden mit

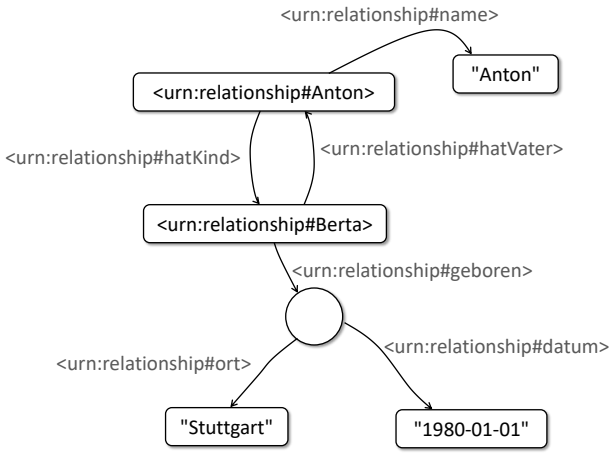

Abbildung 2.1: Visualisierung des Graphen, der von den Triples in Tabelle 2.1 beschrieben wird.

der Dateiendung „.nt“ versehen. Die beispielhaften Triples aus Tabelle 2.1 sind in Quellcode 2.1 dargestellt.

$$\text{ntripleDoc} ::= \text{triple?}\ (\text{EOL triple})^*\ \text{EOL?} \tag{2.1}$$

$$\text{triple} ::= \text{subject predicate object “.“}$$

```
1 <urn:relation#Anton> <urn:relation#name> "Anton" .
2 <urn:relation#Anton> <urn:relation#hatKind> <urn:relation#Berta> .
3 <urn:relation#Berta> <urn:relation#hatVater> <urn:relation#Anton> .
4 <urn:relation#Berta> <urn:relation#geboren> _:Geburtstag .
5 _:Geburtstag <urn:relation#ort> "Stuttgart" .
6 _:Geburtstag <urn:relation#datum> "1980-01-01" .
```

Quellcode 2.1: Beispiel einer N-Triples-Datei

Eine zweite Syntax, mit der RDF-Graphen beschrieben werden können, heißt „Terse RDF Triple Language“ (TTL) und wird auch als „Turtle“ bezeichnet [vgl. Bec+14]. Turtle-Dateien besitzen die Datei-Endung „.ttl“. Die Sprache, die von der TTL-Grammatik beschrieben wird, ist eine Obermenge der N-Triples-Sprache. Das heißt, jede gültige N-Triples-Datei ist auch eine gültige Turtle-Datei. Zusätzlich erlaubt die Turtle-Syntax weitere Schreibweisen, die es ermöglichen, Graphen übersichtlicher und mit weniger Text darzustellen. Durch syntaktische Äquivalenzumformungen lässt sich jede Turtle-Datei wieder auf eine Liste von Triples zurückführen. Somit kann jeder beliebige Graph sowohl mit N-Triples als auch mit Turtle beschrieben werden. Im Folgenden werden einige wichtige Syntax-Merkmale genauer vorgestellt.

Namespaces Die URIs in einer Datei beginnen häufig mit dem gleichen Zeichenfolge. Benannte Knoten unterscheiden sich meist nur am letzten Abschnitt des URIs. Namespaces ermöglichen, eine abgekürzte Schreibweise für URIs zu verwenden, um die Datei übersichtlicher zu machen. Ein Namespace wird mit dem Schlüsselwort `@prefix` festgelegt, wie in Quellcode 2.2 zu sehen ist.

```
1 @prefix rel: <urn:relation#> .
2
3 rel:Anton rel:name "Anton" .
4 rel:Anton rel:hatKind rel:Berta .
5 rel:Berta rel:hatVater rel:Anton .
6 rel:Berta rel:geboren _:Geburtstag .
7 _:Geburtstag rel:ort "Stuttgart" .
8 _:Geburtstag rel:datum "1980-01-01" .
```

Quellcode 2.2: TTL-Datei mit Namespace

Liste von Prädikaten Wenn in einer Turtle-Datei mehrere Triples mit dem gleichen Subjekt vorkommen, können diese Triples zusammengefasst werden. Dazu wird das erste Triple nicht mit einem Punkt, sondern mit einem Semikolon abgeschlossen. In der nächsten Zeile kommen dann nur noch Prädikate und Objekt vor. Das Subjekt wird aus der vorangegangenen Zeile wiederverwendet (siehe Quellcode 2.3).

```
1 rel:Anton rel:name "Anton" ;
2           rel:hatKind rel:Berta .
```

Quellcode 2.3: TTL-Datei mit einer Liste von Prädikaten

Verschachtelung von Blank Nodes Blank Nodes werden häufig eingesetzt, um komplexe Strukturen darzustellen. Im Beispiel setzt sich der Geburtstag aus Ort und Datum zusammen. Blank Nodes werden immer in einem bestimmten Kontext verwendet und verlieren ohne diesen Kontext ihre semantische Bedeutung. Der Knoten `_:Geburtstag` beispielsweise ist bedeutungslos, wenn man nicht weiß, dass er zu Berta gehört. Aus diesem Grund ist es sinnvoll, dass der Knoten nicht durch einen URI auffindbar ist. Auf ihn kann nur zugegriffen werden, indem man vom Knoten Berta entlang dem Prädikat `rel:geboren` navigiert.

Auch für Blank Nodes gibt es eine Kurzschreibweise. Im Beispiel wird anstatt `_:Geburtstag` ein Paar eckiger Klammers `[]` geschrieben. Alle Prädikate, die von dem Blank Nodes ausgehen, werden zusammen mit dem Objekt in die eckigen Klammern geschrieben (siehe Quellcode 2.4)

```
@prefix rel: <urn:relation#> .

rel:Anton rel:name "Anton" ;
          rel:hatKind rel:Berta .
rel:Berta rel:hatVater rel:Anton ;
          rel:geboren [ rel:ort "Stuttgart" ;
                        rel:datum "1980-01-01" ] .
```

Quellcode 2.4: TTL-Datei mit verschachtelter Blank Node

RDF-Listen Es gibt die Möglichkeit, von einem Knoten mehrere ausgehende Kanten zu spezifizieren, die den gleichen URN haben (siehe Quellcode 2.5 oben). Diese Modellierungmöglichkeit kann sehr einfach eingesetzt werden, aber hat den Nachteil dass die Reihenfolge der Objekte nicht spezifiziert ist. Das heißt, wenn ein Computerprogramm die Turtle-Datei einliest, ist die Reihenfolge der Elemente zufällig bzw. abhängig von der Implementierung.

Möchte man eine Liste definieren, bei der die Reihenfolge festgelegt ist, kann man RDF-Listen einsetzen (siehe Quellcode 2.5 unten). Bei der Syntax werden runde Klammern eingesetzt. Wenn Die TTL-Datei geparst wird, ist die Liste als Binärbaum dargestellt (siehe 2.2). Jeweils ein Ausgang des Knotens zeigt auf ein Listenelement, der andere Ausgang auf den Rest der Liste. Das Ende der Liste wird durch ein spezielles Objekt (rdf:nil) markiert.

```
# Knoten mit drei ausgehenden Kanten mit gleichen URIs
rel:Anton rel:name "Anton" ;
          rel:hatKind rel:Berta ;
          rel:hatKind rel:Caesar ;
          rel:hatKind rel:Dora .

# Knoten mit RDF-Liste
rel:Anton rel:name "Anton" ;
          rel:hatKinder ( rel:Berta rel:Caesar rel:Dora ) .
```

Quellcode 2.5: Knoten mit mehreren ausgehenden Kanten mit dem gleichen URI

2.4 BAMM Aspect Meta Model

2.5 Software Defined Vehicle

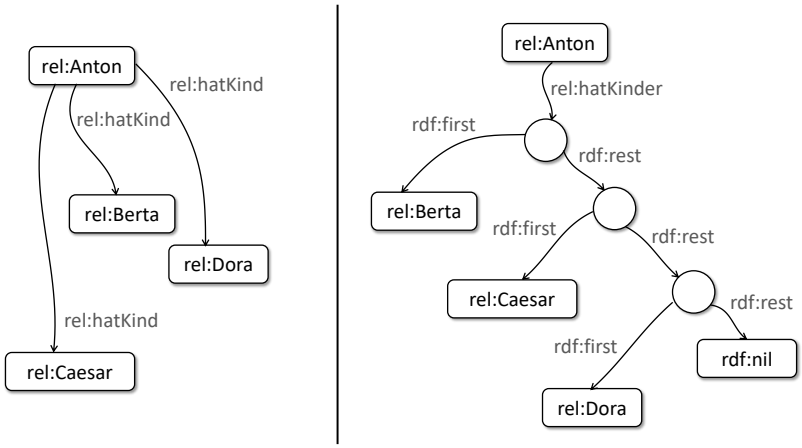

Abbildung 2.2: Vergleich der beiden Graphen bei mehrfacher Verwendung eines Prädikats (links) und beim Einsatz von von RDF-Listen (rechts).

3. Konzeption

4. Implementierung

5. Zusammenfassung und Ausblick

Literaturverzeichnis

[Ban+19] Andrew Banks u. a., Hrsg. *MQTT Version 5.0*. 2019.

[Bec14] David Beckett, Hrsg. *RDF 1.1 N-Triples*. 2014. URL: https://www.w3.org/TR/n-triples/ (besucht am 06. 07. 2022).

[Bec+14] David Beckett u. a., Hrsg. *RDF 1.1 Turtle*. 2014. URL: https://www.w3.org/TR/rdf11-concepts/ (besucht am 05. 07. 2022).

[BL91] Tim Berners-Lee. *The Original HTTP as defined in 1991*. 1991. URL: https://www.w3.org/Protocols/HTTP/AsImplemented.html (besucht am 15. 07. 2022).

[CWL14] Richard Cyganiak, David Wood und Markus Lanthaler, Hrsg. *RDF 1.1 Concepts and Abstract Syntax*. 2014. URL: https://www.w3.org/TR/rdf11-concepts/ (besucht am 05. 07. 2022).

[Gro14] RDF Working Group. *RDF*. 2014. URL: https://www.w3.org/2001/sw/wiki/RDF (besucht am 04. 07. 2022).

[OMP20] Open Manufacutring Platform OMP. *Accelerating Manufacturing Innovation at Scale: Solving mutual challenges through open collaboration*. 2020. URL: https://open-manufacturing.org/wp-content/uploads/sites/101/2020/06/omp_accelerating_manufacturing_at_scale_061620.pdf (besucht am 17. 07. 2022).

[Soc99] The Internet Society. *Hypertext Transfer Protocol – HTTP/1.1*. Hrsg. von R. Fielding u. a. 1999. URL: https://datatracker.ietf.org/doc/html/rfc2616 (besucht am 17. 07. 2022).

[SR14] Guus Schreiber und Yves Raimond, Hrsg. *RDF Primer*. 2014. URL: https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/ (besucht am 04. 07. 2022).

[W3O15] World Wide Web Consortium W3O. *Semantic Web*. 2015. URL: https://www.w3.org/standards/semanticweb/ (besucht am 04. 07. 2022).

[Wri+22] A. Wright u. a. *JSON Schema: A Media Type for Describing JSON Documents*. 2022. URL: https://json-schema.org/draft/2020-12/json-schema-core.html (besucht am 17. 07. 2022).