The Spirits of Web of Things Past, Present, and Yet to Come

Matthias Kovatsch, Summer School on AI for Industry 4.0, 27 Jul 2020
Dr. Matthias Kovatsch
Principal Researcher

Education
• 2003 – 2008 Dipl.-Ing., FAU Erlangen-Nürnberg, Germany
• 2009 – 2014 Dr. sc., ETH Zurich, Switzerland

Work Experience
• 2006 – 2009 Working Student, Fraunhofer IIS, Germany
• 2011 ~ 2015 Visiting Researcher, RISE SICS, Sweden (multiple visits)
• 2014 – 2015 Visiting Researcher, Samsung Electronics, South Korea
• 2016 – 2018 Senior Research Scientist, Siemens AG, Germany
• 2019 – now Principal Researcher, Huawei Technologies, Germany

Roles
• Eclipse IoT Working Group Member (inactive)
• IETF IoT Directorate Member
• W3C Web of Things Interest Group & Working Group Co-Chair (inactive)
• OPC Foundation Field Level Communication Initiative Steering Committee Member

Projects
• Eclipse Thingweb (node-wot)
• Eclipse Californium
• Contiki Erbium
• Firefox Copper (deprecated)
The Spirits of Web of Things

Past

- Web Presences
- Putting Things to REST
- Constrained RESTful Environments

Present

- W3C Standardization
- Thing Description
- node-wot

Yet to Come

- More Bindings
- More Semantics
- Better Actions
The Spirits of Web of Things

Past
• Web Presences
• Putting Things to REST
• Constrained RESTful Environments

Present
• W3C Standardization
• Thing Description
• node-wot

Yet to Come
• More Bindings
• More Semantics
• Better Actions
A Little History

Connecting the physical world with the virtual world

Web Presences
People, Places, Things

2002
“Cooltown”

Internet of Things

UbiComp

QR Codes
Virtual Representations
Web Presences for People, Places, Things

WoT Ideas from 2002 ... 2006

Physical Web: URIs via BLE beacon

A Little History

Connecting the physical world with the virtual world

IoT

1999

“Smart Dust”

2001

UbiComp

RFID

QR Codes

Connected Devices

2003

TinyOS

Contiki

Virtual Representations

2002

“Cooltown”

2007

Web of Things

Web Presences

People, Places, Things

Usability Interoperability
Putting Things to REST – Towards WoT

- Use Representational State Transfer, the architectural style of the Web, to communicate with Things
- Web resources allow loose coupling between devices and applications
- HTTP enables interoperability and libraries available for most platforms

A Little History

Web Presences
People, Places, Things
2002
“Cooltown”

Usability Interoperability
2007

Web of Things

Gateways
2008

IP-based IoT
2010

Constrained RESTful Environments

RFID
VR Codes
Virtual Representations
2003
Connected Devices

6LoWPAN

Gateways
2008

IP-based IoT
2010

Constrained RESTful Environments

Wireless Sensor Networks

TinyOS
Contiki

IoT
RFID
QR Codes
Virtual Representations
2003
Connected Devices

6LoWPAN

IP-based IoT
2010

Constrained RESTful Environments

Web server with lots of adaptation code
Constrained Application Protocol (CoAP)

- New Web protocol for low-power networks and resource-constrained devices
- Designed from scratch following the REST architectural style
- Transparent mapping to HTTP
- Additional features for IoT applications

Why the Web?

- Internet of Things
  - Domain expertise
  - Embedded developers
  - Optimized protocols and formats
    - Silos with high integration costs

- World Wide Web
  - Interoperability and usability
  - Web developers
  - HTTP, JSON, scripting
    - Application mashups

- Web of Things
  - Take patterns that worked for the Web
  - Adapt and apply them to the IoT
But this an **AI** Summer School?!
Spirit of the Past: Digitalization

• All these technologies form the foundation to enable AI
  • Connected devices are required to collect the data for data-driven machine learning
  • Proper protocols and APIs are required to enable automated control and optimization
  • Developers are provided to carry out the digitalization at scale

• Digitalization allows to monitor and quantify processes in real-time
• “Industry 4.0” describes the digitalization of industries
The Spirits of Web of Things

**Past**
- Web Presences
- Putting Things to REST
- Constrained RESTful Environments

**Present**
- W3C Standardization
- Thing Description
- node-wot

**Yet to Come**
- More Bindings
- More Semantics
- Better Actions
W3C Standardization Activity

W3C WoT Community Group (CG)
No charter needed

- Started summer 2013
- ~300 participants
- Free discussion (no membership needed)

- W3C WoT Workshop, Berlin, 2014
- Identify stakeholders for standards work

- Believe in benefits of Web technology for IoT
- Web standards are horizontal and neutral
W3C Standardization Activity

**W3C WoT Interest Group (IG)**
https://w3c.github.io/wot/charters/wot-ig-2019.html
- Started spring 2015
- ~200 participants
- Informal work, outreach
- Exploration of new building blocks
- “PlugFests” validation with running code
- “OpenDays” with also external speakers
- Liaisons and collaborations with other organizations and SDOs

**W3C WoT Working Group (WG)**
https://www.w3.org/2020/01/wot-wg-charter.html
- Started end of 2016
- ~100 participants
- Normative work
- Work on deliverables
- W3C Patent Policy for royalty-free standards
- Only W3C Members and Invited Experts
Describe Existing IoT Ecosystems

Provide **semantic metadata** that uniformly describes how to interact with Things

Form an implicit interaction model, which is usually not well documented

The IoT has a plethora of protocols, often dialects due to custom options for protocols such as HTTP, CoAP, MQTT, etc.

**Any IoT Device**
- Application
- SDK
- Data Model
- Protocol

Define a **common runtime** similar to the Web browser to implement Thing behavior

Every SDK and library is different, so that application development is expensive

Capture each protocol once in a uniform template that describes how to **configure protocol stacks** (e.g., CoAP or MQTT) to send the message expected by the Thing
Describe Existing IoT Ecosystems

**WoT Thing Description (TD)**

JSON-LD representation format to describe Thing instances with metadata. Uses formal interaction model and domain-specific vocabularies to uniformly describe Things, their capabilities, and how to use them.

An *index.html* for Things

Properties

Events

Actions

“Interaction Affordances”

**Any IoT Device**

- Application
- SDK
- Interaction Model
- Protocol

The IoT has a plethora of protocols, often dialects due to custom options

**Define a common runtime similar to the Web browser to implement Thing behavior**

Every SDK and library is different, so that application development is expensive

Capture each protocol once in a uniform template that describes how to configure protocol stacks (e.g., CoAP or MQTT) to send the message expected by the Thing
Affordances

- “Affordance refers to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used.” – Donald Norman on everyday things

- “… the simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions.” – Roy Fielding on hypermedia
Describe Existing IoT Ecosystems

The IoT has a plethora of protocols, often dialects due to custom options. To capture each protocol once in a uniform template that describes how to configure protocol stacks (e.g., CoAP or MQTT) to send the message expected by the Thing.

Define a common runtime similar to the Web browser to implement Thing behavior.

Capture each protocol once in a uniform template that describes how to configure protocol stacks (e.g., CoAP or MQTT) to send the message expected by the Thing.

Every SDK and library is different, so that application development is expensive.

WoT Thing Description (TD)

JSON-LD representation format to describe Thing instances with metadata. Uses formal interaction model and domain-specific vocabularies to uniformly describe Things, their capabilities, and how to use them.

An index.html for Things

Properties
Events
Actions

“Interaction Affordances”
Describe Existing IoT Ecosystems

**WoT Thing Description (TD)**

JSON-LD representation format to describe Thing instances with metadata. Uses formal interaction model and domain-specific vocabularies to uniformly describe Things, their capabilities, and how to use them.

**Any IoT Device**

- Application
- SDK
- Interaction Model
- Protocol Bindings

**WoT Binding Templates**

Capture how the formal Interaction Model is mapped to concrete protocol operations (e.g., CoAP) and platform features (e.g., OCF). The templates are done once per ecosystem and require a vocabulary for each base protocol (e.g., HTTP in RDF).

Define a **common runtime** similar to the Web browser to implement Thing behavior.

Every SDK and library is different, so that application development is expensive.

An *index.html* for Things

**Properties**

**Events**

**Actions**
Describe Existing IoT Ecosystems

WoT Thing Description (TD)

JSON-LD representation format to describe Thing instances with metadata. Uses formal interaction model and domain-specific vocabularies to uniformly describe Things, their capabilities, and how to use them.

WoT Scripting API

Common JavaScript object API for an IoT runtime system similar to the Web browser. Enable portable scripts that implement the behavior of Things and Consume across different vendors, devices, and environments. Behavior must also be identifiable through domain-specific vocabulary terms.

WoT Binding Templates

Capture how the formal Interaction Model is mapped to concrete protocol operations (e.g., CoAP) and platform features (e.g., OCF). The templates are done once per ecosystem and require a vocabulary for each base protocol (e.g., HTTP in RDF).
Describe Existing IoT Ecosystems

**WoT Thing Description (TD)**

JSON-LD representation format to describe Thing instances with metadata. Uses **formal interaction model** and domain-specific vocabularies to uniformly describe Things, their capabilities, and how to use them.

**WoT Scripting API**

Common JavaScript object API for an IoT runtime system similar to the Web browser. Enable **portable scripts** that implement the behavior of Things and Consume across different vendors, devices, and environments. Behavior must also be identifiable through domain-specific vocabulary terms.

**WoT Binding Templates**

Capture how the **formal Interaction Model** is mapped to concrete protocol operations (e.g., CoAP) and platform features (e.g., OCF). The templates are done once per ecosystem and require a vocabulary for each base protocol (e.g., HTTP in RDF).
W3C WoT Architecture

- **Cloud**
  - WoT Servient
    - Behavior
    - Affordances
    - Data Schemas
    - Security Configuration
    - Protocol Bindings

- **Edge**
  - WoT Servient
    - Behavior
    - Affordances
    - Data Schemas
    - Security Configuration
    - Protocol Bindings

- **Local Network**
  - Existing Device
    - +
    - Thing

- **Optional Remote Access and Synchronization**
  - Thing-to-Cloud
  - Thing-to-Gateway

- **Seamless Web Integration**
  - Thing
    - Behavior
    - Affordances
    - Data Schemas
    - Security Configuration
    - Protocol Bindings

- **Direct Thing-to-Thing Interaction**
  - Thing + Consumer
    - Behavior
    - Affordances
    - Data Schemas
    - Security Configuration
    - Protocol Bindings

- **Complement Existing Devices**
  - Protocol

- **Data Schemas**
  - Thing
    - = Thing Description (TD)

= Thing Description (TD)
W3C WoT Thing Description

```json
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        { "cov": "http://proto.example.org/coap-binding#",
          "iot": "http://schema.example.org/" }
    ],
    "@type": ["Thing"],
    "id": "urn:dev:ops:32473-WoTLamp-1234",
    "title": "MyLEDThing",
    "securityDefinitions": {
        "default": { "scheme": "bearer" },
        "dtls": { "scheme": "psk" }
    },
    "security": ["default"],
    "properties": {
        "brightness": {
            "@type": ["iot:Brightness"],
            "description": "Sets the brightness between 0 and 100%",
            "type": "integer",
            "minimum": 0,
            "maximum": 100,
            "iot:Unit": "iot:Percent",
            "forms": [ ...
        }
    }
}
```
Like with HTML forms, the server/Thing can tell the client/Consumer how to create a request.
Combining Existing Standards

**Combining Existing Standards**

**JSON Schema**
- **Description** of existing data formats
- **Validation** of payloads through available implementations
- **Already in use** by industry, e.g., OpenAPI (microservices), Open Connectivity Foundation
Combining Existing Standards

**JSON Schema**
- **Description** of existing data formats
- **Validation** of payloads through available implementations
- **Already in use** by industry, e.g., OpenAPI (microservices), Open Connectivity Foundation

**JSON-LD 1.1**
- **Semantic meaning** through controlled vocabularies enables interoperability
- **Reasoning** through ontologies makes TDs machine-understandable
- **Knowledge Graphs** interlink TDs with all related information
Combining Existing Standards

**Web Links and Forms**

*Uniform REST interface* describes how to interact given an IoT protocol such as HTTP and CoAP, but also MQTT, Modbus, UA Binary, etc.

*URIs* encode the IoT protocol and target address in a simple string

*Media Types* identify the payload format (e.g., application/json)

**JSON Schema**

*Description* of existing data formats

*Validation* of payloads through available implementations

*Already in use* by industry, e.g., OpenAPI (microservices), Open Connectivity Foundation

**JSON-LD 1.1**

*Semantic meaning* through controlled vocabularies enables interoperability

*Reasoning* through ontologies makes TDs machine-understandable

*Knowledge Graphs* interlink TDs with all related information
node-wot: Build Your Own Web of Things

node-wot Servient

Protocol Bindings
- HTTP(S)
- CoAP(S)
- MQTT
- ...

Scripting API

WoT Runtime

Consumer Script

node-wot Servient

Protocol Bindings
- HTTP(S)
- CoAP(S)
- MQTT
- ...

Scripting API

WoT Runtime

Thing Script

Security Metadata

System API

Local Hardware

Proprietary Communication

Legacy Device
Eclipse Thingweb: node-wot & more


https://projects.eclipse.org/projects/iot.thingweb
Independent digitalization led to various siloed ecosystems
• Custom protocols and data models form implicit interaction models
• High integration costs to access and harmonize data
• Documentation usually for human readers only

W3C WoT aims at breaking up the silos for interoperability in the IoT and at making interactions and data machine-understandable through semantic annotations
# The Spirits of Web of Things

<table>
<thead>
<tr>
<th>Past</th>
<th>Present</th>
<th>Yet to Come</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Web Presences</td>
<td>• W3C Standardization</td>
<td>• More Bindings</td>
</tr>
<tr>
<td>• Putting Things to REST</td>
<td>• Thing Description</td>
<td>• More Semantics</td>
</tr>
<tr>
<td>• Constrained RESTful</td>
<td>• node-wot</td>
<td>• Better Actions</td>
</tr>
<tr>
<td>Environments</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Industrial Protocol Bindings
Industrial IoT Ecosystems

**BACnet**
- Building automation
- Protocol with object-based information model

**Modbus**
- Energy systems, supervisory logic
- Simple protocol for addressing registers
Industrial IoT Ecosystems

**OPC Unified Automation (UA)**
- Factory and process automation
- Graph-based information model and communication protocols
  → For management/monitoring

**OPC UA Field-Level Communications (FLC)**
- Extension to cover field controllers and devices
- Integrates TSN, which is configured via NETCONF
  → For real-time applications
OPC UA Binding

• Mapping to Properties, Actions, and Events (with \(\text{opc:methodName}\) field in from)
  • Variable nodes \(\rightarrow\) Properties
  • Method nodes \(\rightarrow\) Actions \hspace{1cm} \text{(node attributes become TD fields)}
  • Node alerts \(\rightarrow\) Events
• DataSchema
  • OPC UA uses \text{binary data types}, hence JSON Schema must be further annotated
  • Would be binding-specific, ergo form field, but form metadata not available to ContentSerdes
  \(\rightarrow\) \(\text{opc:dataType}\) annotation in DataSchema
• Form href URI (UA-Binary over TCP)
  • Adopt opc.tcp schema, but extend with \(;\)-separated query similar to OPC UA tooling
  \(\rightarrow\) \(\text{opc.tcp://localhost:5050/server-path?ns=1;\text{s=mynode}}\)
• Form contentType
  • UA-Binary has no registered mediastype (similar to URI schema, needs a push within OPCF)
  \(\rightarrow\) application/x.opcua

NETCONF Binding

• Mapping to Properties, Actions, and Events (built on RESTCONF)
  • Leaf-nodes → Properties
  • RPCs → Actions
  • Notifications → Events
• DataSchema
  • Mostly works, as YANG is XML-based
  • Must add mechanism for XML node attributes (e.g., )
  → nc:container and nc:attribute annotations – should become general XML mechanism
• Form href URI (XML messages over SSH transport)
  • Similar to RESTCONF URIs, but with support for datastores (RESTCONF has implicit rules)
  → netconf://localhost:830/running/ietf-interfaces:interfaces/interface=eth0/type
• Form contentType
  • Re-usable from RESTCONF
  → application/yang-data+xml
Examples

**OPC UA**

```
"properties": {
    "Velocity": {
        "type": "number",
        "observable": true,
        "opc:datatype": "Double",
        
        "forms": [{
            "href": "opc.tcp://xts.local:5050/ns=1;\s=GVL.OPC_Interface.MOVER[1].Input.Velocity",
            "opc:datatype": "Double",
            "contentType": "application/x.opcua-binary" ]},
        ... },

    "actions": {
        "Execute": {
            "input": {
                "type": "boolean",
                "opc:datatype": "Boolean"
            },
            "output": {
                "type": "boolean",
                "opc:datatype": "Boolean"
            },
            "forms": [{
                "href": "opc.tcp://xts.local:5050/ns=1;\s=GVL.OPC_Interface.XTS.Input.Execute",
                "opc:method": "Call"
            }]
        }
    }
```

**NETCONF**

```
"properties": {
    "admin-control-list": {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "index": {
                    "type": "number",
                    "minimum": 0,
                    "maximum": 127
                },
                "time-interval": {
                    "type": "number",
                    "minimum": 0,
                    "maximum": 4294967295
                },
                "gate-state": {
                    "type": "number",
                    "minimum": 0,
                    "maximum": 255
                }
            }
        }
    },
    "uriVariables": {
        "datastore": {
            "@type": "nc:Target",
            "type": "string",
            "enum": ["candidate", "running", "startup"]
        },
        "interface": {
            "type": "integer",
            "minimum": 0,
            "maximum": 7
        }
    }
```
From TDs to Knowledge Graphs
WoT Thing Description Is a Framework

• Provisioning of domain-specific vocabularies and ontologies
  • schema.org IoT Extension
    • W3C Community Group
  • Bridging existing ontologies, e.g.,
    • SSN
    • eCl@ss
    • Building Topology Ontology
  • Converting existing models, e.g.,
    • OPC UA Companion Specifications
    • OneDM (ZigBee Cluster Lib etc.)
WoT Thing Description Is a Framework

• Management of TD information
  • Thing Directory to be standardized
    • Registration
    • Lookup
  • TDs are Linked Data (JSON-LD 1.1)
    • Thing Directory is a knowledge base
    • Enrich with any data, e.g., maintenance
    • Serialize context-aware TDs, e.g., for admin
  • TDs is a modern version of the I4.0 Asset Administration Shell
    • Describes the interface
    • Can store lifetime data
    • Has no baggage of executable code

Semantic query
“JSON-LD Framing”
Hypermedia-driven Actions
Actions in Thing Descriptions

```
  "actions": {
    "fadeIn": {
      ...
    },
    "fadeOut": {
      ...
    },
    "toggle": {
      ...
    },
    "execute": {
      ...
    }
  },
  ...
```

- All examples just show simple Actions that can be completed in a single step
- This has been sufficient for most use cases considered so far
- Often there is the implicit assumption that a Consumer needs to know in what order to interact with the different affordances to follow a process
What if only the Thing Knows the Process?

Electric Vehicle Charging

Robots (PLCopen)
Let Machines use Things Like We Browse the Web

Client
Entry URI

Resource Directory

Follow links

Submit forms

Action Result

Dynamically extend process flow

Auth-Server

Thing A

Thing B

Thing C

Thing D

Choice & redundancy

48
Spirit of Yet to Come: *Up to You!*

- AI for Industry 4.0 requires industrial protocol bindings for W3C WoT
- W3C WoT only defines the framework and still requires WoT-oriented vocabularies and ontologies as JSON-LD context extension files
  → Industrial Knowledge Graphs can help
- W3C WoT currently only describes simple, single-step interactions, so that complex workflows and processes still need manual programming
  → Action responses with affordances and Hypermedia Agents can help
Contact

Dr. Matthias Kovatsch
Principal Researcher
Huawei, Munich Research Center

matthias.kovatsch@huawei.com

(Note that this is a research view)