FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Standard Behavior Descriptions for the Web of Things

Victor Charpenay
Chair of Technical Information Systems, Friedrich-Alexander University of Erlangen-Nuremberg
05.06.2019

Outline

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Introduction

Introduction

This presentation is about extending the “Behavior” building block of W3C WoT.

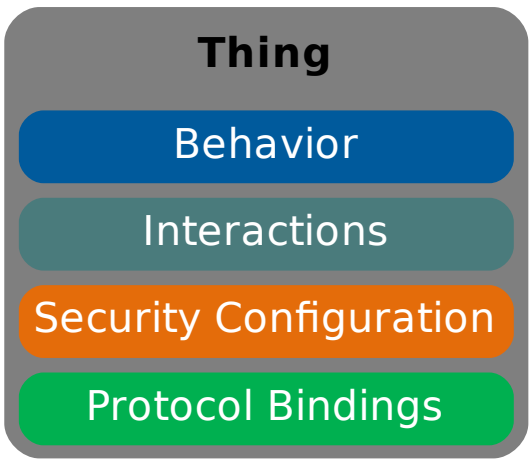


Figure: Building blocks of a WoT runtime (*Source: W3C*)

“the behavior aspect of a Thing includes both lifecycle management (...) but also the **operational behavior** of the Thing.”

Source: `https://www.w3.org/TR/wot-architecture/`

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Describing the Behavior of Things

Web Mash-ups at an Industrial Scale

Problem

WoT building blocks allow for **application mash-ups** driven by interactions between Things. How to scale up from a handful of Things to **complex industrial systems** with 1,000+ Things?

Approach

Interaction cycles ($\sim$ processes) can be described with **WoT scripts**.

Web Mash-ups at an Industrial Scale

Problem

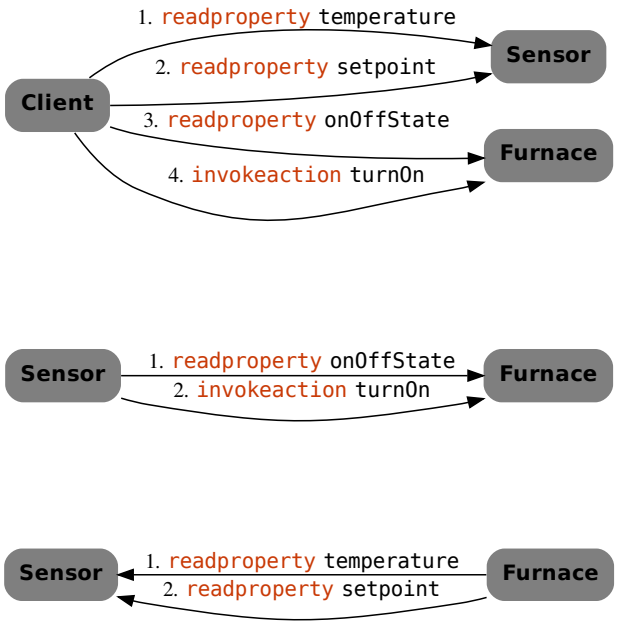
WoT building blocks allow for **application mash-ups** driven by interactions between Things. How to scale up from a handful of Things to **complex industrial systems** with 1,000+ Things?

Approach

Interaction cycles ($\sim$ processes) can be described with **WoT scripts**.

Example

Implementation of a **thermostat** with a temperature **sensor** and a **furnace**




```
var sensor  = wot.consume({...}), // temperature sensor TD
    furnace = wot.consume({...}); // furnace TD

function regulate() {
    var actual  = sensor.properties.temperature,
        desired = furnace.properties.setpoint,
        isOn    = furnace.properties.onOffState;

    if (actual < desired && !isOn)
        furnace.actions.turnOn.invoke();
    else if (isOn)
        furnace.actions.turnOff.invoke();
}

setInterval(regulate, 1000);
```

Figure: Alternative sequences of interactions among elements of a thermostat system; interactions are either mediated (top) or peer-to-peer (middle & bottom)

Why Exposing WoT Scripts?

- For Reusability
  - Similar to a Node-RED node or an `npm` package
- For Scalability
  - Automatic (re)deployment of an application onto a WoT runtime
  - Simulation of the internal behavior of a system in a so-called *Digital Twin*
  - Interaction replay in case of failure or liability testing as required in some industries

Why Exposing WoT Scripts?

- For Reusability
  - Similar to a Node-RED node or an `npm` package
- For Scalability
  - Automatic (re)deployment of an application onto a WoT runtime
  - Simulation of the internal behavior of a system in a so-called *Digital Twin*
  - Interaction replay in case of failure or liability testing as required in some industries

Why Exposing WoT Scripts?

- For Reusability
  - Similar to a Node-RED node or an `npm` package
- For Scalability
  - Automatic (re)deployment of an application onto a WoT runtime
  - Simulation of the internal behavior of a system in a so-called *Digital Twin*
  - Interaction replay in case of failure or liability testing as required in some industries

Why Exposing WoT Scripts?

- For Reusability
  - Similar to a Node-RED node or an `npm` package
- For Scalability
  - Automatic (re)deployment of an application onto a WoT runtime
  - Simulation of the internal behavior of a system in a so-called *Digital Twin*
  - Interaction replay in case of failure or liability testing as required in some industries

Why Exposing WoT Scripts?

- For Reusability
  - Similar to a Node-RED node or an `npm` package
- For Scalability
  - Automatic (re)deployment of an application onto a WoT runtime
  - Simulation of the internal behavior of a system in a so-called *Digital Twin*
  - Interaction replay in case of failure or liability testing as required in some industries

Main Issues

Deployment-specific requirements included in scripts should be **parameterized**.

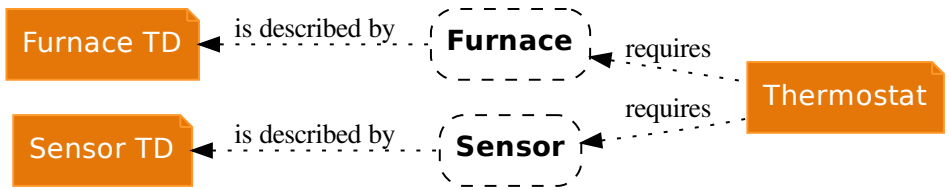


Figure: Script requirements for a thermostat

Parameters to add to the WoT API

- Input requirements: TD templates or shapes or frames
- Contextualization: semantic relation between Things

Main Issues

Deployment-specific requirements included in scripts should be **parameterized**.

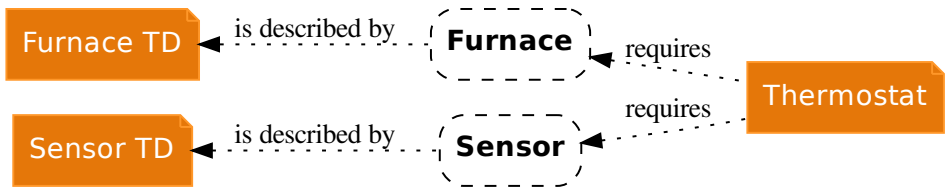


Figure: Script requirements for a thermostat

Parameters to add to the WoT API

- Input requirements: TD templates or shapes or frames
- Contextualization: semantic relation between Things

Main Issues

Deployment-specific requirements included in scripts should be **parameterized**.

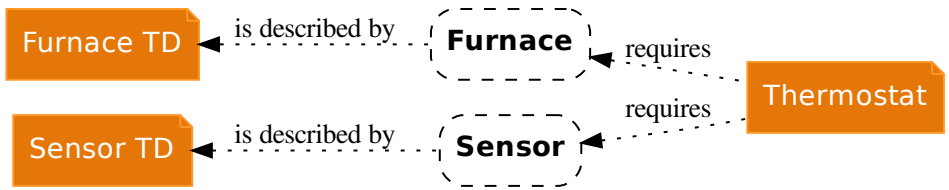


Figure: Script requirements for a thermostat

Parameters to add to the WoT API

- Input requirements: TD templates or shapes or frames
- Contextualization: semantic relation between Things

Input Requirements

Sensor TD **frame** (JSON-LD)

```
{
    "@context": "...",
    "id": "tag:sensor",
    "properties": {
        "temperature": {
            "type": "number"
        },
        "setpoint": {
            "type": "number"
        }
    },
    "actions": {}
}
```

Furnace TD **frame** (JSON-LD)

```
{
    "@context": "...",
    "id": "tag:furnace",
    "properties": {
        "onOffState": {
            "type": "boolean"
        }
    },
    "actions": {
        "turnOn": {},
        "turnOff": {}
    }
}
```

Contextualization

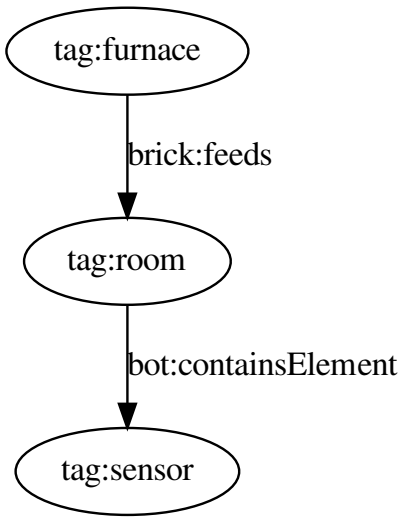


Figure: Ontological expression giving the necessary relation between the temperature sensor and the furnace for a thermostat system

Context **frame** (JSON-LD)

```
{
    "@context": "...",
    "id": "tag:furnace",
    "brick:feeds": {
        "id": "tag:room",
        "bot:containsElement": {
            "id": "tag:sensor"
        }
    }
}
```

→ Can be merged with sensor and furnace TDs in a single frame

Contextualization

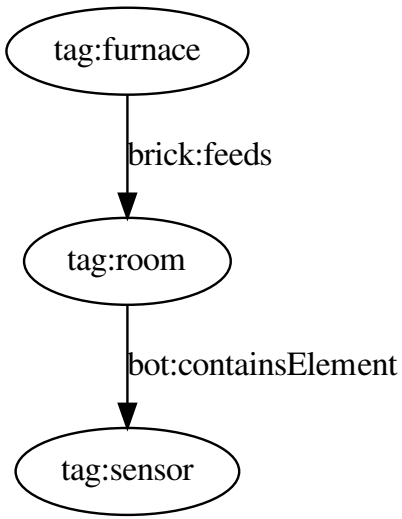


Figure: Ontological expression giving the necessary relation between the temperature sensor and the furnace for a thermostat system

Context **frame** (JSON-LD)

```
{
    "@context": "...",
    "id": "tag:furnace",
    "brick:feeds": {
        "id": "tag:room",
        "bot:containsElement": {
            "id": "tag:sensor"
        }
    }
}
```

→ Can be merged with sensor and furnace TDs in a single frame

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Existing Technologies

WoT-compatible Development Environments

Node-RED

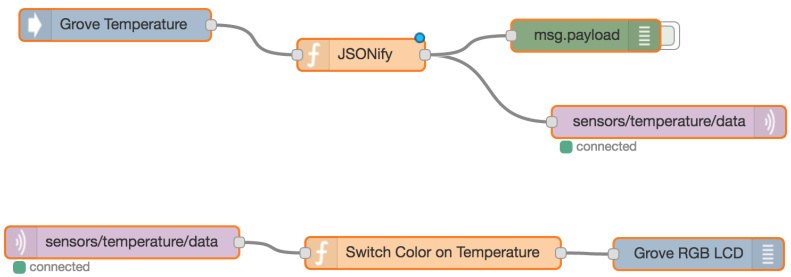


Figure: Temperature measurement and light control with Node-RED nodes and flows (*Source: Intel software*)

Eclipse 4diac (IEC 61499)

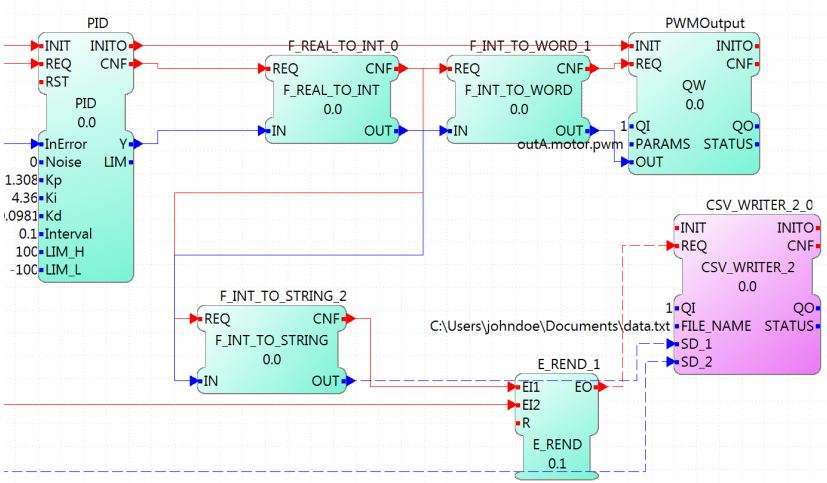


Figure: Motor control and monitoring with 4diac function blocks (*Source: Eclipse*)

Behavior Description Languages

A W3C standard to describe behavior should support all paradigms, which can be (roughly) divided in four categories.

- Process-oriented
  - State-transition machines
  - Business process modeling
- Numeric
  - Transfer functions (e.g. PID)
- Rule-based & knowledge-based
  - Horn logic (rules)
  - Belief-desire-intention model
  - Condition-action rules
- Statistical
  - Bayesian inference
  - Neural networks

Behavior Description Languages

A W3C standard to describe behavior should support all paradigms, which can be (roughly) divided in four categories.

- Process-oriented
  - State-transition machines
  - Business process modeling
- Numeric
  - Transfer functions (e.g. PID)
- Rule-based & knowledge-based
  - Horn logic (rules)
  - Belief-desire-intention model
  - Condition-action rules
- Statistical
  - Bayesian inference
  - Neural networks

Behavior Description Languages

A W3C standard to describe behavior should support all paradigms, which can be (roughly) divided in four categories.

- Process-oriented
  - State-transition machines
  - Business process modeling
- Numeric
  - Transfer functions (e.g. PID)
- Rule-based & knowledge-based
  - Horn logic (rules)
  - Belief-desire-intention model
  - Condition-action rules
- Statistical
  - Bayesian inference
  - Neural networks

Behavior Description Languages

A W3C standard to describe behavior should support all paradigms, which can be (roughly) divided in four categories.

- Process-oriented
  - State-transition machines
  - Business process modeling
- Numeric
  - Transfer functions (e.g. PID)
- Rule-based & knowledge-based
  - Horn logic (rules)
  - Belief-desire-intention model
  - Condition-action rules
- Statistical
  - Bayesian inference
  - Neural networks

Behavior Description Languages

A W3C standard to describe behavior should support all paradigms, which can be (roughly) divided in four categories.

- Process-oriented
  - State-transition machines
  - Business process modeling
- Numeric
  - Transfer functions (e.g. PID)
- Rule-based & knowledge-based
  - Horn logic (rules)
  - Belief-desire-intention model
  - Condition-action rules
- Statistical
  - Bayesian inference
  - Neural networks

Behavior Description Languages

A W3C standard to describe behavior should support all paradigms, which can be (roughly) divided in four categories.

- Process-oriented
  - State-transition machines
  - Business process modeling
- Numeric
  - Transfer functions (e.g. PID)
- Rule-based & knowledge-based
  - Horn logic (rules)
  - Belief-desire-intention model
  - Condition-action rules
- Statistical
  - Bayesian inference
  - Neural networks

Behavior Description Languages

A W3C standard to describe behavior should support all paradigms, which can be (roughly) divided in four categories.

- Process-oriented
  - State-transition machines
  - Business process modeling
- Numeric
  - Transfer functions (e.g. PID)
- Rule-based & knowledge-based
  - Horn logic (rules)
  - Belief-desire-intention model
  - Condition-action rules
- Statistical
  - Bayesian inference
  - Neural networks

Behavior Description Languages

A W3C standard to describe behavior should support all paradigms, which can be (roughly) divided in four categories.

- Process-oriented
  - State-transition machines
  - Business process modeling
- Numeric
  - Transfer functions (e.g. PID)
- Rule-based & knowledge-based
  - Horn logic (rules)
  - Belief-desire-intention model
  - Condition-action rules
- Statistical
  - Bayesian inference
  - Neural networks

Behavior Description Languages

A W3C standard to describe behavior should support all paradigms, which can be (roughly) divided in four categories.

- Process-oriented
  - State-transition machines
  - Business process modeling
- Numeric
  - Transfer functions (e.g. PID)
- Rule-based & knowledge-based
  - Horn logic (rules)
  - Belief-desire-intention model
  - Condition-action rules
- Statistical
  - Bayesian inference
  - Neural networks

Risks & Opportunities

- Do not reproduce the failure of Web service descriptions (OWL-S, WSMO)
  → Focus on **usability** and ECMAScript
- Do not compete with Node-RED
  → Complement it*

(*) Nodes are identified by plain strings in Node-RED (`heater-controller`, `ramp-thermostat`, etc.). TD documents and scripts have full IRI addressing.

Risks & Opportunities

- Do not reproduce the failure of Web service descriptions (OWL-S, WSMO)
  - → Focus on **usability** and ECMAScript
- Do not compete with Node-RED
  - → Complement it*

(*) Nodes are identified by plain strings in Node-RED (`heater-controller`, `ramp-thermostat`, etc.). TD documents and scripts have full IRI addressing.

Risks & Opportunities

- Do not reproduce the failure of Web service descriptions (OWL-S, WSMO)
  - → Focus on **usability** and ECMAScript
- Do not compete with Node-RED
  - → Complement it*

(*) Nodes are identified by plain strings in Node-RED (`heater-controller`, `ramp-thermostat`, etc.). TD documents and scripts have full IRI addressing.

Risks & Opportunities

- Do not reproduce the failure of Web service descriptions (OWL-S, WSMO)
  - → Focus on **usability** and ECMAScript
- Do not compete with Node-RED
  - → Complement it*

(*) Nodes are identified by plain strings in Node-RED (`heater-controller`, `ramp-thermostat`, etc.). TD documents and scripts have full IRI addressing.

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Roadmap & Conclusion

Conclusion

The current proposal is to include a task force on exchanging and packaging WoT Scripts in a potential working group for WoT.

Mission

- Integrate JSON-LD frames in the WoT scripting API
  - For input requirements
  - For contextualization
- Focus on ECMAScript

Final Word

“Any application that can be written in JavaScript, will eventually be written in JavaScript”
— Jeff Atwood (paraphrasing Tim Berners-Lee)

Conclusion

The current proposal is to include a task force on exchanging and packaging WoT Scripts in a potential working group for WoT.

Mission

- Integrate JSON-LD frames in the WoT scripting API
  - For input requirements
  - For contextualization
- Focus on ECMAScript

Final Word

“Any application that can be written in JavaScript, will eventually be written in JavaScript”
— Jeff Atwood (paraphrasing Tim Berners-Lee)

Conclusion

The current proposal is to include a task force on exchanging and packaging WoT Scripts in a potential working group for WoT.

Mission

- Integrate JSON-LD frames in the WoT scripting API
  - For input requirements
  - For contextualization
- Focus on ECMAScript

Final Word

“Any application that can be written in JavaScript, will eventually be written in JavaScript”
— Jeff Atwood (paraphrasing Tim Berners-Lee)

Thanks for listening.

Any questions?