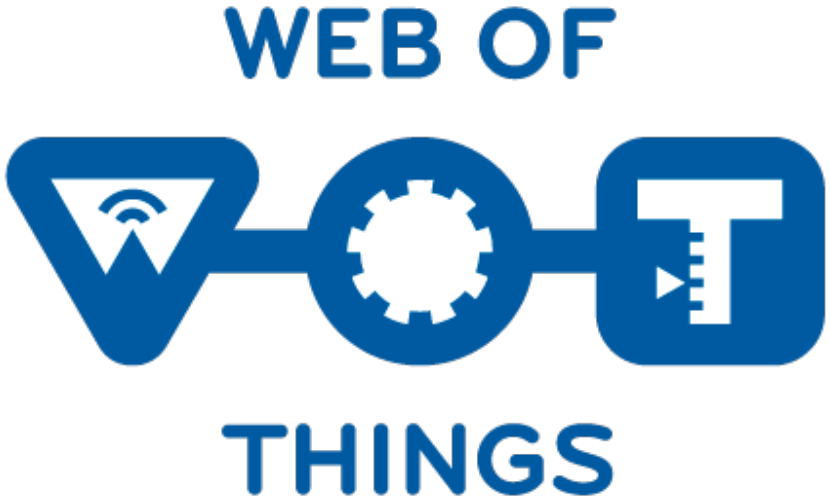


Plugfest Results

Michael McCool

25 March 2021

Summary

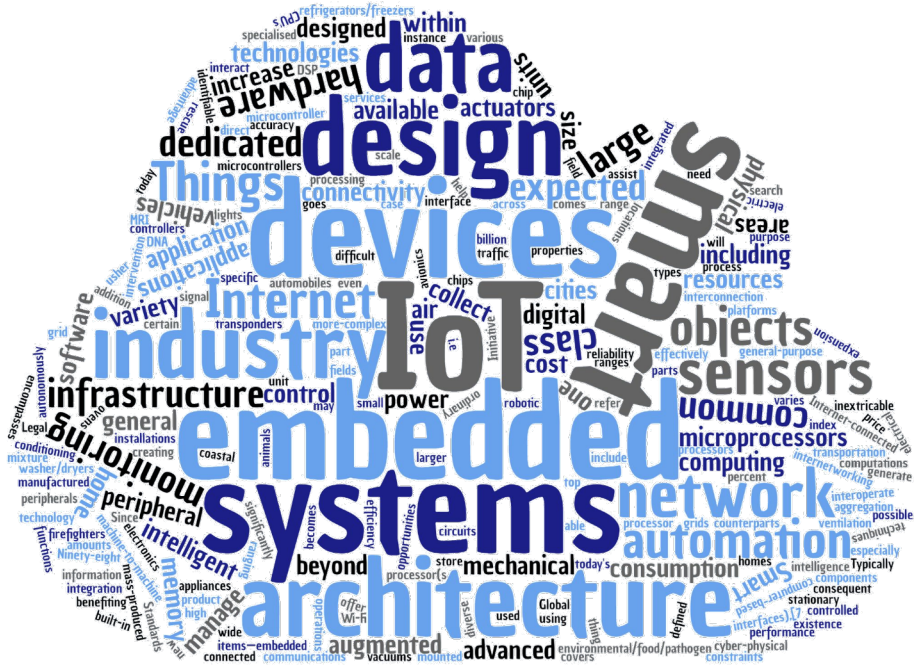
- ediTDor
- WADE + MAGE
- WoT testbench
- RIoT OS
- Node-RED Directory Integration
- OneDM/SDF Examples
- Creating TMs from SDF
- Geolocation Examples (Map app demo update is WIP)

W-ADE and A-MaGe

Fady Salama, Ege Korkan
fady.salama@tum.de, ege.korkan@tum.de

Embedded Systems and Internet of Things

Department of Electrical and Computer Engineering



W-ADE

- Development Environment:
  1. Modify TDs
  2. Invoke Interactions
  3. Test performance
  4. Spin up a Virtual Thing based on TD

- DEMO using Plugfest TDs

A-MaGe

- Atomic Mashup Generator:
  1. Choose, which Things participate
  2. Define Rules and Constraints for Generation
  3. View generated Mashups
  4. Generate System Description
  5. Generate Code!

- DEMO using Plugfest TDs

Findings

- Tested WADE and A-MaGe using:
  - Coap light (Philip Blum)
  - Coffee Machine (Siemens)
  - Industrial Plant (Oracle)
  - Led (Oracle)
  - Camera (Intel)
  - Wot Farm

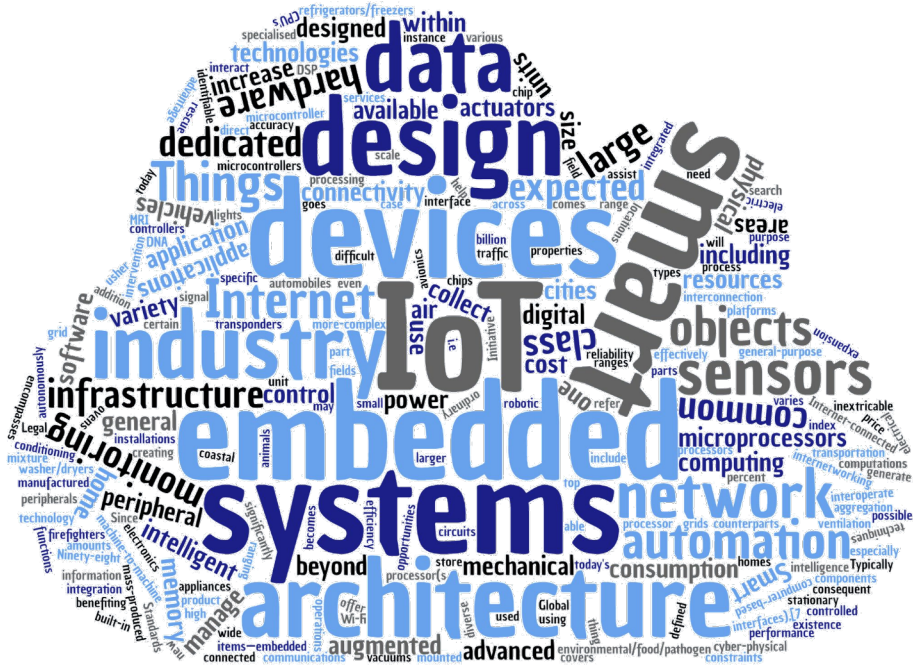
- Findings:
  - Fetching TDs using coap works now.
  - Fetching status is now visible in the status bar
  - Coap light not specifying readOnly or type of property -> W-ADE does not know how to

W3C WoT Plugfest - Testbench

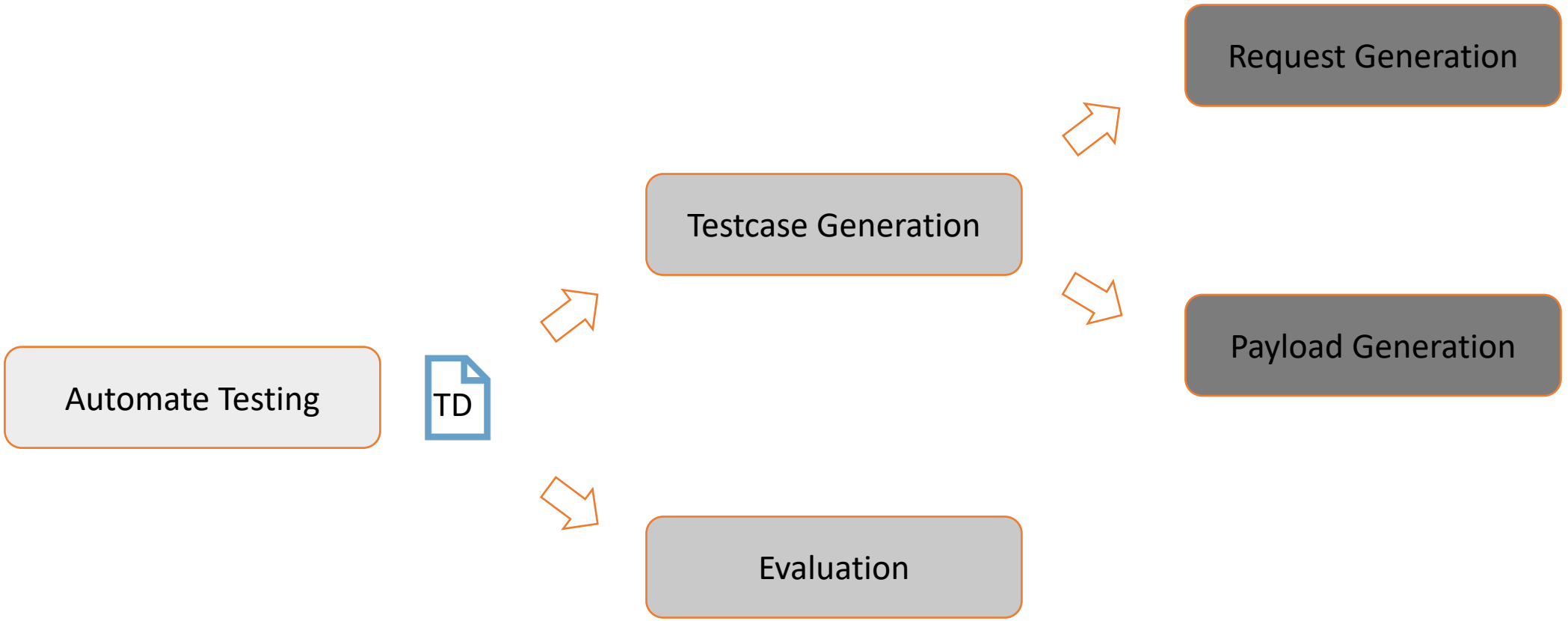
by Marcus Wolf Schmidt
& Ege Korkan

March 5th, 2021

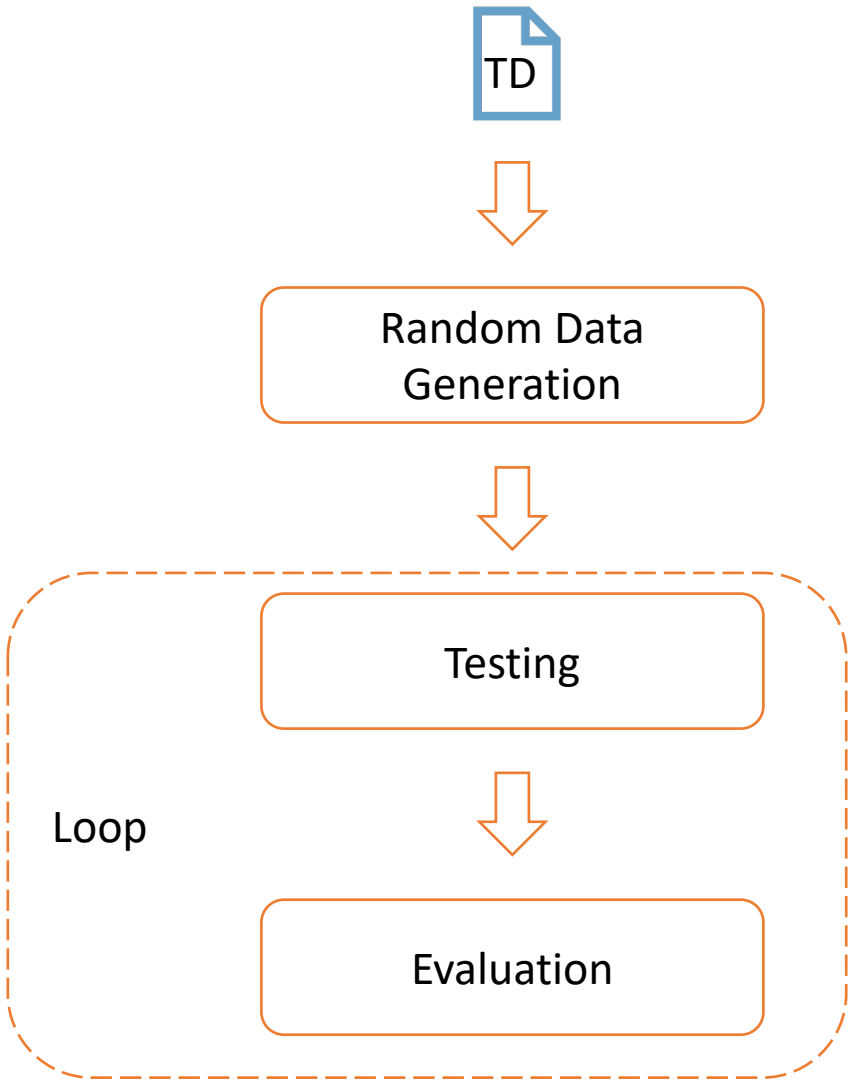
Associate Professorship of Embedded Systems and Internet of Things

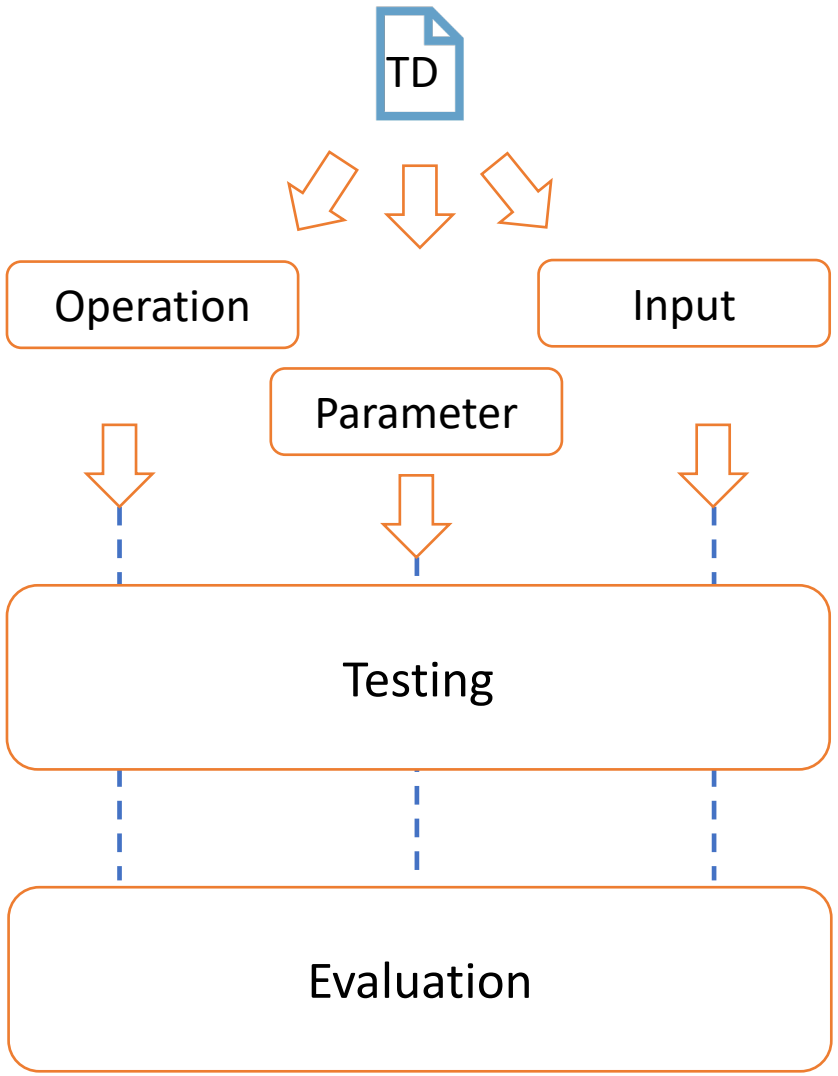


The Testbench: Main Goals



Old vs New Testbench





Test Report

T1 : Operation Coverage

T2: Parameter Coverage

T3: Input Coverage

T4: (Output Coverage)


```
"T1": [
    {
        "name": "bool",
        "passed": true,
        "time": "16:34:17",
        "writeInteraction": {
            "payload": false,
            "result": "OP level writeProperty Success"
        }
    },
    {
        "name": "bool",
        "passed": true,
        "time": "16:34:17",
        "readInteraction": {
            "data": false,
            "result": "OP level readProperty Success"
        }
    },
    {
        "name": "int",
        "passed": true,
        "time": "16:34:17",
        "writeInteraction": {
            "payload": -84478460,
            "result": "OP level writeProperty Success"
        }
    },
```

Things I have interacted with

Things	Reachable	Tested
RIOT OS	True	True
Siemens	True	True
Oracle	Partially True	Partially True
Intel	False	False
Hitachi	False	False
Farmsimulation	True	True
TUM	True	True

Results

Things	Operation (T1)	Parameter (T2)	Input (T3)	Output (T4) (not done yet)	Points of Interest
RIOT OS	Passed	No Parameters	No Inputs	Passed	Stress Testing
Siemens	Passed	Passed	Passed	Passed	Uri Variables
Oracle	Passed	Passed	Passed	Failed	Output Definition
Farmsimulation	Passed	Passed	Passed	Passed	Servient Shutdown
TUM	Passed	Passed	Partially	Passed	Sensehat Library

WoT-Testbench

https://github.com/tum-esi/testbench

OneDM@WoT Plugfest

Summary

Michael J. Koster

Goals

- Explore SDF => TM/TD Generation
- Define how SDF semantic annotation works
- Test the Modbus protocol binding

SDF to TM Conversion

- Inserted WoT DataSchema and Forms construct into SDF as extensions
- Created an sdfThing that customizes its sdfObject components and inserts the WoT extensions
- Generated a TD manually from this file:

https://github.com/mjkoster/onefb-sdk/blob/main/control-models/modbus/modbus-rtu.td.jsonld

- TBD Create a set of transformation rules/handlers
- Extend the conversion tool and framework
  - Read Input => Invoke Transforms => Write Output

Semantic Annotation using SDF

- SDF URIs have json-pointer fragments
- application/sdf+json contentType needs to be indicated where SDF is referenced

```
"@type": {
  "https://onedm.org/playground/#/sdfObject/Voltage",
  "contentType": "application/sdf+json"
}
```

- Semantic annotation for sdfObjects using "links" construct in the TD, since there is no TD Object class
  - We need a link relation like "implements" to describe this pattern

TD Vocabulary – value scaling

- We need to represent the data pattern which encodes a fixed point decimal number using an integer
- Also other similar use cases where a liner re-scaling is needed
- scaleMinimum and scaleMaximum are proposed for SDF
- we could also consider unitMinimum and unitMaximum to make it clear that we are scale converting the raw number to the defined units

Value Scaling (contd)

```
{
  "type": "number",
  "unit": "V",
  "minimum": 0,
  "maximum": 1200,
  "multipleOf": 1,
  "scaleMinimum": 0,
  "scaleMaximum": 120000
}
```

- The JSON validator works on the raw values and the application applies a linear scale interpolation
- This is how the node-red value scaling node works
- We also need other types of scale

Modbus binding

- Hosted 2 modbus devices for the test case and plugfest
- Modbus vocabulary suggestions
  - use address and quantity instead of offset and length
  - review "entity" concept to collapse read/write/obs
- Modbus URI construction feedback
- Modbus requires read multiple entities
  - Should we use array property or TD readmultiple?
  - Entity blocks need to be defined separately in the TD
  - Entity size needs to be defined in a dataschema extension
  - How do we semantically identify the individual entities?

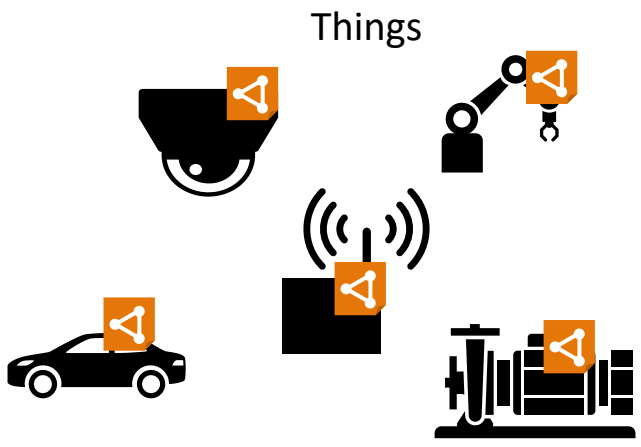
Node-RED – WoT-Discovery Integration

Kunihiko Toumura

2021.3.5

Overview

- In last year plugfest, we developed an Auto-population server which uses Linksmart Thing Description Directory (TDD) and DNS-SD/mDNS.
- On this plugfest, we integrate query interface and node generator into Node-RED, so that the flow developer can issue a query to TDD, and install a corresponding node on-the-fly.



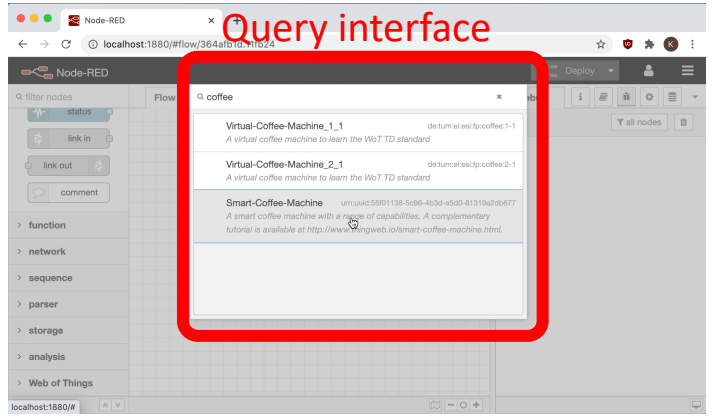
Sync script
Collects
TDs
(once an hour)

Thing
Description
Directory

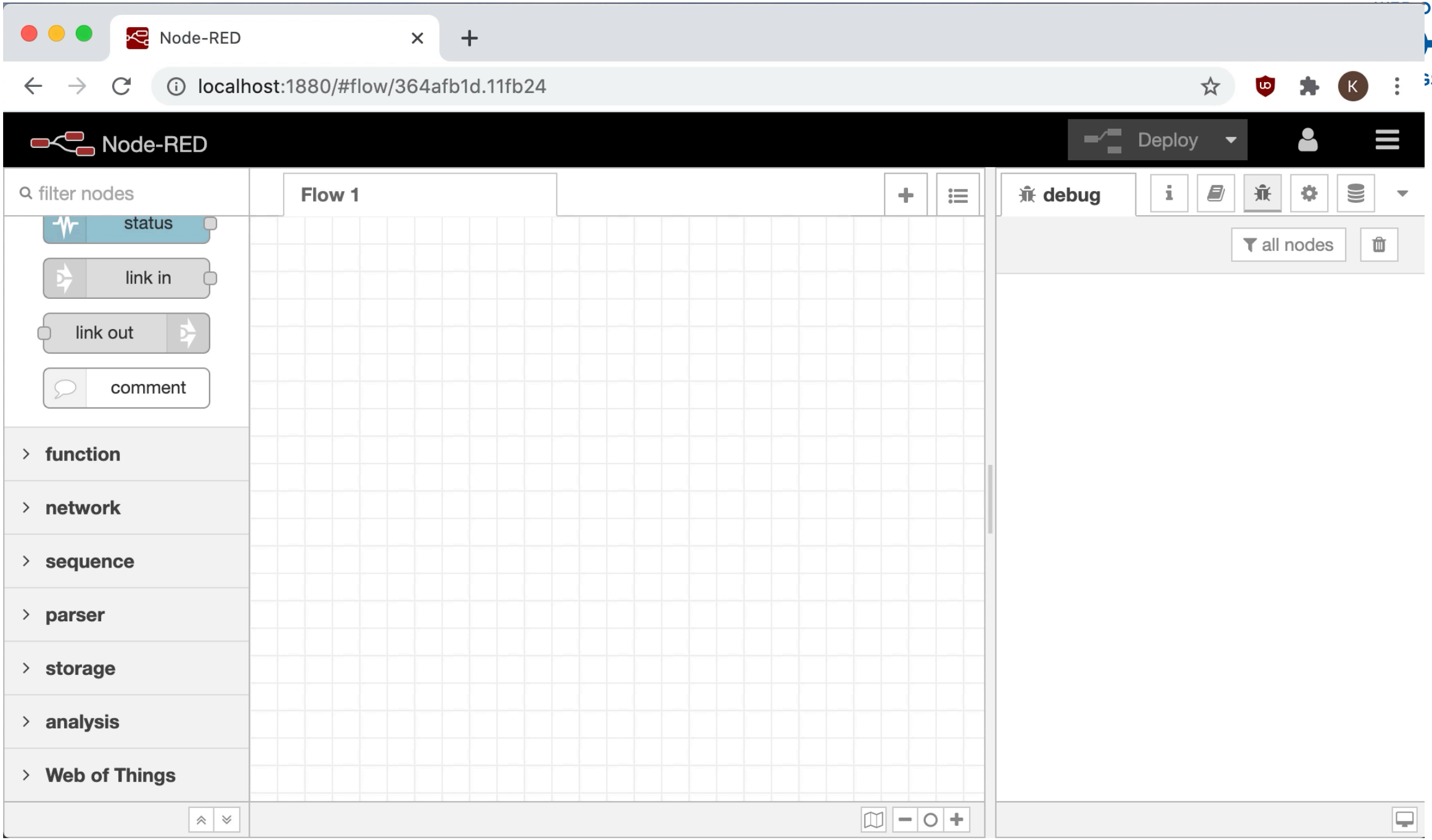
Retrieve
TDs
using
JSONpath
Query

Node-RED

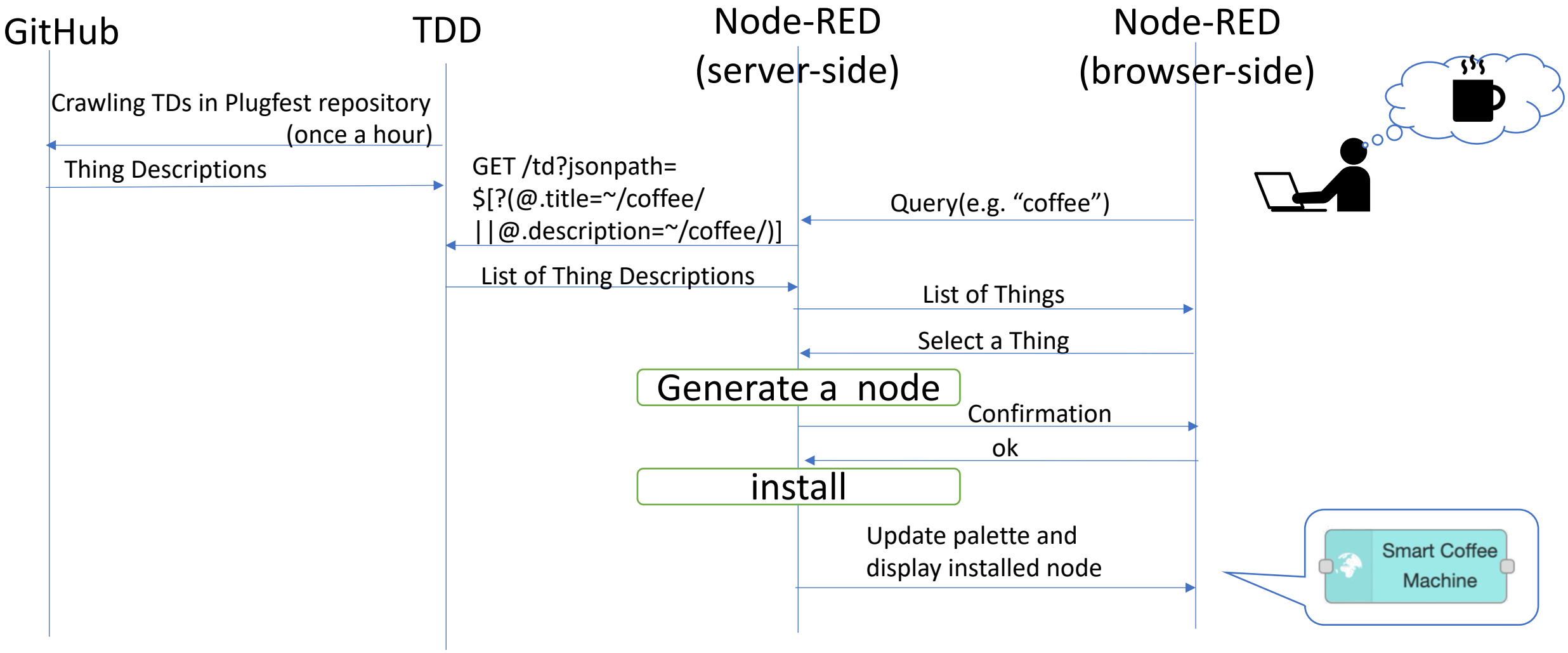
Integrated
Query interface



Generating
Node from TD
(Node Generator)


Node-RED
localhost:1880/#flow/364afb1d.11fb24
Node-RED
Deploy
filter nodes
status
link in
link out
comment
function
network
sequence
parser
storage
analysis
Web of Things
Flow 1
debug
all nodes

Appendix: Sequence diagram



Creating TMs from SDF

- **Generic feedback:**
- defining TMs for device classes was quite easy and straightforward
- almost all new TM features could be tested that are provided in the latest TD draft
- the edi{TD}or tool with its new feature was quite helpful for creating such models (also see #115) --> new TM JSON Schema was used
- reusing existing Model

Topics for discussions:

•a major problem is the current usage of the **required** term at interaction level: First of all it is confusing mainly for TD/TM processor that the **required** term is showing up at interaction level. Validator showing an error since it is not a normal interaction definition.

•as alternative, we should evaluate the **required** approach from SDF with JSON Pointer

•another topic should be clarified, if **required** can be used to point to an extended TM interaction. This may be helpful when TD are generated (a generator knows, what kind of interactions shall be adopted from the extended TM)

•beside of the **extend** feature we should introduce an import mechanism to **import** peaces of other TM definitions --> topic for vF2F

•we have no media type definition for TM --> question is this needed or shall we use application/td+json or application/jsonld

•we need also clarification about the a common file