Orchestration of Network-accessible Components with Condition-Action Rules

Andreas Harth
Joint work with Tobias Käfer

INSTITUTE AIFB, CHAIR FOR WEB SCIENCE

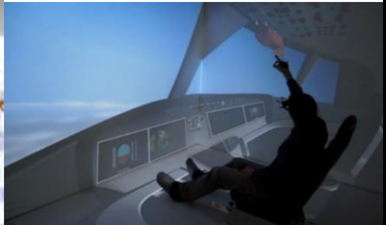
Motivation: System Interoperation

- Many scenarios require the combination of data and functionality from different components
- Integration architectures include wrappers (aka administration shells, lifting/lowering) to provide access to components via common interface
- Applications are built in programming language such as Python or JavaScript

- Standardising interfaces to components reduces cost
- Specifying applications in a high-level abstraction increases flexiblity

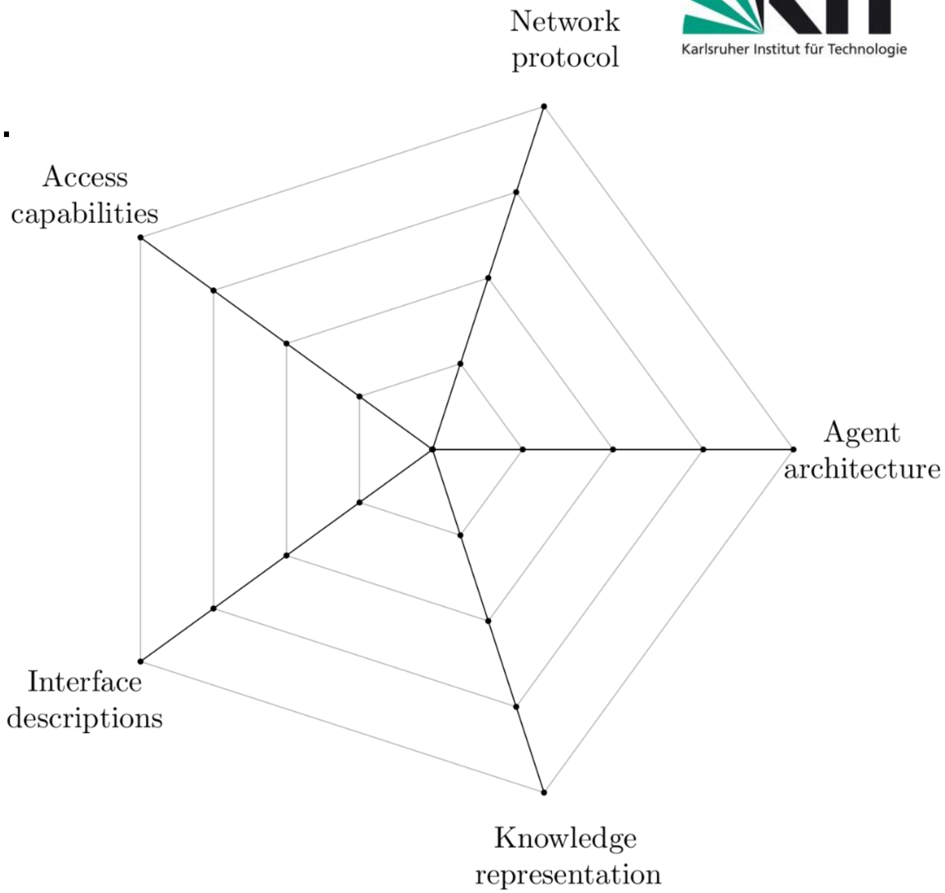
http://ivision-project.eu/

http://arvida.de/



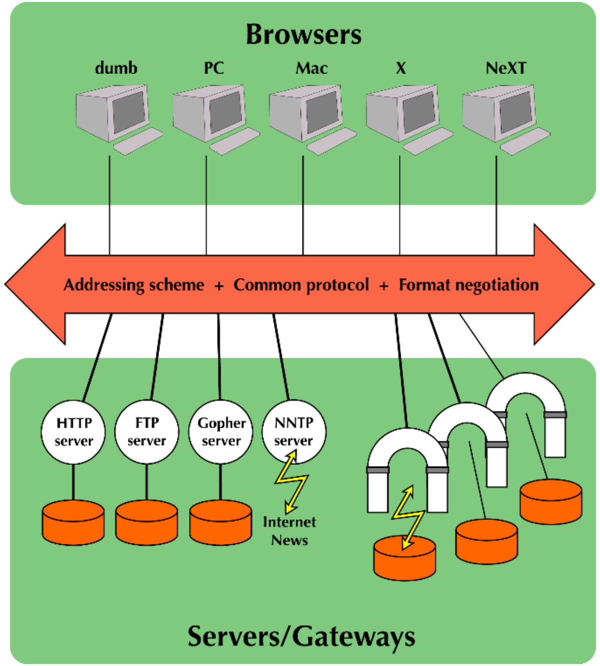
Many Choices…

- Network protocol: resource-based vs. event-based vs. publish-subscribe
- Access capabilities: get/set state vs. query
- Interface descriptions: none vs. input/output/precondition/effect
- Knowledge representation: XML/JSON vs. OWL 2 profiles
- Agent architecture: simple reflex agents vs. goal-based learning agents



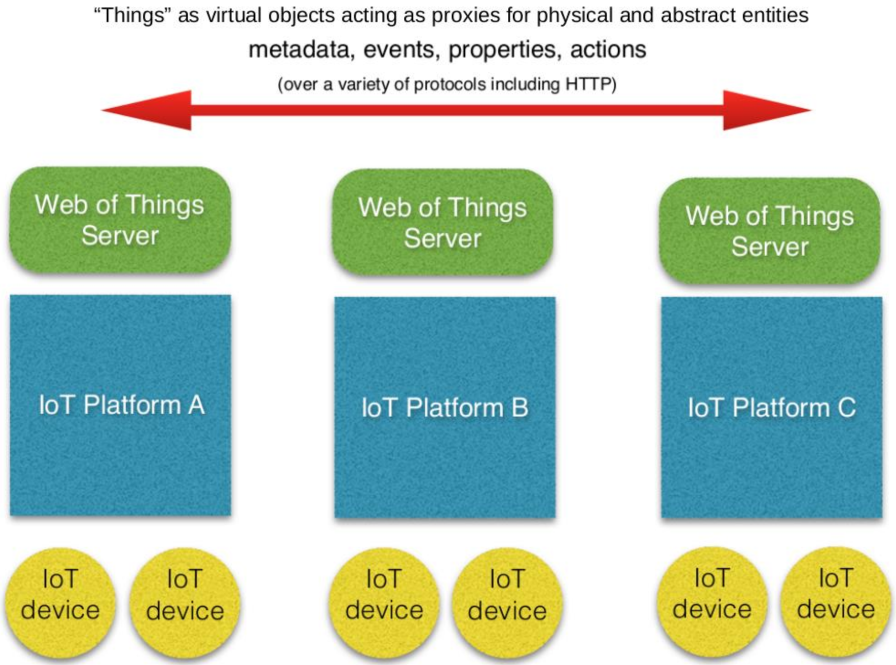
Uniform Interfaces (Web Architecture)

1990



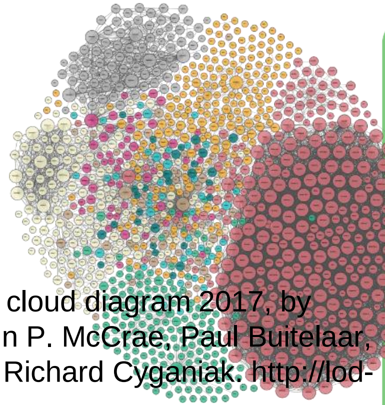
[Berners-Lee 1996]

2015

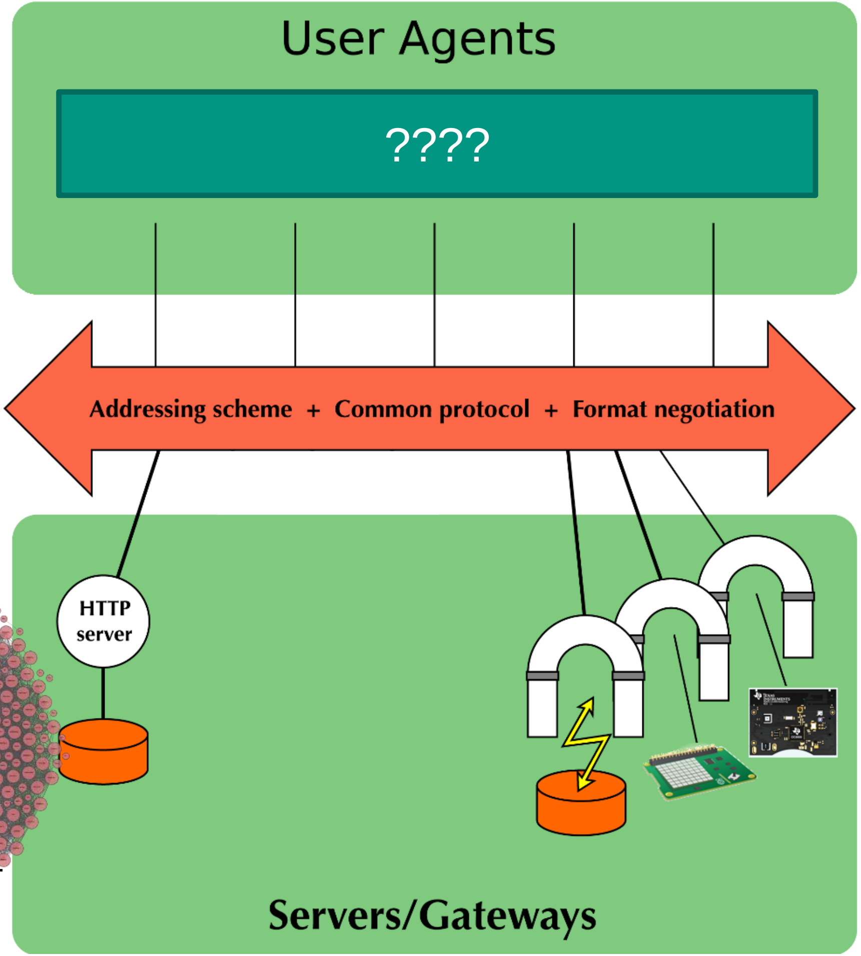


[Raggett 2015]

Web of Data/ Web of Things Architecture



Linking Open Data cloud diagram 2017, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/

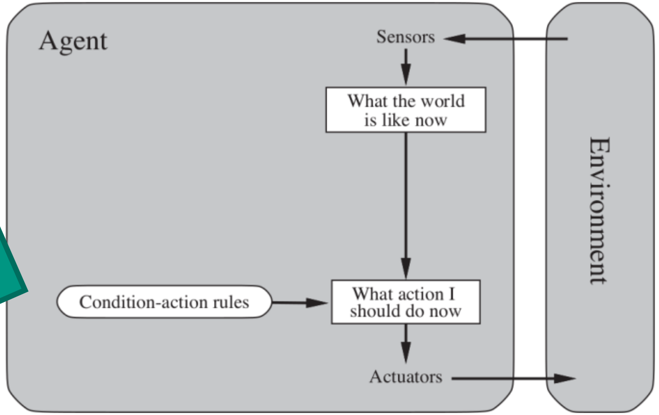


Adapted from https://www.w3.org/DesignIssues/diagrams/history/Architecture_crop.png

Cognitive Architectures

- SOAR (initially: State, Operator, Apply, Result),
- ACT-R (Adaptive Control of Though – Rational)
- Goal: to create „intelligent agents“
- In the following, we only consider user agents that are
  - „simple reflex agents“ (Russel & Norvig, see figure),
  - aka „tropistic agents“ (Genesereth & Nilson)
- We explain how to use rules to control the agent‘s behaviour
- Sense: safe HTTP methods (GET)
- Act: unsafe HTTP methods

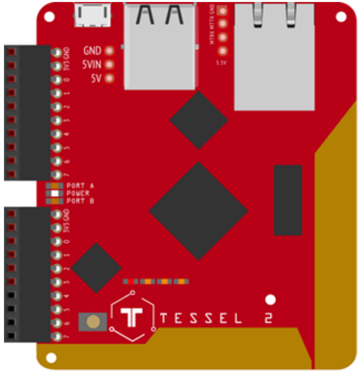
Russel and Norvig, Artificial Intelligence – A Modern Approach, Third Edition, 2010



Resource-based Interface to Devices

- Tessel 2, WiFi and Ethernet interfaces
- Read-Write Linked Data interface to four LEDs at https://github.com/kaefer3000/t2-rest-leds

Resource	URI
Index page	`http://t2-rest-leds.lan/`
The Tessel 2	`http://t2-rest-leds.lan/#tessel2`
Set of LEDs	`http://t2-rest-leds.lan/leds/#bar`
LED 0	`http://t2-rest-leds.lan/leds/0#led`
LED 1	`http://t2-rest-leds.lan/leds/1#led`
LED 2	`http://t2-rest-leds.lan/leds/2#led`
LED 3	`http://t2-rest-leds.lan/leds/3#led`



Device Described in RDF

- HTTP GET on http://t2-rest-leds.lan/ returns RDF with links to LEDs:

```
@prefix sosa:http://www.w3.org/ns/sosa/> .
<#tessel2> a sosa:Platform ;
           sosa:hosts <leds/#bar> .
```

- Querying with SPARQL (e.g. using roqet, # apt-get install rasqal-utils):

```
SELECT ?y
FROM <http://t2-rest-leds.lan/>
WHERE { ?x sosa:hosts ?y . }
```

?y
http://t2-rest-leds.lan/leds/#bar

Link to LEDs

- GET on http://t2-rest-leds.lan/leds/ returns links to individual LEDs:

```
@prefix sosa: <http://www.w3.org/ns/sosa/> .
<#bar> a sosa:Platform ;
       sosa:hosts <0#led> , <1#led> , <2#led> , <3#led> .
```

- SPARQL query:

```
SELECT ?x ?y
FROM <http://t2-rest-leds.lan/leds/0>
FROM <http://t2-rest-leds.lan/leds/1>
WHERE { ?x <https://w3id.org/saref#hasState> ?y . }
```

?x	?y
http://t2-rest-leds.lan/leds/0#led	saref:OFF
http://t2-rest-leds.lan/leds/1#led	saref:OFF

Getting and Setting the State of an LED

- HTTP GET on http://t2-rest-leds.lan/leds/0 returns the current state:

```
@prefix saref: <https://w3id.org/saref#> .
<#led> a saref:LightingDevice ;
         saref:hasState saref:Off .
```

- PUT on http://t2-rest-leds.lan/leds/0 with the following body sets the current state:

```
@prefix saref: <https://w3id.org/saref#> .
<#led> saref:hasState saref:On .
```

Hello World^H^H^H^H^H^H IoT: Blinking Light

```
@prefix http: <http://www.w3.org/2011/http#> .
@prefix httpm: <http://www.w3.org/2011/http-methods#> .
@prefix saref: <https://w3id.org/saref#> .

[] http:mthd httpm:GET ; http:requestURI <http://t2-rest-leds.lan/leds/0> .

{ <http://t2-rest-leds.lan/leds/0#led> saref:hasState saref:Off .
} => {
  [] http:mthd httpm:PUT ; http:requestURI <http://t2-rest-leds.lan/leds/0> ;
     http:body { <http://t2-rest-leds.lan/leds/0#led> saref:hasState saref:On . } .
} .

{ <http://t2-rest-leds.lan/leds/0#led> saref:hasState saref:On .
} => {
  [] http:mthd httpm:PUT ; http:requestURI <http://t2-rest-leds.lan/leds/0> ;
     http:body { <http://t2-rest-leds.lan/leds/0#led> saref:hasState saref:Off . } .
} .
```

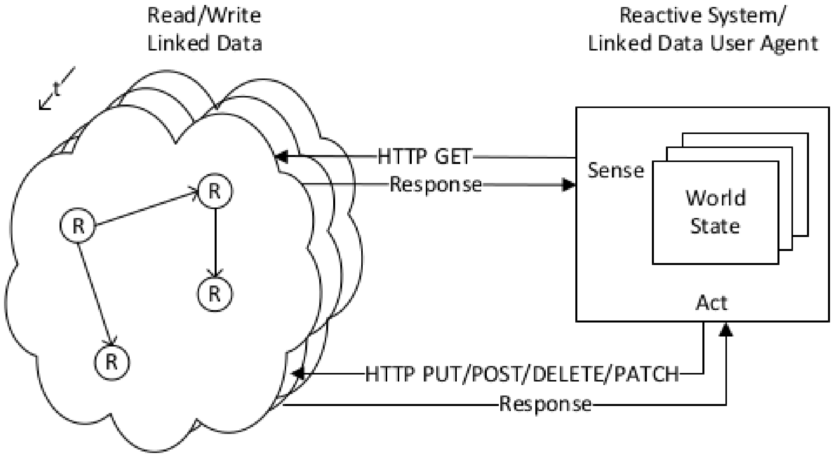
Link-Following to Access Resource State

```
@prefix http: <http://www.w3.org/2011/http#> .
@prefix httpm: <http://www.w3.org/2011/http-methods#> .
@prefix saref: <https://w3id.org/saref#> .

# GET index page
[] http:mthd httpm:GET ;
   http:requestURI <http://t2-rest-leds.lan/> .

# for each triple with predicate sosa:hosts
{
  ?x sosa:hosts ?y .
} => {
# do a GET on the object of the triple
  [] http:mthd httpm:GET ;
     http:requestURI ?y .
} .
```

Linked Data-Fu Architecture

- Approach for accessing, integrating, querying and manipulating resource state on the Web of Things
- The language allows developers to specify interactions using rules
- The engine executes desired interactions in parallel

- **Request rules** specify how and when to interact with resources, i.ie., retrieve the state of resources (sense) or manipulate the state of resources (act)
- **Derivation rules** support reasoning constructs, e.g., transitivity, reflexivity of properties



Conclusion

- We have designed and implemented a rule-base language to specify user agents on Read-Write Linked Data
- Rule-based programs encode a state machine that emits HTTP requests (user agents)
- The program operates in a sense-act cycle:
  - First acquire the state of resources via GET
  - Then decide which action to carry out
- Graph-structured data model (RDF), vocabulary description languages (RDFS) and ontology languages (OWL) provide basis for data modelling, integration and exchange
- Rules and state machines provide rigorous formal background for specifying and reasoning with application behaviour
- A (subset of) Semantic Web and Linked Data technologies provides a solid basis for specifying application behaviour
- We conduct research on advanced topics, for example pushing application behaviour to components

References

- Andreas Harth. “Orchestration of Network-accessible Components with Condition-Action Rules”. Position paper for the Workshop on IoT Semantic/Hypermedia Interoperability, July 2017, Prague, Czech Republic. http://harth.org/andreas/2017/wishi/paper.html
- Andreas Harth and Tobias Käfer. “Tutorial on Rule-based Processing of Dynamic Linked Data”. European Semantic Web Conference (ESWC 2017), May 28 — June 1, 2017 in Portorož, Slovenia. http://harth.org/andreas/2016/eswc-tut/
- Andreas Harth, Tobias Käfer. “Towards Specification and Execution of Linked Systems”. 28. GI-Workshop Grundlagen von Datenbanken, May 24 - 27, 2016, Nörten-Hardenberg, Germany. http://ceur-ws.org/Vol-1594/paper12.pdf
- Tessel2 REST+Linked Data interface: https://github.com/kaefer3000/t2-rest-leds
- Linked Data-Fu: https://linked-data-fu.github.io/