



Institute of Automation

# Implementing Semantic Agents with Node-RED

Introduction to and Application Of Node-RED and Semantic Web Technologies

- Node-RED as an exemplary low-code environment & edge-capable runtime environment

“20 years ago, enterprise systems were implemented by 100s of consultants and experts at a multi-million dollar cost, now with no code/low code, the same systems can be implemented by smaller players at a fraction of the price and in a fraction of the time.”

*Simon Chan, Founder and CEO, DigiVue Consulting*

“While I really appreciate hand coding for very specific things, for a lot of other things it just seems dumb and irrelevant work,”

*Jan Bolhuis, IT Regiemanager, Master Software.*

“Low-Code/No-Code will disrupt [the] entire pattern [of software development], as enterprises realize they can be even more successful with their digital transformations if they do away with hand-coding altogether, adopting Low-Code/No-Code across their organizations instead.”

*Jason Bloomberg, IT industry analyst, Forbes contributor*

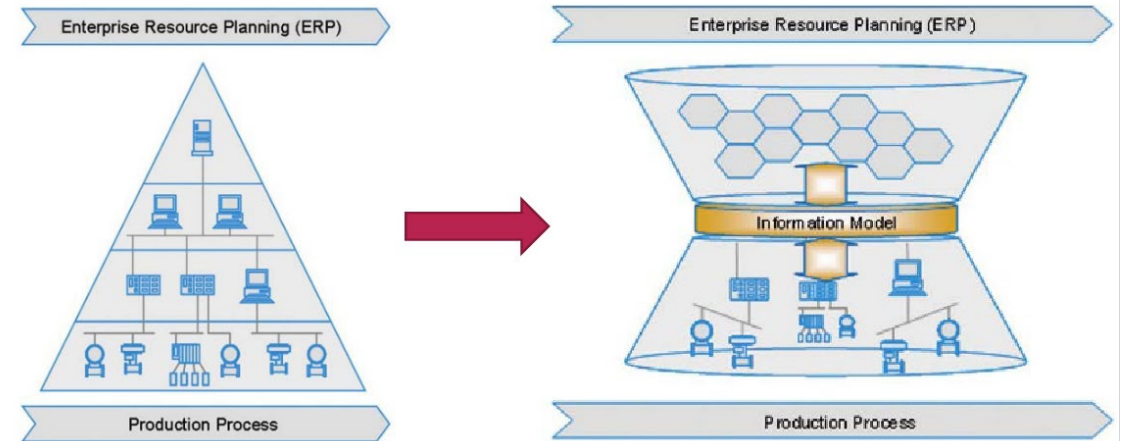
- Introduction to semantic web technologies

"A new form of web content that is meaningful to computers will unleash a revolution of new possibilities"  
*Tim Berners-Lee*



- Introduction to semantic web technologies

The classic automation pyramid provides a hierarchical structure, whose monolithic information technology connection generates many disjoint "data sources"



B. Vogel-Heuser, G. Kegel, K. Bender, K. Wucherer: Global Information Architecture for Industrial Automation. atp - edition (1), 2009, S. 108–115.

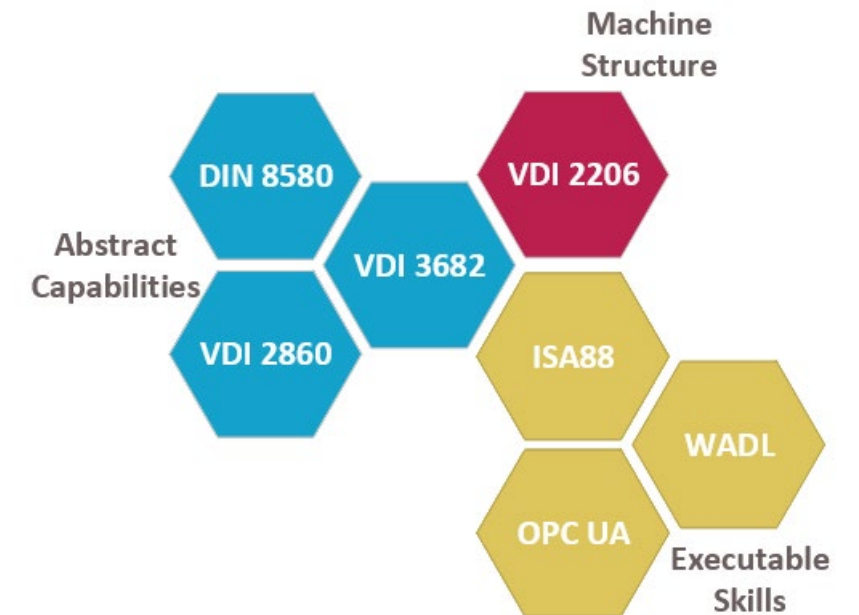
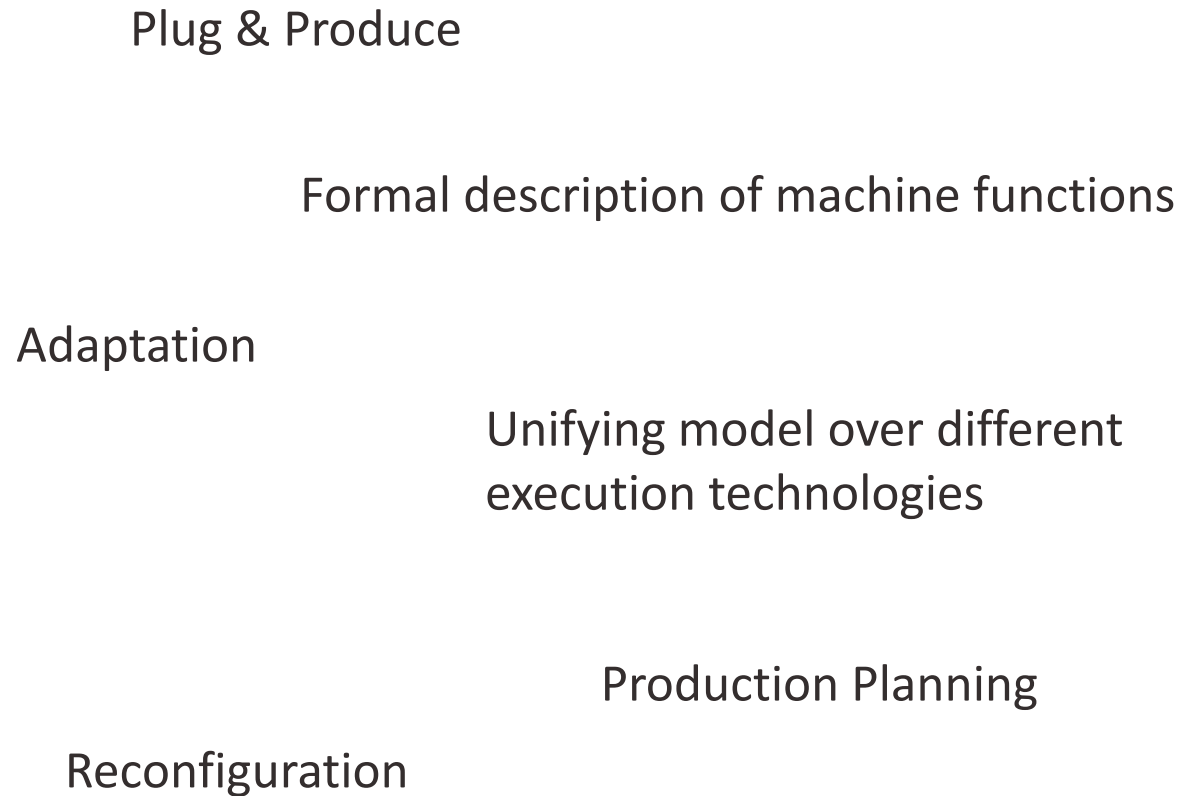


This is to change within the framework of Industry 4.0 by networking the "data sources". Networking encompasses both the entire life cycle of products, production facilities and orders as well as the connection of all necessary value creation processes"

*A. Fay et al.: Durchgängigkeit in Wertschöpfungsketten von Industrie 4.0. In: VDI (Hrsg.): AUTOMATION, 2016.*

<https://www.bitkom.org/Themen/Digitale-Transformation-Branchen/Industrie-40/Perspektive-der-Arbeit.html>

- Capability and Skill-based Engineering



Köcher, Aljosha, et al. "A formal capability and skill model for use in plug and produce scenarios." *2020 25th IEEE ETFA*. Vol. 1. IEEE, 2020

- Introduction & Scenario Description
- Implementation Part 1 (Basics + HTTP)
- Ontologies, SPARQL & Capabilities
- Implementation Part 2 (Products and Agent Registration)



## Session 1

## Coffee Break

- Thoughts on Negotiations & Semantics for Agents
- Implementation Part 3 (Message Processing)
- Implementation Part 4 (Bid Selection)



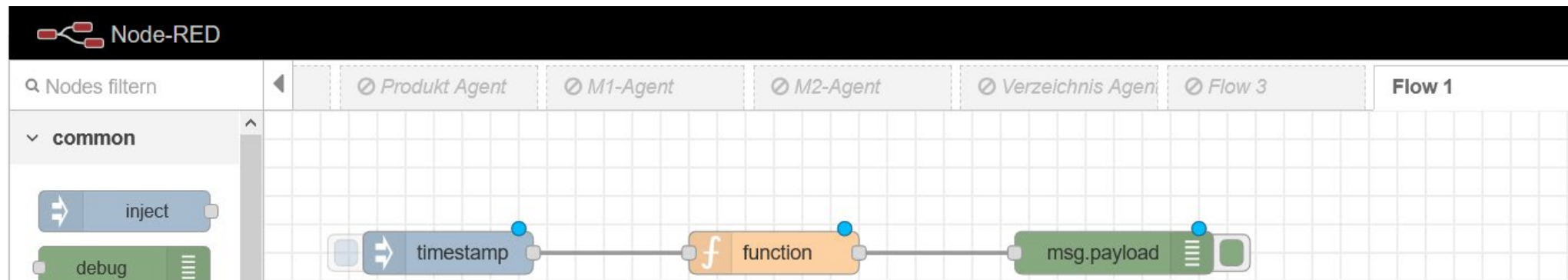
## Session 2

- We have designed this workshop as a crash course that you can follow along
- BUT there are quite a lot of requirements:
  - Agent basics
  - Node-RED basics + Node-RED installed
  - JavaScript basics
  - Semantic web basics + GraphDB installed
- In case you find yourself "crashed":
  - Feel free to ask questions
  - You can find everything on Github:  
<https://github.com/hsu-aut/ISSIA22-node-red-agents>
  - Checkout individual aspects again later



<https://twitter.com/carsparkedwrong>

- Start Node-RED via Command line (> node-red) and open <http://localhost:1880> in your browser



- Flow basics
  - Message-based
  - Nodes: Inject, function, delay
  - Deploy
  - Debugging: Node.warn() or debug messages

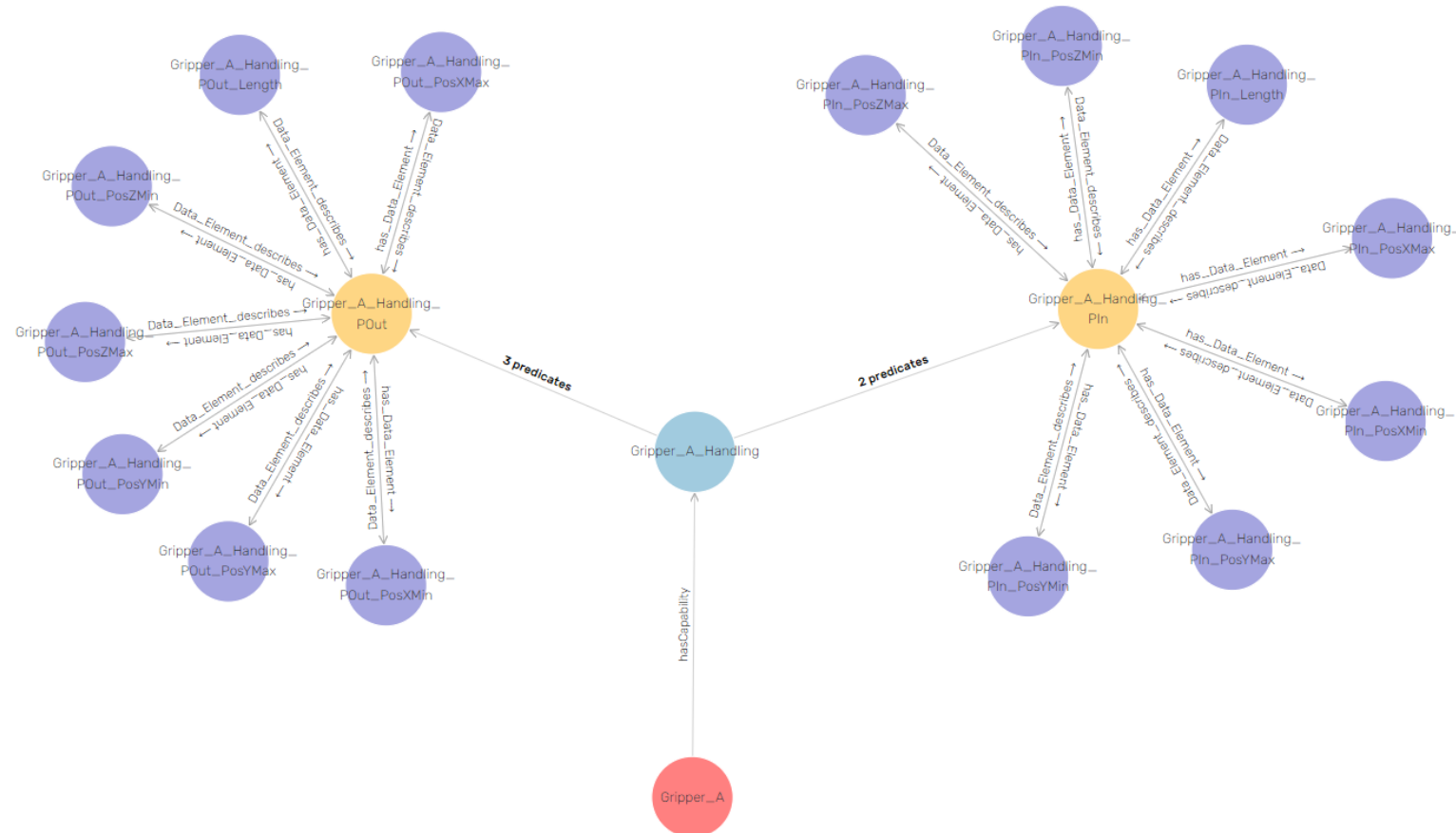


GraphDB
FREE

- Import
- Explore
- SPARQL**
- Monitor
- Setup
- Help

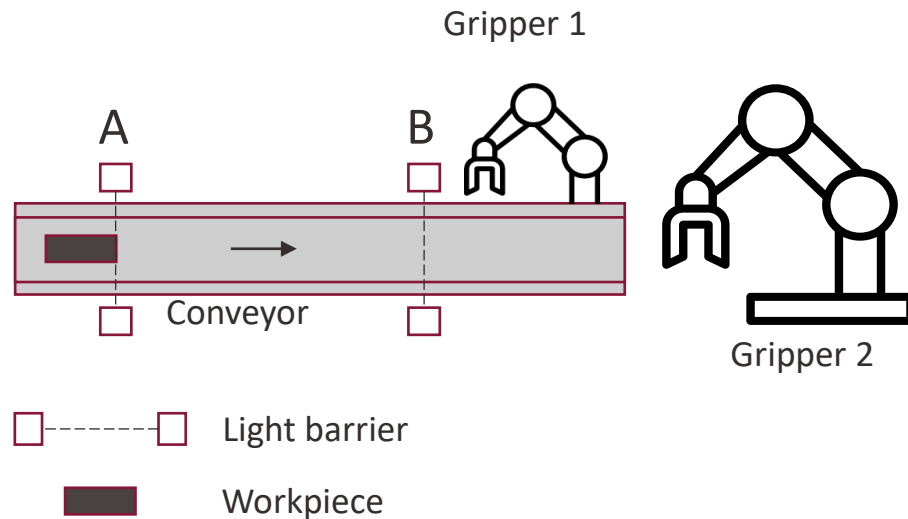
## Visual graph ?

AgentSummerSchool

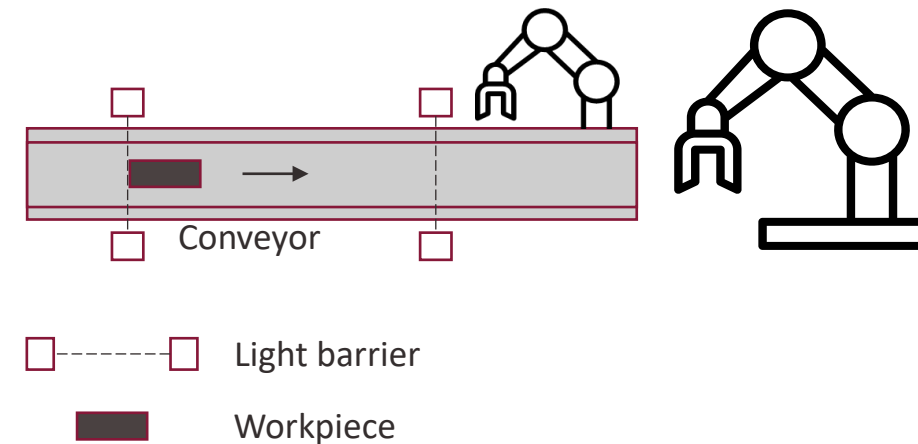


- Distributed Control Architecture → Reconfigurable and Robust

1 Workpiece presence is registered by sensor and ID is detected



2 Presence not detected anymore



## Short Digression

3

A asks Conveyor what distance was covered during detection time → Save id & length in GraphDB

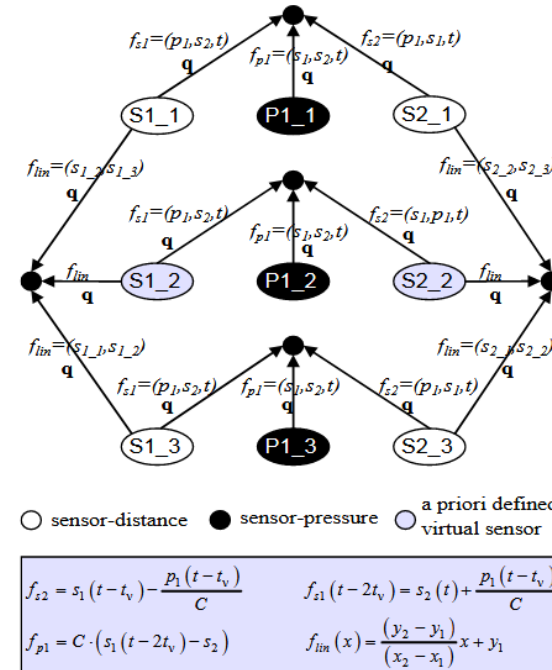
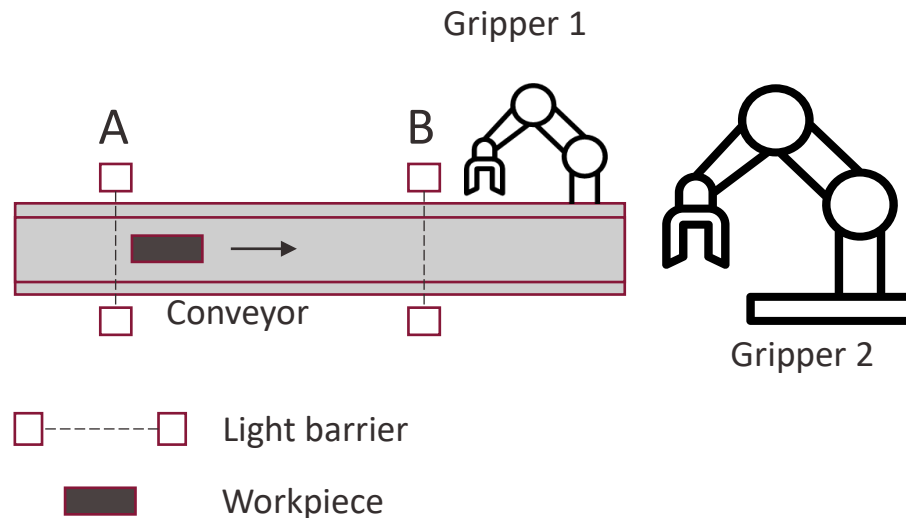


Fig. 4. Analytical dependencies of sensor values (P-pressure, S-distance) using the material property (C).

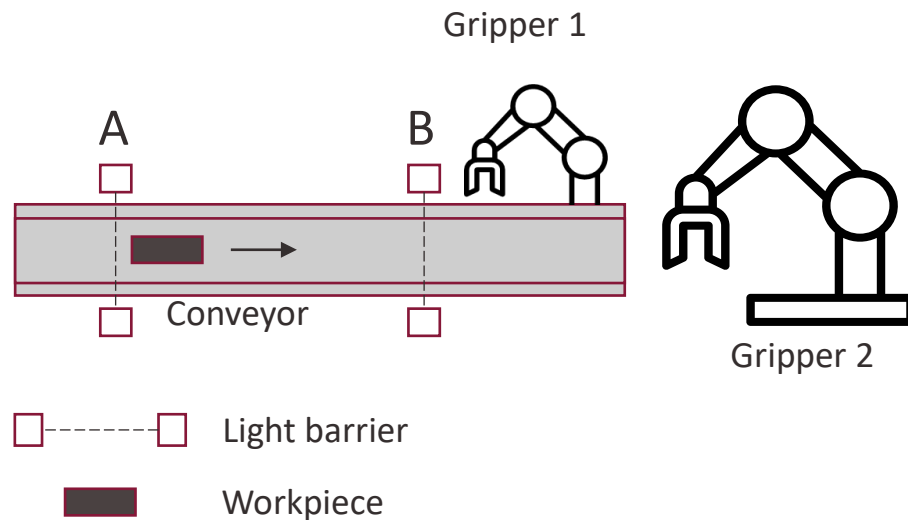
	S1,1	S1,2	S1,3	P1,1	P1,2	P1,3	S2,1	S2,2	S2,3
S1,1	x_s1,1	x_virt_s1,1 (1)		x_virt_s1,1 (2)			x_virt_s1,1 (3)		
S1,2	x_virt_s1,2 (1)	x_s1,2			x_virt_s1,2 (2)			x_virt_s1,2 (3)	
S1,3		x_virt_s1,3 (1)	x_s1,3			x_virt_s1,3 (2)			x_virt_s1,3 (3)
P1,1				x_p1,1			x_virt_p1,1 (1)		
P1,2					x_p1,2			x_virt_p1,2 (1)	
P1,3						x_p1,3			x_virt_p1,3 (1)
S2,1				x_virt_s2,1 (1)			x_s2,1	x_virt_s2,1 (2)	
S2,2					x_virt_s2,2 (1)			x_virt_s2,2 (2)	x_s2,2
S2,3						x_virt_s2,3 (1)		x_virt_s2,3 (2)	x_s2,3

Fig. 6. Matrix of as representation of the knowledge base

A. Wannagat, B. Vogel-Heuser: Increasing Flexibility and Availability of Manufacturing Systems - Dynamic Reconfiguration of Automation Software at Runtime on Sensor Faults, IFAC Proceedings, Volume 41, Issue 3, 2008, pp. 278-283,

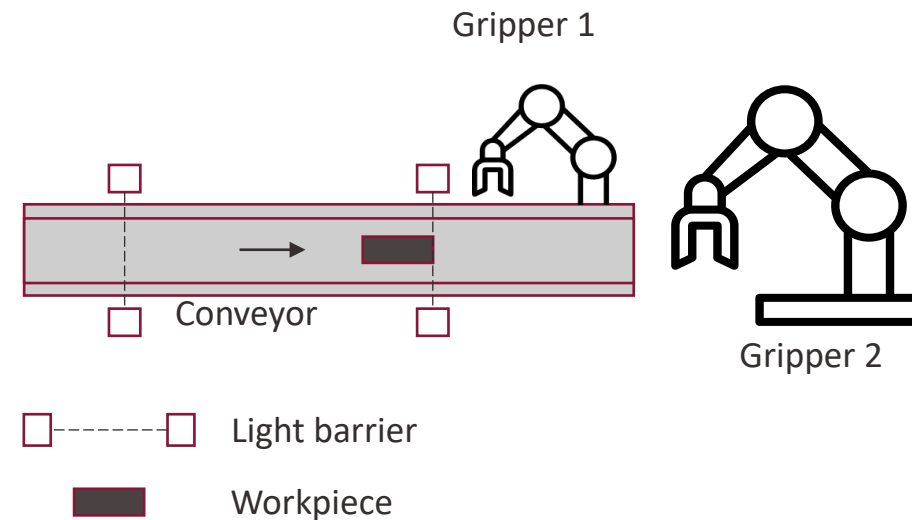
3

A asks Conveyor what distance was covered during detection time → Save id & length in GraphDB

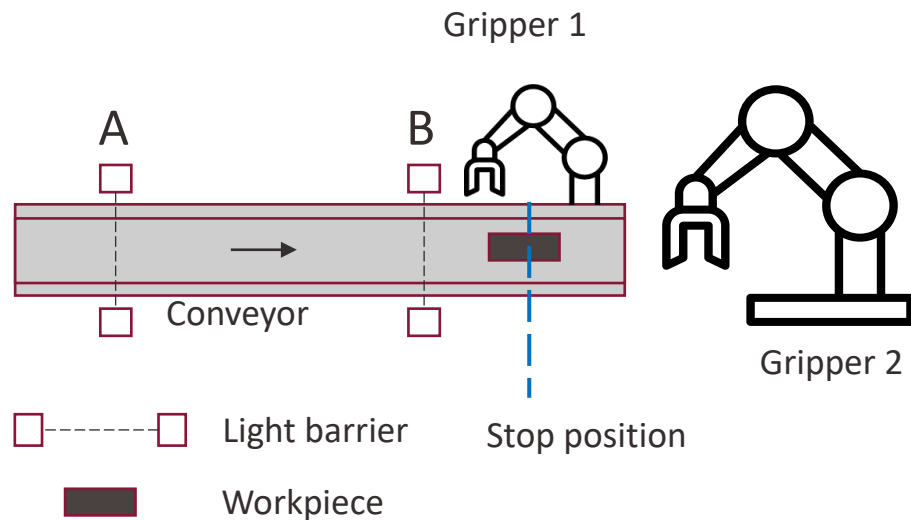


4

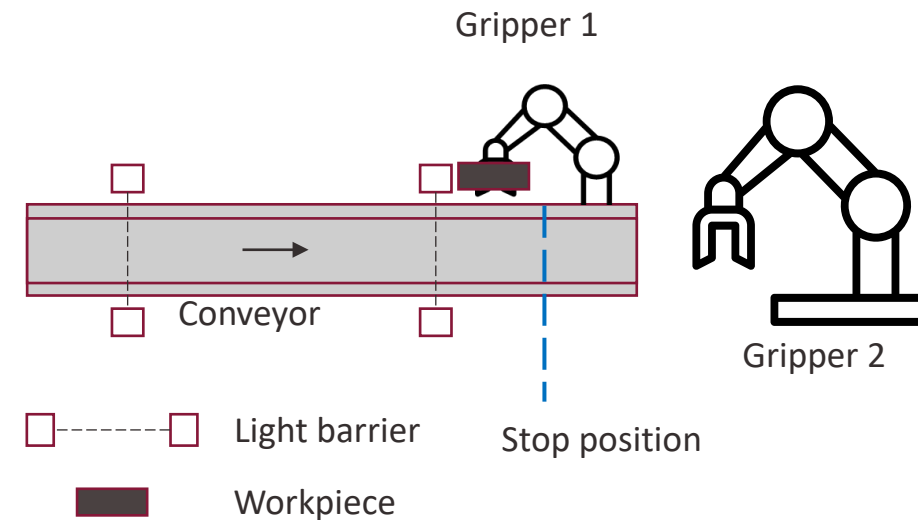
B detects ID and asks for length from GraphDB



- 5 B requests stop from Conveyor at the correct position (e.g. center of the workpiece)

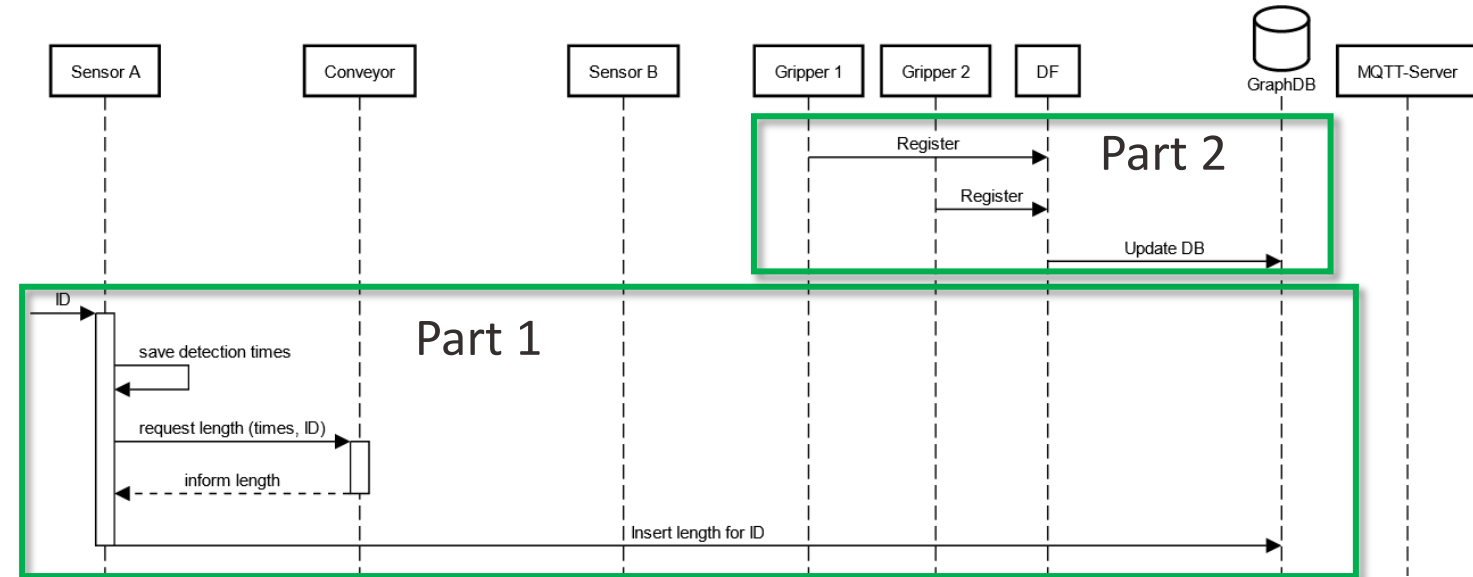


- 6 B selects appropriate Gripper, workpiece is picked up by Gripper

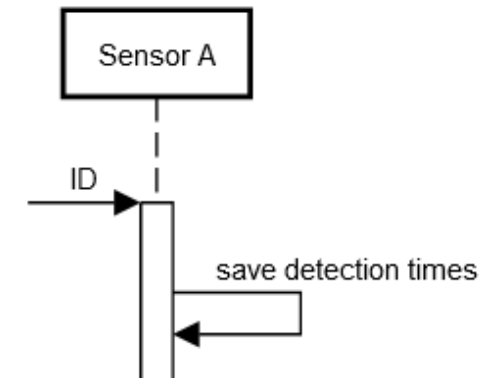


ID (Barcode) of workpiece is recognized. Delay = time of detection

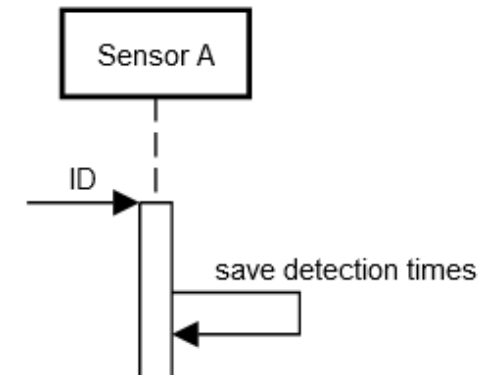
Conveyor-Agent knows speed of and controls conveyor. Using the time instances the conveyor agent can calculate the distance covered



- Import the template flow for agent "Sensor A"
- An Inject node needs to generate a message with...
  - ... an empty JSON object ( `{}` ) as its `msg.payload`
  - ... a `msg.delay` of `"500 + $random() * 500"` (as an expression)
    - This delay is used to simulate the time it takes the workpiece to pass through Sensor A
- Connect your Inject node with the given node "generate random ID and time"



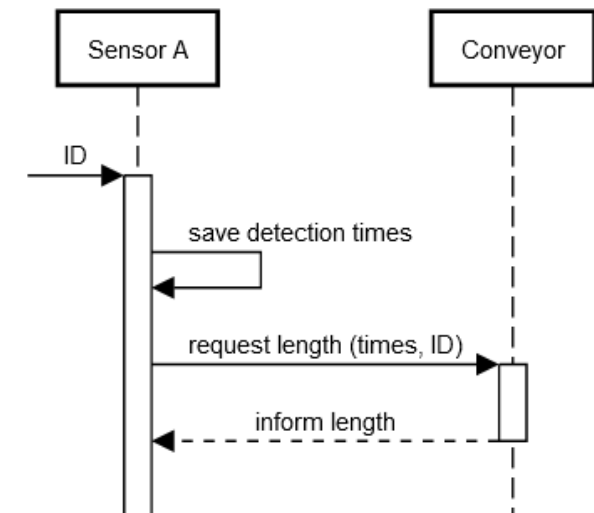
- A Delay node (between "generate random ID and time" and "time\_difference") is going to simulate the workpiece passing through the sensor
  - Add a Delay node
  - This node needs to use our randomly generated msg.delay (is defined in the inject node)
- Implement the function "time\_difference"
  - In this function, define a variable "sensorA\_timeEndDetection" which adds the delay to the initial detection time ("sensorA\_timeStartDetection")
  - Make sure to store your variable inside msg.payload
- Check the current msg.payload with a Debug node behind "time\_difference"





- Let's take a look at the node to get workpiece lengths from the conveyor...

- Communication via HTTP



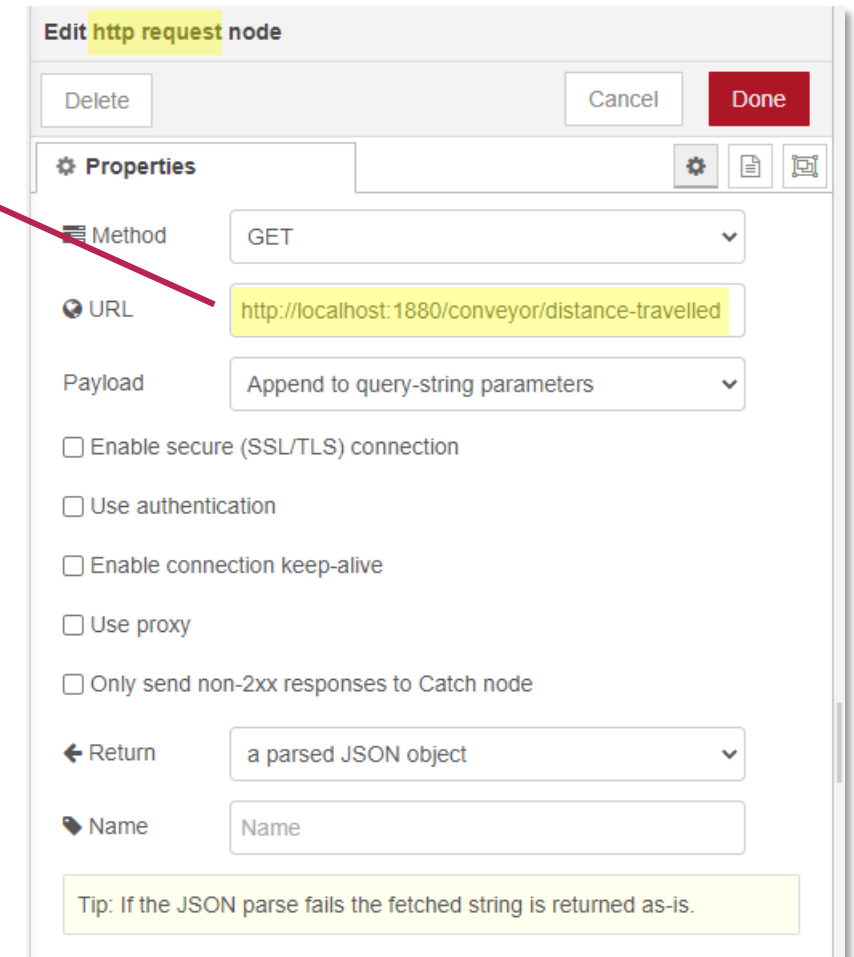
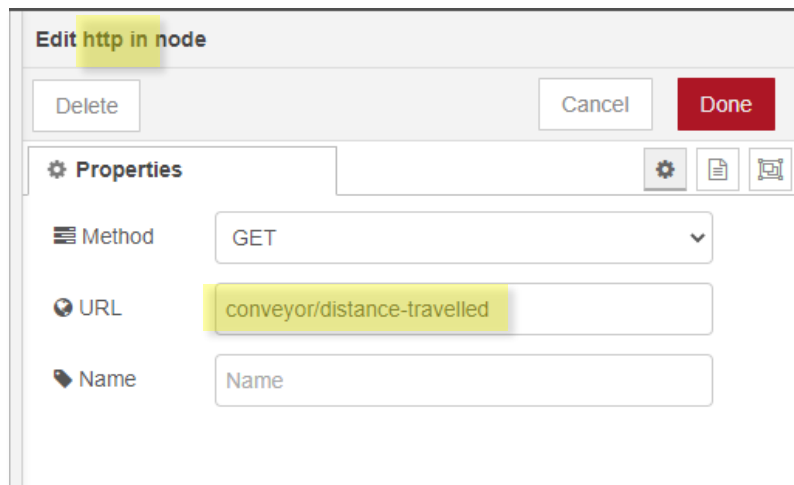
- HTTP-Nodes

- http-request: To request a web service

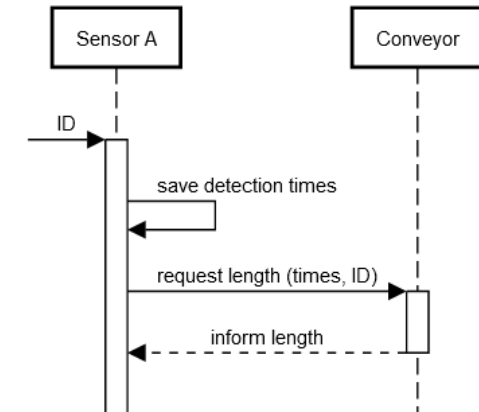
- HTTP Method and URL (here "localhost" + port + "service name") required
    - URL may be set dynamically: `msg.url = "http://localhost:1880/" + someVariable;`

- http-in: To provide a web service

- Method and URL (more precise: path relative to "localhost:1880") required
    - Important: A http-response node is required to send a response and end the request



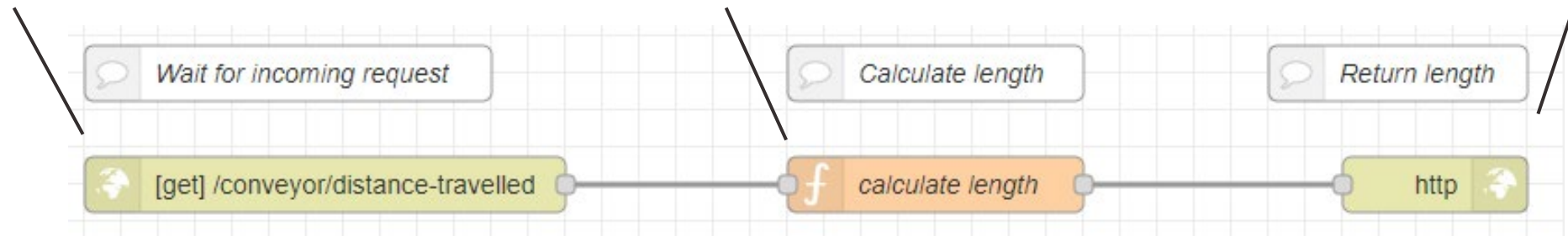
- Next steps:
  - How can we calculate the workpiece length?
  - Which information is required?
- Import the conveyor agent flow and extend it by implementing the function "calculate length via timestamps"



Provided HTTP endpoint at which a service is available

HTTP request triggers this function. We will insert our logic here

Return calculated information

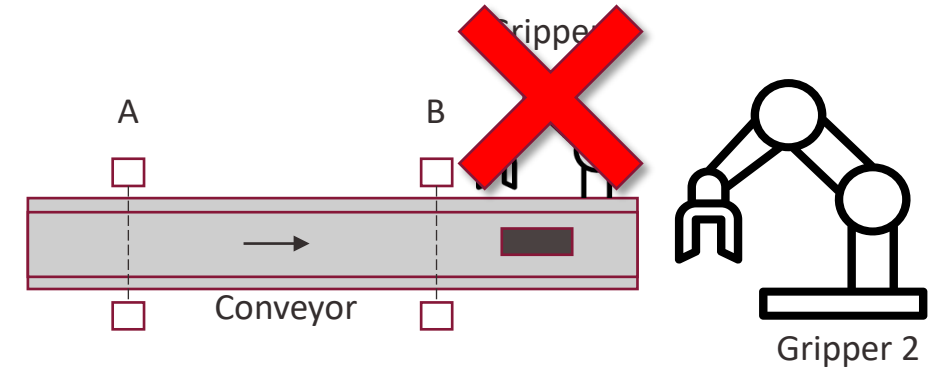


# Part 1 – How we could continue...



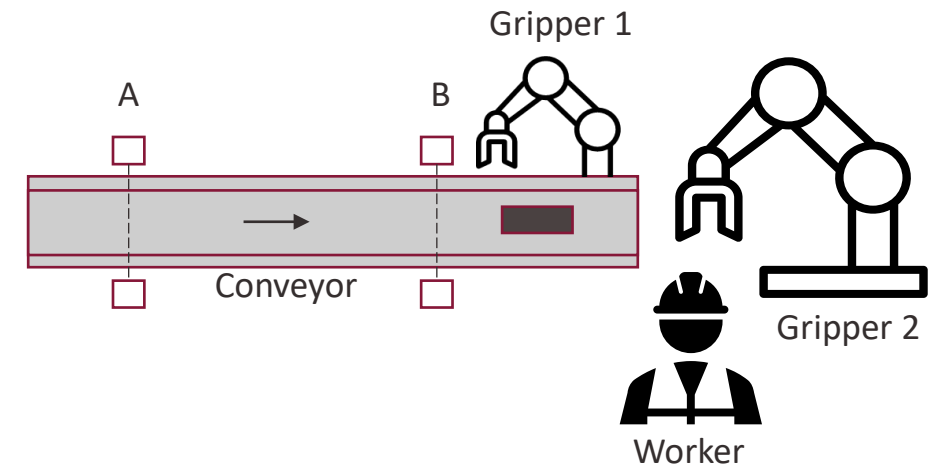
- Continue as before:

- Implement functionality of Sensor B and "hard-wire" it to Sensor A (B requests length from A)
- Add two gripper flows + a simple selection rule in Sensor B to determine a matching gripper



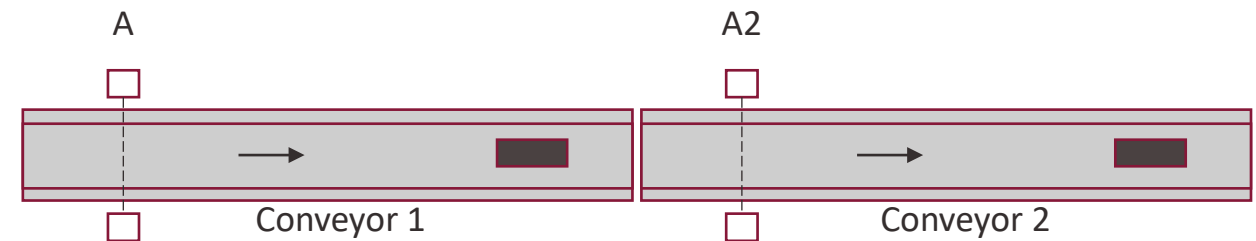
- Downsides of this approach

- Error-prone:
  - Everything hard-coded, errors are hard to find
  - No explicit agent / capability description
- Inflexible
  - No chance to react on breakdowns etc.
  - Not possible to add / remove agents in some kind of "registry"



- Next steps: Using an ontology

- Functions to dynamically register / unregister agents
- Using an ontology as a shared "language" between agents
- Using explicit rules to find agents

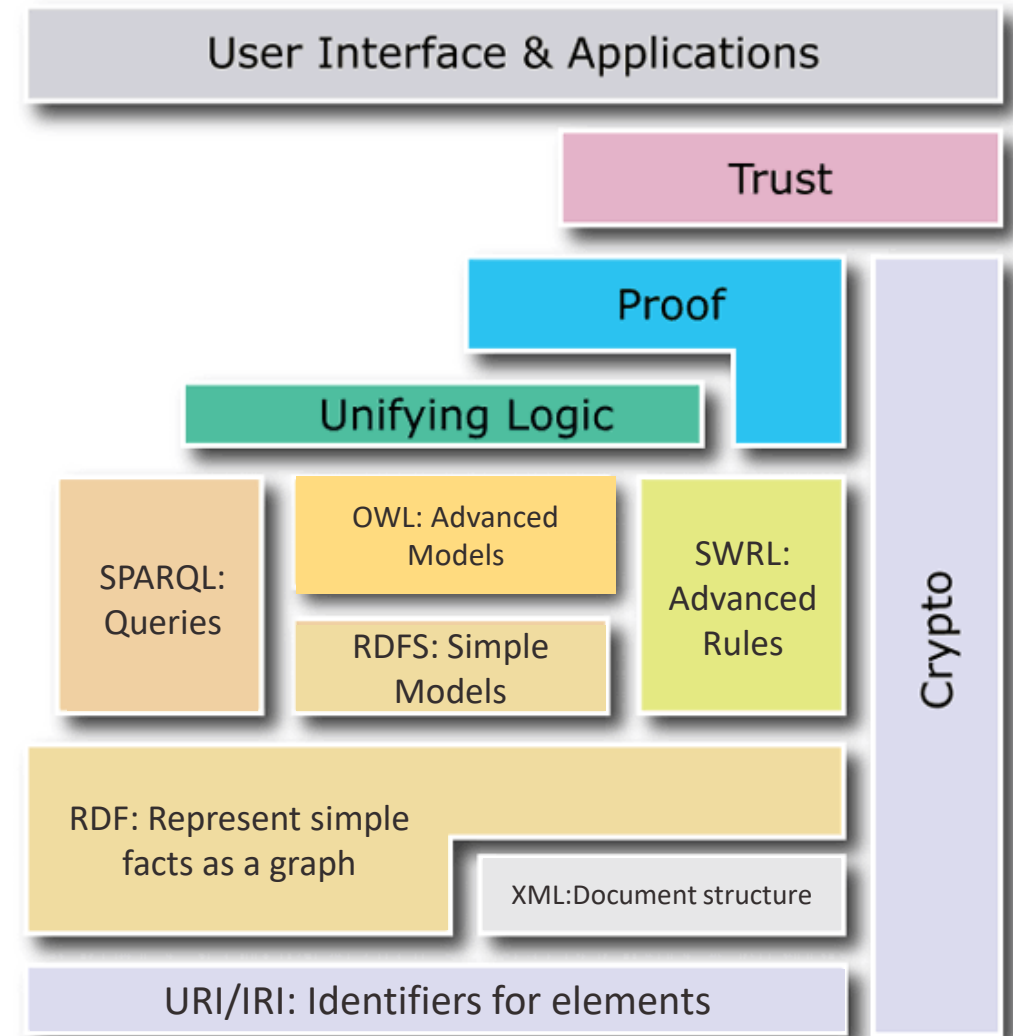




Institute of Automation

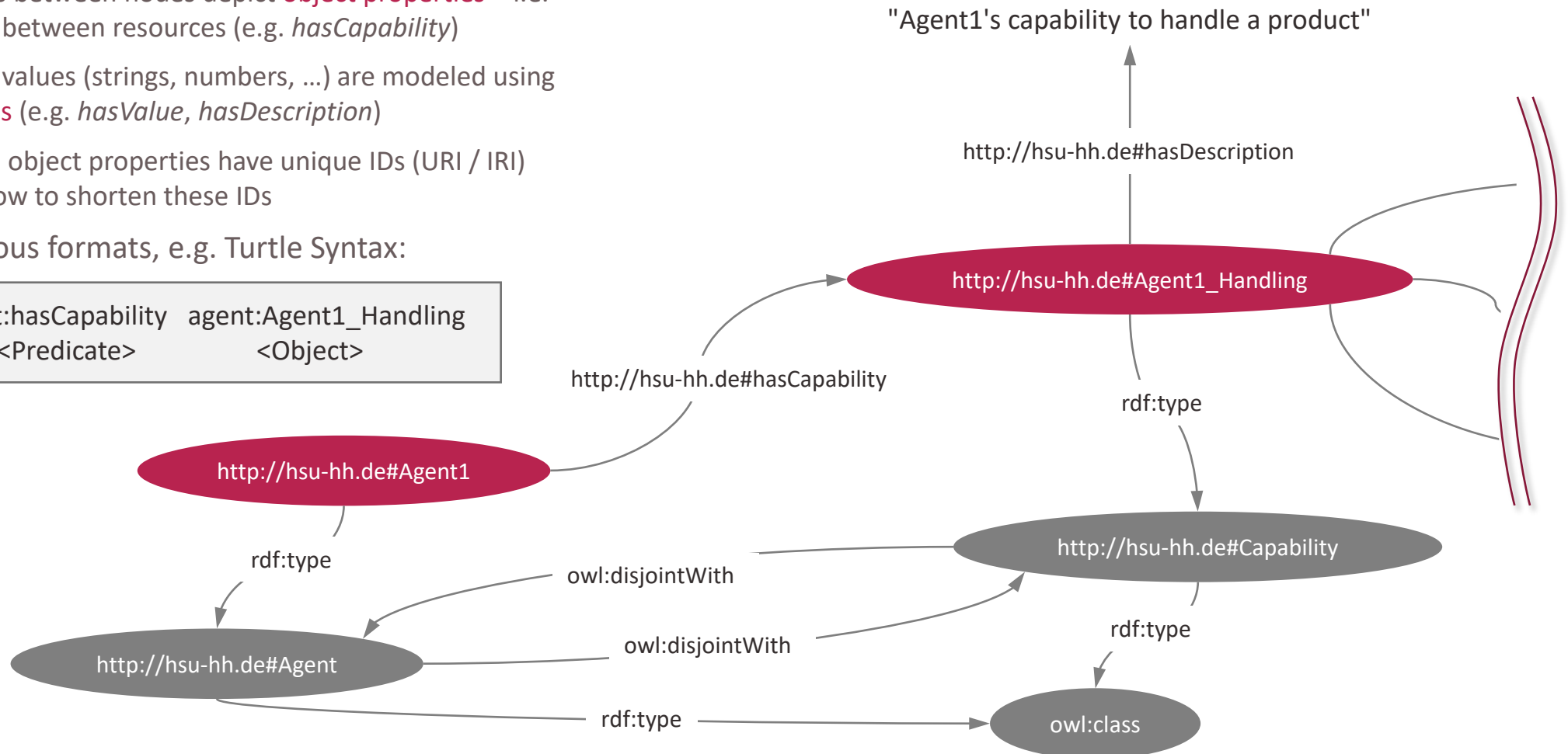
# Preparation for Part 2: An Introduction to Semantic Web Technologies

- More advanced W3C Standards:
  - “[Semantic Web Rule Language](#)” (SWRL) is a language to define rules that go beyond OWL's rule functionalities
  - “[Shapes Constraint Language](#)” (SHACL) to define constraints that have to be satisfied by an RDF graph
- W3C standards of the Semantic Web:
  - “[SPARQL Protocol And RDF Query Language](#)” (SPARQL) is the semantic web's query language used to retrieve information
  - “[Web Ontology Language](#)” (OWL) is used to model complex ontologies with advanced expressions about terms and their relations
  - “[RDF Schema](#)” (RDFS) adds ways to model a schema on top of RDF graphs which allows to define classes and class hierarchies
  - “[Resource Description Framework](#)” (RDF) is a basic technology of the semantic web used to represent simple facts about resources and relations as a graph
- Fundamental standards:
  - “[Extensible Markup Language](#)” (XML) is a markup language to model hierarchically structured Data using *tags*
  - [Uniform Resource Identifiers](#) or Internationalized Resource Identifiers are used for IDs that uniquely define elements

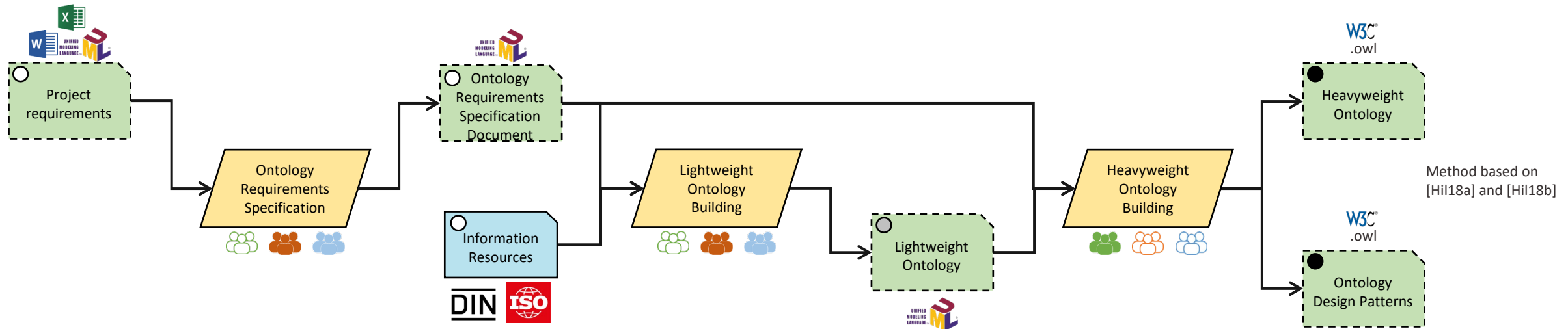


- Ontologies can be visualized as directed graphs:
  - Resources** (e.g. *Agents*, *Capabilities*, *Products*) are nodes
  - Directed edges between nodes depict **object properties** – i.e. the relations - between resources (e.g. *hasCapability*)
  - Primitive data values (strings, numbers, ...) are modeled using **data properties** (e.g. *hasValue*, *hasDescription*)
  - Resources and object properties have unique IDs (URI / IRI)  
→ Prefixes allow to shorten these IDs
- Stored using various formats, e.g. Turtle Syntax:

```
agent:Agent1 agent:hasCapability agent:Agent1_Handling  
<Subject>    <Predicate>      <Object>
```



- ... is time-consuming and complicated if done right
- There is a whole research area around the question "How to develop ontologies?"



- To get good ontologies with manageable effort, reuse is important
- One essential way to increase reuse: Create ontologies based on standards
  - Standards typically contain general knowledge widely applicable to a range of projects
  - Standards are typically agreed upon by a large community

[Hil18a] Hildebrandt, Constantin et al. (2018): Ontology Engineering for Collaborative Embedded Systems – Requirements and Initial Approach. In: Ina Schäfer und Dimitris Karagiannis (Hg.): Fachtagung Modellierung 2018.

[Hil18b] Hildebrandt, Constantin et al. (2018): Ontology Building for Cyber-Physical Systems: A domain expertcentric approach. In: 2018 14th IEEE CASE. Munich, Germany, 20-24.08.2018.



## ■ VDI 3682 – Formalized Process Description

- Created by the *German Association of Engineers*
- Very lightweight modeling language to describe both technical and non-technical processes

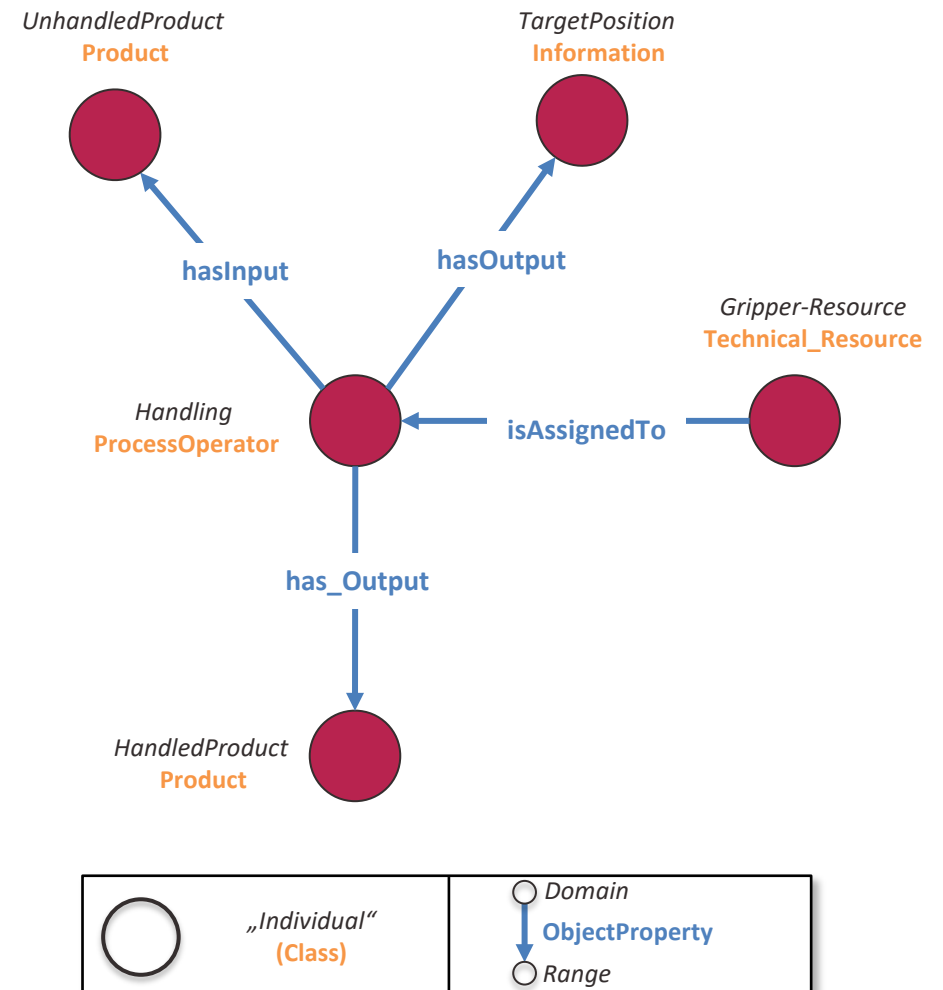
## ■ Easy-to-use graphical representation:

- See online modeling editor at [demo.fpbjs.net](https://demo.fpbjs.net)

## ■ Information model

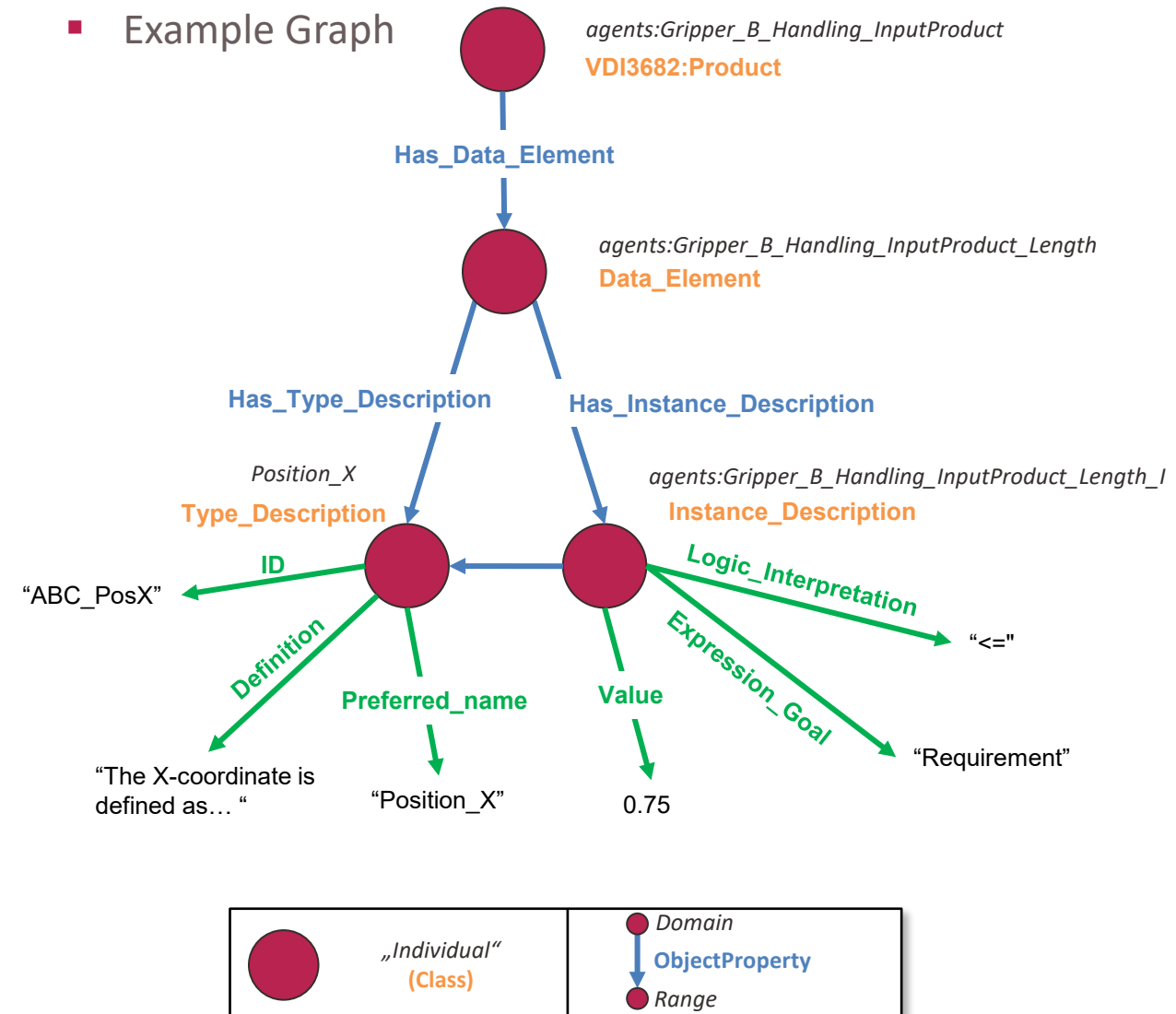
- A process operator transforms input elements into output elements
- Types of input / output:
  - Product
  - Information
  - Energy
- Every process operator is assigned to a technical resource
- A complete process is confined within a system border
- Process operators are chained through their inputs / outputs

## ■ Example Graph



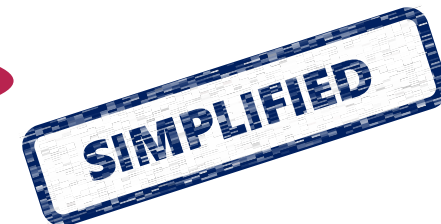
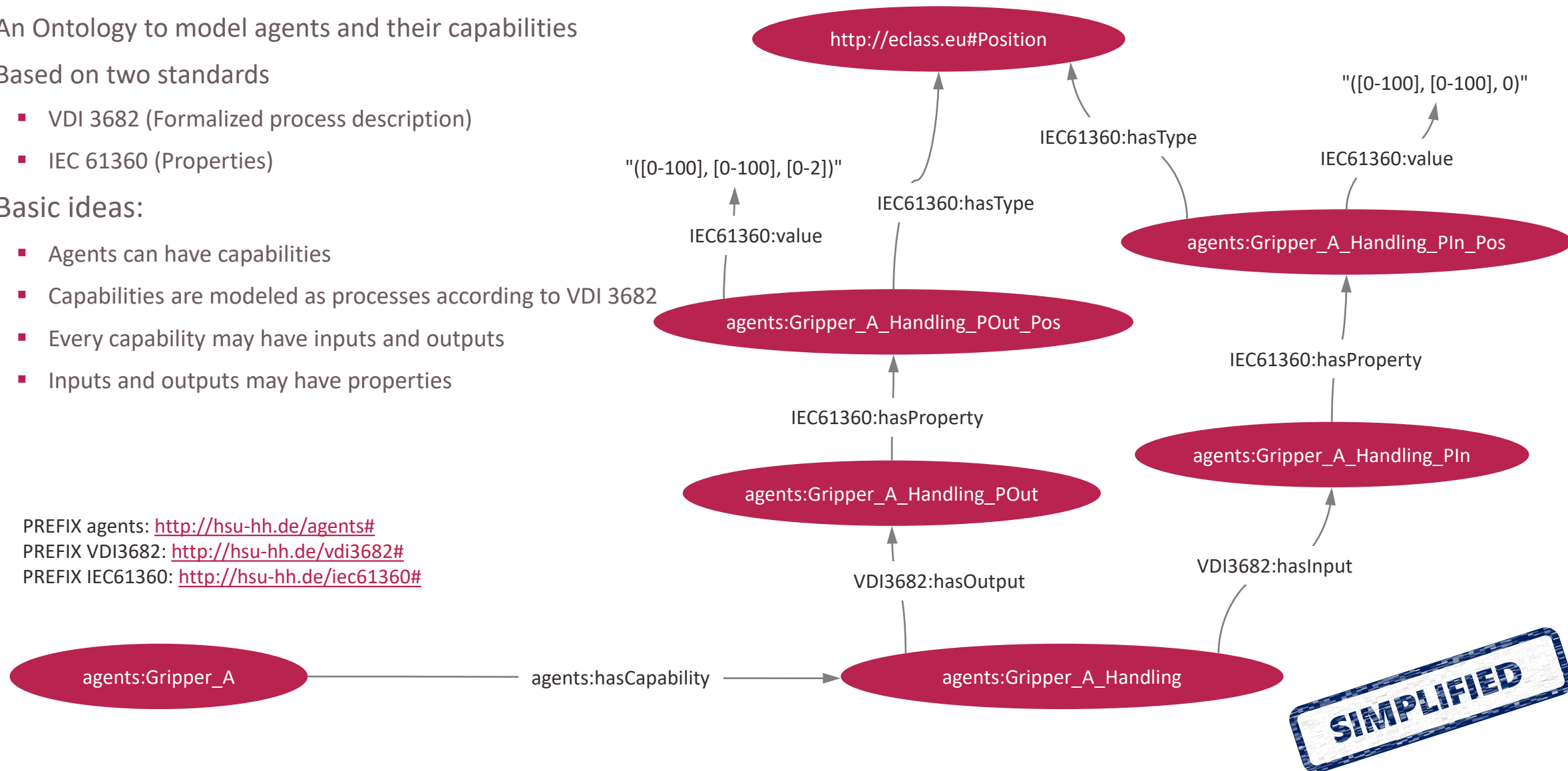
- IEC 61360 uses “Data Elements” to characterize objects
- Data Elements are described by a **type description** and an **instance description**.
- Data Elements can represent anything that should be represented as a complex or simple data type (e.g. Temperature or Pressure)
- The type description represents the generally applicable parts of the description (e.g. ID, definition, unit of measure)
- Instance descriptions describe the value of a specific data element in a given context:
  - Requirement
  - Assurance
  - Actual (=Measured) Value
- Examples: Length of input WP is required to be  $\leq 1$

## Example Graph



- An Ontology to model agents and their capabilities
- Based on two standards
  - VDI 3682 (Formalized process description)
  - IEC 61360 (Properties)
- Basic ideas:
  - Agents can have capabilities
  - Capabilities are modeled as processes according to VDI 3682
  - Every capability may have inputs and outputs
  - Inputs and outputs may have properties

PREFIX agents: <http://hsu-hh.de/agents#>  
PREFIX VDI3682: <http://hsu-hh.de/vdi3682#>  
PREFIX IEC61360: <http://hsu-hh.de/iec61360#>



- Defined in the [W3C SPARQL Query standard](#)
- A SELECT query consists of three parts:
  - Prefix definition (header)
  - Definition of variables to be selected (**SELECT**)
  - Definition of a graph pattern to find (**WHERE**)
- Definition of variables to be selected
  - Within the **SELECT** clause
  - Variables are declared using a **?** + user-defined name
  - Variables are separated by spaces
- Defining a graph pattern
  - Within the **WHERE { }** clause
  - Using a syntax similar to Turtle
- The SPARQL endpoint receiving the query checks all its triples and returns all results that match the given graph pattern

## Example data in graph

agent:Gripper_A	agent:hasCapability	agent:Gripper_A_Handling
agent:Sensor_A	agent:hasCapability	agent:Sensor_A_DetectProduct
agent:Sensor_A	agent:hasCapability	agent:Sensor_A_MeasureLength

## SPARQL query

# Prefix definition

PREFIX agent: <<http://www.hsu-hh.de/aut/ontologies/agent-summer-school#>>

# Definition of variables to be selected

```
SELECT ?agent ?capability WHERE {  
  ?agent agent:hasCapability ?capability. # Definition of graph pattern  
}
```

## Return

?agent	?capability
agent:Gripper_A	agent:Gripper_A_Handling
agent:Sensor_A	agent:Sensor_A_DetectProduct
agent:Sensor_A	agent:Sensor_A_MeasureLength

- Defined in the [W3C SPARQL Update Standard](#)
- SPARQL Updates don't return information but instead change existing graphs
  - A **INSERT** operation adds triples if they don't exist yet
    - **INSERT DATA** adds triples without constraints
    - **INSERT { ... } WHERE { ... }** adds triples to all results found with the **WHERE { ... }** clause
  - A **DELETE** operation deletes triples from a graph
    - **DELETE DATA** removes exactly the specified triples (no variables allowed)
    - **DELETE { ... } WHERE { ... }** deletes the structure defined inside the **DELETE { ... }** clause for every match of the **WHERE { ... }** clause
- **INSERT** and **DELETE** operations can be used to (dynamically) change contents of the graph

## Example data in graph

agent:Gripper_A	rdf:type	agent:Agent
agent:Sensor_A	rdf:type	agent:Agent

## SPARQL Update

### # Prefix definition

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX rdfs: <https://www.w3.org/2000/01/rdf-schema#>

PREFIX agent: <http://www.hsu-hh.de/aut/ontologies/agent-summer-school#>

### # Definition of the graph (part) to be inserted

INSERT { ?x rdfs:comment "This is an agent..." }

### # Definition of the variables to insert / append to

WHERE { ?x rdf:type agent:Agent. }

## Resulting example data in graph

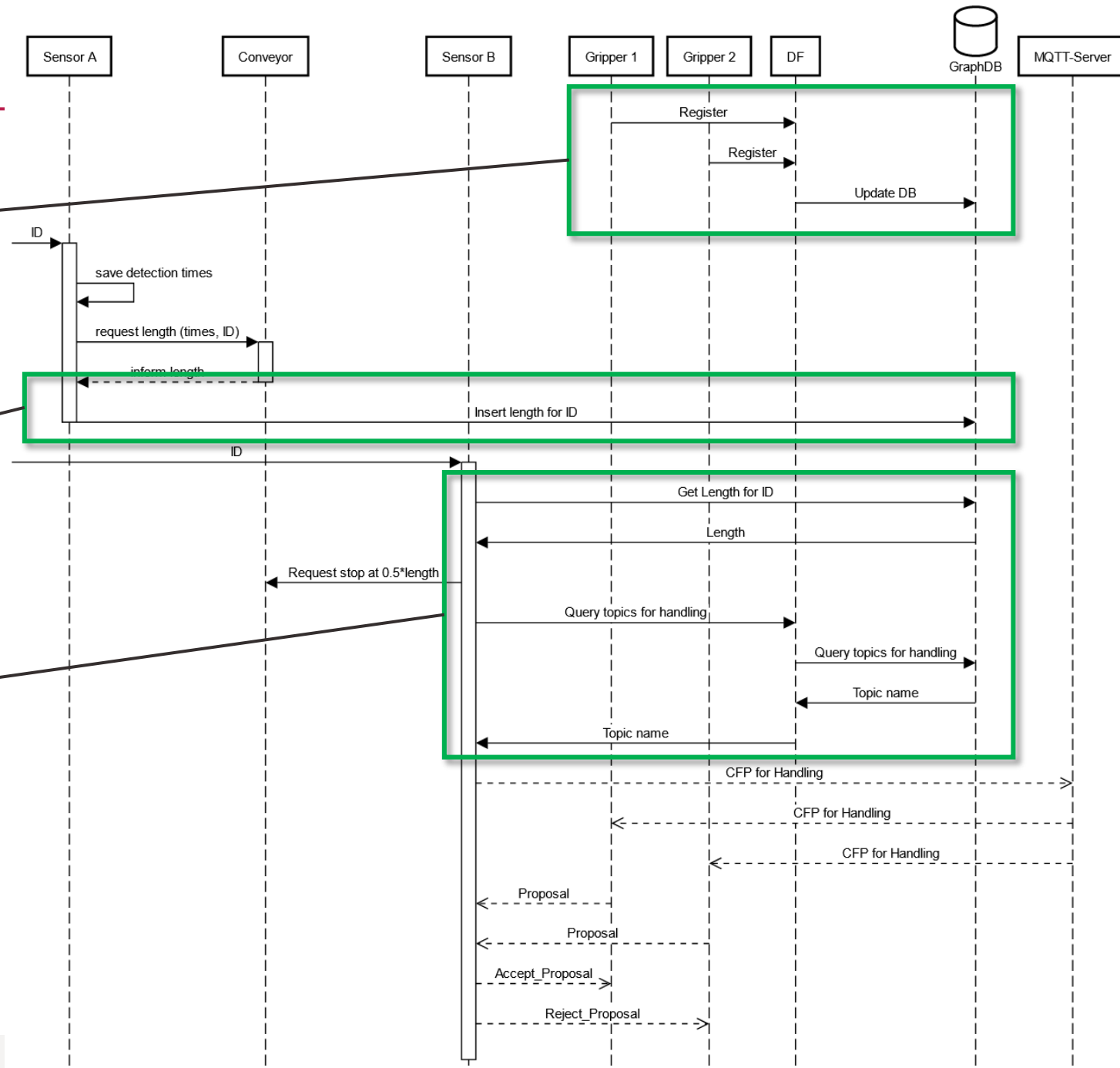
agent:Gripper_A	rdf:type	agent:Agent.
agent:Gripper_A	rdfs:comment	"This is an agent..."
agent:Sensor_A	rdf:type	agent:Agent.
agent:Sensor_A	rdfs:comment	"This is an agent..."

# Part 2 – Using an Ontology

2: Resources register themselves with their capabilities in an ontology (via the DF)

1: Store workpiece information in an ontology so that it is accessible to other agents

3: Sensor B agent queries for length and applicable resources (using SPARQL)



- Sensor A needs to store workpiece information locally
- How can we store workpiece information inside an ontology? → With a SPARQL INSERT

## # Prefix definition

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX agents: <http://www.hsu-hh.de/aut/ontologies/agent-summer-school#>

## # Actual INSERT operation

INSERT DATA {

```
agents:Product_`${msg.payload.barcode}` a agents:Product;  
agents:hasProductId "`${msg.payload.barcode}`";  
agents:hasLength `${msg.payload.wp_length}`;  
agents:timeAdded `${msg.payload.sensorA_timeStartDetection}`.
```

}

How can we create unique product individuals?

How can we add the ID, length and detection time **of the current product**?

By dynamically composing the INSERT query using JavaScript expressions

## Part 2 – Add / Delete agents with capabilities



- How can we register / unregister agents with their capabilities? → With a SPARQL INSERT

# Prefix definition not shown

# Actual INSERT operation

INSERT DATA {

```
agents:Gripper_A rdf:type agents:GripperAgent ;
  agents:hasCapability agents:Gripper_A_Handling .
agents:Gripper_A_Handling VDI3682:hasInput agents:Gripper_A_Handling_PIn ;
  VDI3682:hasOutput agents:Gripper_A_Handling_POut;
  rdf:type agents:Handling.
agents:Gripper_A_Handling_PIn rdf:type agents:Product ;
  DINEN61360:has_Data_Element agents:Gripper_A_Handling_PIn_Length , ...
agents:Gripper_A_Handling_PIn_Length rdf:type DINEN61360:Data_Element,
  DINEN61360:Instance_Description ;
  DINEN61360:has_Instance_Description agents:Gripper_A_Handling_PIn_Length ;
  DINEN61360:has_Type_Description agents:Length ;
  DINEN61360:Expression_Goal "Requirement" ;
  DINEN61360:Logic_Interpretation "<=" ;
  DINEN61360:Value 1 .
```

...

Defines agent and links it to its capability

Defines capability and links it to its inputs and outputs

Link input product with its data elements (only one shown here)

Definition of a data element (only one shown here)

To define:

- Input:
  - X, Y, Z
  - Max. Length
- Output (irrelevant for this workshop):
  - X, Y, Z
  - Length





Institute of Automation

<< Coffee Break >>



Institute of Automation

# Session 2

Parts 3 and 4

- Introduction & Scenario Description
- Implementation Part 1 (Basics + HTTP)
- Ontologies, SPARQL & Capabilities
- Implementation Part 2 (Products and Agent Registration)



Session 1

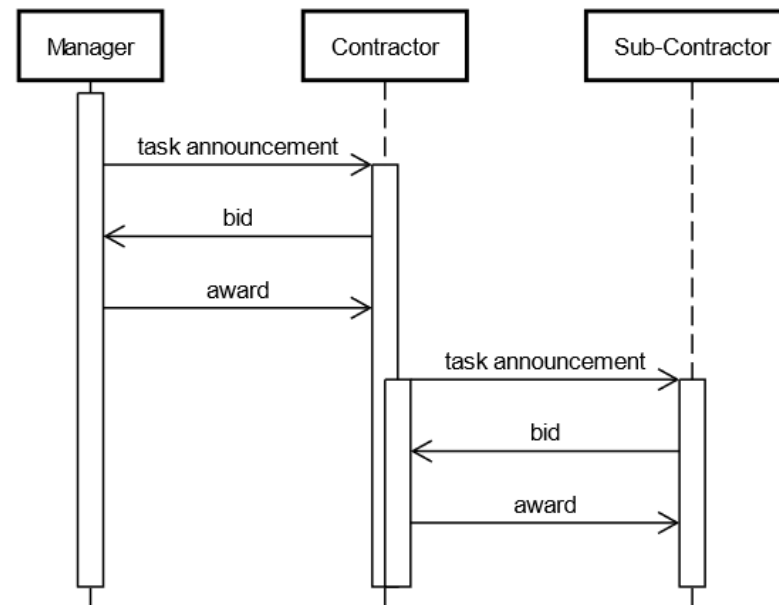
## Coffee Break

- Thoughts on Negotiations & Semantics for Agents
- Implementation Part 3 (Message Processing)
- Implementation Part 4 (Bid Selection)



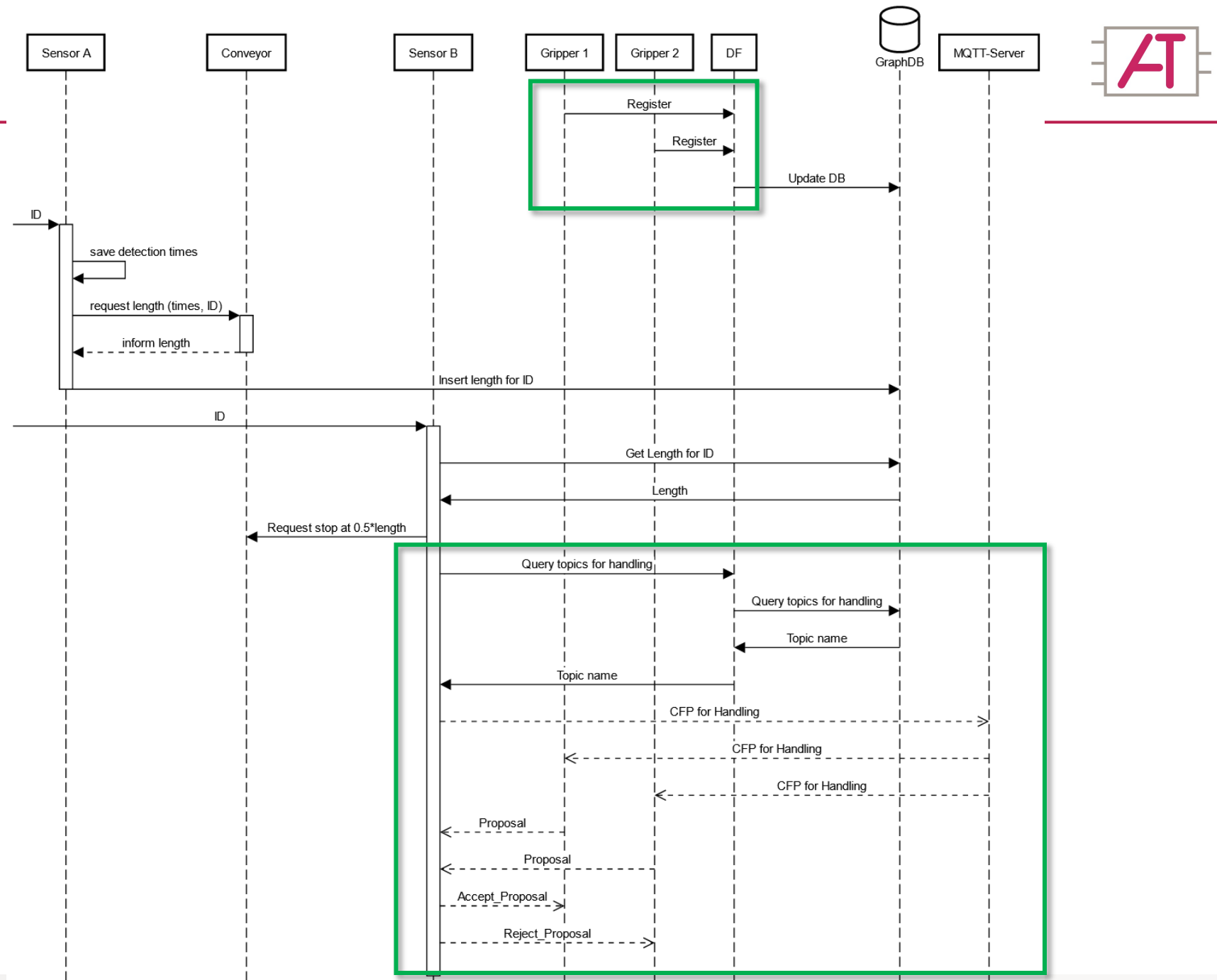
Session 2

- In small groups:
  - What is the goal & advantage of a negotiation at this point? What are disadvantages?
  - What influences are / aspects can be accounted for?
  - How does a simple negotiation i.e. auction look like?

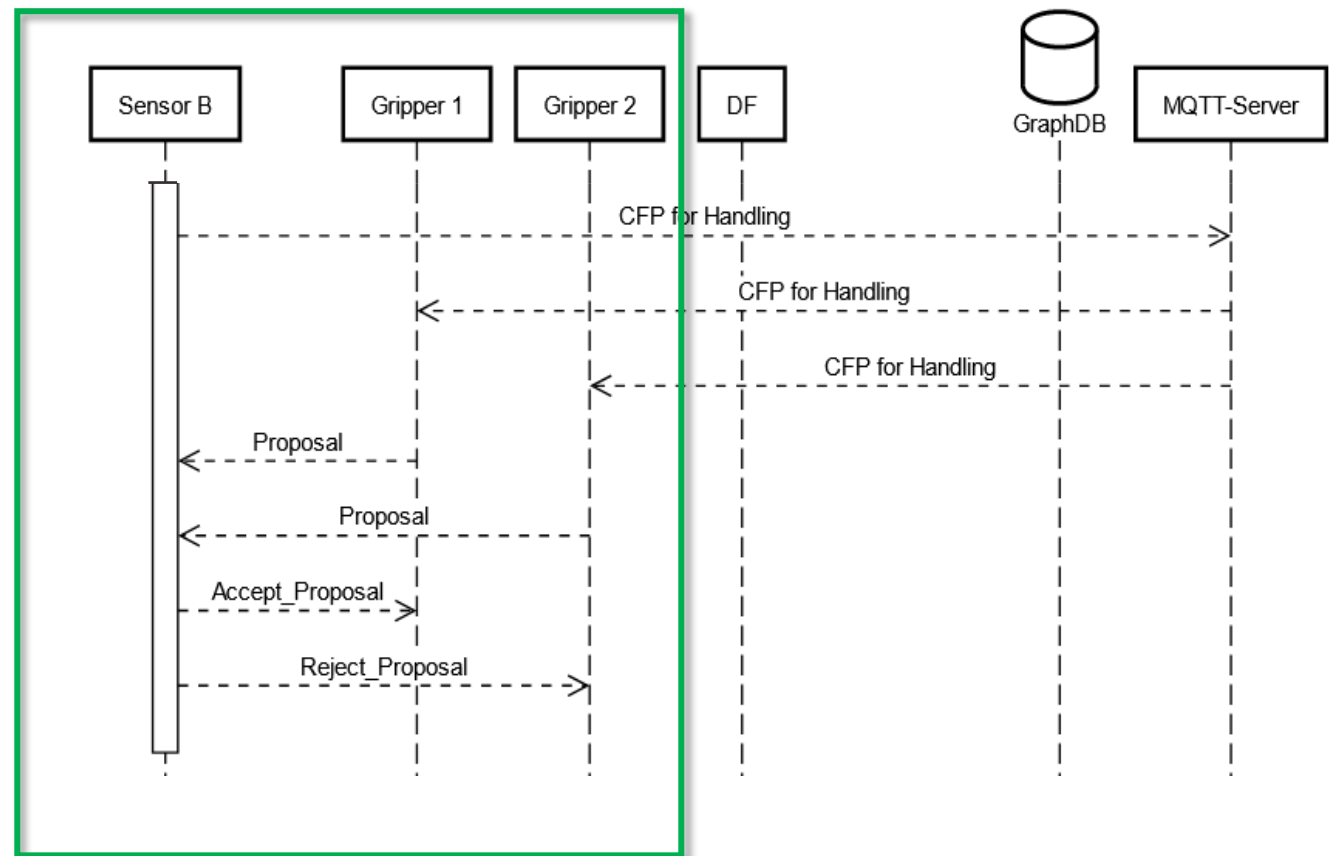


Based on Smith, R. G., "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, 1980.

# Overview



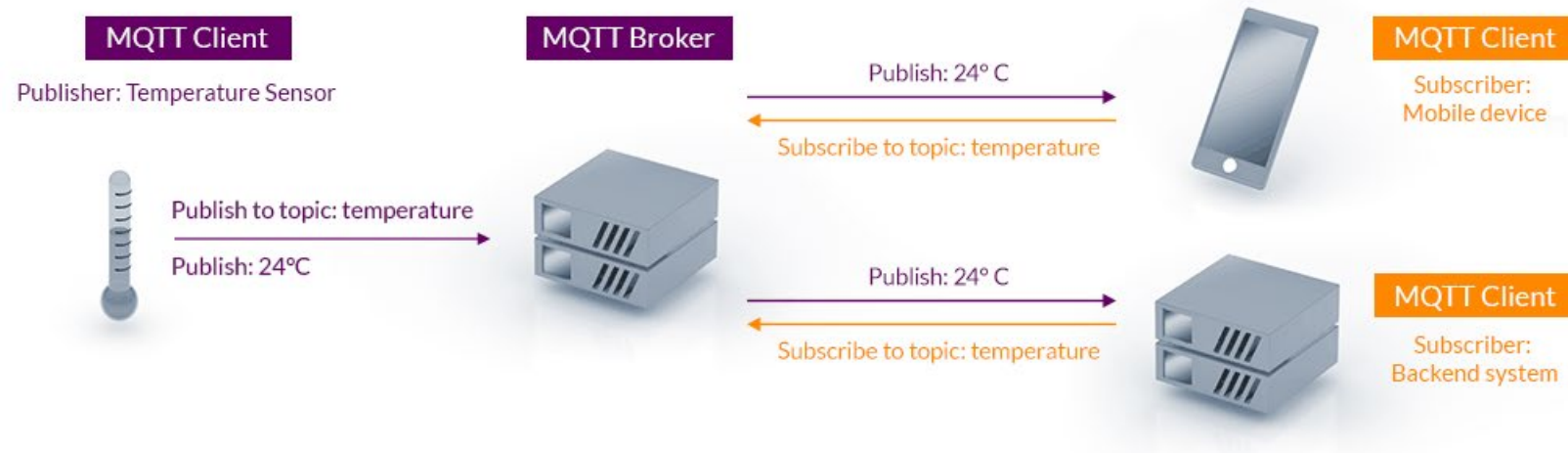
- In small groups:
  - What has to happen within the software at this point (to optimize the negotiation's outcome and minimize its duration)?



## ■ What is MQTT?

- „MQTT [...] is a lightweight, publish-subscribe, machine to machine network protocol. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth.” Wikipedia

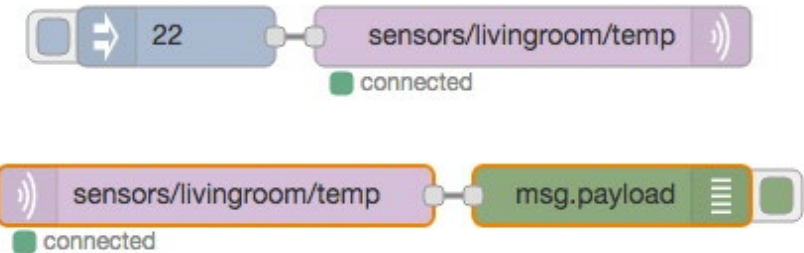
### MQTT Publish / Subscribe Architecture



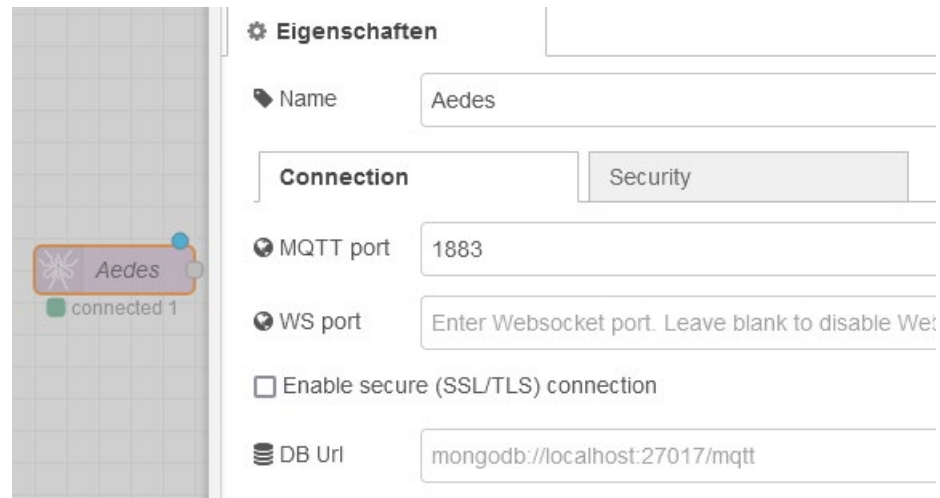
Source: <https://mqtt.org/>

- Node-RED provides the in and out nodes
- But this requires a separate MQTT server like Mosquitto
- Other solution: Aedes  
(<https://flows.nodered.org/node/node-red-contrib-mqtt-broker>)

## Example



<https://cookbook.nodered.org/mqtt/connect-to-broker>





- Long-standing guideline: FIPA ACL

- Conversation ids
- Performatives
- Content language

- However, it is a bit outdated and lacks support for, e.g. „event notification at the low control level, integration with legacy systems and a combination of services and agent-based systems”

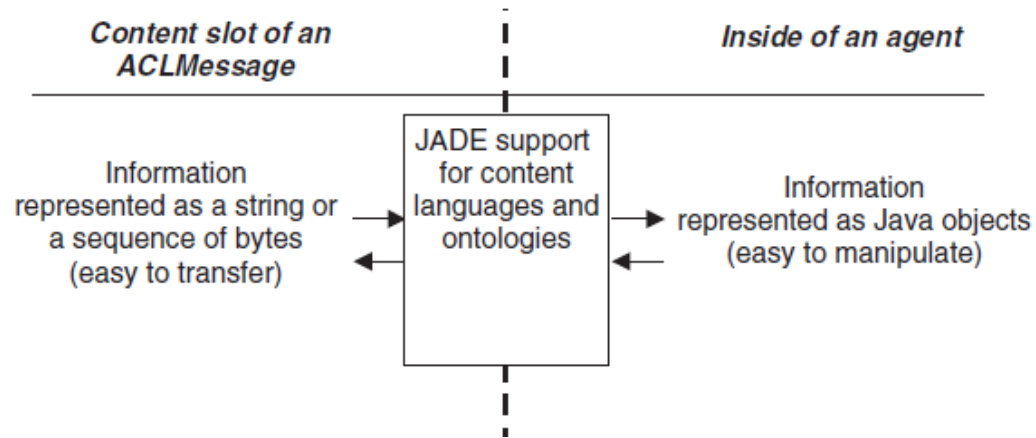
I. Seixas and P. Leitão, “Standards Compliance in Industrial Agents Applications,” in IEEE IECON, Nov. 2013.

Parameter	Category of Parameters
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

Table 1: FIPA ACL Message Parameters

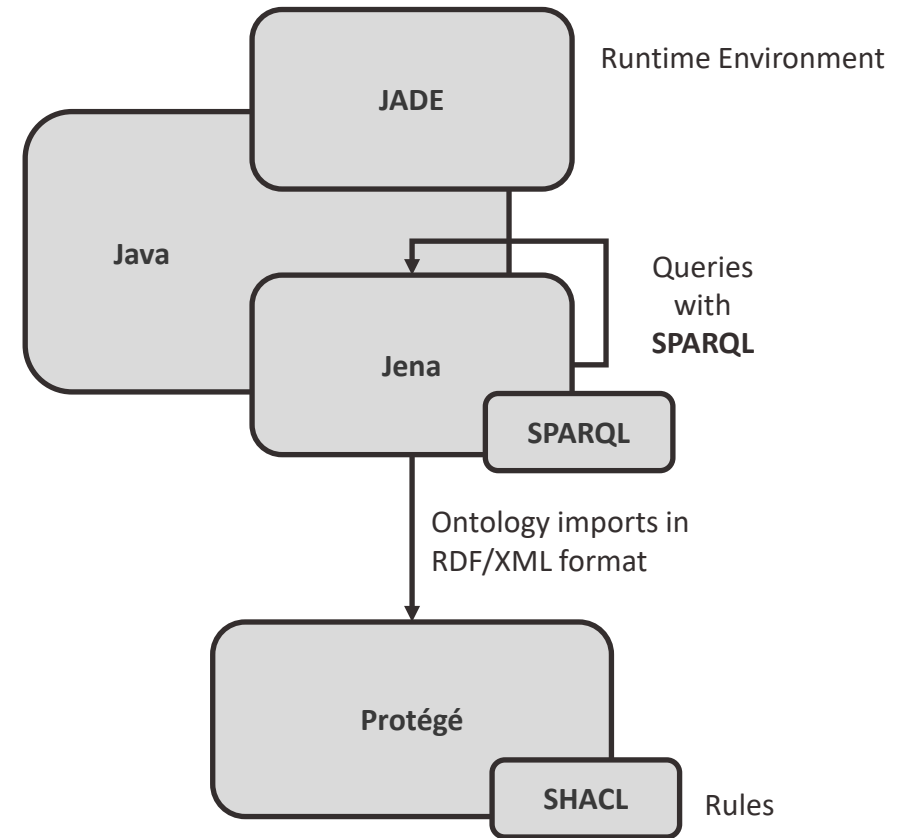
Source: <http://www.fipa.org/specs/fipa00061/SC00061G.html>

- Case 1: Message passing and interpreting

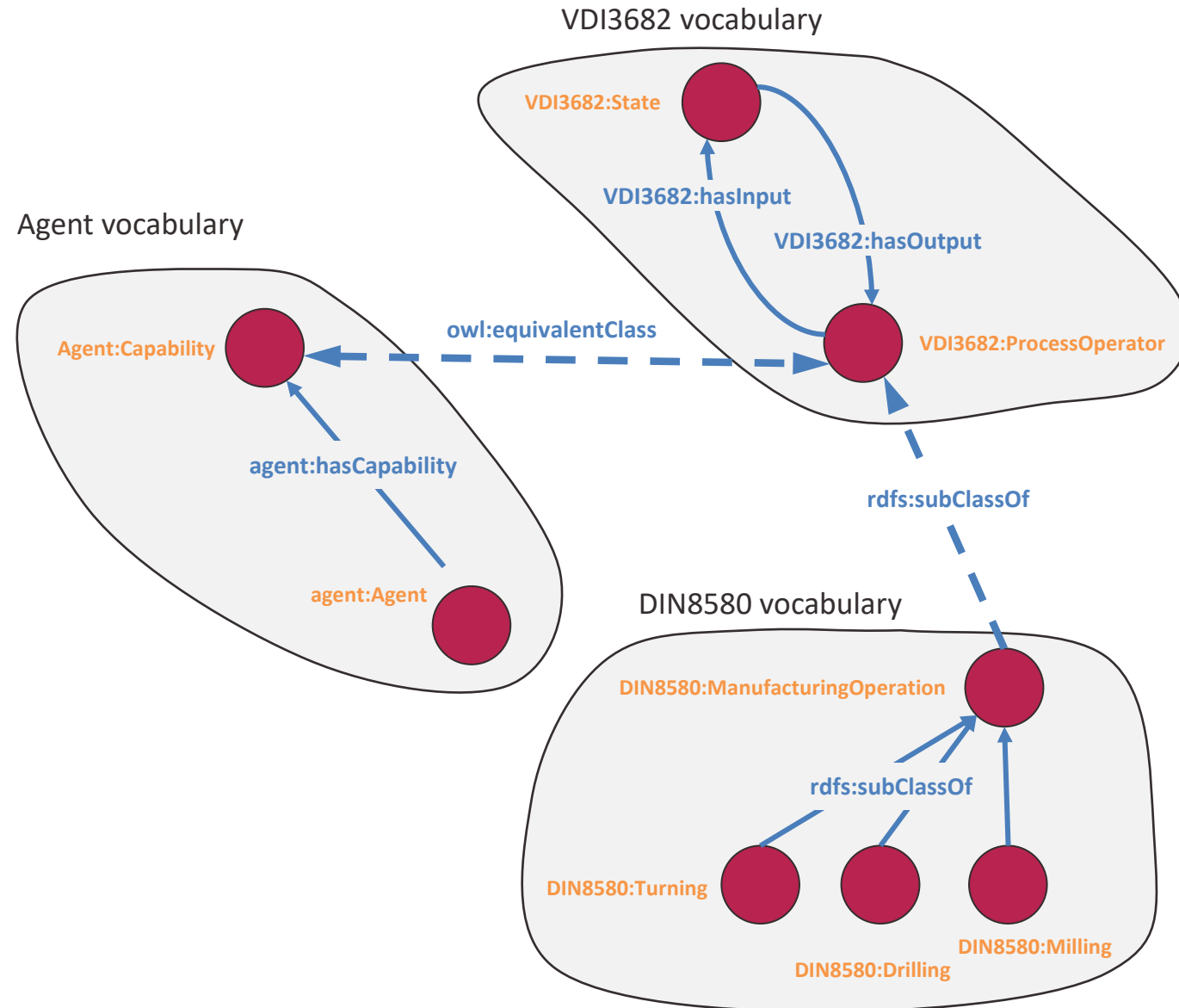


Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing multi-agent systems with JADE*. John Wiley & Sons.

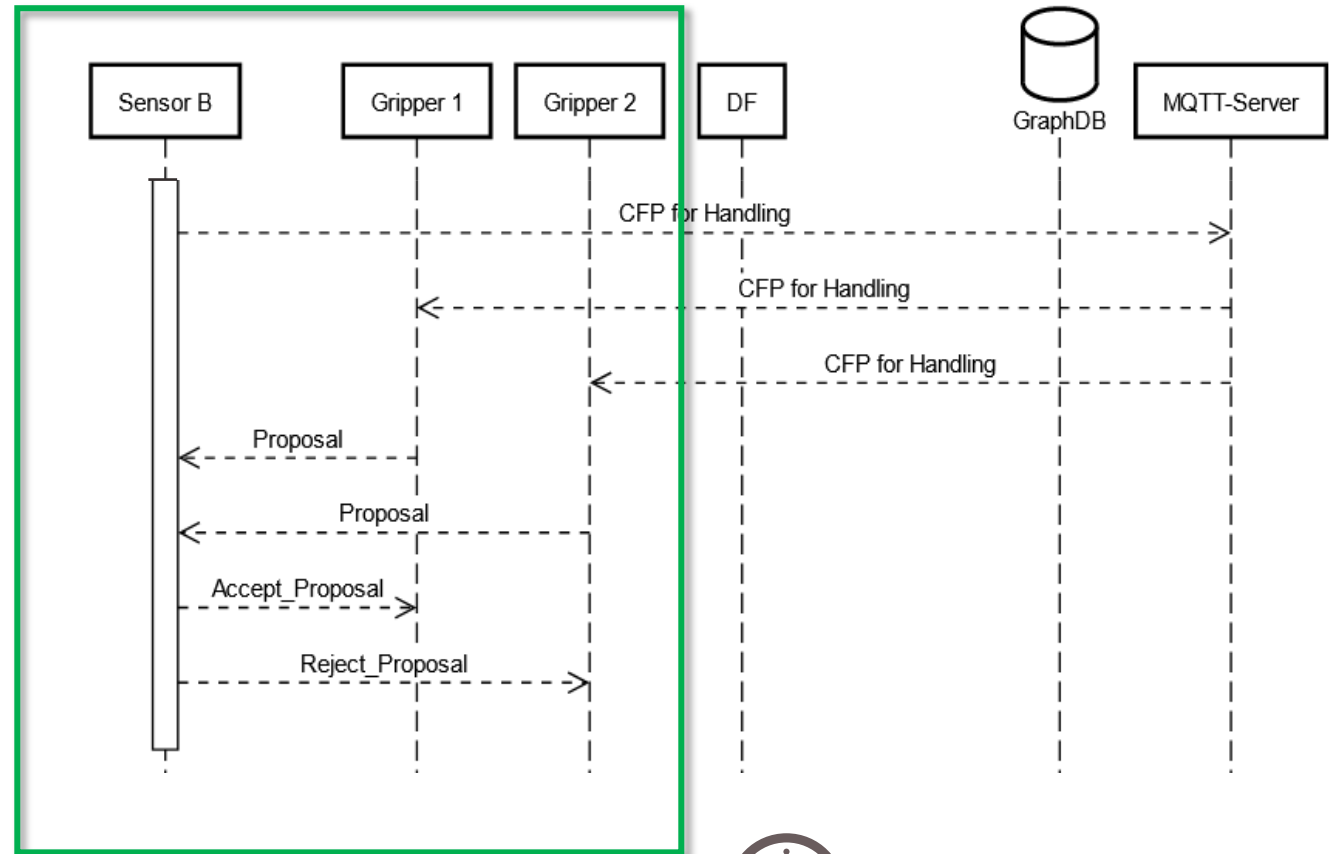
- Case 2: Agent knowledge bases



- Ontologies are well suited to define a shared vocabulary by explicitly creating relations between different terms
  - Defining new relations → so-called "ontology alignment"
    - Define equivalent classes
    - Subclassing
    - Define new relations
  - Using rules that allow inferring new facts
    - "Everything that has a capability is an agent"
    - "Every process operator that manipulates X, Y and Z coordinate of an input is a handling operation"



- In small groups:
  - What should happen (internal logic) with the different performatives?
- Implement different performative handling



15 Minutes (in groups)

- When is a gripper able to handle a workpiece?

➤ If X, Y, Z of the workpiece are within the gripper's work area and workpiece length  $\leq$  max. length

PREFIX agents: <http://www.hsu-hh.de/aut/ontologies/agent-summer-school#>

PREFIX VDI3682: <http://www.hsu-ifa.de/ontologies/VDI3682#>

PREFIX DINEN61360: <http://www.hsu-ifa.de/ontologies/DINEN61360#>

```
SELECT ?elem ?type ?expGoal ?expression WHERE {  
  agents:Gripper_A agents:hasCapability ?capability.  
  ?capability VDI3682:hasInput ?elem.  
  ?elem DINEN61360:has_Data_Element ?de.  
  ?de DINEN61360:has_Type_Description ?type;  
    DINEN61360:has_Instance_Description ?inst.  
  ?inst DINEN61360:Value ?val;  
    DINEN61360:Logic_Interpretation ?logInt;  
    DINEN61360:Expression_Goal ?expGoal.  
  
  BIND(CONCAT(STR(?expGoal), STR(?logInt), STR(?val)) AS ?expression)  
}
```

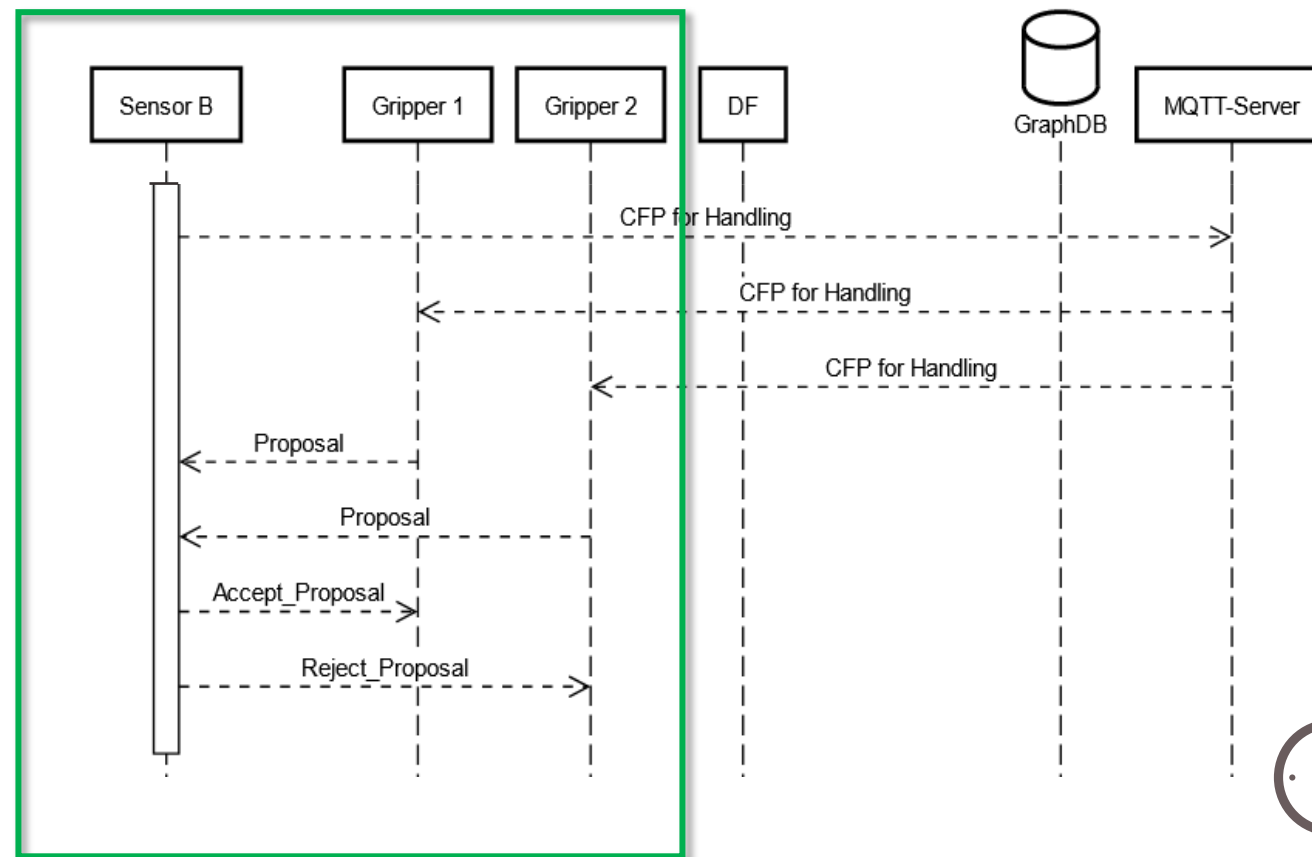
Find types and instance descriptions of all input data elements

Find value, logic interpretation (e.g. " $\leq$ ") and expression goal ("Requirement")

Create an expression by concatenating expression goal, logic interpretation and value.  
→ Example: "Requirement  $\leq$  75"

This expression is used in Node-RED:  
1. Replace "Requirement" with the current value  
2. Evaluate the expression using "eval"

- Implement selection process (cheapest offer)



10 Minutes (in groups)

- Basics of Node-RED (low-code and edge-capable)
  - MAS can be developed very quickly while being integrated with handy APIs etc.
- HTTP & MQTT
  - Both are easily usable within Node-RED and both have their advantages / disadvantages
- Basics of the Semantic Web
  - Short introduction into possible applications of ontologies
- Code is available on Github  
(<https://github.com/hsu-aut/ISSIA22-node-red-agents>)
  - Feel free to use that as a playground /starting point for future applications

