

# Semantic Support Points for on the Fly Knowledge Encoding in Heterogenous Systems

Daniel Spieldenner<sup>1</sup><sup>a</sup>, André Antakli<sup>1</sup><sup>b</sup>, Torsten Spieldenner<sup>2</sup><sup>c</sup> and Harkiran Sahota<sup>2</sup><sup>d</sup>

<sup>1</sup>Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Campus D3 2, Saarbrücken, 66123, Saarland, Germany

<sup>2</sup>Lappeenranta-Lahti University of Technology LUT, Yliopistonkatu 34, Lappeenranta, 53850, Finland  
{daniel.spieldenner, andre.antakli}@dfki.de, tor.spiel@gmail.com, fourth\_author@student.lut.fi

Keywords: Intelligent Systems, Semantic Web, Interoperability, Multi Agent Systems.

Abstract: Connecting participants in heterogenous environments and allowing for seamless interaction is a challenging task that requires profound knowledge of the system envolved and data exchanged. The knowledge on how to integrate and use certain systems is usually given in the form of manuals, code documentation or needs to be derived by the developer by understanding and interpreting source code or data sets. With this work, we propose an approach to provide this knowledge about data via so called semantic support points, making use of Semantic Web technologies and appropriate ontologies, in a resource efficient fashion by creating semantic representations only when and where needed. Instead of just providing semantic meta data to describe actors in the environment, we make it possible to embed actual data values from data sources like databases and payloads exchanged between systems into a virtual semantic cloud, allowing for interactions purely on the semantic data representations and propagating computation results back to the system layer automatically.


## 1 INTRODUCTION


Every day our world is becoming more and more complex, systems are growing in size, data is produced in unprecedented amounts and exchanged in increasingly smart environments, from smart homes equipped with devices ready for the internet of things (Khanna and Kaur, 2020; Nižetić et al., 2020; Lee and Lee, 2015) to smart cities (Wang et al., 2020; Jafari et al., 2023; Khan et al., 2020) to the vision of national or even international data spaces. (Halevy et al., 2006; Franklin et al., 2005; Curry, 2020; Solmaz et al., 2022; Zillner et al., 2021)


Maintaining these systems, consisting of often very heterogenous devices, services and data sources, some of them decades old from times long before concepts like interoperability or the *Semantic Web* (Lassila et al., 2001) were a thing, poses an enormous challenge, making it necessary to understand data, transform data to be understood by the recipient and making data discoverable and available in general.


This is especially true for complex intelligent systems, which consist of a range of actors working together autonomously. For instance, in a smart home, sensors and control units collaborate to maintain the resident’s preferred room temperature and lighting. Similarly, an entire shop floor with machines forming an intricate production line operates in a coordinated manner. However, every participant in even the most complex system was designed by someone with a certain intention about the task the participant is expected to perform, under which conditions, with certain prerequisites and expected outcomes. Usually this knowledge is ”encoded” in manuals, software documentation, code comments or what we would consider ”common knowledge”, for example the fact that a thermometer is supposed to measure temperature.

We aim to bring this implicit knowledge contained by design within the actors themselves to the environment itself, allowing the participants to emanate this ”knowledge” when and where needed as requested and to weave somewhat of a volatile, connected fabric of semantics above the actual system layer, as suggested by (Spieldenner, 2023), expressing and encoding expert knowledge concerning the available systems, the intended interpretation of data values, capabilities of actors and the possible connection be-

<sup>a</sup> <https://orcid.org/0000-0002-8262-7733>

<sup>b</sup> <https://orcid.org/0000-0002-4066-7649>

<sup>c</sup> <https://orcid.org/0000-0003-3034-9345>

<sup>d</sup> <https://orcid.org/0000-0003-1160-1904>

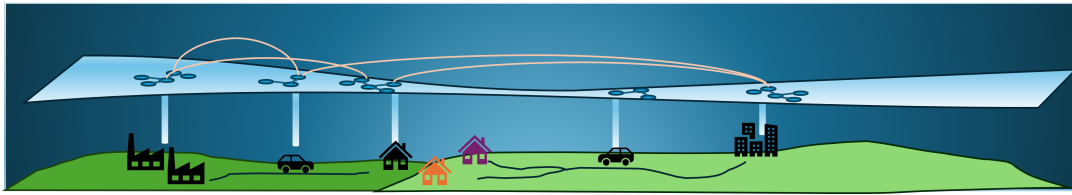


Figure 1: Our vision of semantic support points, spanning a layer of semantics abstracting from the world below.

tween participants with the help of ontologies (Breitman et al., 2007) and principles of the *Semantic Web*. (Lassila et al., 2001)

With such a representation at hand, we would allow any system to benefit from concepts like semantic communication (Lan et al., 2021; Qin et al., 2021), semantic interoperability (Heiler, 1995) as well as reasoning and semantic queries.

In this paper we present a method to bridge the gap between a world consisting of legacy data sources, devices and services without any semantic information attached and the aforementioned semantic data fabric, using *semantic support points*: federated, light weight, runtime configurable and non-intrusive micro services tailored to each individual participant.

Each of these micro services have access to individual distributed data transformation definition files, creating a semantic representation of individual or aggregated data values at runtime on request as well as data selection and transformation on the semantic level for transformation back to non-semantic data representations like JSON.

Pairs of support points allow for bidirectional communication between what we consider the *system layer*, i.e. the collection of participants without any semantic information attached, and arbitrary actors on the *semantic layer*, interacting directly and only with knowledge graphs provided by the semantic support points. The knowledge graphs will be derived from non-semantic data emitted from non-semantic interfaces or data sources, and enriched with human written expert knowledge by providing transform rules between non-semantic data formats to RDF, and selection and transform rules to extract data from knowledge graphs and create precisely defined JSON objects accordingly.

This approach enables concepts like semantic interoperability, semantic queries and reasoning on knowledge graphs in any environment, even if it was initially designed without any attempt to provide semantic capabilities.

As a proof of concept, we demonstrate the proposed approach by applying it to a real life example of an automotive production line, using semantic support points to extract information from heterogeneous data sources, transforming it to semantic data

and feeding it to a multi agent system, using the semantic nature of the mapped data values for deriving optimized production plans by using a reasoning engine. The results of the agent frameworks’ plan optimization are then propagated back to a range of systems via non-semantic input tailored specifically to each participant’s interface and data model.

The paper is structured as follows:

An overview over the relevant background and related approaches aiming to realize a concept of a semantic data integration is given in section 2. In section 3, we introduce our own concept of *semantic support points*, minimal service implementations allowing to access transform rules based on semantic concepts about systems and data sources to generate knowledge graphs including knowledge about the system interfaces themselves as well as the data exchanged. Section 4 demonstrates how to put the concept into practical use by using semantic support points to enhance a system representing an automotive production line by adding a semantic multi agent system, using the semantic representation of actors on the system level and the data they provide to optimize production plans with the help of semantic reasoning. The results of the exemplary realization and their relevance as proof of concept are summarized in 5 before we finally conclude in section 6, discuss open questions and provide an outlook on possible future work.

## 2 RELATED WORK

The idea of creating a semantic abstraction layer on top of an existing architecture involves a wide range of research domains and technologies. We will make use of semantic web concepts to describe data values and system interfaces, take some inspiration from ambient intelligent systems and employ multi agent systems to interact autonomously with the environment.

In the following, we provide background information to these topics and conclude with a brief overview over existing semantic layer approaches.

```

@prefix ex: <http://www.example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<ex:placeholder> a <rdfs:Resource>;
  <rdfs:label> "Example" .

```

Figure 2: Example for a very simple RDF graph in turtle syntax.

## 2.1 RDF, OWL and the Semantic Web

In order to include expert knowledge directly into our semantic layer, we will make use of *semantic web* (Lassila et al., 2001) technologies, namely *RDF graphs* (Manola et al., 2004) as data format, making use of *OWL ontologies* (Breitman et al., 2007; McGuinness et al., 2004) to define the concepts and relations relevant in a given environment, both well established W3C standards.

RDF allows us to make statements about things in the world in terms of triples, consisting of a subject, predicate and object. The place of the subject position in such a triple is taken by a *resource* with a *unique resource identifier* (URI). This resource does not only represent an entity in the abstract, digital world, but can directly relate to a physical thing in the real world, providing an address where to access information about this entity via the URI.

A RDF triple on its own does not necessarily express any knowledge yet and can be not much more than arbitrary strings, providing no additional valuable information aside from existing links between resources. Using appropriate ontologies by including their respective namespaces and asserting resources in our RDF graph to be members of classes defined there, we can include these semantics, modelled by domain experts, into our knowledge graph, and with that to interfaces and data values of a participant in our system. In order to define these ontologies, we make use of the *Web Ontology Language* (OWL), allowing us to define concepts like sub class relationships, class unions, exclusions, properties and so on.

## 2.2 Applications of the Semantic Web Idea

Thinking about systems consisting of actors with each actor provided with the capability to express knowledge about itself, making it in some sense intelligent, brings the idea of *Ambient intelligence* to mind, as Weber et al described it as *"the vision of a future filled with smart and interacting everyday objects offers a whole range of fascinating possibilities"* (Weber et al., 2005; Chung et al., 2020), or Dunne et al.

put it in a recent survey concerning the current state of ambient intelligence research: *"Aml can be synonymous with terms such as smart homes, smart environments, and intelligent environments. It is an umbrella term for a set of technologies that are embedded into the physical surroundings — seamlessly — to create an invisible user interface augmented with AI"* (Dunne et al., 2021).

Furthermore, encoding actual expert knowledge within the ambient intelligent environment by putting into effect principles of the semantic web is a widely regarded as worthwhile approach (Santofimia et al., 2011; Razzak et al., 2013; Ngankam et al., 2022; Padilla-Cuevas et al., 2021).

This gives rise to the need for flexibly maintainable, quick to set up and extend, systems, allowing for a seamless integration of both knowledge as such as well as actors to work on that knowledge.

While the semantic, ontology side of view is examined in detail, providing rather a theoretical concept of knowledge encoding, the practical application of these concepts to existing systems is often lacking.

Concepts like the *Semantic Web of Things* (Ruta et al., 2012) aim to bridge the gap between semantically annotated data on the one hand and a real system world on the other hand by introducing the idea of including knowledge about every participant via per device knowledge base generation, united in what Rute et al. refer to as *"ubiquitous knowledge base"*. The most practical approach we are aware of to actually putting the idea of the semantic web of things into action was proposed by Antoniazzi et al. (Antoniazzi and Viola, 2019)

Building on the W3C vision<sup>1</sup> of the Web of Things (Zeng et al., 2011), Charpenay investigated methods to semantically describe web things (Charpenay et al., 2016) and actual semantic data integration. (Charpenay et al., 2018). Semantic data integration is realized by an automatic transform step to JSON-LD, based on the keys of the original JSON object and the ontology to be used for annotation.

While this again shows the interest in and the need for semantic annotation not only on a device interface meta data level, but actual semantic data integration, for our vision this approach is still too limited in two regards: first, the focus on W3C WoT poses a restriction when it comes to system representation as the assumption that participants can be expressed in terms of sensors, actions and so on does not necessarily hold in every case. Furthermore, the key based JSON to JSON-LD approach does not provide the level of freedom we would like to realize when it comes to including actual human expert knowledge into the semantic

<sup>1</sup><https://www.w3.org/WoT/>

representation generated.

Vdovjak et al presented their vision of a *semantic layer* in (Vdovjak and Houben, 2002), however in our vision of enriching a system with semantic information, we aim for giving the user, in our case system maintainers and developers, more control over expressing their intentions, making the construction of the layer as such a more interactive process.

Lu and Ashgar suggested a semantic communication layer for cyber physical systems (Lu and Ashgar, 2020), proposing an architecture consisting of a *physical layer* representing the actual physical things, a *cyber layer* sensing the environment of a thing and planning future behavior, propagating messages through a *semantic layer* to enrich it with semantic information and finally transmit it through a *communication layer* to other cyber physical systems. While this work emphasizes the importance of considering semantic data layers in future applications, the approach focusses more on a centralized communication platform design and secure recommendation, rather than providing an actual permanently present knowledge based representation of a world possibly consisting of more than just cyber physical systems.

Entirely abstracting from any specific data formats, ontologies, middlewares or other implementations, Spieldenner described the idea of a *semantic medium*, a virtual, semantic reflection of a collection of systems, enhancing the options of multi agent systems to act within the system by exploiting the semantic information available, and making actual changes in the "real world" by directly linking outcomes in the semantic world to actual actions in the real world. (Spieldenner, 2023)

However, to the best of our knowledge, until today there still is a lack of actual implementations allowing actors to interact on the semantic level, exploit knowledge derivation and reasoning capabilities or allow for actual data exchange down to the system level or dynamic embedding of actual data values into the ubiquitous knowledge graph.

### 3 THE SEMANTIC SUPPORT POINT CONCEPT

We want the semantic support points to enable semantic integration and data exchange with the following properties:

**Distributed.** Rather than one monolithic middleware handling all data transform and transmission on one platform, we aim to provide *semantic support points* on top of the system layer, acting like a

source of semantic data to be discover- and queryable for participants acting on the semantic level.

**Decoupled.** Two separate semantic support points should always be decoupled from each other, in the sense that losing access to one support point does not influence the sanity of the other. Each support point should always be able to perform on its own, containing all necessary data to add its specific semantic information to the abstraction layer.

**Non-Invasive.** Adding semantic information to an existing system via an abstraction layer should be seen as an optional data offer for actors aiming for exploiting available semantic information, e.g. for reasoning to generate new knowledge, achieving interoperability by working on semantic interface abstractions or to have access of semantic data querying capabilities. To this end, semantic access points should be realized in such a way that they provide interfaces that existing systems are able to use (e.g. HTTP endpoints, event streams, message queues etc ...)

**Non-Persisting.** No state and no data should be stored in a specific support point. All data transformation and access should happen on demand, while necessary data transform rules are provided as necessary via *transform rule injection routes* into each semantic support point.

**Domain Independent.** The semantic support point implementation itself should be minimal and independent. All (semantic) information needed to describe the abstracted interface or data point should be provided via external sources. Domain independence along with the non-invasive, non-persisting and non-invasive properties ensure that the concept of semantic support points can be applied to any environment, in any domain, to an arbitrary extent, without the necessity of hosting or configuring a pre-implemented middleware beyond providing appropriate semantic transform rules to the semantic support points.

Any semantic support point implementation should be kept as minimal as possible, providing only the most basic, same features per support point. First of all, connection components that allow for data flow into the component as well as out of it. This dataflow is routed through a transform processing step, with transform rules provided via an external injection point. Receiving transform rules from outside sources ensures distributed and non-persisting properties of the support point, as no data is stored within any support point at any time and can be routed as needed. Further, it ensures the domain independence, as the

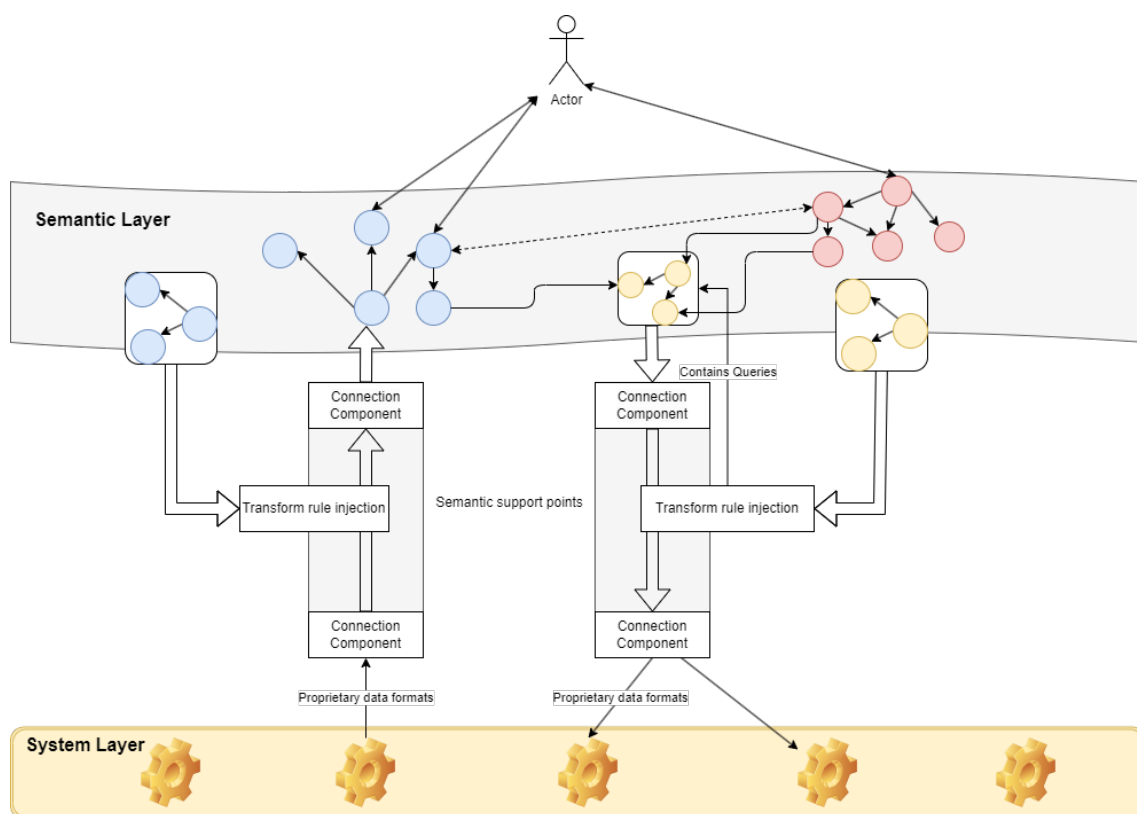


Figure 3: High level overview of the semantic support point concept.

domain solely depends on the transformation route, not on the support point implementation.

The connection point can implement any communication protocol necessary to connect the systems on the system layer it provides semantic support for as needed, be it HTTP, event driven systems, message buses or whatever else might exist out there.

This is true for both transformation directions. In case of a support point generating RDF from non-RDF sources, the outgoing connection component must provide interfaces for any actors that want to interact with it on a semantic manner. The semantic layer as such is non-persisting and provides data only as needed.

See figure 3 for a schematic overview of the semantic support point concept. Data is emitted in semantic format, interpretable as knowledge graph, to be provided to an actor intending to process semantically annotated data. This data processing may include operations like reasoning, combining information from several knowledge graphs, or linking concepts from a knowledge graph emitted by a semantic support point to a larger ontology (red knowledge graph in the diagram).

In order to propagate results of computations on the semantic level back to the system layer, again a

transform rule is used that may contain additional information on how to query the dataset provided as result of the actor's processes. The sub knowledge graph resulting from this querying step is then routed back through a support point and transformed to a data object according to the transformation rules provided via the injection route.

Notice that transformation rules, as they are written in RDF, can be understood as part of the virtual semantic layer and as such can provide additional information for actors acting on the semantic level, e.g. for service discovery and composition by exploiting information about the handling of certain data values contained in the transform file knowledge graph.

This concept of semantic support points fulfills the intended properties listed above: it is *distributed* in the sense that the minimal support point implementations can be hosted independent from each other where needed and where possible, *decoupled* in the sense that one support point crashing does not affect the runtime of any other support point, or any available transformation rules, *non-invasive* in the sense that existing interfaces do not need to be changed but can connect to support points to send and receive data as-is, *non-persisting* as no knowledge graphs that are results of semantic transformation are stored, and

as actors that interact with the semantic layer don't work on a persisted knowledge graph directly, but via the connection components of semantic support points and *domain independent* in the sense that domain is defined by the separately provided transformation rules, not by the support point implementations themselves.

## 4 USING SEMANTIC SUPPORT POINTS: AN EXAMPLE

In order to demonstrate how to combine the capabilities micro transformation service mapping data from and to RDF to a full abstraction layer we give an example making use of a semantic multi agent system.

Each agent maintains an individual view onto the world in its own knowledge base. This has multiple benefits compared to trying to render the knowledge of the entire world continuously and storing it in one central spot.

First of all it ensures encapsulation between multiple agents acting potentially simultaneously.

Second the we do not need to store a large amount of data representing the entire world at once, which would require not only a large data storage to do so, but also long computation times to process the queries to find relevant data. Instead, each agent can decide when and which data it needs. Imagine it as asking someone who has knowledge you need to explain it to you, or as only looking at the pages of a documentation that cover the points of interest for you.

### 4.1 Mapping Between Structured Data and RDF

In the following we provide some inspiration on how to realize the concept of a semantic support point in a real world application.

For mapping non-semantic data into a RDF knowledge graph, we entirely rely on freely available 3rd party libraries, while for the mapping back from a semantic representation to a explicitly specified interface, we propose our own solution.

#### 4.1.1 From Non-Semantic Data Formats RDF

In order to generate RDF graphs from different data formats, several approaches exist (Hildebrand et al., 2019; Méndez et al., 2020; Sahoo et al., 2009), however due to its wide acceptance and the benefit of being able to manage mapping files in a non-central, flexible distributed way, we aim for using *RML* (Dimou et al., 2014).

Mapping files written in RDF allow, in our opinion, the best trade off between complexity and the flexibility for system maintainers or service developers to freely express their intentions and their interpretation of the role of their system in terms of the generated RDF graph. Moreover, the possibility to store mapping files in a federated manner, making it easy to add mappings for new concepts, updated existing mappings to reflect changes in the environment and remove knowledge that is not longer needed at runtime support the idea of a seamless integration of new concepts, weaving our linked data fabric quilt patch by patch without having to interfere with the underlying physical world.

In order to provide RML mapping functionality to our system, we wrap *CARML* <sup>2</sup> into a micro service offering either an HTTP endpoint for direct calls or capabilities to connect to an event stream. Via what we call an *mapping file injection route* mapping files to be used for mapping the incoming data are received from a configurable remote endpoint, the resulting RDF graph can either be forwarded via HTTP to a pre configured external endpoint or written to an event based message bus.

#### 4.1.2 From RDF to Explicitly Specified Non-Semantic Data Formats

In order to access the semantic abstraction layer and propagating knowledge from there back to specific systems, we aim to use a similar approach as RML provides, but in the opposite direction.

While JSON-LD (Sporny et al., 2020) as out of the box available serialization format seems like an obvious choice at first glance, it does not provide the functionality we need. First of all, we are not interested in just generating a JSON representation from an entire RDF graph, which would be the case if we just take the JSON-LD representation of some given RDF, but we want to be able to select the information we are interested in. This requires at least enabling SPARQL queries as a preprocessing step. Also, we need more control over the structure of the generated JSON object, the key labels, value types and possible addition of literals not included in the RDF dataset.

Say we want to generate a simple JSON object like given in 4 as an example with the goal to propagate a measured room temperature to some fictional smart thermostate in order to decide whether the heater should switch off or continue heating. The measured room temperature was published by the temperature sensors to our semantic abstraction layer, however the information that this value now should be forwarded

---

<sup>2</sup><https://github.com/carml/carml>

```

{
  "id": "wekbnfi35r",
  "targetDevice": "thermostate",
  "temperature": 21
}

```

Figure 4: JSON example including a "type" not derivable from the RDF dataset.

to a thermostate is knowledge of the recipient system's developer, not something a third party would have published as semantic data.

This step of adding additional information in the process of making the semantically available data useful for the actual physical systems is something that can not be accomplished by just serializing existing RDF data to JSON-LD.

JSON schema (Pezoa et al., 2016) is not a feasible approach as it is cumbersome to describe larger objects and it lacks proper integration of semantic data. (Pezoa et al., 2016). Working on and with the semantic abstraction layer is supposed to happen in terms of expressing interpretations and intentions, instead of being a purely technical task. Therefore, mapping rules are provided and applied on a semantic level, analogously to RML mapping files.

While approaches on how to derive non-semantic data objects from RDF knowledge graphs are investigated (Allocca and Gougousis, 2015; Grassi et al., 2023), for our intentions they lack the possibility to enforce restrictions on the structure of the generated JSON or the exact definition of key names to use.

For this reason, we developed *POSER*<sup>3</sup> (Spieldenner, 2022) to provide data mapping from RDF graphs into a JSON object, following the structure given by transformation rules formulated in terms of a minimal JSON ontology. Briefly summarized, the intended outcome is described in terms of a nested hierarchy of resources representing JSON objects and datatypes, along with a datatype header defining queries and subgraph matches to extract them from a given knowledge graph and put them in relation to the JSON structure defined in terms of the JSON ontology.

These transformation rules can be injected into a semantic support point, analogously to RML mapping file in the inverse direction.

RDF data is pushed by or read from an actor on the semantic level side by the incoming connector component, filtered or queried according to the datatype header, written into a JSON object constructed according to the rules in the transformation file and then provided to systems on the system layer level via the outgoing connection component.

<sup>3</sup><https://github.com/spidan/poser>

## 4.2 Integration of a Semantic MAS

As an application example to serve as proof of concept, we show an integration of a semantic multi agent system into an automotive production system, aggregating data from heterogenous data sources, exploiting the semantic representation by performing reasoning operations and deriving optimized production plans that are again provided to different elements on the system level.

Examples for agent systems capable of working on semantic data are the semantic web-based MAS platform SEAGENT (Dikenelli et al., 2005) introduced in 2005, the system of Sabbatini et al. (Sabbatini et al., 2022), in which OWL and RDF knowledge graphs are used to train machine learning systems to realize learning agents in the Semantic Web or (Ciordea et al., 2018) in which a web-based MAS for manufacturing is introduced, interacting with Linked-Data and Web-of-Things environments to develop new behaviors from the semantic environment. However, as these systems are often designed with specific applications in mind tailored for specific applications and require adaptation to work fully exploit the benefits of semantic data sources, such as deduction of new facts via reasoning or logic operations on the semantic data set, we decided to use our own semantic Multi Agent System *AJAN* (Antakli et al., 2023).

For additional information on (MAS) and its capabilities, we'd like to refer you to previous works where it has been used in various Semantic Web and non-Semantic Web based environments. For example, in a smart living environment in which agents use the W3C Web of Things (WoT) architecture (see (Alberternst et al., 2021)), to optimize the production of a virtual factory floor using an *AJAN*-based MAS (see (Spieldenner and Antakli, 2022)), to coordinate language courses (see (Antakli et al., 2023)), or to control simulated human-robot collaboration scenarios (see (Antakli et al., 2019)).

## 4.3 Application Scenario

The EU-funded AIToC<sup>4</sup> project developed an assistance system (see Figure 5) for on-site workers and production planners in the automotive sector. A key component is the Operation Reasoner (see Figure 6), an agent-based planner using the MAS-framework *AJAN*<sup>5 6</sup> to generate assembly plans for visualization

<sup>4</sup>EU-Project AIToC: [aitoc.eu/](http://aitoc.eu/)

<sup>5</sup>*AJAN* on GitHub: [github.com/aantakli/AJAN-service](https://github.com/aantakli/AJAN-service)

<sup>6</sup>Operation Reasoner *AJAN*-model (ZIP-File) including RML and *POSER* mapping descriptions: [github.com/aantakli/AJAN-](https://github.com/aantakli/AJAN-)



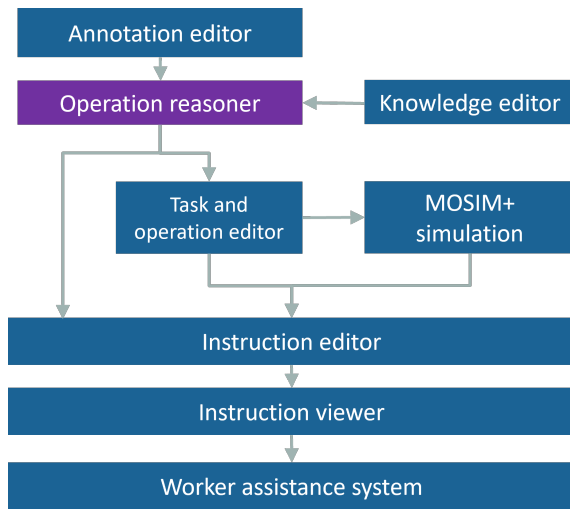


Figure 5: Assistance Pipeline.

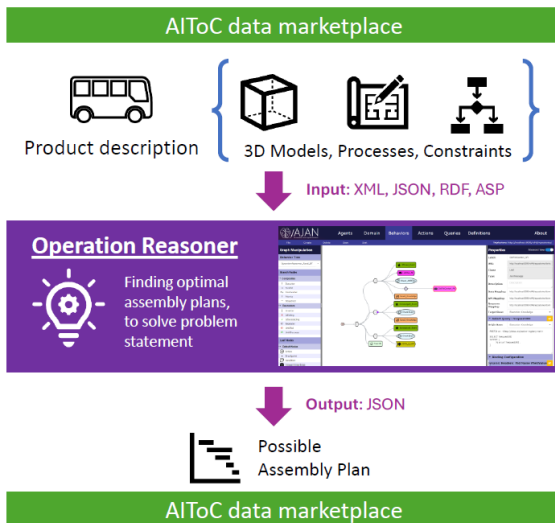


Figure 6: Operation Reasoner.

and verification.

The Operation Reasoner is a Web Service combining general process rules and specific product annotations to derive required assembly plans using the Answer Set Programming (ASP)(Vladimir, 2008) solver clingo<sup>7</sup>. Therefore, the Operation Reasoner uses previously defined product descriptions as an input for clingo and derives possible assembly plans from resulting stable models. The product descriptions are stored in the project data marketplace, the first point of exchange for the various services of the different project partners to exchange data with each other. These descriptions including annotated 3D models of

packages/blob/main/packages/Operation.Reasoner.ajan

<sup>7</sup>ASP Solver clingo: <https://potassco.org/clingo/>

assembly parts (originated from the Annotation editor) and process definitions and constraints (originated from the Knowledge editor), are defined in JSON respectively AutomationML (XML), RDF and ASP. For the use within the Operation Reasoner, product descriptions are translated into RDF, the data format internally used by our MAS, using RML.

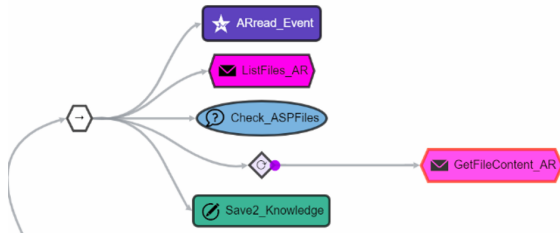
The fact that the required aggregated and homogenized planning problem is contained in the agent's knowledge enables the AJAN-based ASP-reasoning(Antakli et al., 2021) of instructions in the next step. In order to make the generated assembly plans available to other components and services within the project architecture, they must be translated back into JSON, generally used by services from the project service market place, and stored in the data marketplace. The Operation Reasoner uses the approach presented in 4.1.2 to translate the RDF-based data into the target data format. Based on the resulting possible plans, other services such as a motion synthesis based Simulation can intuitively display such assembly plans in a 3D simulated factory or via the Instruction Viewer to a real worker on site for assistance.

As an example of picking up data from the system layer, the processing of the semantic layer and the forwarding of the results to the system layer, we delve a bit deeper in showing the first step in the planning process of an Operation Reasoner agent: the reading of heterogeneous product description information.

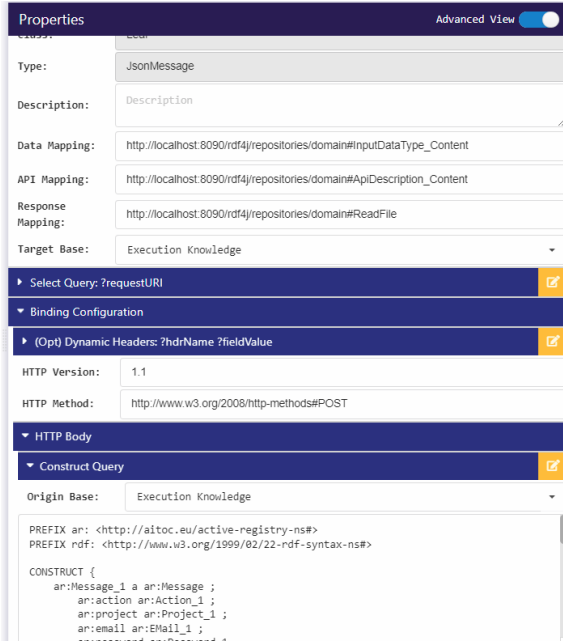
In Figure 7a, the BT to fulfill this task of collecting data from the real world of an Operation Reasoner agent in charge of, is presented. First, the agent receives an instruction (purple colored BT node) to generate optimal assembly plans, by pointing it to multiple data points from the system layer, which consists the needed information to perform the reasoning and planning task. The querying of this information is done by the pink colored BT nodes, where the *transform rule injection routes*, as defined in 4.1, are given via the node property fields: *Data Mapping*, *Endpoint Mapping* and *Response Mapping*. This is done in form of endpoints to read the respective mapping data from, as shown in Figure 7b. Via a *SPARQL Construct query*, the information relevant for an API call on system level, is collected from the agent's knowledge base, transformed to a proper JSON object matching the recipients API and performed as an HTTP POST request.

In order to ensure the correct data is provided to the JSON API, during the transformation step we provide a definition of datatypes we are interested in (see figure 8). An example of the JSON object to be generated is given in figure 9. For the sake of brevity, we





(a) Behavior Tree executing collecting data for plan generation and data forwarding.



(b) Properties for the Behavior Tree given above.

Figure 7: Behavior tree and its execution properties.

did not include the full API transform definition used by the transform engine. The response of this request is again received by the agent and mapped back to RDF via the RML service, allowing the actual assembly planning task in the next step.

## 5 RESULTS

By realizing the concept of semantic support points, we were able to efficiently abstract from data provided by heterogeneous systems, providing a semantic representation and relations inbetween data to be consumed by and worked on a semantic multi agent system. For example, worker instructions were generated by collecting data from different sources, like properties of tools and parts from 3D models, process descriptions in ASP and general properties of

```
: InputDataType_Content {
  json: EntryPoint a ar: Message;
  ar: action ar: Action;
  ar: project ar: Project;
  ar: file ar: File;
  ar: email ar: EMail;
  ar: password ar: Password .
```

```
ar: Action ar: value iots: Number .
ar: Project ar: value iots: Number .
ar: File ar: value iots: Number .
ar: EMail ar: value iots: Number .
ar: Password ar: value iots: Number .
}
```

Figure 8: Data types to be used to determine the specific values in a generated JSON object.

```
{
  "action": "getFileAsJSON",
  "project": 17,
  "file": "context_information.lp",
  "email": "X.X@example.com",
  "password": "asfkalnkknakfsnfjb55"
}
```

Figure 9: Example of JSON object to be generated.

the environment given in JSON, transformed to RDF, aggregated, used in a planning and reasoning engine and finally the results being returned to different systems like animation synthesis or plan generation UI in JSON (see figure 10 for an UI example).

Notice how the interplay between RML and JSON mappings, along with the concept and importance of the on-demand semantic layer, becomes especially apparent here. We cannot rely on the information gathered in the first step being available in one specific data format. Instead, it is provided by a plethora of heterogeneous sources: some in JSON or XML format, some consisting of ASP rules, and others directly providing RDF data. This diversity makes a certain level of interoperability necessary.

This is, in this case, achieved by exploiting the semantic interoperability capabilities of the semantic layer, making the data available in an aligned format as soon as the agent requests it.

Further more, in order to perform the reasoning tasks for plan generation, we are required to have data available in RDF format, with the results being provided in RDF again. This constitutes processes on a purely semantic level with no direct relation to interfaces in the physical world, making a concise, configurable and query enriched transform step from RDF to JSON necessary.

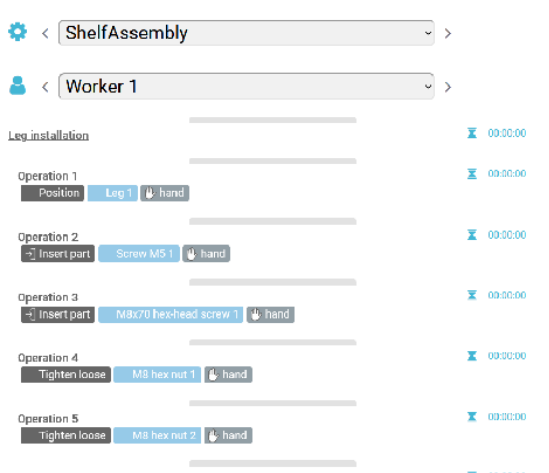


Figure 10: UI to model instructions for workers, pre-configured by results of reasoning on agent level converted back to JSON.

This demonstrates the idea behind and the power of the semantic layer by moving beyond simple interoperable data exchange in an IoT environment, which is already a complex task. By combining raw data with expert knowledge about data interpretation (the data transform step), we fully exploit the possibilities of this approach. Additionally, we incorporate actual intention.

## 6 CONCLUSION AND DISCUSSION

We have introduced the concept of federated semantic support points to abstract from interfaces in a heterogeneous, non-semantic environment, allowing for a unified view on the underlying system world. Minimal stateless semantic transformation services that represent these support points are used to generate a RDF graph from non-RDF data, using established mapping technologies, and again propagating the results of actions performed on the data fabric back to the system layer by querying and transforming data according to semantic descriptions of both structure and data types of the consuming interfaces.

The data fabric created by the semantic support points is non-persistent, generated on demand by actors wishing to consuming the semantic data representation on request. Data is encoded on the fly from data emitted by the system layer and results are immediately propagated back to the system layer to be consumed by recipients. Transformation rules are stored and managed separately from actual semantic support point implementations, making our approach highly

flexible and domain independent.

In contrast to semantic interoperability approaches common in the dataspace or semantic web of things world, not only the participants and entire data objects are semantically described in terms of static meta data, but each data value itself can be embedded into the semantic fabric at runtime.

As a proof of concept we demonstrated on-demand data aggregation and using a multi agent system capable of working on semantic linked data to derive instruction suggestions for workers in an automotive factory. Based on data like product descriptions, 3D models, pre-defined processes and constraints we generated a unified semantic view enriched with expert knowledge to derive optimized production plans that could not only be immediately consumed and virtually be tested by a semantic multi agent system, but also rendered to a human readable representation for actual factory workers.

For this proof of concept, we decided to follow a plug in like approach, extending the agent system itself with the capabilities to generate a semantic view on the world and propagate changes made by the agents back to the system layer. However, extracting the actual support point functionality to fully cover the concept presented in section 3 would only require minor code changes and hosting of the support points as separate services. In our case it was more efficient to realize a plugin approach, as the MAS was the only actor working on the semantic layer. We consider this as another proof of flexibility of the approach when it comes to actual implementation details.

We are highly interested in exploring the possibilities provided by mapping files that are already written in RDF and with that contain semantic information and expert knowledge when it comes to service discovery and composition. In addition with, possibly distributed, service registries containing meta data based information about available services, these semantic instructions given in the mapping files, seem like a promising way to generate another semantic meta data layer on top of the actual linked data fabric, allowing for knowledge based service discovery and composition, strengthening the idea of an intention driven data exchange even more, moving towards a system providing a certain level of *pragmatic interoperability* (Neiva et al., 2016)

Further work is necessary in terms of runtime behavior and resources needed for time or resource critical applications. To our anecdotal findings, the mapping instructions provided are the main impact on resources needed in terms of triple stores while resources needed to handle the semantic representations generated at runtime are proportional to the size of the

non-semantic data payloads emitted and consumed on system level. As this dynamic data is discarded as soon as a certain process ends, the data fabric itself only creates negligible overhead.

However, for our use cases these considerations were not relevant at this point and will be subject to future work.

## ACKNOWLEDGEMENTS

This work has been supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) in the projects *ForeSightNEXT* (01MK23001G) and *TwinMap* (13IK028J).

## REFERENCES

- Alberternst, S., Anisimov, A., Antakli, A., Duppe, B., Hoffmann, H., Meiser, M., Muaz, M., Spieldenner, D., and Zinnikus, I. (2021). From things into clouds—and back. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CC-Grid)*, pages 668–675. IEEE.
- Allocca, C. and Gougousis, A. (2015). A preliminary investigation of reversing rml: From an rdf dataset to its column-based data source. *Biodiversity data journal*, (3).
- Antakli, A., Kazimov, A., Spieldenner, D., Rojas, G. E. J., Zinnikus, I., and Klusch, M. (2023). Ajan: An engineering framework for semantic web-enabled agents and multi-agent systems. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 15–27. Springer.
- Antakli, A., Spieldenner, T., Rubinstein, D., Spieldenner, D., Herrmann, E., Sprenger, J., and Zinnikus, I. (2019). Agent-based web supported simulation of human-robot collaboration. In *Proc. of the 15th Int. Conf. on Web Information Systems and Technologies (WEBIST)*, pages 88–99.
- Antakli, A., Vozniak, I., Lipp, N., Klusch, M., and Müller, C. (2021). Hail: Modular agent-based pedestrian imitation learning. In *Proc. 19th Int. Conf. on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*. Springer.
- Antoniazzi, F. and Viola, F. (2019). Building the semantic web of things through a dynamic ontology. *IEEE Internet of Things Journal*, 6(6):10560–10579.
- Breitman, K. K., Casanova, M. A., and Truszkowski, W. (2007). Ontology in computer science. *Semantic Web: Concepts, Technologies and Applications*, pages 17–34.
- Charpenay, V., Käbisch, S., and Kosch, H. (2016). Introducing thing descriptions and interactions: An ontology for the web of things. In *SR+ SWIT@ ISWC*, pages 55–66.
- Charpenay, V., Käbisch, S., and Kosch, H. (2018). Semantic data integration on the web of things. In *Proceedings of the 8th International Conference on the Internet of Things*, pages 1–8.
- Chung, E., Lefrançois, M., and Boissier, O. (2020). Increasing interoperability in the web of things using autonomous agents (position paper). In *18e Rencontres des Jeunes Chercheurs en Intelligence Artificielle 2020 (RJCIA)*.
- Ciorteza, A., Mayer, S., and Michahelles, F. (2018). Repurposing manufacturing lines on the fly with multi-agent systems for the web of things. In *AAMAS*, pages 813–822.
- Curry, E. (2020). *Real-time linked dataspace: Enabling data ecosystems for intelligent systems*. Springer Nature.
- Dikenelli, O., Erdur, R. C., and Gumus, O. (2005). Seagent: a platform for developing semantic web based multi agent systems. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1271–1272.
- Dimou, A., Vander Sande, M., Slepicka, J., Szekely, P., Mannens, E., Knoblock, C., and Van de Walle, R. (2014). Mapping hierarchical sources into rdf using the rml mapping language. In *2014 IEEE International Conference on Semantic Computing*, pages 151–158. IEEE.
- Dunne, R., Morris, T., and Harper, S. (2021). A survey of ambient intelligence. *ACM Computing Surveys (CSUR)*, 54(4):1–27.
- Franklin, M., Halevy, A., and Maier, D. (2005). From databases to dataspace: a new abstraction for information management. *ACM Sigmod Record*, 34(4):27–33.
- Grassi, M., Scrocca, M., Carenini, A., Comerio, M., and Celino, I. (2023). Composable semantic data transformation pipelines with chimera.
- Halevy, A., Franklin, M., and Maier, D. (2006). Principles of dataspace systems. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–9.
- Heiler, S. (1995). Semantic interoperability. *ACM Computing Surveys (CSUR)*, 27(2):271–273.
- Hildebrand, M., Tourkogiorgis, I., Psarommatis, F., Arena, D., and Kiritsis, D. (2019). A method for converting current data to rdf in the era of industry 4.0. In *Advances in Production Management Systems. Production Management for the Factory of the Future: IFIP WG 5.7 International Conference, APMS 2019, Austin, TX, USA, September 1–5, 2019, Proceedings, Part I*, pages 307–314. Springer.
- Jafari, M., Kavousi-Fard, A., Chen, T., and Karimi, M. (2023). A review on digital twin technology in smart grid, transportation system and smart city: Challenges and future. *IEEE Access*, 11:17471–17484.
- Khan, H. H., Malik, M. N., Zafar, R., Goni, F. A., Chofreh, A. G., Klemeš, J. J., and Alotaibi, Y. (2020). Challenges for sustainable smart city development: A conceptual framework. *Sustainable Development*, 28(5):1507–1518.

- Khanna, A. and Kaur, S. (2020). Internet of things (iot), applications and challenges: a comprehensive review. *Wireless Personal Communications*, 114:1687–1762.
- Lan, Q., Wen, D., Zhang, Z., Zeng, Q., Chen, X., Popovski, P., and Huang, K. (2021). What is semantic communication? a view on conveying meaning in the era of machine intelligence. *Journal of Communications and Information Networks*, 6(4):336–371.
- Lassila, O., Hendler, J., and Berners-Lee, T. (2001). The semantic web. *Scientific American*, 284(5):34–43.
- Lee, I. and Lee, K. (2015). The internet of things (iot): Applications, investments, and challenges for enterprises. *Business horizons*, 58(4):431–440.
- Lu, Y. and Asghar, M. R. (2020). Semantic communications between distributed cyber-physical systems towards collaborative automation for smart manufacturing. *Journal of manufacturing systems*, 55:348–359.
- Manola, F., Miller, E., McBride, B., et al. (2004). Rdf primer. *W3C recommendation*, 10(1-107):6.
- McGuinness, D. L., Van Harmelen, F., et al. (2004). Owl web ontology language overview. *W3C recommendation*, 10(10):2004.
- Méndez, S. J. R., Haller, A., Omran, P. G., Wright, J., and Taylor, K. (2020). J2rm: An ontology-based json-to-rdf mapping tool. In *ISWC (Demos/Industry)*, pages 368–373.
- Neiva, F. W., David, J. M. N., Braga, R., and Campos, F. (2016). Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature. *Information and Software Technology*, 72:137–150.
- Ngankam, H. K., Pigot, H., and Giroux, S. (2022). Ontodomus: a semantic model for ambient assisted living system based on smart homes. *Electronics*, 11(7):1143.
- Nižetić, S., Šolić, P., Gonzalez-De, D. L.-d.-I., Patrono, L., et al. (2020). Internet of things (iot): Opportunities, issues and challenges towards a smart and sustainable future. *Journal of cleaner production*, 274:122877.
- Padilla-Cuevas, J., Reyes-Ortiz, J. A., and Bravo, M. (2021). Ontology-based context event representation, reasoning, and enhancing in academic environments. *Future Internet*, 13(6):151.
- Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M., and Vrgoč, D. (2016). Foundations of json schema. In *Proceedings of the 25th international conference on World Wide Web*, pages 263–273.
- Qin, Z., Tao, X., Lu, J., Tong, W., and Li, G. Y. (2021). Semantic communications: Principles and challenges. *arXiv preprint arXiv:2201.01389*.
- Razzak, F. et al. (2013). The role of semantic web technologies in smart environments.
- Ruta, M., Scioscia, F., and Di Sciascio, E. (2012). Enabling the semantic web of things: framework and architecture. In *2012 IEEE Sixth International Conference on Semantic Computing*, pages 345–347. IEEE.
- Sabbatini, F., Ciatto, G., and Omicini, A. (2022). Semantic web-based interoperability for intelligent agents with psyche. In *Explainable and Transparent AI and Multi-Agent Systems: 4th Inter. Workshop (EXTRAAMAS 2022)*, pages 124–142. Springer.
- Sahoo, S. S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., Sequeda, J., and Ezzat, A. (2009). A survey of current approaches for mapping of relational databases to rdf. *W3C RDB2RDF Incubator Group Report*, 1:113–130.
- Santofimia, M. J., Fahlman, S. E., Del Toro, X., Moya, F., and Lopez, J. C. (2011). A semantic model for actions and events in ambient intelligence. *Engineering Applications of Artificial Intelligence*, 24(8):1432–1445.
- Solmaz, G., Cirillo, F., Fürst, J., Jacobs, T., Bauer, M., Kovacs, E., Santana, J. R., and Sánchez, L. (2022). Enabling data spaces: Existing developments and challenges. In *Proceedings of the 1st International Workshop on Data Economy*, pages 42–48.
- Spieldenner, D. (2022). Poser: A semantic payload lowering service. In *WEBIST*, pages 249–256.
- Spieldenner, T. (2023). Linked data as medium for distributed multi-agent systems.
- Spieldenner, T. and Antakli, A. (2022). Behavior trees as executable representation of milner calculus notations. In *2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE.
- Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., and Lindström, N. (2020). Json-ld 1.1. *W3C Recommendation, Jul*.
- Vdovjak, R. and Houben, G.-J. (2002). Providing the semantic layer for wis design. In *Advanced Information Systems Engineering: 14th International Conference, CAiSE 2002 Toronto, Canada, May 27–31, 2002 Proceedings 14*, pages 584–599. Springer.
- Vladimir, L. (2008). What is answer set programming. In *AAAI*, volume 8.
- Wang, J., Yang, Y., Wang, T., Sherratt, R. S., and Zhang, J. (2020). Big data service architecture: a survey. *Journal of Internet Technology*, 21(2):393–405.
- Weber, W., Rabaey, J., and Aarts, E. H. (2005). *Ambient intelligence*. Springer Science & Business Media.
- Zeng, D., Guo, S., and Cheng, Z. (2011). The web of things: A survey. *J. Commun.*, 6(6):424–438.
- Zillner, S., Gomez, J. A., García Robles, A., Hahn, T., Le Bars, L., Petkovic, M., and Curry, E. (2021). Data economy 2.0: From big data value to ai value and a european data space. In *The elements of big data value: Foundations of the research and innovation ecosystem*, pages 379–399. Springer International Publishing Cham.