ISSN 2277-2685
IJESR/JULY 2023/ Vol-13/Issue-3/1-9
Mrs. B KEERTHI / International Journal of Engineering & Science Research

Methods for Matching Semantic Things with Relevant Services in Industrial IoT Settings: A Critical Review

Mrs. B KEERTHI, Mrs. G MANJULA, Mrs. T REDDI RANI,
Assistant Professor[1,2] , Associate Professor[3]
Department of ECE,

Viswam Engineering College (VISM) Madanapalle-517325 Chittoor District, Andhra Pradesh, India

Abstract

Industrial Internet of Things Platforms enable theuse of available data to improve production and business processes.However, the data exchange and provisioning between thedata sources and platform services remain a challenge as suchplatforms are usually vendor specific, proprietary, and associatedwith specific IoT hardware. Therefore, we propose a detaileddescription of an open-source software-based solution, Thing toService Matching (TSMatch) which performs fine-grained semanticmatching between available IoT data and services. Moreover,the paper presents the actual implementation of the proposedsolution in 2 different Aerospace production environments, anda performance evaluation in a testbed environment.

Index Terms—IIoT data sources, semantic matching, use cases,IIoT Application

I. INTRODUCTION

Industrial Internet of Things (IoT) platforms bring in multiplebenefits, being one of them the capability to makeuse of data available in industrial environments to improveproduction and business processes. However, the processingand application of the collected data remains a challenge. IoTplatforms rely on IoT infrastructures with a high number ofsensors and actuators. The setting up of these infrastructuresand the adequate support by an IoT platform is not trivial,as such platforms are usually vendor specific, proprietary, andassociated with specific IoT hardware or specific cyber physicalsystems. The data is then carried to the Cloud, bringing inadditional costs and the need for additional, specialized manualintervention, making the whole process time consuming, proneto errors, and/or costly [1].

The IoT landscape today is fragmented due to the vendorbasedapproach; therefore, interoperability is a key aspectto tackle in regards to IoT, be it from a communicationperspective or from an application perspective. This workfocuses on the second challenge, and aims at contributing witha solution to assist interoperability and data exchange betweenIoT Things (data sources) and services provided by an IoTplatform in an automated way.

We propose an open-source software-based solution, Thingto Service Matching (TSMatch). TSMatch performs finegrainedsemantic matching between collected IoT data andservices. The goal of TSMatch is to ease the process ofintegrating existing IoT infrastructures with external servicesby automating the matching and provisioning process. Thepaper contributions are (i) providing a description of the opensourceTSMatch1 engine, (ii) demonstrating the feasibilityof deploying TSMatch in manufacturing scenarios based onrealistic operation conditions, and (iii) evaluating TSMatch inan experimental IIoT environment (testbed).

The paper is organized as follows. After this introduction,related work is presented in section II while section III introducesthe main software components of TSMatch. A detaileddescription of the implementation is provided in section IV.The paper also demonstrates the feasibility of deployment inproduction scenarios and integration with other IoT platformssuch as EFPF2 as demonstrated in section V. TSMatch isthen evaluated in an experimental environment, in regards toadded processing delay, and time to completion aspects. Theresults and discussions are presented in section VI. Finally, aconclusion and an outline of the future work are presented insection VII.

II. RELATED WORK

Several approaches are used in service matching, rangingfrom logical [2] to non-logical [3] semantic based techniquesor a hybrid (combining logical and non-logical techniques)approaches [4]. For example, Kovacs et al. describe thetechnical approach and the system architecture,for realizinga world-wide semantic interoperability solution combiningthe FIWARE NGSI and oneM2M context interfaces. Thoughthe approach is promising, this is a conceptual paper, whichdoes not address the application of the concept in realisticor in experimentalenvironments [5]. Moreover, Cassar et al.propose a matchmaking solution based on both a semanticand probabilistic matchmaking. The

*Corresponding Author

proposed approach wasevaluated by calculating its precision and the NormalisedDiscounted Cumulative Gain, considering an OWL-S datasetconsisting of 1007 Web service descriptions. Even though theproposed approach exhibits higher performance than existingmethods, the concept has beenevaluated based on simulations[6].
In prior work, we have briefly presented the methodologyand the TSMatch concept. The current work builds on ourprior work and addresses the specification and implementationas well as validation of TSMatch in realistic environments [7].

III. THING TO SERVICE MATCHING ARCHITECTURE

TSMatch is a software-based solution that supports thesemantic matchmaking of IoT data to services. The maingoal of TSMatch is to automate the data supply between IoTdata sources and services, while satisfying the service needs.For that, TSMatch currently follows a semantic similaritymatchmaking approach. Figure 1 illustrates the key actors inTSMatch. IoT Things are represented, standing for a semanticrepresentation of different sensors. the TSMatch Engine iscurrently a server-side component of TSMatch that can reside,for instance, on an Edge server, on an IoT gateway, or even ona Cloud server. The Thing registry corresponds to a databasesystem where Things descriptions are periodically stored. TheTSMatch Client represents the end-user application, that canrun in any end-user system, e.g., an Android phone. The usercorresponds to the end-user of an IoT service, i.e., a person,an organization or even a specific software.

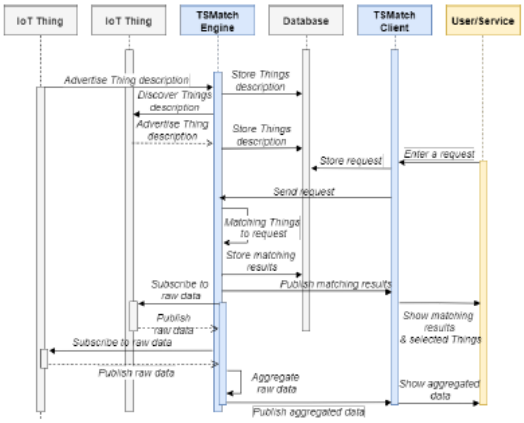

Fig. 1. Sequence diagram of the various actors involved in TSMatch.

As illustrated in Figure 1, a first process relates with theIoT Things advertising their descriptions when powered on.TSMatch Engine subscribes to Things events and registerstheir descriptions in the Things registry. This process isexternal to the TSMatch semantic matchmaking engine andin our case is supported via Coaty3

The TSMatch Engine can also request IoT Things descriptionsdirectly from a mediating entity such as a broker. In otherwords, it can be configured as a subscriber. Via a separateprocess, a user performs a request for a service, e.g., measuretemperature on a specific room using the TSMatch clientwhich forwards the request to the TSMatch Engine. Basedon the request and the available IoT Things descriptions,the TSMatch selects from the registry a specific set of IoTThings to consider to answer the service, based on similaritymatching. Upon success, the results of the matching areforwarded to the TSMatch Client.

IV. TSMATCH IMPLEMENTATION ASPECTS

This section describes the ongoing TSMatch implementationof each component and of the respective interfaces.TSMatch is developed based on a micro-service architectureand therefore all of its components are currently dockerized.On this specific implementation instantiation, the TSMatchengine is deployed on an IoT gateway, while the client resideson an Android platform. The communication across the IoTinfrastructure and TSMatch components is based on Wi-Fi.We describe next each component.

A. IoT Things

The IoT Things have been modelled based on the OGCSensor Thing API [8]). The Thing description contains informationsuch as the sensors properties and location. Advertisementis performed via Coaty, application-layer communicationsoftware. Coaty provides a way for IoT Things to advertisetheir semantic descriptions via multicast, notify subscriberswhen the IoT Thing is no longer available, by sending adeadvertized event. When a discover event is sent by theTSMatch Engine to discover available Things, a response issent containing the Thing description.

IoT Things correspond to semantic descriptions of cyber-physicalsystems based on heterogeneous hardware, e.g., asensor, a single-board computer (SBC), programmable logiccontroller (PLC) connected with sensors via its Input/outputinterface. In terms of software, the IoT cyber-physical systemis composed of the sensor driver and a Coaty agent tomanage the IoT Thing functionalities, such as advertising itsdescription.

B. TSMatch Engine

The TSMatch engine corresponds to the server-side implementationof the semantic matchmaking between IoT Thingsdescriptions and a service descriptions. The engine handlesservice requests transmitted via the TSMatch Client. It alsohandles queries to the TSMatch Thing registry and thenhandles the matchmaking between the properties/attributes ofthe Things descriptions to the semantic description of services.The matchmaking is based on semantic similarity based on theSørensen–dice coefficient and term frequency–inverse documentfrequency. Based on the matching, a set of availableThings are selected, then grouped and an aggregated objectrepresenting this grouping is again stored in the database. Thematching result is then sent to the TSMatch Client to be shownto the user both in case a match was found or not. When amatch is found, the engine triggers an event which starts asubscription to the observations of the selected sensors andcalculates the mean value of the observation based on theselected location. The updates are then sent to the TSMatchClient. If a request is deleted, TSMatch unsubscribes from theIoT Things observations (from a broker) and stops sendingupdates to the TSMatch client

The TSMatch Engine has been implemented in Node.js, Javascript runtime environment and Typescript language5. Forthe implementation of Sørensen–dice similarity we have reliedon the Nebulous Plasma Muffin npm-string-similarity packageSand on the natural packages.

C. TSMatch Client

The TSMatch Client handles the service semantic description,either by creating one based on specific requests from theuser (e.g., monitor temperature) or by getting remote servicedescriptions. We have implemented the TSMatch Client as amobile application developed using the React Native Javascriptframework8. The client currently runs on Android 4.1 andabove versions. When the TSMatch client starts, it performsa subscription to an MQTT broker. It allows the user to seea list of available Things and their descriptions, to describethe requests, and to visualize the data updates based on theprovided request.

D. Things Registry, Discovery, and Communication Aspects

The registration, announcement, discovery, and communicationfrom IoT Things to the end-user (subscriber) has beensupported by Coaty v2.0. The communication is supportedby an MQTT broker, based on Mosquitto v2.0.11. The IoTdescriptions are stored in a local registry based on PostgreSQL13.2 database to store data as JSONB data type for binary storageand retrieval of JSON objects. The Official PostgreSQLDocker image9 has been used.

V. TSMATCH APPLIED TO EFPF USE-CASES

The European Connected Factory Platform for Agile Manufacturing(EFPF) ecosystem is a federated digital platformthat offers innovation solutions from Industry 4.0, IoT, AI, BigData and Digital Manufacturing domains. At the core of EFPFis an interoperable 'Data Spine' that provide open interfacesto support the integration of distributed systems and platformswith their toolset and services. The individual componentssupported in EFPF are provided by several partners andcommunicate through the central Data Spine. Therefore, theEFPF ecosystem follows a Service-oriented architecture (SOA)style. The main elements in the EFPF federation are the DataSpine, the EFPF Web-based platform (which provides unifiedaccess to various tools and services through a Web-basedportal), the base digital platforms (four base platforms fundedby the European Commission's Horizon 2020), and externalplatforms (platforms connected to the EFPF ecosystem thataddresses the specific needs of connected smart factories).

TSMatch is currently one of the components integrated inEFPF. Its integration has been tested in production scenarios,together with the IoT automation platform SymphonyFactory Edition (one of the External Platforms within theEFPF federation) provided by the EFPF partner Nextworks.Symphony is a complete IoT Platform characterized by amodular architecture supporting a wide range of heterogeneoushardware, e.g., IoT sensors and actuators.

TSMatch has been used for data exchange and provisioningbetween available IoT data in the production environments andthe IoT Symphony platform, while utilizing the EFPF DataSpine. In EFPF, TSMatch has been integrated and deployed inthree different aerospace manufacturing use-cases, which wererederived from requirements obtained from Walter Otto MfullerGmbH & Co.KG11 (WOM) and Innovint Aircraft InteriorGmbH(IAI). The use-cases are: 1) Control of temperatureand humidity in a manufacturing area to ensure consistentquality and environmental conditions required for componenttolerances (WOM); 2) Continuous survey of raw materialstored in a freezer to avoid scrapping in case of excessivelyhigh temperature (IAI); 3) Remote monitoring of a vacuumforming machine to enable immediate actions when pressurevalues fall out of tolerance (IAI). In all three use-cases, thegoal has been to secure the stability and quality of manufacturingprocesses by monitoring the relevant parametersand providing alarms in case defined thresholds are underrunor exceeded. The environmental parameters to be monitoredare measured through IoT Things and shared to the externalplatform Symphony via TSMatch.

As illustrated in Figure 2, the TSMatch Client is employedby the Symphony Factory Connector13 using service requestsfor IoT Things. A Cloud instance of the Symphony IoTautomation platform ingests the information provided by TSMatchthrough the EFPF Data Spine which offers serviceswith interoperable security features. The visual monitoring,sensor data and event storage, signal analysis, and alarmsystems associated with the Things are provided by the Symphonyinternal modules, in particular: sensor data has beenremotely collected through the Symphony HAL (a softwaremodule that primarily abstracts the low-level details of variousheterogeneous fieldbus technologies and provides a commoninterface to its users) and stored by the Symphony DataStorage. The Symphony Event Reactor handles events andalarms through combining information coming from differentsources and data brokers (e.g., TSMatch) to determine actionsto be taken including control actions on field-level devices,notifications (e-mails and SMS), and alarms (via stack lightwhich provides visual and audible indications). The real-timedata, along with the status of the thresholds and the alarms, canbe also visualized and handled through the Symphony GUI inthe Cloud instance of the platform.

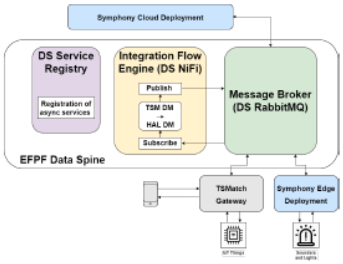

Fig. 2. EFPF interconnected components on the manufacturing use-cases.

As illustrated in Figure 3, a successful deployment has beendone on production sites. The EFPF components, includingTSMatch, have therefore been validated based on the definedrequirements, usability aspects, as well as ease of installation/configuration. After 6 months of deployment, traceabilityand detection of abnormalities has been also achieved, whichincreased the reliability of the manufacturing process, reduceddelivery delays, and minimized rejects/waste occurrences.

Fig. 3. Installation of the use-cases.

VI. PERFORMANCE EVALUATION AND RESULTS

A. TSMatch Testbed

Fig. 4. The TSMatch demonstrator at the fortiss IIoT Lab.

Based on the operational requirements derived from theTSMatch integration on realistic manufacturing use-cases, wehave set up a TSMatch testbed illustrated in Fig. 4 on thefortissIIoT Lab. The testbed is composed of the followingcomponents, as illustrated in Figure 5:

- 2 IoT Things and 10 virtual IoT Things: Each IoT Thingis connected to 5 sensors: temperature, humidity, sound,air quality, and particulate matter sensors. Each virtualIoT Thing is connected to virtual IoT sensor, running inseparate Docker container as specified in section IV.

- TSMatch Engine, message Broker, and database: TSMatchEngine is containerized, while the Mosquittomessage bus and PostgreSQL database are deployed ascontainers in the fortiss IoT gateway .

- IoT service request simulator: It aims at simulating externalIoT services by sending service requests to TSMatchEngine using an MQTT client15. The service request wasused instead of the TSMatch Client to enable control overthe time interval range between the service requests sent

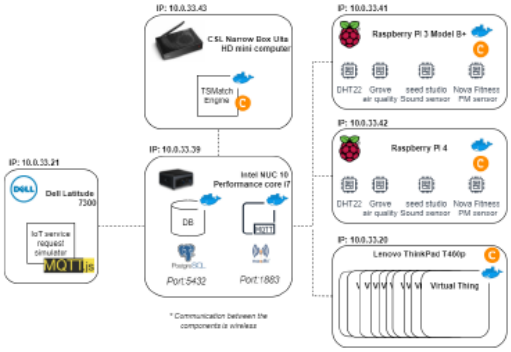

Fig. 5. TSMatch components and experimental interconnections.

The hardware specifications for each of the devices that ispart of the demonstrator are given in Table I.

B. Performance Evaluation Parameters

The evaluation performance focused on assessing latency,based on two delay definitions: i) processing time (PT) and ii)Time to Completion (TTC).

PT refers to the time required by the TSMatch Engine toprocess a request and complete a matching process. Therefore,the processing time is measured on the application layer,based on the difference between the instant when the TSMatchEngine receives a service request, and the instant when the TSMatchEngine generates a matching result. PT comprises thetime required to query IoT descriptions on the local database;aggregate the data derived from multiple IoT observations.

TTC refers to the required time to complete a servicerequest. It is measured on the application layer as the timeinterval (seconds) between the instant when a request issent by the service, until the instant when a first update (aJSON file containing the aggregated observation data whichhas around 365 bytes in size) is received by the service

TABLE I
HARDWARE SPECIFICATIONS OF THE DEMONSTRATOR.

Device	Model	CPU	Memory
IoT Thing 1	Raspberry Pi 3B+	BCM2837B0	1GB SDRAM 32GB Micro SD
IoT Thing 2	Raspberry Pi 4	BCM2711	4GB SDRAM 32GB Micro SD
Sound Sensor	Grove sound sensor v1.6	-	-
Temperature Humidity Sensor	AM2302 - DHT22		
Air Quality Sensor	Grove air quality v1.3	-	-
Particle Sensor	Nova PM SDS011	-	-
10 Virtual IoT Things	Lenovo Thinkpad T460p	i7-6700HQ @2.60 GHz	16 GiB, Samsung SSD 860 500GB
TSMatch	CSL Narrow Box Mini Computer	Intel Celeron CPU J3455 @1.50GHz	4 GiB
fortiss IoT Gateway	Intel NUC NUC10i7FNH	Intel Core i7-10710U @1.10GHz	16GiB
Service Request Simulator	Lenovo Thinkpad T460p	i7-6700HQ @2.60 GHz	16 GiB, Samsung SSD 860 500GB

C. Scenarios

To get an estimate on the number of services connected toeach IoT environment, we analysed the various pilot scenariosimplemented using EFPF platform. In the pilot scenarios,7 SMEs from three domains i.e. aerospace, furniture, andcircular economy provided various requirements based onwhich a set of services were selected and integrated [9], [10],[11]. After analysing the various use cases, we found out thatthe maximum number of services, which consume IoT data,used is 5 and estimated that each service would send between3 to 5 requests, resulting in a maximum of 25 requests perIoT environment. This estimate was also confirmed by thepilots. Therefore, in the evaluation 25 requests will be sentsequentially and simultaneously.

Simultaneous scenario:

emulates an IoT environment connectedto many services. In this scenario, 25 clients send 1service request each, resulting in ”25 simultaneous requests”as illustrated in Table II.

TABLE II
EVALUATION SCENARIOS.

Scenarios/TI range	Sequential Scenario		Simultaneous Scenario
	Fixed TI (s)	*Variable TI (s)*	*25requests*
High	120	[50-120]	N.A
Medium	25	[25-50]	N.A
Low	0.015	[0.015-0.03]	N.A

D. Results and Discussion

1) **Sequential Scenario Results**: We assessed the TSMatchperformance first based on the sequential scenario, having asbaseline for comparison the performance of Coaty. Figure 6provides results for fixed service interarrival times: the topchartprovides results for PT and TTC for both TSMatch andCoaty with small time intervals; the bottom-one for large timeintervals between service requests.

Based on the first diagram, we can see that for fixed lowtime interval the PT for TSMatch increases on average by38.3ms, while the TTC increases on average by 30ms, whencompared with Coaty. For mid-size intervals, the PT increaseson average by 5.2ms while the TTC increases on average by26.5ms when compared to Coaty. The bottom chart showsthat for large time intervals between requests, the PT andTTC respectively increase in average 15.4ms and 90.7ms whencompared with Coaty. These results show that while additionalprocessing time is required by TSMatch, as expected, due tothe matchmaking and aggregation functions, the added delayis quite low, being compatible with the strict requirementsof industrial environments. We can also observe in high timeintervals TSMatch has lower PT and TTC compared to Coatymainly due to code optimization and the fact that TSMatchhas sufficient time to process each request.

Fig. 6. Evaluation results for sequential scenario with fixed time intervals.

On a second round of experiments, we analyzed the impactof variable time intervals between sequential service requests,being the results depicted in Figure 7, holding again threedifferent diagrams: the top one holds results for low servicerequest interrarrival time intervals; the bottom one holdsresults for large interarrival time intervals, for both TSMatchand Coaty.

The top diagram shows that for variable low interarrival service request time intervals, the PT for TSMatch is loweron average by 50.8ms while the TTC is lower on average by140.8ms compared to Coaty. The second diagram shows thatfor variable medium time interval, the PT increases on averageby 14.5ms while the TTC increases on average by 32.3mscompared to Coaty. The third diagrams illustrates that the PTand TTC for TSMatch increased by an of average 36.7msand 104.6ms respectively compared to Coaty. We can observethat TSMatch is sensitive to variability in time intervals. Itperforms better in variable low time interval when comparedto coaty; however, it performs significantly worse in scenarioswith more variable time intervals (bottom chart). This couldindicate that further code optimization

might be required toaccommodate variability. Overall the increase in PT and TTCremains acceptable for industrial environments, taking intoaccount the additional functionalities provided by TSMatch

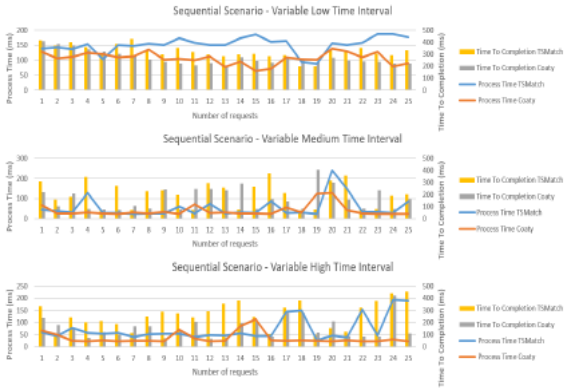

Fig. 7. Evaluation results for sequential scenario with variable time intervals.

Fig. 8. Evaluation results for simultaneous scenario.

2) **Simultaneous Scenario**: A third round of experimentshas been run to assess the performance of TSMatch in environmentswith simultaneous requests. For this experimentationround, the aim is to compare the results with the outcomesobtained in the TSMatch sequential scenarios. Figure 8 depictsPT and TTC results of the simultaneous scenario as explainedin sub-section VI-C. The mean PT value is 74.8ms while themean TTC value is 237.04ms as illustrated in Table III**. Bycomparing the TSMatch** results of the simultaneous and thesequential scenarios, we can observe that the results achievedfor both PT and TTC are similar to the results obtained inthe sequential scenario, when the service request interarrivaltime was low. This is justified due to the higher frequency ofrequests that need to be processed in a shorter time frame,resulting in a higher PT and TTC.

VII. SUMMARY AND FUTURE WORK

This paper describes a novel solution, TSMatch, for semanticmatchmaking of IoT data sources and service descriptions,based on semantic similarity. We describe the TSMatchsoftware-based architecture and explain its usage in industrial

TABLE III
MEAN PT AND TTC OF SEQUENTIAL AND SIMULTANEOUS SCENARIOS.

	Sequential Scenario						Simultaneous Scenario
	Fixed TI			Variable TI			
	Low Range	*Medium Range*	*High Range*	*Low Range*	*Medium Range*	*High Range*	
Mean PT(ms)	151.88	32.76	33.72	55.8	53.76	72.84	74.8
Mean TTC(ms)	321.56	170	111.08	127.8	198.88	260.96	237.04

environments within the context of the European EFPF project.Moreover, the paper also provides information on the currentopen-source TSMatchimplementation, and provides a firstperformance evaluation of TSMatch in terms of processingtime and time to completion of requests.

The future work will focus on improving the semanticmatching engine, in particular by integrating a more intelligentparsing and also learning, to enable the selection of the ”most”adequate set of IoT Things whose aggregated data can servea specific semantic request.

ACKNOWLEDGMENT

This project has been funded by the H2020 EFPF project,grant agreement number 825075.We thank the EFPF partners',the SMEs that have supported the use-cases (WOM & IAI),and the fortiss team supporting the IIoT Lab.

REFERENCES

[1] A. Verma, and S. Kaushal," Cloud computing security issues andchallenges: a survey," In International Conference on Advances inComputing and Communications Springer, Berlin, pp. 445-454, July.2011.

[2] A. Segev and E. Toch, "Context-Based Matching and Ranking ofWeb Services for Composition," in IEEE Transactions on ServicesComputing, vol. 2, no. 3, pp. 210-222, July-Sept. 2009.

[3] H. Fethallah, A. Chikh , and A. Belabed. "Automated discovery of webservices: an interface matching approach based on similarity measure."In Proceedings of the 1st International Conference on Intelligent SemanticWeb-Services and Applications, pp. 1-4, 2010.

[4] M. Klusch and K. Patrick, "isem: Approximated reasoning for adaptivehybrid selection of semantic services," In Extended Semantic WebConference, Springer, Berlin, Heidelberg pp. 30-44, 2010 .

[5] E. Kovacs, M. Bauer, J. Kim, J. Yun, F. Le Gall and M. Zhao,"Standards-Based Worldwide Semantic Interoperability for IoT," inIEEE Communications Magazine, vol. 54, no. 12, pp. 40-46, December2016.

[6] G. Cassar, P. Barnaghi, W. Wang and K. Moessner, "A Hybrid SemanticMatchmaker for IoT Services," 2012 IEEE International Conference onGreen Computing and Communications, pp. 210-216, 2012.

[7] N. Bnouhanna, R. C. Sofia, and A. Pretschner,, "IoT Thing To ServiceSemantic Matching," 2021 IEEE International Conference on PervasiveComputing and Communications Workshops and other Affiliated Events(PerCom Workshops), pp. 418-419, March 2021.

[8] S. Liang, C.Y. Huang, and T. Khalafbeigi. "OGCSensorThings API Part1: Sensing, Version 1.0.", 2016.

[9] I. Martens, D9.1 - Implementation and Validation through Pilot-1.[Deliverable] https://www.efpf.org/deliverables, 2021.

[10] I. Martens, D9.2 - Implementation and Validation through Pilot-2.[Deliverable] https://www.efpf.org/deliverables, 2021.

[11] I. Martens, D9.3 - Implementation and Validation through Pilot-3.[Deliverable] https://www.efpf.org/deliverables, 2021.