From user intents to context graphs: Ontology-mediated intent-based data operations in the computing continuum

Lorenzo Balzotti*[1], Donatella Firmani[1], Francesco Leotta[2], Andrea Piermartini[1], Jacopo Rossi[2], Verena Pietsch[3], Valerio Frascolla[4], Hui Song[5], Dumitru Roman[5,6], Rustem Dautov[5], Ioan Toma[7], Hyunwhan Joe[8], Hong-Gee Kim[8] and Alexandre Ulisses[9]

[1]*Department of Statistical Sciences, Sapienza University of Rome, Italy*
[2]*Department of Computer, Control and Management Engineering, Sapienza University of Rome, Italy*
[3]*Fill GmbH, Gurten, Austria*
[4]*Intel Deutschland GmbH, Feldkirchen, Germany*
[5]*SINTEF AS, Oslo, Norway*
[6]*Bucharest University of Economic Studies, Bucharest, Romania*
[7]*Onlim GmbH, Vienna, Austria*
[8]*Seoul National University, Seoul, Republic of Korea*
[9]*EVS Broadcast Equipment, Porto, Portugal*

Abstract

Operating data-intensive applications across the cloud–edge–IoT continuum increasingly resembles a knowledge engineering problem: stakeholders express *what* they want in natural language, while systems must decide *how* to realize it over dynamic, heterogeneous, multi-owner infrastructures. We argue that intent handling should be engineered as a model-driven transformation chain, from an underspecified natural-language request, to a structured intent model, to an executable deployment model, to an explanation model, mediated at every step by a knowledge graph acting as *context graph*. We present a reference conceptualization that operationalizes this view, developed within a European research project on intent-based data operations: five interconnected conceptualizations (intents, users, resources, applications, workflows) built by systematic reuse of the TM Forum Intent Ontology, FOAF/ORG, SOSA, and W3C PROV, in which the agentic AI components performing the transformations (a conversational front-end, a translation component, and an explanation component) are themselves first-class citizens of the model. We report an experience-based validation in a real smart-manufacturing scenario, where an LLM-based agent equipped with graph-querying tools resolved ten underspecified user intents into container deployment decisions: measured against ground-truth deployment sets, the pipeline achieved perfect precision and recall on all intents, including intents that were textually identical across users but required different resolutions, and infeasible intents that had to be recognized as such. The results support our central claim: in intent pipelines, the knowledge graph is the component responsible for ensuring correctness.

Keywords

intent-based systems, context graphs, model-driven engineering, computing continuum, agentic AI, knowledge graphs

1. Introduction

The computing continuum, the coordinated use of cloud data centers, edge nodes, and IoT devices, promises scalable, energy-aware, and low-latency data processing [1]. It also introduces an operational complexity that traditional configuration-centric management struggles to absorb: resources are volatile, geographically dispersed, owned by different organizations, and heterogeneous in capabilities and constraints. As applications grow in scale and autonomy, asking human operators to reason at the level of container images, node selectors, and API endpoints becomes unsustainable.

Knowledge Graphs and Model-driven Systems Engineering (KGMDSE), in ISWC 2026 Workshops Joint Proceedings, October 25–26, 2026, Bari, Italy

lorenzo.balzotti@uniroma1.it (L. Balzotti*); donatella.firmani@uniroma1.it (D. Firmani); leotta@diag.uniroma1.it (F. Leotta); andrea.piermartini@uniroma1.it (A. Piermartini); j.rossi@diag.uniroma1.it (J. Rossi); verena.pietsch@fill.co.at (V. Pietsch); valerio.frascolla@intel.com (V. Frascolla); hui.song@sintef.no (H. Song); dumitru.roman@sintef.no (D. Roman); rustem.dautov@sintef.no (R. Dautov); ioan.toma@onlim.com (I. Toma); hyunwhanjoe@snu.ac.kr (H. Joe); hgkim@snu.ac.kr (H. Kim); a.ulisses@evs.com (A. Ulisses)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-05

CEUR-WS.org/Vol-4313/kgmdse-paper9.pdf

Intent-based approaches respond to this pressure by inverting the interaction paradigm: stakeholders declare desired outcomes, performance envelopes, privacy or cost constraints, sustainability preferences, and the system takes responsibility for interpretation and operationalization. The recent maturation of Large Language Models (LLMs) has made the natural-language entry point of this paradigm practically viable [2], and intent-based networking has already produced industrial-grade artifacts such as the TM Forum Intent Ontology (TIO) [3]. Yet, outside networking, and in particular for *data operations* across the continuum, the paradigm remains underexplored.

Our goal is to show that **intent handling is, at its core, a form of model-driven engineering**: the lifecycle of an intent—elicitation, disambiguation, enrichment, translation, execution, and reporting—is a chain of model transformations. What distinguishes it, is that (i) the source model is an informal natural-language artifact, (ii) the transformations are increasingly performed by LLM-based agents rather than deterministic transformation engines, and (iii) the "platform model" (the continuum) changes underneath the transformation at runtime. A knowledge graph is the natural mediating artifact that makes this chain traceable, verifiable, and explainable.

This paper presents:

- a reading of the intent lifecycle as a four-model transformation chain in which a knowledge graph mediates every transformation (Section 3);
- an ontology view on intent-based data operations in the cognitive computing continuum [4, 5] organized around five interconnected conceptualizations, in which the agentic components performing the transformations are modeled as first-class citizens with explicit delegation and provenance (Section 4);
- an experience-based validation in a real smart-manufacturing scenario, in which an LLM-based agent grounded on the instantiated knowledge graph resolves ten underspecified user intents into container deployment decisions, evaluated against ground truth with precision and recall (Section 5).

This work identifies a concrete role for ontologies beyond data integration: *transformation context* that grounds and constrains what LLM-based components may infer.

2. Background and related efforts

Intent representation. The most mature line of work on intent modeling comes from network management. The TM Forum Intent Ontology [3] decomposes an intent into *expectations* (desired outcomes), *targets* (the subjects the intent applies to), and *conditions* (the constraints under which fulfillment is acceptable), and pairs intents with *intent reports* that document fulfillment. Knowledge-graph-enabled architectures have combined this style of intent modeling with digital twins of the network, so that intents, network state, and policies coexist in a single graph and can be validated before deployment [6, 7]. Other efforts focus on the natural-language side of the problem, refining free-text requests into structured tuples or RDF graphs and mapping them onto SDN-style APIs [8, 9], or applying rule-based reasoning to translate declarative intents into concrete configurations [10]. These works demonstrate the feasibility of intent translation, but they are largely confined to the networking domain and rarely treat the translation pipeline itself as a first-class modeled artifact.

Semantic models for the continuum. A complementary body of work provides ontologies for cloud–edge–IoT infrastructures: semantic descriptions of computational resources and IoT devices for fault handling [11]; ontology-assisted service placement with energy and network constraints [12]; semantic layers over deployment topologies (e.g., TOSCA-based) for matchmaking and validation [13]; QoS-aware models for hybrid edge environments [14]; and marketplace-oriented models for multi-cloud service selection [15, 16]. These models are rich on the *resource* and *deployment* side but tend to treat requirements as already formalized inputs: the human stakeholder, their goals, and the process by which vague goals become precise specifications remain outside the model.

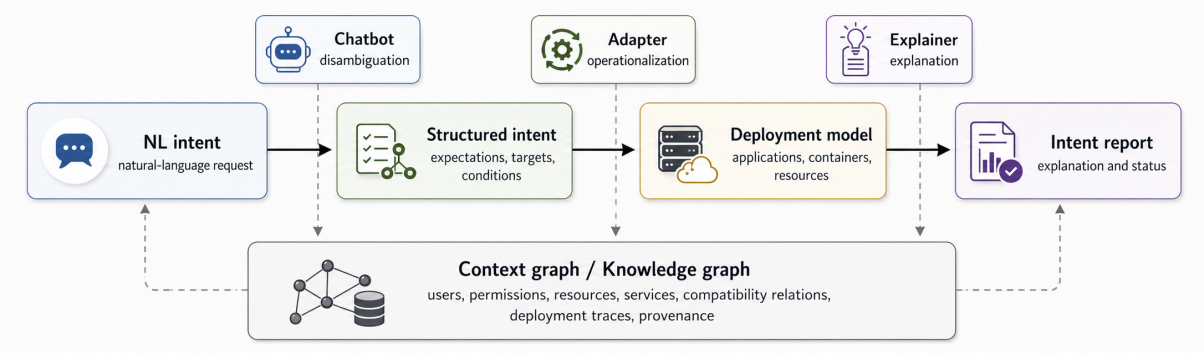


Figure 1: Intent handling as a context-graph-mediated transformation chain.

The gap. What is missing, in our reading, is a semantic model that has an impact both on users, roles, permissions, goals expressed in natural language and on the resources, applications, deployments, that also models the components that interpret, translate, and explain intents. This is simultaneously a knowledge representation problem (what should the knowledge graph contain?) and a systems-engineering problem (how are transformations specified, validated, and traced?).

3. The intent lifecycle as a model transformation chain

Consider a single request in an industrial setting: *"I want to analyze possible leakages in my production process."* This sentence does not mention any specific machine, monitoring service, software component, or deployment location. However, an intent-based system must translate it into a concrete operational decision, such as: *deploy the containers implementing the energy-monitoring service on the edge node attached to machine MA-01.*

We interpret this process from a model-driven perspective, as a chain of four models connected by three transformations:

1. **NL intent.** The user's request in natural language, stored together with the relevant conversational context.
2. **Structured intent.** A machine-readable intent expressed according to an intent metamodel. The transformation from (1) to (2) performs *disambiguation and enrichment*: it resolves expressions such as "my machine" into concrete identifiers, infers the intended service category from the described problem, and asks for missing information when needed.
3. **System intent / deployment model.** An executable specification describing which applications and containers must be deployed, where they must run, and under which constraints. The transformation from (2) to (3) performs *operationalization*: it matches the intent requirements with resource capabilities, service–machine compatibility, and user permissions.
4. **Intent report.** A structured explanation of how the system interpreted the intent and why it was, or was not, fulfilled. The report is organized around expectations, conditions, and targets. The transformation from (3), together with execution logs, to (4) performs *explanation generation* and provides feedback to the stakeholder.

Figure 1 summarizes this process and highlights the role of the context graph in supporting each transformation. Two observations follow. **First, every transformation in this chain depends on context.** Disambiguation requires information about which machines the user owns. Operationalization requires machine types, service compatibility, container requirements, and current resource properties. Explanation requires the full derivation history. In classical MDE, this context is typically captured in metamodels, transformation rules, and trace models. In the intent setting, it should be represented in a knowledge graph instantiated from a shared ontology—a context graph. The graph is not only documentation *about* the system; it is an operational input *to* every transformation and, at the same

time, the place where each transformation records its traces. **Second, the transformation engines are increasingly agent-based.** The steps above are often performed by LLM-based components, such as a conversational agent, a retrieval-augmented translator, and a multi-agent explainer. Unlike deterministic transformation engines, these components are stochastic and less transparent. This makes the context graph even more important: it is the mechanism used to *constrain* what agents may infer through grounding, *record* what they did through provenance, and *verify* whether the outcome satisfies the intent through validation.

4. A reference conceptualization in five conceptualizations

The ontology we designed to operationalize this view is organized around five interconnected conceptualizations; deliberately, none of them is sufficient alone, and the value lies in their explicit interconnection.

Intent conceptualization. Intents are modeled by direct structural reuse of the TM Forum Intent Ontology: each intent carries expectations, targets, and conditions with their original semantics, which keeps the model interoperable with existing TMF-based tooling [17]. Our extensions do not touch the internal anatomy of an intent; they contextualize it. An intent is linked to the resources and applications it concerns, to the manager components that handle it, and through derivation links to the sub-intents it is refined into.

User conceptualization. Stakeholders, like production managers, technicians, developers, administrators, but also organizations, are modeled by reusing lightweight identity vocabularies (FOAF [18] for persons and agents, the W3C Organization ontology [19] for collective actors). Users carry roles and permissions that bound which intents they may express and which resources those intents may touch. Permissions are thus not an afterthought enforced at deployment time, but part of the context consulted during intent interpretation.

Resource conceptualization. Continuum resources are aligned with SOSA's notion of *platform* [20], an entity that hosts other entities, capturing the hosting role of cloud, edge, and device nodes, while dynamic characteristics (throughput, availability, connection density, latency) are modeled as observable properties monitored at runtime. A pragmatic distinction separates *business resources* (shared infrastructure, governed by organizational policies and permissions) from *customer resources* (user-owned devices and industrial machines).

Application conceptualization. Applications are the executable means by which intents are fulfilled. We align them with PROV activities [21]: they occur over time, use resources, generate data products, and are associated with the agents that triggered their deployment. Applications carry lifecycle status and performance-related properties (latency, bandwidth requirements, acceptable packet loss) so that intent conditions can be matched against application capabilities during operationalization.

Workflow conceptualization. The most distinctive choice is that the translation machinery itself is part of the model. A *workflow* aggregates four components:

- a **chatbot**, which supports dialogue-based disambiguation and elicitation;
- an **adapter**, which translates clarified intents into system intents and is implemented as a retrieval-augmented multi-agent pipeline;
- an **explainer**, which generates intent reports from intent hierarchies, conversations, and execution logs;
- a **business knowledge** component, which provides the knowledge graph store together with query and update capabilities.

Each of the agentic components is modeled as a PROV software agent acting on behalf of the user. Modeling the machinery turns the pipeline from an opaque implementation detail into an inspectable engineering artifact.

5. Validation in an industrial scenario

We validated the approach in a real scenario provided by an industrial partner of the project: an international machine-engineering company whose customers operate high-tech production machines. Machines can be equipped with digital services like production-data recording, machine-state reporting, energy tracking, KPI analysis, alarm analysis, machining-program history, G-code analysis, tool-wear monitoring, each implemented as a set of software components requiring specific containers. Which services run on which machines depends on the machine type; which containers a component requires depends on the (component, machine-type) pair. Knowledge about machines, services, components, containers, and users exists partly as structured data and partly as textual documentation.

Instantiating the five-conceptualization conceptualization on this scenario is direct: production managers and technicians are *users* with permissions scoped to their production lines; their machines are *customer resources*; edge servers and cloud instances are *business resources*; services, components, and containers are *applications* with lifecycle status; user requests are *intents* with expectations (e.g., performance targets), targets (machines, services, actions), and conditions (time, resource, cost constraints); the chatbot–adapter–explainer triad and the knowledge store constitute the *workflow*. The resulting knowledge graph encodes, among other relations, user-to-machine ownership, machine-to-service compatibility, service-to-component composition, and component-to-container requirements: exactly the traversal paths needed to resolve an underspecified intent.

5.1. Experimental setup

Realistic user requests are short and underspecified: users rarely mention machine identifiers, IP addresses, or service names, especially when they own a single machine. We therefore evaluated the pipeline on ten user-centric intents expressed by three distinct users, deliberately including (i) intents with no explicit machine or technical parameter (e.g., *"I want to analyze possible leakages"*); (ii) intents that are textually identical or near-identical across users but must resolve differently because the users own different machines; and (iii) intents that are *infeasible* for the requesting user's machine and must be recognized as such rather than force-fitted to a deployment.

The pipeline was implemented as a ReAct-style agent built with DSPy [22], powered by a compact commercial LLM and equipped with two tools querying the knowledge graph: one retrieving service descriptions, the other retrieving the containers needed to deploy a given service on the requesting customer's machine. For each intent, the ground truth GT is the set of containers strictly necessary to fulfill it, derived from the compatibility and requirement relations of the scenario; the workflow output WO is the set of containers the agent decided to deploy. We evaluate the workflow output against the ground truth using precision and recall over the selected container sets.

Table 1 reports, for each intent, the requesting user, a paraphrase of the request, the cardinalities of ground truth and workflow output, and the resulting precision and recall. The three users are denoted as U1, U2, and U3. User U1 owns one machine (MA-01), U2 owns two machines (MA-02 and MA-03), and U3 owns one machine (MA-04). The pipeline achieved perfect precision and recall on all ten intents, including the two infeasible ones (intents 7 and 9), for which it correctly deployed nothing.

5.2. Discussion

Three aspects of these results are worth stressing beyond the aggregate scores.

Identical requests, different resolutions. Intents 4 and 5 differ only in the referenced machine, and intents 1 and 2 express the same underlying goal for different users; however, the correct container sets differ substantially (from 4 to 12 containers), because each user owns a machine of a different type with

Table 1
Evaluation on the ten user-centric intents of the industrial scenario. $|GT|$ and $|WO|$ are the cardinalities of the ground-truth and deployed container sets; container identifiers are anonymized. Intents 7 and 9 are infeasible for the requesting user's machine and must yield an empty deployment.

#	Intent (paraphrased)	User	$\|GT\|$	$\|WO\|$	Prec. / Rec.
1	Analyze possible leakages	U1	8	8	1.0 / 1.0
2	Leakage suspected on machine MA-02; analyze the process	U2	12	12	1.0 / 1.0
3	No machining-program change should go unnoticed on MA-02	U2	13	13	1.0 / 1.0
4	Replace tools based on their condition (MA-02)	U2	8	8	1.0 / 1.0
5	Replace tools based on their condition (MA-03)	U2	4	4	1.0 / 1.0
6	Optimize the G-code on machine MA-03	U2	5	5	1.0 / 1.0
7	Optimize the G-code (no machine specified; infeasible)	U1	0	0	1.0 / 1.0
8	Deploy the machine-state service	U1	2	2	1.0 / 1.0
9	Detect vibrations in my machine (infeasible)	U3	0	0	1.0 / 1.0
10	Many messages on my machine; analyze them all	U3	1	1	1.0 / 1.0

a different compatibility profile. The agent produced the correct and distinct deployments in each case. The disambiguating information like ownership, machine type, and service compatibility was available to the LLM *only* through the knowledge graph: this is the empirical core of our claim that the context graph, not the language model, carries the burden of correctness.

Infeasibility as a first-class outcome. Intents 7 and 9 request services that are not available for the user's machine. The agent correctly returned an empty deployment instead of selecting an unsuitable service. This was possible because the decision was grounded in the compatibility relations stored in the knowledge graph.

Scope and threats to validity. The evaluation involves a single (real) scenario, ten intents, and one agent configuration; the perfect scores should be read as evidence of feasibility on a realistic but bounded task, not as a general performance claim. The compatibility structure of the scenario, while genuinely industrial, is discrete and fully known; noisier settings (drifting resource properties, partially unknown compatibility) will stress the pipeline differently. Moreover, precision/recall over deployment sets does not measure the quality of the disambiguation dialogue or the faithfulness of the generated explanations, which require dedicated evaluation instruments.

6. Future works

Our experience suggests three lessons for future work.

(1) Explicit checks for agentic transformations. In our scenario, the compatibility relations in the graph worked as checks on the adapter's output: the agent should only produce deployments that are compatible with the selected machine. This idea could be made explicit, for example by using SHACL shapes as pre- and post-conditions over the intent graph, and competency questions as acceptance tests. In this way, LLM-based transformation steps could be verified automatically, instead of relying only on whether they happen to produce the correct result.

(2) Enterprise models as additional context. Some intents in our scenario refer implicitly to business or production processes, such as *"my production process"*. These processes are often already described in enterprise models, for example in BPMN diagrams or similar notations. Connecting intent ontologies with goal and process modeling languages would make these existing models useful sources of context for intent interpretation.

(3) Continuous satisfaction of intents. Our evaluation considers each intent at one point in time. However, the computing continuum changes continuously: nodes can fail, latency can vary, and services can be redeployed. For this reason, the context graph should be kept synchronized with the current state of the system, for example using digital-twin-style mechanisms. Intent reports should also be updated when conditions change, so that the system can explain whether an intent is still satisfied

over time.

7. Conclusions

We presented an ontology-mediated approach to intent-based data operations in the computing continuum, framed as a model-driven transformation chain in which a knowledge graph serves as the context that grounds, constrains, and traces every transformation, including those performed by LLM-based agents, which the model reifies as software agents with explicit delegation. The validation on a real smart-manufacturing scenario shows that the approach resolves underspecified, ambiguous, and even infeasible user intents into correct deployment decisions, with perfect precision and recall on the evaluated intent set, precisely because the decisive knowledge resides in the graph rather than in the language model. We believe that intent ontologies are a useful starting point for connecting knowledge graphs with agentic AI systems. They help structure user goals and system decisions, while knowledge graphs provide the runtime context needed to ground, validate, and explain those decisions.

Declaration on generative AI

During the preparation of this work, the authors used ChatGPT and Claude in order to: Grammar and spelling check, paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

Acknowledgments

This work was supported by the HORIZON Research and Innovation Action 101135576 INTEND "Intent-based data operation in the computing continuum".

References

[1] O. Vermesan, V. Frascolla, A. Pagani, A. Mihovska, B. Debaillie, C. O. Murchu, D. Buleandra, F. Chinello, F. Fischer, G. Suciu, et al., Edge IoT Industrial Immersive Technologies and Spatial Computing Continuum, AIOTI Association, 2024.

[2] S. Parikh, M. Tiwari, P. Tumbade, Q. Vohra, Exploring zero and few-shot techniques for intent classification, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track), ACL 2023, Toronto, Canada, July 9-14, 2023, Association for Computational Linguistics, 2023, pp. 744–751. doi:10.18653/V1/2023.ACL-INDUSTRY.71.

[3] TMForum-TIO, TR292 TM Forum Intent Ontology (TIO) v3.6.0, TM Forum, 2024. URL: https://www.tmforum.org/resources/introductory-guide/tr292-tm-forum-intent-ontology-tio-v3-6-0/.

[4] D. Firmani, F. Leotta, J. G. Mathew, J. Rossi, L. Balzotti, H. Song, D. Roman, R. Dautov, E. J. Husom, S. Sen, V. Balionyte-Merle, A. Morichetta, S. Dustdar, T. Metsch, V. Frascolla, A. Khalid, G. Landi, J. Brenes, I. Toma, R. Szabó, C. Schaefer, C. Udroiu, A. Ulisses, V. Pietsch, S. Akselsen, A. Munch-Ellingsen, I. Pavlova, H. Kim, C. Kim, B. Allen, S. Kim, P. Eberechukwu, INTEND: intent-based data operation in the computing continuum, in: Proceedings of the Research Projects Exhibition Papers Presented at the 36th International Conference on Advanced Information Systems Engineering(CAiSE 2024), Limassol, Cyprus, June 3-7, 2024, volume 3692 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 43–50. URL: https://ceur-ws.org/Vol-3692/paper6.pdf.

[5] R. Dautov, H. Song, D. Roman, E. J. Husom, S. Sen, V. Balionyte-Merle, D. Firmani, F. Leotta, J. G. Mathew, J. Rossi, L. Balzotti, A. Morichetta, S. Dustdar, T. Metsch, V. Frascolla, A. Khalid, G. Landi, J. Brenes, I. Toma, R. Szabó, C. Schaefer, S. Kim, C. Udroiu, A. Ulisses, V. Pietsch, S. Akselsen, A. Munch-Ellingsen, I. Pavlova, D. Petrova-Antonova, H. Kim, C. Kim, B. Allen, S. Kim, P. Eberechukwu, INTEND: human-like intelligence for intent-based data operations in

the cognitive computing continuum, in: A. Rula, E. Sallinger, O. Savkovic, I. G. Ciuciu, I. Toma, J. X. Parreira, R. Prodan, H. Song, A. Soylu (Eds.), Companion Proceedings of the 8th International Joint Conference on Rules and Reasoning co-located with 20th Reasoning Web Summer School (RW 2024) and 16th DecisionCAMP 2024 as part of Declarative AI 2024, Bucharest, Romania, September 16-18, 2024, volume 3816 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3816/paper54.pdf.

[6] X. Chang, C. Yang, H. Wang, Y. Ouyang, R. Dong, J. Guo, Z. Ji, X. Liu, Kid: Knowledge graph-enabled intent-driven network with digital twin, in: 2022 27th Asia Pacific Conference on Communications (APCC), 2022, pp. 272–277. doi:10.1109/APCC55198.2022.9943779.

[7] K. Mehmood, K. Kralevska, D. Palma, Knowledge-based intent modeling for next generation cellular networks, in: 2023 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), 2023, pp. 181–186. doi:10.1109/MeditCom58224.2023.10266653.

[8] Y. Ouyang, F. Li, C. Yang, R. Song, X. Liu, Z. Ji, Ontology-based network intent refinement framework, in: 22nd IEEE International Conference on Communication Technology, ICCT 2022, Nanjing, China, November 11-14, 2022, IEEE, 2022, pp. 1394–1399. doi:10.1109/ICCT56141.2022.10072810.

[9] M. Kiran, E. Pouyoul, A. Mercian, B. Tierney, C. Guok, I. Monga, Enabling intent to configure scientific networks for high performance demands, Future generation computer systems 79 (2018) 205–214. doi:10.1016/J.FUTURE.2017.04.020.

[10] P. Alcock, C. Rotsos, N. J. P. Race, Oz: Towards an extensible intent handler architecture with semantic reasoning, in: NOMS 2024 IEEE Network Operations and Management Symposium, Seoul, Republic of Korea, May 6-10, 2024, IEEE, 2024, pp. 1–5. doi:10.1109/NOMS59830.2024.10575364.

[11] I. B. Sánchez, A. Fernández, H. Billhardt, S. Ossowski, Problem detection in the edge of IoT applications, International Journal of Interactive Multimedia and Artificial Intelligence (IJIMAI) 8 (2023) 85. URL: https://doi.org/10.9781/ijimai.2023.07.007. doi:10.9781/IJIMAI.2023.07.007.

[12] M. Karanik, I. Bernabé-Sánchez, A. Fernández, Ontological modeling and clustering techniques for service allocation on the edge: A comprehensive framework, Electronics 13 (2024) 477.

[13] J. Gorroñogoitia, D. Radolović, Z. Vasileiou, G. Meditskos, A. Karakostas, S. Vrochidis, M. Bachras, The SODALITE Model-Driven Approach, in: SpringerBriefs in Applied Sciences and Technology, Springer, 2022, pp. 23–52. doi:10.1007/978-3-031-04961-3_3.

[14] S. Ghrab, I. Lahyani, S. Yangui, M. Jmaiel, A core IoT ontology for automation support in edge computing, Service Oriented Computing and Applications 17 (2023) 25–37. URL: https://doi.org/10.1007/s11761-022-00356-2. doi:10.1007/S11761-022-00356-2.

[15] N. Bassiliades, M. Symeonidis, P. Gouvas, E. Kontopoulos, G. Meditskos, I. P. Vlahavas, PaaSport semantic model: An ontology for a platform-as-a-service semantically interoperable marketplace, Data & Knowledge Engineering 113 (2018) 81–115. doi:10.1016/J.DATAK.2017.11.001.

[16] M. M. Al-Sayed, H. A. Hassan, F. A. Omara, CloudFNF: An ontology structure for functional and non-functional features of cloud services, Journal of Parallel and Distributed Computing 141 (2020) 143–173. doi:10.1016/J.JPDC.2020.03.019.

[17] TMForum-API, TMF921 Intent Management API User Guide v5.0.0, TM Forum, 2024. URL: https://www.tmforum.org/resources/specifications/tmf921-intent-management-api-user-guide-v5-0-0/.

[18] M. Graves, A. Constabaris, D. Brickley, Foaf: Connecting people on the semantic web, in: Knitting the Semantic Web, Routledge, 2013, pp. 191–202.

[19] W. W. W. Consortium, The organization ontology (2014). URL: https://www.w3.org/TR/vocab-org/.

[20] K. Janowicz, A. Haller, S. J. D. Cox, D. L. Phuoc, M. Lefrançois, SOSA: A lightweight ontology for sensors, observations, samples, and actuators, Journal of Web Semantics 56 (2019) 1–10. doi:10.1016/J.WEBSEM.2018.06.003.

[21] P. Missier, K. Belhajjame, J. Cheney, The W3C PROV family of specifications for modelling provenance metadata, in: Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013, ACM, 2013, pp. 773–776. doi:10.1145/2452376.2452478.

[22] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T. Joshi, H. Moazam, H. Miller, M. Zaharia, C. Potts, DSPy: Compiling declarative language model calls into self-improving pipelines, CoRR abs/2310.03714 (2023). URL: https://doi.org/10.48550/arXiv.2310.03714. arXiv:2310.03714.