# KEMASS: Knowledge-Enhanced Multi-Agent (digital) twin for energy Scheduling Support

Guillaume Muller[1][0000−0003−3740−9232], Somsakun Maneerat[2][0000−0001−5745−0879], and Alan Adamiak[2]

[1] Mines Saint-Etienne, Institut Henri Fayol, Saint-Etienne, France
[2] EDF, R&D, SEQUOIA, Lab Paris-Saclay, Palaiseau, France
somsakun.maneerat@edf.fr

**Abstract.** The transition to decentralized energy distribution, where any node can function as a consumer and/or producer, presents challenges in the design and testing of control algorithms, particularly in maintaining production. The existing energy scheduling model, assuming uniformity, struggles to capture the unique dynamics and constraints of individual production units. This paper introduces KEMASS, generating a Multi-Agent System using Ontologies and Knowledge Graphs to tailor optimization algorithms for power plants. Implemented in a specific energy production valley, KEMASS closely simulates the actual system, optimizing energy schedule while considering local constraints. Although not a complete Digital Twin for Energy Scheduling Support, KEMASS, with its dynamic Knowledge Graphs and Ontologies, is more adaptable to evolving into one compared to other systems. The use of Knowledge Representation technologies makes it suitable for various applications.

**Keywords:** Multi-Agent System · Ontology · Knowledge Graph · Multi-Agent System Engineering · Simulation · Energy · Real-world scenarios

## 1 Introduction

The energy sector is shifting from conventional to renewable energy sources, leading to a decentralized production landscape with diverse stakeholders. This transition to a complex and decentralized energy system (a.k.a. SmartGrid) poses growing challenges in balancing supply and demand on electrical grids, as well as in designing management and optimization algorithms for companies in the energy sector.

The existing forecasting and optimization model for electricity production, assuming uniformity across facilities, fails to capture sector dynamics and constraints unique to each production unit. Disparities between production schedules and actual capacities require frequent adjustments and human verification. The optimization algorithm needs customization for each power plant type, and coordinating diverse units involves interacting with multiple systems. Engineers designing new SmartGrid algorithms need to test them under various conditions (e.g. extreme cases). However, as the actual energy system must remain in production, testing on it is nearly impossible.

If the engineers would have a Digital Twin (DT) of the system at their disposal, they would be able to set up such specific scenarios, run some simulations, collect data, select the best solutions and apply them directly to the actual system, when required. In this paper, we propose KEMASS, a methodology/implementation, that generates a Multi-Agent System (MAS) based on Knowledge Representations (KR) of the various components of the actual system, represented with Ontologies[3] and Knowledge Graphs[4].

Choosing MAS as the virtual counterpart in the DT is straightforward due to the SmartGrid's decentralized nature and its autonomous components. Using Knowledge Representation to generate the MAS is also logical, as engineers in the energy domain can easily write KR for their expertise, despite not being Multi-Agent System programming experts.

The paper is organized as follows: Section 2 discusses related works and limitations of current approaches; Section 3 details the ontology construction and the KEMASS system; Section 4 presents the results from running the MAS for an energy production in a French valley; and Section 5 discusses KEMASS's strengths, weaknesses, and potential extensions.

## 2 Related Works

There are three main approaches to mixing Multi-Agent Systems and Knowledge Representations in the literature:

- **External once-and-for-all generation of the MAS**: [12,6,7] provide external means to generate the MAS, once and for all, from the KRs (e.g. a toolchain that generates code, or Protégé plugins for MAS mapping).
- **External specifications for the MAS**: [17] uses KGs to formulate the specifications of the system. The description of the agents is "external", like an API. The representation of the system can be manipulated, e.g. by the agents themselves, to "understand" how to call a service or communicate with another entity, but they do not need to know how the service/agent will do its job.
- **Runtime generation of Agents**: [10,4,14] generally deploy special "constructor" agents/services in the MAS platform/language, which use the representation of the system, to generate agents, from the KRs, at runtime. In this strategy, it is important to model the "internal" functioning of the agents (i.e. "executable code"), in order to completely instantiate the agents. The approach allows for some level of verification of the MAS specifications.

[3] An Ontology is a formal and explicit specification of a shared conceptualization. It is a way of representing knowledge in a machine-readable format that can be understood by both humans and machines. Ontologies define *concepts* and *categories*, as well as their *properties* and *relations* [11].

[4] Knowledge Graphs (KGs) are some sort of instantiation of an Ontology on a set of individual data points. A KG is a graph-based database that represents knowledge in a structured and semantically rich format. It is a way of organizing and representing data that allows for more efficient and effective processing of information [11].

A true Digital Twin [9] requires connections in **both directions** between the physical system and the virtual one. The first approach lacks the possibility to have a backward connection from the running virtual system to the physical system. The second approach lacks the internal representation of the agents' functioning, thus does not completely implement the connection from the physical system to the virtual system. Only the third approach is opened to the creation of a true DT. We thus focus mostly on this approach.

To further describe the connection between KR and MAS, we will rely on the "Vowels" paradigm [2]), that identifies five components in MAS: **A**gents, **E**nvironment, **I**nteractions, **O**rganization, **U**ser, and discuss the use of KRs at each of these levels.

Some works, such as [16,5,8] employ KRs at the **A**gent level for reasoning, while others use them at the **I**nteraction level as a "common language", to smooth inter-agent communications [7,13,15,16], particularly as a means of interoperability between different sources of information [1,3]. As emphasized in [13], these uses of KRs in MAS are related. Indeed, if **A**gents communicate using KRs, they already manipulate elements on which they can reason. Thus, it is reasonable to design the reasoning process of these agents based on the same KRs [13]. Moreover, if **A**gents' reasoning is based on KRs, they are most probably already manipulating concepts from the **O**rganization/**E**nvironment levels, in their KRs. The prospect of designing the complete MAS using these KRs is thus in close reach. At best, papers in the literature are using Ontologies as the unique Knowledge Representations to describe the actual system. Ontologies can represent the structure of the system, including classes and their relations. However, Ontologies do not provide sufficient detail to fully instantiate an agent, including configurations/values of its properties, from real-world data, a task at which Knowledge Graphs excel. This is the main contribution of KEMASS, our model. This is also, the specific means that will allow the MAS to communicate back with the physical system, thus opening the door the the generation of an actual DT.

## 3 Methodology

The objective of KEMASS is to automatically generate Multi-Agent Systems (MAS) from energy system descriptions using Knowledge Representations. The paper specifically focuses on developing a tool to assess the feasibility of production schedules for hydropower plants in a French valley, taking into account both local and global constraints. The modeling incorporates absolute timesteps, each lasting 30 minutes, to ensure synchronous agent operation.

Our methodology comprises two components: (1) domain representation and (2) automatic MAS generation, as illustrated in Figure 1.

### 3.1 Domain representation

Domain knowledge is represented through the creation of an Ontology and a Knowledge Graph (Figure 2). These Knowledge Representation elements serve

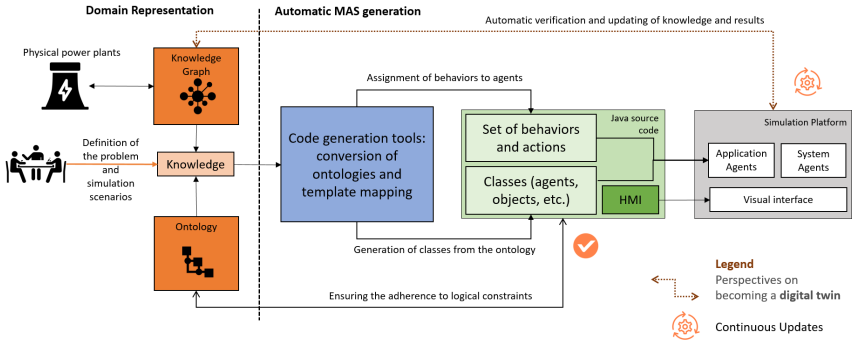


**Fig. 1.** The Methodology for automatically Generating a Multi-Agent System platform from Knowledge Representation and its prospective evolution into a Digital Twin.

as a resource generator for our Multi-Agent Systems simulation. An example of this is the transformation of the blue classes in the Hydropower Ontology into MAS agents.

***Agents and their organization:*** Local constraints modeled at the **Production facility**[5] include unit activation times, operating duration adjustments, capacities, water flow rates, and power output. This Production facility entity is a subclass of **Physical object**, a broad category covering shared characteristics between **Production facilities** and **Reservoirs**.

At the regional level, constraints revolve around the spatial position of **Physical Objects** in a **Valley**, impacting flow, operational capacity, and collaboration sequencing.

At a global scale, a **Programming Entity** manages production scheduling of **Productions Facilities** in **Valleys**, specifying time slots and power demand for each.

***Agent behaviors and interactions:*** Agent behaviors are intricately integrated into the template linked to ontology classes, providing flexibility for modeling within our implementation. The dynamic interaction and coordination among all agents, as conditioned by the ontology, are illustrated in Figure 3. The MAS starts with the reactive agent **Programming Entity** transmitting daily production schedules for coordination to each **Production facility** agent. The **Production facility**, in turn, initiates actions through a Finite State Machine (FSM) encompassing main states of the plant: *shutdown*, *restart*, and *production*. In the *shutdown* state, the agent remains inactive, periodically checking for a need to resume activity. During the *restart* phase, the agent takes some duration to find the best operating point based on instant water flow before resuming *production*. The agent communicates flow information to upstream and

[5] Bold text marks Agent names.

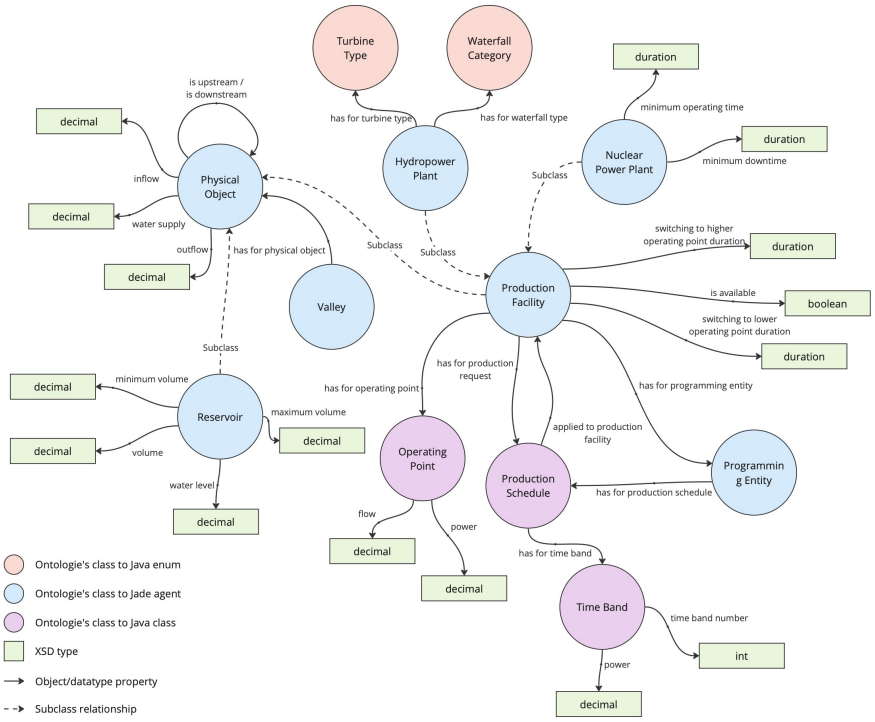


**Fig. 2.** The Hydropower Ontology and its corresponding Java mapping components.

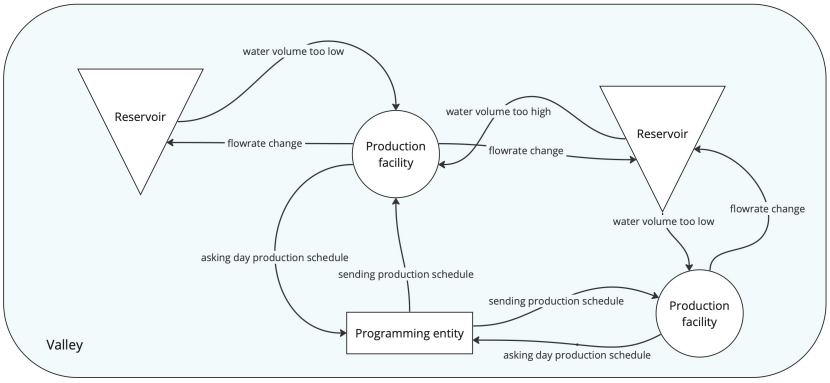


**Fig. 3.** Exchanged messages between agents.

downstream agents classified as **Physical objects**, records production and flow rates, conveys efficiency information during operating point changes, and notifies of *shutdown*. **Reservoir** agents update water volume, adjust inflow/outflow based on messages, and introduce random adjustments in the inflow with a cyclic behavior.

### 3.2 Automatic MAS generation

This section presents the KEMASS methodology, for generating and running a Multi-Agent System from Knowledge Representation implemented as Ontologies and Knowledge Graphs.

The section centers on the KR-to-MAS conversion methodology, which occurs in two primary stages involving three programmatic objects known as (see Figure 4): *Main* (the algorithm launcher), *ConstructAgent*, and *ClassBuilder*. The initial step involves the generation of Java classes from the ontology, while the second step instantiates Jade agents from a knowledge graph.

#### Generation of Agents' class from ontologies

*The first phase of our KR-to-MAS conversion algorithm* is class creation. Managed by an agent builder, this stage combines the loading of ontologies into memory and the generation of functional classes through the *ClassBuilder* utility. Ontology files are converted into RDF4J[6] models. The ontologies are split into two RDF4J models, one for the main ontology, and the second for the imported ones. This solution is paving the way for the selection of relevant classes in the construction of the final MAS.

***Conversion rules:*** After initializing the algorithm, the constructor agent relies on a class dedicated to conversion, the *ClassBuilder*. The *ClassBuilder* plays a crucial role in translating ontological concepts into executable Java code. Ontology classes, which may include Jade agents, Java base classes or Enums, are generated according to parameters defined in a specific configuration file. The `SubClassOf` properties are used to establish hierarchical links between classes, while taking into account Java's inheritance constraints. Data type properties (`owl:DatatypeProperty`) are translated into variables according to the XSD to Java conversion rules, and object properties (`owl:ObjectProperty`) are converted into variables, using the class defined by the "range" property (`rdfs:range`) as the type. Cardinality constraints, in particular the "minimum 1" constraint, guide the algorithm by signaling the representation of properties in the form of lists (`ArrayList`). Finally, class indivuals (`owl:classIndividual`) are used to represent the elements of an Enum. At the end of this step, the ontology conversion function is invoked recursively for each object property, when the variable's type class is defined in external ontologies. For more detail of the KR-to-MAS conversion, see Table 1 in the Appendix.

[6] https://www.rdf4j.org/

***Templates to create source code:*** Once all the classes have been parsed, the algorithm uses the Apache FreeMarker template engine to translate the data into Java code. Each class refers to a template determined by information in the configuration file, facilitating the conditional integration of class-specific elements. Behaviors, currently coded manually in additional template, are then integrated into the generated Java source code.

Once the source files have been written [7], the Java Compiler receives a path module containing only the classes and functions of the dependencies essential to the program. In this way, the MAS agents are compiled and the class names passed on to Jade.

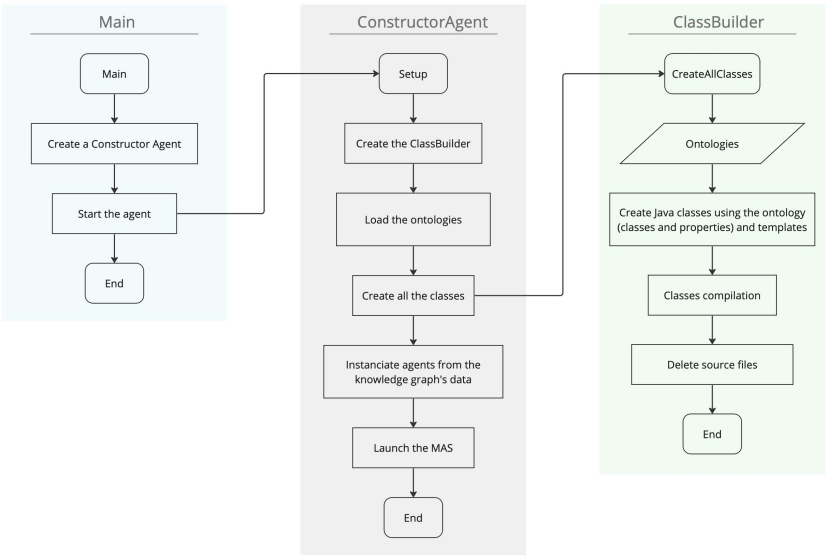


**Fig. 4.** The agent generation process.

### Agent instantiation from Knowledge Graph

*This step involves executing specific SparQL queries overseen by the constructorAgent.* The queries provide the flexibility to integrate constraints customized for the specific context. In this study, limitations are applied to select only a single valley.

Initially, (1) a query is executed to obtain the complete list of classes present in the knowledge graph. Then, (2) for each identified class, a new query is formulated to retrieve all related properties, including their cardinality where ap-

[7] The source codes are compiled in a single batch, a strategy designed to anticipate the potential problems of circular dependencies common when using ontologies.

plicable. When properties exhibit cardinality, a specialized function is invoked to significantly reduce the number of queries, ensuring optimal performance. Each property is then stored in a Java variable of type Map, where the variable name serves as the key and its type as the value. If the property corresponds to a class, a third SparQL query is deployed to retrieve all the elements necessary for agent instantiation. (3) The instantiation phase concludes by launching an agent of the appropriate class and passing the Map of variables as an argument. This latter is used at the agent's startup to initialize its variables using the *Reflection API*. This pragmatic approach ensures efficient agent initialization, taking into account context-specific dependencies in the Multi-Agent System (MAS).

## 4 Results

**Note on the results:** *We ran KEMASS on data from an actual valley in France. However, releasing the actual data to the public would put the Energy Grid at risk of a dangerous attack, as such precise data is critical to its functioning. As a consequence, we designed an artificial hydraulic valley (see Fig.5) containing 1 reversible plant (which can consume electricity to pump water back into its reservoir) and 5 hydroelectric plants, among which two are run-of-river. To complement these plants, we linked them to their respective reservoirs. To add to the complexity of the representation, one of the reservoirs has two upstream plants. The results presented in this section are based on the artificial valley, but demonstrate the same behavior from KEMASS.*

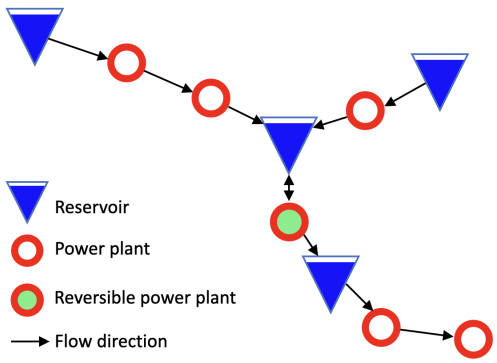


**Fig. 5.** Fake valley structure

.

Simulations record what happens at every moment, such as the amount of electricity produced or the quantity of water in the reservoirs. These recordings help us observe how things unfold during the simulation.

An evaluation was conducted on the simulation's compliance with the production schedule, focusing on water flow and produced energy. The graphs (see Figure 6) revealed that the simulation was not perfectly aligned with the production schedule due to specific rules considered by our simulator, such as the required wait times after certain changes. Our field experts clarified that this observation aligns with the daily routine of the energy company, which often involves manually adjusting production scheduling. This highlights the role of our simulator, which acts as a digital tool to verify the compliance of the production schedule with local rules.

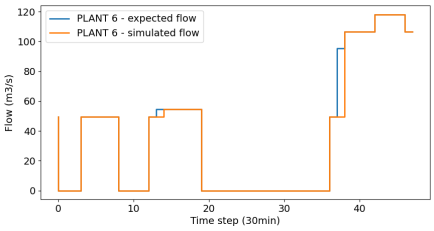


**Fig. 6.** Verification of a power plant's flow regulation compliance with its production schedule.

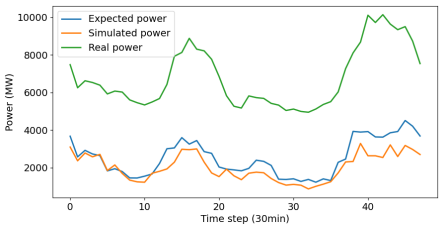


**Fig. 7.** Comparison between daily hydroelectric production in France among RTE data, our simulation output, and production scheduling.

To validate our approach, we analyzed simulation results, focusing on assessing their resemblance to reality in terms of produced energy. Due to limited access to actual factory data, a direct comparison at the agent level (Production Facility) was not feasible. However, partial validation was achieved at the aggregated level by comparing simulated total hydroelectricity production in France, considering all modeled Production Facilities, with data from RTE's ÉCo2mix[8]. The simulation results align with similar trends, despite producing less electricity than reported by RTE (see Figure 7). Expert discussions confirmed the realism of these results, highlighting that the energy company manages only a subset of power plants (around 16%), while others face unmodeled local constraints in our simulator, thus maintaining coherence with real-world constraints in KEMASS's results.

## 5 Discussions

Based on the results, KEMASS demonstrates significant potential as confirmed by domain experts and its ability to simulate a trend similar to real-world produced energy data. However, it is acknowledged to be in the early stages of

[8] https://odre.opendatasoft.com/explore/dataset/eco2mix-national-tr/information/?disjunctive.nature

development, and its validation has been partial. This section delves into both current limitations and ongoing enhancements.

*Validation:* KEMASS involved testing it in three scenarios: using real and synthetic data from a hydropower valley and data from another valley, resulting in realistic outcomes. However, further validation is essential using a broader range of real-world data in terms of volume and variety to enhance its reliability and robustness. To ensure the scalability of KEMASS, we are expanding our current Knowledge Representations by incorporating additional energy types (e.g., nuclear plants) and incorporating data from diverse valleys simultaneously. Additionally, we plan to extend the application of KEMASS to different domains, such as healthcare or manufacturing, for comprehensive validation.

*Internal Agent model:* The modeling of agent behaviors is currently expressed through a template system based on Finite State Machines (FSM). Despite its utility, this approach lacks the expressiveness inherent in a Turing Machine, consequently affording agents a less generic foundation for potential evolution, as articulated by domain experts. To address this limitation, our objective is to advance a more expressive and user-friendly representation of agent behaviors, leveraging Knowledge Representation. The adaptability of our system to this evolutionary trajectory is underpinned by its KR-to-MAS auto-generate system.

*Time representation:* Our application uses absolute time with a fixed timestep, which may not fit other applications. KEMASS can be modified to fit different time scales (even at the same time), by simply ensuring: 1) message ordering in agents and 2) correct time scale representation(s) in Knowledge Representation.

*Full Digital Twin:* KEMASS's main contribution is the combined use of Ontologies and Knowledge Graphs, allowing for a potential feedback loop between the virtual and physical systems (as shown in Figure 1). This could enable KEMASS to generate a full Digital Twin of any system. To ensure bidirectional data flow, KEMASS's `ConstructorAgent`(s) would need to update KRs and MAS as changes occur.

## 6 Conclusion

This paper introduced KEMASS, a methodology that enables the generation and execution of a Multi-Agent System directly from a domain representation, leveraging the synergy of Ontologies and Knowledge Graph.

Our application of KEMASS to an electric production scenario in a hydraulic valley yielded promising initial findings, demonstrating the system's ability to closely replicate the outcomes of the original system and pinpoint disparities in production scheduling versus actual plant capacity.

However, for comprehensive validation, KEMASS necessitates further testing with additional empirical data. Two ongoing improvements to KEMASS are

underway: (i) refining agent behaviors using a more complex and flexible system based on Knowledge Representation, and (ii) establishing synchronization between MAS updates and KR updates to achieve an authentic Digital Twin of the modeled system

While initially designed for energy scheduling support, we have meticulously crafted each process to be as generic as possible. As a result, our system holds potential for application in diverse domains that can be represented with Ontologies and Knowledge Graphs, requiring minimal modifications.

**Acknowledgement.** *First part removed for the sake of anonymity.*

**NOTE:** Generative AI tools have been used in the process of creating this paper, for writing improvements (ChatGPT) and for finding additional references (ResearchRabbit).

## References

1. Cranefield, S., Purvis, M., Nowostawski, M., Nowostawski, M., Hwang, P.: Ontologies for interaction protocols. In: Tamma, V., Cranefield, S., Finin, T.W., Willmott, S. (eds.) Ontologies for agents: theory and experiences. pp. 1–17. Birkhäuser Basel (2005)
2. Demazeau, Y.: From interactions to collective behaviour in agent-based systems. In: European Conference on Cognitive Science. vol. 95 (1995)
3. Dikenelli, O., Erdur, R.C., Özgür Gümüs: 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. pp. 1271–1272. AAMAS '05, Association for Computing Machinery, New York, NY, USA (07 2005)
4. Donzelli, C., Kidanu, S.A., Chbeir, R., Cardinale, Y.: Onto2MAS: An ontology-based framework for automatic multi-agent system generation. In: 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS). pp. 381–388. IEEE (01 2016)
5. Drozdowicz, M., Ganzha, M., Paprzycki, M., Olejnik, R., Lirkov, I., Telegin, P.N., Senobari, M.: Ontologies, agents and the grid: An overview. Parallel, distributed and grid computing for engineering pp. 117–140 (01 2009)
6. Freitas, A., Bordini, R.H., Meneguzzi, F., Vieira, R.: Towards integrating ontologies in multi-agent programming platforms. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2015, Singapore, December 6-9, 2015 - Volume III. vol. 3, pp. 225–226. IEEE (2015)
7. Freitas, A., Bordini, R.H., Vieira, R.: Designing multi-agent systems from ontology models. In: Weyns, D., Mascardi, V., Ricci, A. (eds.) Engineering Multi-Agent Systems - 6th International Workshop, EMAS 2018, Stockholm, Sweden, July 14-15, 2018, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11375, pp. 76–95. Springer International Publishing (07 2018)
8. Freitas, A., Schmidt, D., Schmidt, D., Schmidt, D.N., Panisson, A.R., Meneguzzi, F., Vieira, R., Bordini, R.H.: Semantic representations of agent plans and planning problem domains. In: Dalpiaz, F., Dix, J., van Riemsdijk, M.B. (eds.) Engineering Multi-Agent Systems: Second International Workshop, EMAS 2014, Paris, France, Revised Selected Papers 2. pp. 351–366. Springer International Publishing (05 2014)

9. Fuller, A., Fan, Z., Day, C., Barlow, C.: Digital twin: Enabling technologies, challenges and open research. IEEE access **8**, 108952–108971 (2020)
10. Girardi, R., Leite, A.: A knowledge-based tool for multi-agent domain engineering. Knowledge Based Systems **21**(7), 604–611 (2008)
11. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., et al.: Knowledge graphs. ACM Computing Surveys (Csur) **54**(4), 1–37 (2021)
12. Mascardi, V., Ancona, D., Barbieri, M., Bordini, R.H., Ricci, A.: Cool-agentspeak: Endowing agentspeak-dl agents with plan exchange and ontology services. Web Intelligence and Agent Systems: An International Journal **12**(1), 83–107 (2014)
13. Mathieu, P., Routier, J.C., Secq, Y.: Towards a pragmatic use of ontologies in multi-agent platforms. In: Palade, V., Howlett, R.J., Jain, L. (eds.) International Conference on Knowledge-Based Intelligent Information & Engineering Systems. pp. 1395–1402. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
14. Poveda, G., Schumann, R.: An ontology-driven approach for modeling a multi-agent-based electricity market. In: Multiagent System Technologies: 14th German Conference, MATES 2016, Klagenfurt, Österreich, September 27-30, 2016. Proceedings 14. pp. 27–40. Springer (2016)
15. Sheldon, F.T., Elmore, M., Potok, T.E.: An ontology-based software agent system case study. In: 2003 International Symposium on Information Technology (ITCC 2003), 28-30 April 2003, Las Vegas, NV, USA. pp. 500–506. IEEE Computer Society, Los Alamitos, CA, USA (04 2003)
16. Tran, Q.N., Low, G.: MOBMAS: A Methodology for Ontology-Based Multi-Agent Systems development. Information and Software Technolology **50**(7-8), 697–722 (2008)
17. Trojahn, C., Quaresma, P., Vieira, R.: Conjunctive queries for ontology based agent communication in mas. In: 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, 2008, Volume 2. pp. 829–836 (2008)

# Appendix

**Table 1.** The mapping between XSD types and Java objects.

| Type XSD | Type Java |
|---|---|
| anyURI | java.net.URI |
| base64Binary | String |
| boolean | boolean |
| byte | byte |
| date | java.time.LocalDate |
| dateTime | java.time.LocalDateTime |
| dateTimeStamp | java.time.LocalDateTime |
| dayTimeDuration | java.time.Duration |
| decimal | double |
| double | double |
| duration | java.time.Duration |
| ENTITIES | ArrayList<String> |
| ENTITY | String |
| float | float |
| gDay | int |
| gMonth | java.time.Month |
| gMonthDay | java.time.MonthDay |
| gYear | java.time.Year |
| gYearMonth | java.time.YearMonth |
| hexBinary | String |
| ID | String |
| IDREF | String |
| IDREFS | ArrayList<String> |
| int | int |

| Type XSD | Type Java |
|---|---|
| integer | int |
| language | String |
| long | long |
| Name | String |
| NCName | String |
| negativeInteger | long |
| NMTOKEN | String |
| NMTOKENS | ArrayList<String> |
| nonNegativeInteger | long |
| nonPoritiveInteger | long |
| normalizedString | String |
| NOTATION | String |
| positiveInteger | long |
| QName | String |
| short | short |
| string | String |
| time | java.time.LocalTime |
| token | String |
| unsignedByte | int |
| unsignedInt | long |
| unsignedLong | long |
| unsignedShort | int |
| yearMontDuration | java.time.Duration |