Mapping SysML v2 to NGSI-LD: Enhancing Energy Systems Modeling

Christoph Klaassen
TU Wien
Institute of Energy Systems and Thermodynamics
Getreidemarkt 9
1060 Vienna, Austria
christoph.klaassen@tuwien.ac.at

Lukas Kasper
TU Wien
Institute of Energy Systems and Thermodynamics
Getreidemarkt 9
1060 Vienna, Austria
lukas.kasper@tuwien.ac.at

René Hofmann
TU Wien
Institute of Energy Systems and Thermodynamics
Getreidemarkt 9
1060 Vienna, Austria
rene.hofmann@tuwien.ac.at

***Abstract*—The world is transitioning towards green energy to address the challenges of climate change and reduce greenhouse gas emissions. However, the complexity of Multi Energy Systems (MES) requires advanced tools for efficient simulation, operation, and maintenance. This study introduces a novel parser developed using the Python open-source tool SLY, which maps critical elements from System Modeling Language version 2 (SysML v2) specifications to runtime model Next Generation Service Interface - Linked Data (NGSI-LD). This approach enables seamless communication between different energy system components. It improves the co-simulation of energy networks by integrating MES and supporting the mapping of the distributed nature of energy resources. The parser successfully demonstrates the mapping of essential SysML v2 elements to NGSI-LD, showcasing the potential for SLY-based tools to significantly contribute to the field of energy systems modeling. Our study highlights the importance of open-source tools in addressing the challenges of modern energy systems and represents a step forward in developing such tools for energy systems.**

***Index Terms*—Multi Energy Systems, SysML v2, NGSI-LD, SLY, Open-Source Tools, Simulation**

I. Introduction

Modern energy systems are complex, comprising diverse components such as heat and power plants, renewable electricity production units, storage systems, and transmission and distribution networks [1]. Integrating Multi Energy Systems (MES) and the distributed nature of energy resources is challenging [2]. These systems operate in a dynamic environment, influenced by renewable energy integration, demand fluctuations, and grid stability [3]. The rapid implementation of hydrogen energy systems is essential to meet the growing demand for clean and sustainable energy [4]. Therefore, it is necessary to have efficient tools for modeling, simulation, and optimizing these complex systems [5]. The integration of hydrogen into the energy system requires an in-depth understanding of the individual components and their interactions to ensure optimal system configuration and operation [6]. Therefore, there is a crucial need for adaptable and interoperable open-source software solutions, which demand advanced tools that seamlessly integrate various energy carriers and facilitate dynamic simulations [7]. Existing open-source software provides solutions for modeling and analyzing energy networks; however, they do not represent the actual component from a system model perspective [8]. Those systems often lack the possibility of sector coupling tasks, which could be a future application domain of those tools. They are usually uniquely tailored to a specific energy carrier [9].

Digital Twins (DT) can support this and play a pivotal role in a better understanding of those energy systems [10]. To develop the digital object, a digital model of the physical object needs to be created. A system model designed to compile semi-automatic runtime models combines a data protocol for connections and sensors with an interface for physical object information transfer [11]. In the context of MES, a DT is created by using system models of the different components in the interested physical energy system and mapping those to a runtime model, representing a digital counterpart to the MES [12]. This mapping serves as a basis for the connection from the physical object to the digital model already at an early design stage. The runtime model serves as a functional data protocol possibility when sensors are connected.

This research paper develops and applies a novel parser designed to bridge the gap between system models and runtime models to enable a better understanding of the components and a coupled-component MES. The goal is to reach an easy handling tool for mapping system models to runtime models. The contribution of this paper is to present an open-source SLY-based parser for this task. First, we present the required and used open-source tools. Then, we describe the architecture and implementation of the parser. Challenges and further considerations are then described. A proof-of-concept use of the parser in a possible hydrogen MES is given, followed by a result section. The discussion section highlights alternative approaches, and in the conclusion, an outlook is given. The proof-of-concept SLY-parser script is under an open-source BSD 3-Clause License publicly available.[1]

II. SysML v2, NGSI-LD, and SLY

System Modeling Language is a standard in model-based system engineering used in various engineering disciplines,

[1]https://gitlab.tuwien.ac.at/iet/public/sysmlv2mapngsi-ld

including energy systems [13]. The updated System Modeling Language version 2 (SysML v2) is still in development but features multiple improvements. SysML v2 language provides a standardized framework for describing system architectures, requirements, and behavior. However, a data transfer or information exchange is challenging. Next Generation Service Interface - Linked Data (NGSI-LD) is a semantic data model designed for context representation in a linked data format. It enables seamless communication and interoperability across diverse domains [14]. It distinguishes between *Entities*, types of attributes (*Properties*, *GeoProperty*, and *Relationships*), and *Context* data. For a clear model identification, an NGSI-LD Uniform Resource Name (URN) is created. NGSI-LD builds the backbone of the very well-known open-source Fiware ecosystem. Fiware has evolved to better support linked data in the Internet of Things applications [15].

[16] proposes a solution that bridges Open Data portals and NGSI-LD-based data. By integrating Comprehensive Knowledge Archive Network-based Open Data portals with NGSI-LD standards, the solution ensures comprehensive dataset descriptions and high-quality metadata. Their proposed approach has been validated through real-world implementation, achieving excellent ranking in terms of Findable, Accessible, Interoperable, and Reusable data principles. Another approach shows semantic enrichment and data comprehensibility. NGSI-LD enables a linkage to semantic data. The insight data enrichment focuses on specific domains. It can enhance NGSI-LD *Entities* containing numerical *Properties* within predefined categories. For example, it supports *SmartDataModel* types like AirQualityObserved, SoundPressureLevel, Temperature, and TrafficFlowObserved. NGSI-LD contributes to improving data comprehension and citizen-centered initiatives. Non-existing NGSI-LD *SmartDataModel* types can be created easily, stored, and then reused, publicly.[2] As smart cities evolve, this symbiosis between semantic data and smart city ecosystems sets the stage for future exploration and refinement [17]. However, it is not designed for modeling complex system architectures, unlike a typical system modeling language. The challenge lies in bridging the gap between these two domains. SysML v2 models capture the intricacies of system components from an early design stage. In contrast, runtime models offer an extensible format for representing context information and integrating runtime sensor data from those components. The heart of this research lies in developing a Python-based parser using the SLY library.[3] SLY is an open-source lexer and parser generator that enables efficient parsing of complex specifications.[4] It improves the extraction of critical language elements with the possibility of directly mapping them into runtime models and emerges as a powerful tool in this context.

[2]https://smartdatamodels.org/
[3]https://github.com/dabeaz/sly
[4]https://sly.readthedocs.io/en/latest/

The following list provides the main reasons for that:

- Robust Parsing Capabilities: SLY can extract critical elements and organize them for further processing.
- Mapping Essentials: Our novel parser leverages SLY to map essential system elements to runtime constructs. This mapping ensures that crucial information, such as system architecture, component interactions, and behavior, is accurately represented in the runtime models/format.
- Seamless Communication: The parser facilitates seamless communication between different energy system components and vectors by translating system models to runtime models. Hence the exchange of context information between system components can be achieved.

Next, the SLY-based parser has several practical applications:

- Multiple energy carrier systems: Energy systems often involve multiple carriers (e.g. electricity, different types of gas). The parser enables their integration within a standardized runtime model. Disparate energy vectors can “communicate” harmoniously to represent sector coupling approaches.
- Distributed energy resources (DERs): As DERs become more prevalent, like solar installations and home batteries, the parser ensures their interoperability by mapping system models to functional runtime models. DERs can dynamically adjust their behavior based on context information shared via the runtime model. Since the implementation is highly flexible, integrating different types of energy resources and consumers is possible [18]. The proof-of-concept has been modeled with different components and energy conversion units.
- Extension for co-simulation [19]: Co-simulation of energy networks, where different component models interact in a simulated environment, benefits from accurate context representation in the runtime models. However, runtime models based on system models are quite complex to create. The SLY-based parser can provide a strong foundation for more accurate and insightful co-simulation results.

III. Architecture and Implementation

Fig. 1 shows the semi-automatic processes needed to create the runtime models, based on the system model of the MES. Domain experts generate the system model by representing the physical objects. This system model is transferred to the SLY-based parser, which then creates the runtime models.

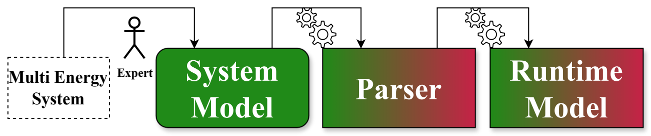


Fig. 1. Semi-automatic toolchain for creating runtime models based on the system model of the MES by integrating the SLY-based parser.

A. *Architecture*

The SLY-parser translates system models into runtime models, ensuring seamless communication between different energy system components in an open-source ecosystem. The SLY-based parser extracts relevant system information, such as SysML v2 *parts* and inner *subparts*, *connections*, and expected engineering input and output data through *ports*. It then translates this data to NGSI-LD *Entities*. Each system *part* becomes a digital representation with associated *attributes*. This includes dissecting composite systems into single *Entities*, representing *attributes*, *connections*, and storing relevant engineering information in it. The linguistic analysis process, starts with text scanning and is followed by tokenization. The parser subsequently processes these tokens following the defined grammar rules.

The mapping logic is crucial in architecture engineering, as it establishes the mapping process between system elements and runtime constructs. The architecture comprises modules for both lexer and parser functionalities, ensuring seamless integration of the two processes. For instance, a system *part* can be transformed into a NGSI-LD runtime *Entity*, and its *attributes* can be turned into runtime *Properties*. In SysML v2, nested *parts* are defined as a single *Entity* with a "hasSubpart"-*Relationship*. It is crucial to accurately map this definition to the runtime model. This is because *subparts* may indirectly represent possible physical connections to their main part, which is essential for system engineering. However, the *connection* element also exists in SysML v2. Therefore, a separate "connection"-*Relationship* was defined. Furthermore, the parser should be designed to be compatible with a larger ecosystem that includes already existing system- and runtime models, APIs, and visualization tools. The result is a well-rounded solution that meets the needs of a wide range of users.

B. *Implementation steps*

Regarding SysML v2, the various elements, such as *parts*, *connections*, and *ports*, must be clearly defined in the system model to enable a correct representation. NGSI-LD uses a linked data format that represents context information in a standardized way. The parser ensures that SysML v2 elements are correctly transformed into NGSI-LD *Entities*, *Properties*, and *Relationships*. The first step is to define the grammar rules for the system model shown in Listing 1.

```
program      ::= statements
statements   ::= statement statements
             | empty
statement    ::= ATTRIBUTE IDENTIFIER DOUBLEPOINT IDENTIFIER
             SEMICOLON
             | ATTRIBUTE IDENTIFIER SEMICOLON
             | PART IDENTIFIER DOUBLEPOINT IDENTIFIER
             "{" statement "}"
             | PART IDENTIFIER DOUBLEPOINT
             IDENTIFIER SEMICOLON
             | PART IDENTIFIER SEMICOLON
empty        ::=
```

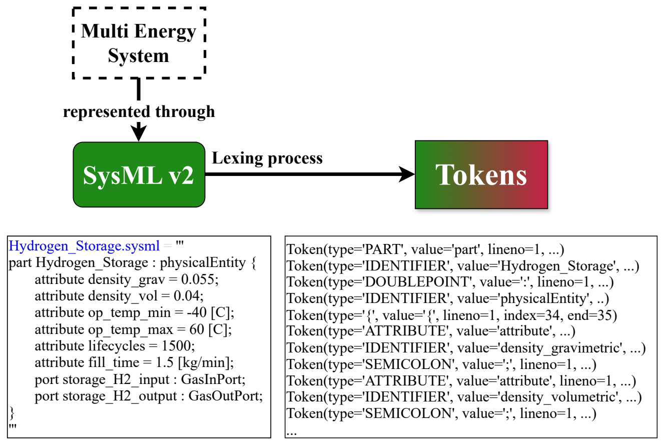
Listing 1: Snippet of the generic grammar based on the developed parser.

These rules describe the syntax and structure of the system models, based on predefined tokens. Table I shows the corresponding mapping logic.

SysML v2	mapping token	NGSI-LD type
parts	"part"	*Entity*
attributes	"attribute"	*Property*, *Relationship*
connections	"connection"	*Entity*
ports	"port"	*Entity*, *Relationship*

TABLE I
SYSML V2 TO NGSI-LD TRANSFORMATION RULES (NOT COMPLETE)

It tokenizes the input with the help of the lexer and breaks it down into meaningful units. Once the relevant elements have been extracted, the parser processes these tokens according to the defined grammar rules. Fig. 2 outlines this process, where relevant elements such as system components, relationships, and behavioral descriptions are identified.



```
Hydrogen_Storage.sysml = '''
part Hydrogen_Storage : physicalEntity {
      attribute density_grav = 0.055;
      attribute density_vol = 0.04;
      attribute op_temp_min = -40 [C];
      attribute op_temp_max = 60 [C];
      attribute lifecycles = 1500;
      attribute fill_time = 1.5 [kg/min];
      port storage_H2_input : GasInPort;
      port storage_H2_output : GasOutPort;
}
'''
```

```
Token(type='PART', value='part', lineno=1, ...)
Token(type='IDENTIFIER', value='Hydrogen_Storage', ...)
Token(type='DOUBLEPOINT', value=':', lineno=1, ...)
Token(type='IDENTIFIER', value='physicalEntity', ..)
Token(type='{', value='{', lineno=1, index=34, end=35)
Token(type='ATTRIBUTE', value='attribute', ...)
Token(type='IDENTIFIER', value='density_gravimetric', ...)
Token(type='SEMICOLON', value=';', lineno=1, ...)
Token(type='ATTRIBUTE', value='attribute', lineno=1, ...)
Token(type='IDENTIFIER', value='density_volumetric', ...)
Token(type='SEMICOLON', value=';', lineno=1, ...)
...
```

Fig. 2. A snippet of the lexing process SysML v2 *Hydrogen_Storage-part* and its related tokens.

Combined with the grammar they are mapped to corresponding NGSI-LD constructs, shown in Fig. 3.

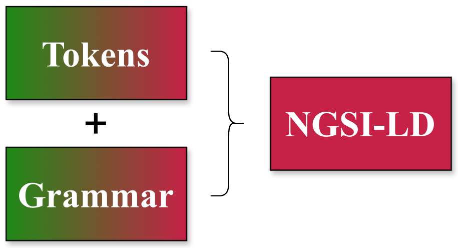


Fig. 3. Based on the extracted tokens and the defined grammar the NGSI-LD model is created.

This workflow already demonstrates the flexibility of the parser approach, as you can define utterly independent syntax and structure rules. The parser is upward compatible to further SysML v2 development, as new rules can be applied to the lexer to implement the latest features of SysML v2 in the mapping process.

C. Challenges and Considerations

Mapping system model semantics to runtime models requires careful consideration. Some system elements in a runtime model may not have direct equivalents, necessitating creative solutions. The system models can be intricate, with various levels of abstraction and dependencies. The SysML v2 system language has several concepts representing a physical object. Some SysML v2 concepts, like *generalization* and *specialization*, need to be further investigated. Those represent abstractions of existing elements, which describe and model actual devices of a MES. Mapping higher abstraction concepts still needs to be justified, as the mapping process focuses on the actual system components instantiated. NGSI-LD only knows the constructs *Entity, Property*, and *Relationship*. Therefore, the parser needs to handle different constructs in SysML v2 consistently. The following implementation solutions tackle these challenges:

- *parts:* are converted to NGSI-LD *Entities*, including context information. For a clear identification, a NGSI-LD URN is created. All created URNs include the described SysML v2 element. The parser also includes a separate NGSI-LD "type"-*Property*, for a closer type specification. If the *part* is specified by a *part definition* in the SysML v2 model, its definition name is used. Otherwise, the value **part** is set in the NGSI-LD model.
- *attributes:* are either *Properties* of the created *Entities*, which can include number or string as a DataType, or *Relationships* to other NGSI-LD *Entites*, via URNs.
- Nested parts:
  - The NGSI-LD *Entity*, representing the outer *part*, has a *Relationship* attribute "hasSubpart", referring to all other NGSI-LD *Entities*, representing the inner *parts*, via URNs.
  - Vice-versa, inner *parts* in the SysML v2 model were transferred to single NGSI-LD *Entities*, including a *Relationship* attribute "isSubpart" referring to the NGSI-LD *Entity* representing the outer *part*, also via URN.
- *ports:* There are currently two ways how *ports* are transferred to NGSI-LD:
  - A single NGSI-LD *Entity* with a defined "port"-type based on the system model description is created. It includes a *Relationship* attribute "belongsToPart" with a URN to the *Entity*, representing the SysML v2 *part* using it.
  - The NGSI-LD model of the SysML v2 *part*, which uses the *port*, includes also a "port"-*Relationship*, referring to the NGSI-LD port representation via URN.
- *connections:* In SysML v2 *connections* represent physical connections. Therefore this concept was mapped to a single NGSI-LD *Entity*, including *Relationship* attributes *from* and *to* based on the SysML v2 connected *parts*. The mapping considers two ways in the SysML v2 model, creating a *connection*: either through a **connection definition** or through a **connect** command.

This study highlights the possible modeling of a MES. Based on the systems' physical characteristics and engineering constraints, a system model was created. The developed prototype parser generates NGSI-LD runtime models, which can be used for detailed modeling of MES, as the approach is extendable.

IV. Proof of concept

In the proof-of-concept for the parser, the mapping of a hydrogen-based MES to their respective runtime models is shown. Component data of actual units were used to model the system's specifications and stored in a *MES_model.sysml* file. Fig. 4 shows the system, which comprises an electrolyzer, hydrogen storage, and a fuel cell, including component capacities, power and gas flows. The complexity of a model regarding different energy vectors increases very quickly. Therefore, the physical relationships and processes are simplified for reasons of clarity, with the possibility of extending the approach. The simulation and investigation of the described MES are out of scope.

The SysML v2 model of the MES is shown in Fig. 5. The system model consists of several *parts* including physical *connections* (green), *parts*, and subpart membership (red). The individual *port* and *item* instances required to model the processes are shown in a combined *Port Def* layer and *Item* layer. The *H2_System-part* includes information about the *connections* and the *flows* between the single components. Its *subparts* include a *Electrolyzer-*, *Hydrogen_Storage-* and *Fuel_Cell-part*. The engineering constraints of each physical object are covered as *attributes*, e.g. power and capacity of the components, modulation- and pressure ranges, and fill-time as well as a number of life cycles for the hydrogen storage. A *Power-* and *Gas-item*, were defined for modeling different energy carriers. Based on those *item definitions*, an **Item Layer** was implemented to connect the components via *ports*. An *input-* and *output-port* were instantiated and connected to each component. The *ports* are defined in a **Port Definition Layer**. The SysML v2 is processed by the parser, resulting in the creation of NGSI-LD models, stored as *model[number].json* files.

V. Results

This work introduces an open-source prototype parser that aims to facilitate the transfer of system models to runtime models. The open-source SLY-parser uses the system model for creating the single NGSI-LD models. Based on the SysML v2 elements in Table I, *parts* of the SysML v2 model were mapped to single NGSI-LD *Entities*. The NGSI-LD *Entity*, representing the *Hydrogen_System-part*, refers to all *Entities*, representing the previously assembled *parts* in SysML v2. It has a URN for each subpart. The *subparts* are mapped

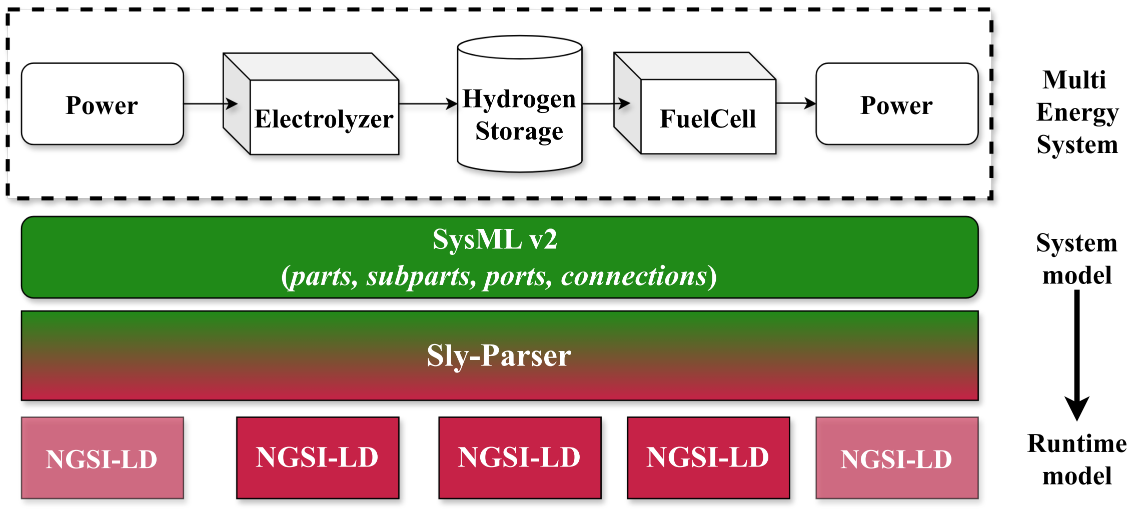


Fig. 4. Overview of the hydrogen-based MES with its generic SysML v2 system model and the related NGSI-LD runtime models.

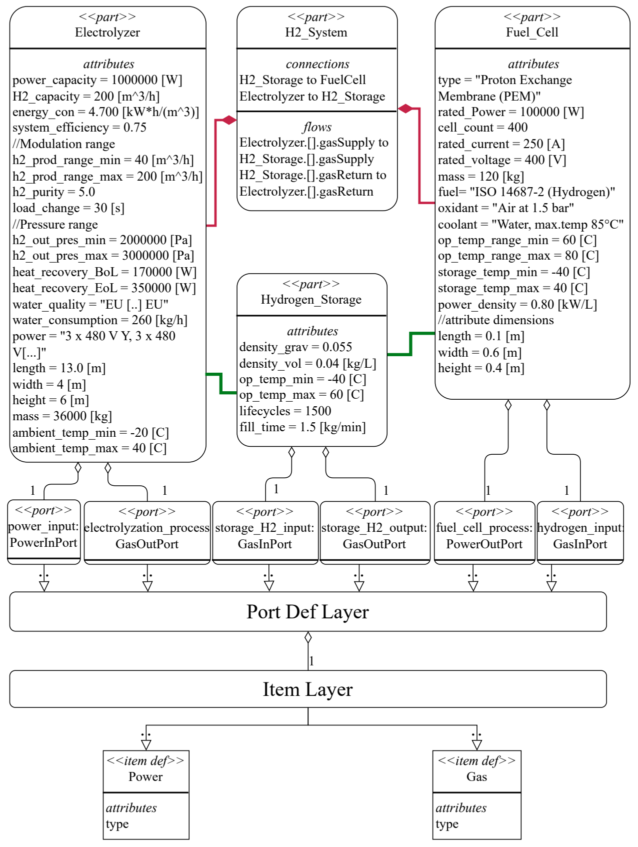


Fig. 5. Components of the MES shown in Fig. 4, modeled in SysML v2.

to unique NGSI-LD models, with a reference to the main *part* included. *Attributes* have been mapped accordingly to *Properties* and *Relationships*. *Ports* were included in the NGSI-LD *Entities* as *Relationships*, through URN-based *Properties*. For each *connection* and *port*, a single NGSI-LD *Entity* is created. The NGSI-LD snippets of the *Hydrogen_System*, the *Hydrogen_Storage* and an example *Port* are shown by Listing 2-4.

```
{
  "@context": [...],
  "id": "[...]:Part:[Hydrogen_System]",
  "type": "physicalEntity",
  "name": {
    "type": "Property",
    "value": "Hydrogen_System"
  },
  "hasSubpart": {
    "type": "Relationship",
    "object": [
      "urn:ngsi-ld:Part:[Electrolyzer]",
      "[...]:Part:[Fuel_Cell]",
      "[...]:Part:[Hydrogen_Storage]"
    ]
  }
}
```

Listing 2: *Hydrogen_System* as the main system *part* in NGSI-LD format (*model_3.json*).

```
{
  "@context": [...],
  "id": "urn:ngsi-ld:Port:[...]-988",
  "type": "Port",
  "belongsToPart": {
    "type": "Relationship",
    "object": [
    "[...]:Part:[Hydrogen_Storage]"
    ]
  },
  "attributes": [],
  "inItems": [],
  "outItems": []
  }
}
```

Listing 3: *Port* in NGSI-LD format (*model_11.json*).

The novel approach enhances energy system modeling by combining open-source methodologies with an existing Python

```
{
  "@context": [...],
  "id": "[...]:Part:[Hydrogen_Storage]",
  "type": "physicalEntity",
  "name": {
    "type": "Property",
    "value": "Hydrogen_Storage"
  },
  "isSubpartFrom": {
    "type": "Relationship",
    "object": "[...]:Part:[Hydrogen_System]"
  },
  "density_gravimetric": {
    "type": "Property",
    "value": 0.055
  },
  ....
  "Port": {
    "type": "Relationship",
    "object": [
      "[...]:Port:[...]-988"
    ]
  }
}
```

Listing 4: *Hydrogen_Storage* in NGSI-LD format as inner *part* of *Hydrogen_System* (*model_2.json*).

library. Its architecture highlights the capabilities of the SLY package, providing a robust foundation for the parser's functionality. It promotes interoperability between system models and runtime models. It allows the transfer of specific SysML v2 elements, like *parts*, inner *parts*, *connections*, and *ports*, to single NGSI-LD runtime *Entities*. The parser was used in a proof-of-concept use for MES. Based on the SysML v2 system model, NGSI-LD runtime models were successfully created.

VI. Discussion

The SLY-based approach is promising in transferring system models to runtime models. The runtime model is needed to establish a mapping from the physical object to the digital representation. SysML v2 and NGSI-LD represent both well-known standards and have their proven scientific reason to co-exist. SysML v2 as a system modeling language and NGSI-LD as a functional runtime model representation. The MES SysML v2 model provides the system specifications, possible *connections*, and *ports*. The NGSI-LD runtime models, created by the SLY-parser represent the runtime instantiations. As there exist other system modeling languages and tools, as well as different runtime model options, this work helps by conceptualizing mapping approaches for other underlying model types. This would result in a completely new implementation, however the SLY-based architecture stays the same. New lexing tokens and a new grammar definition would be necessary. During a newly defined mapping process, concepts that are not clear to transfer would also need a closer investigation of how to handle them. The complexity of this paper's proof-of-concept of the parser, including a complex system model representing a MES, highlights the flexible possibilities with such a tool. Concerning simulation, accurate context representation enables dynamic simulations of energy networks. Co-simulation capabilities benefit from the NGSI-LD format since the runtime model can be a foundation for actual behavioral models to monitor, predict, and maintain the energy system. The approach of *SmartDataModels* makes the reuse of already existing models possible. With the Python package *pysmartdatamodels*[5], a library for NGSI-LD models is provided. Since the SLY parser is under the proof-of-concept stage, it doesn't have any system checking available at the moment. Therefore it is assumed to use correct SysML v2 models.

VII. Conclusion

An open-source prototype SLY-based parser was developed to map SysML v2 models to NGSI-LD models. In a proof of concept, a hydrogen-based MES was used to create runtime models based on its system model. As we strive for a sustainable energy future, such open-source tools and *SmartDataModels* can be useful in existing tool chains to shape our understanding and decision-making processes in MES and the DT domain. Another useful approach that can be thought of alongside direct mapping is graph transformations. It enables seamless information integration within system models. In future work, we will investigate how predefined runtime models can represent the SysML v2 modeling concepts, since the mapping approach seems promising for semi-automatic DT development.

References

[1] W. Yu, P. Patros, B. Young, E. Klinac, and T. G. Walmsley, "Energy digital twin technology for industrial energy management: Classification, challenges and future," *Renewable and Sustainable Energy Reviews*, vol. 161, p. 112407, Jun. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S136403212200315X

[2] G. Paludetto, E. Bionda, and F. Soldan, "MESP - An Interoperable Platform for Multi-Energy Systems," in *2022 AEIT International Annual Conference (AEIT)*, Oct. 2022, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9951779

[3] J. Lowitzsch, C. E. Hoicka, and F. J. van Tulder, "Renewable energy communities under the 2019 European Clean Energy Package – Governance model for the energy clusters of the future?" *Renewable and Sustainable Energy Reviews*, vol. 122, p. 109489, Apr. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032119306975

[4] J. O. Abe, A. P. I. Popoola, E. Ajenifuja, and O. M. Popoola, "Hydrogen energy, economy and storage: Review and recommendation," *International Journal of Hydrogen Energy*, vol. 44, no. 29, pp. 15 072–15 086, Jun. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S036031991931465X

[5] M. O. Nachawati, G. Bullegas, A. Vasilyev, J. Gregory, A. Pop, M. Elaasar, and A. Asghar, "Towards an Open Platform for Democratized Model-Based Design and Engineering of Cyber-Physical Systems," *Modelica Conferences*, pp. 102–114, 2022. [Online]. Available: https://ecp.ep.liu.se/index.php/modelica/article/view/629

[6] X. Dong, Z. Xu, J. Wu, K. Liu, and X. Guan, "Optimal Coordination of Hydrogen-Based Integrated Energy Systems Considering Thermal Dynamics of Fuel Cells," in *2022 4th International Conference on Smart Power & Internet Energy Systems (SPIES)*, Dec. 2022, pp. 1630–1635. [Online]. Available: https://ieeexplore.ieee.org/document/10082142

[7] G. Paludetto, E. Bionda, and F. Soldan, "MESP - An Interoperable Platform for Multi-Energy Systems," in *2022 AEIT International Annual Conference (AEIT)*, Oct. 2022, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9951779

[5]https://pypi.org/project/pysmartdatamodels/

[8] L. Thurner, A. Scheidler, F. Schäfer, J. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, “pandapower — An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems,” *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, Jan. 2018.

[9] S. Ershadi-nasab, E. K. Farahani, and M. Sharifzadeh, “Multi-Vector Internet of Energy (IoE),” in *Industry 4.0 Vision for the Supply of Energy and Materials: Enabling Technologies and Emerging Applications*. Wiley, 2022, pp. 159–187, conference Name: Industry 4.0 Vision for the Supply of Energy and Materials: Enabling Technologies and Emerging Applications. [Online]. Available: https://ieeexplore.ieee.org/document/9779356

[10] T. Testasecca, M. Lazzaro, and A. Sirchia, “Towards Digital Twins of buildings and smart energy networks: Current and future trends,” in *2023 IEEE International Workshop on Metrology for Living Environment (MetroLivEnv)*, May 2023, pp. 96–101. [Online]. Available: https://ieeexplore.ieee.org/document/10164035

[11] U. Cali, B. D. Dimd, P. Hajialigol, A. Moazami, S. N. G. Gourisetti, G. Lobaccaro, and M. Aghaei, “Digital Twins: Shaping the Future of Energy Systems and Smart Cities through Cybersecurity, Efficiency, and Sustainability,” in *2023 International Conference on Future Energy Solutions (FES)*, Jun. 2023, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10182868

[12] L. Zhang, L. Zhou, and B. K. Horn, “Building a right digital twin with model engineering,” *Journal of Manufacturing Systems*, vol. 59, pp. 151–164, Apr. 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0278612521000455

[13] J. Marco and N. D. Vaughan, “Architectural modelling of an energy management control system using the SysML,” *International Journal of Vehicle Design*, vol. 55, no. 1, p. 1, 2011. [Online]. Available: http://www.inderscience.com/link.php?id=38044

[14] ETSI, “ETSI GS CIM 006 (V1.2.1): Context Information Management (CIM); NGSI-LD Information Model,” 2023.

[15] M. Bauer, “FIWARE: Standard-based Open Source Components for Cross-Domain IoT Platforms,” in *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*, Oct. 2022, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10152259

[16] L. Martín, J. Lanza, V. González, J. R. Santana, P. Sotres, and L. Sánchez, “A Connector for Integrating NGSI-LD Data into Open Data Portals,” *Sensors*, vol. 24, no. 5, p. 1695, Jan. 2024, number: 5 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/24/5/1695

[17] V. González, L. Martín, J. R. Santana, P. Sotres, J. Lanza, and L. Sánchez, “Reshaping Smart Cities through NGSI-LD Enrichment,” *Sensors*, vol. 24, no. 6, p. 1858, Jan. 2024, number: 6 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/24/6/1858

[18] L. Tian, L. Cheng, Y. Wan, K. Yuan, R. Liu, and K. Wu, “From Distributed Energy Resources to Virtual Power Plants: A Cyber-Physical System Solution for Integrating Demand-side in Smart Grid,” in *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*, Oct. 2020, pp. 3463–3467.

[19] B. Pesendorfer, E. Widl, W. Gawlik, and R. Hofmann, “Co-simulation and control of power-to-heat units in coupled electrical and thermal distribution networks,” in *2018 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. IEEE, 2018, pp. 1–6, num Pages: 6. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8405396