Automated Generation of Asset Administration Shell with Large Language Models: Interoperability in Digital Twins with Semantic Node

Yuchen Xia
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Stuttgart, Germany
yuchen.xia@ias.uni-stuttgart.de

Zhewen Xiao
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Stuttgart, Germany
st165303@stud.uni-stuttgart.de

Nasser Jazdi
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Stuttgart, Germany
nasser.jazdi@ias.uni-stuttgart.de

Michael Weyrich
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Stuttgart, Germany
michael.weyrich@ias.uni-stuttgart.de

***Abstract*— This work proposes and evaluates an LLM-system to process the technical data for creating digital twin models in the context of industry 4.0. The integration of Asset Administration Shells (AAS) within Industry 4.0 frameworks marks a pivotal advance in digital transformation, aiming to foster interoperability among diverse assets in smart manufacturing. Despite the potential, the manual creation of AAS models is time-consuming and costly, hindering widespread adoption. Addressing this challenge, our study introduces a novel approach utilizing Large Language Models (LLMs) to automate the generation of AAS instances, significantly enhancing efficiency and reducing manual effort. We develop a novel theoretical framework centered on a data structure called "semantic node", designed to capture and process the semantics of information using LLMs. Through a comprehensive case study, we demonstrate the utility of our approach in generating AAS instance models from technical data sheets of automation components with a significant reduction in effort and an effective generation rate of 62-79%, as validated by human evaluation. Our findings reveal the potential of LLMs in achieving semantic interoperability and streamlining the digital transformation process in manufacturing. An in-depth investigation of Retrieval-Augmented Generation (RAG) mechanisms further indicates a "Cheat Sheet Effect" based on our evaluation result, offering insights into the effective design of LLM systems for technical concept understanding. This research not only underscores the feasibility of automating AAS instance creation but also lays a groundwork for enhancing semantic interoperability in digital twins with application of LLM technology. The tool and evaluation results is released here: https://github.com/YuchenXia/AASbyLLM .**

Keywords*— *Asset Administration Shell, Large Language Models, Semantic Node, Digital Twins, Interoperability, Retrieval-Augmented Generation. Introduction

I. INTRODUCTION

The Asset Administration Shell is a key concept within the framework of Industry 4.0 to enable interoperability in the digital transformation process. AAS provides a standardized digital representation of an asset, which can be a physical object (e.g., a machine, a component, or a product) or a non-physical entity (e.g., a service, a process, or software) [1]. Serving as a solution for implementing digital twin, the primary goal of AAS is to facilitate seamless communication and data exchange between different assets, systems, and stakeholders in a smart manufacturing environment.

While the user may have a wealth of data, the efforts spent on transforming the company specific data into the standardized AAS form are expensive, and the cost-benefit ratio for adapting AAS shall be improved to encourage the companies to use the AAS[2]. The automation of the translation process – the process from existing vendor-specific information models into the standardized AAS-model – offers a viable solution to improve efficiency and reduce costs. In general, creating digital twins remains a challenging task, requiring manual effort. Researchers [3][4][5]advocate the invention of new solutions for automated AAS creation based on available heterogeneous engineering information.

This paper begins with reviewing existing approaches employed to convert diverse, heterogeneous information into AAS models. It provides a comprehensive review of their applied methods, along with an in-depth discussion on the strengths and weaknesses.

Drawing insights and departing from these existing methodologies, we propose a novel theoretical framework that utilizes the langue language model to analyze the semantics of textual data and translate them into a self-contained semantic unit called "semantic node". We provide a narrative of development in LLM-technology to justify why the LLM can be used to process the semantics of textual data.

Diving deeper into the practical value of our proposed framework, we conducted a case study using the semantic node and LLM agent system to automate the generation of AAS instance models from raw text obtained from technical data sheets of automation components. Furthermore, we developed a software called "AASbyLLM", where users can use this implemented AI-tool via web-browser to efficiently generate AAS instance model with their own data.

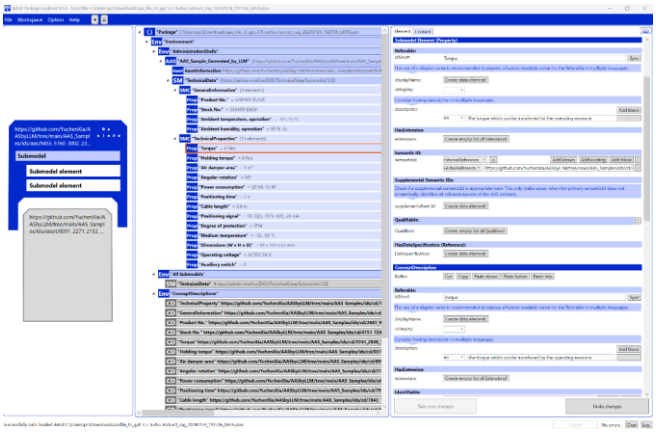

Figure 1 A screenshot of generated AAS instance model by AASbyLLM software

Main message and contribution of the paper are:

- **Theory:** We conceptualize a data structure unit called "semantic node" for capturing the essential semantics of a piece of information and showing how it can be utilized in a LLM agent system for processing information and generating AAS instance models. A general approach for creating LLM agents system is articulated and realized.

- **Utility:** By manipulating the "semantic node", an information processing system powered by LLMs can generate high quality AAS models automatically with 62-79% effective generation rate based on human evaluation, and this indicates a significant effort reduction comparing to manual creation of AAS instances. We publish our evaluation dataset and host a web-service for an end-to-end testing of the thoroughness of this approach.
- **Observation & Insights**: We performed model-level comparative analysis and ablation study in our evaluation. Our statistical analysis revealed the conditional effectiveness and limitations of RAG mechanisms, leading us to propose the "Cheat Sheet Effect of RAG" hypothesis to elucidate these findings. This provides insight into how to design an LLM-system effectively.

II. Related Works to AAS Instance Creation

Based on our literature review, we identified 18 papers closely related to the automated creation of AAS instances, summarized in Table 1. After examining the broad-spectrum methods used to create AAS model and their commonalities, we distilled the promising aspects of these studies into two key points:

1) Rule-Based Automated Conversion Methods

The most methods focus on the automated conversion between AAS and various information sources through precisely defined and clear-cut mappings and explicitly outlined target and source models. This approach ensures structured data translation but requires significant initial setup for rule definition and managing the precise mappings between semantics between source and target models.

Table 1 Literature review on automated AAS model generation

Paper	Purpose	Transformation methods	Transformed data type / source format	Semantics in data
Category 1: Using pre-defined rule-based mapping				
[6]	Model Transformation for AAS	Model Transformation Language based on OCL	Software model / UML	Rely on static model semantics in the context of model driven software engineering
[7][8]	Interoperable digital twins in IIoT systems	Detailed mapping model for conversion to AAS format	Proprietary information model / UML	Reference on ECLASS dictionary for data definition
[9]	AAS creation from heterogeneous Data	Python implementation for extracting and mapping engineering information	Engineering information / PDF, STP, XML, XML/AutomationML, RDF	Mentioned semantic web and semanticID in AAS with ECLASS reference
[10]	Mapping between AAS submodels and skill ontology CaSkMan	Rule-based mappings using RML and RDFex	CaSkMan ontology / RDF, OWL	Formal semantic in ontology
[11]	Semantic interoperability for AAS-based digital twins using ontologies	Ontology Modeling Language (OML) to map AAS models to OWL	Concepts in Manufacturing Resource Capability Ontology / OWL, UML	Connecting ontology vocabularies semantics with system models
[12]	Interoperability between DTDL and AAS	Mapping between the Digital Twin Definition Language (DTDL) and AAS	DTDL metamodel elements / UML	Semantic annotation in JSON-LD and AAS
[13]	Generation of digital twins for information exchange	Formalizes the AAS meta-model in a domain-specific language and an intermediate representation	AAS model / XSD, JSON schema, RDF	Mapping based on semantic equality
[14]	Improve semantic discoverability	Establish a set of rules for converting RDF-based models into AAS models	RDF-based models / RDF, SHACL, OWL, SPARQL	RDF as a semantic base for AAS
[15]	Generating test cases to verify AAS server implementations	Transform AAS-based plant models into MaRCO Ontology instances	AAS model / UML, RDF, OWL, SPARQL	Semantic interoperability based on ontologies
[16]	Provide data in a standardized and semantically described manner	Mapping semantic descriptions from OPC UA information models into the AAS model	OPC UA information models / OPC UA nodes, JSON	Specified semantics in OPC UA standards
[17]	Model the semantics of the current operation, status and configurations of assets.	A mapping ruleset to create the semantic AAS as an RDF graph aligned with the RAMI4.0 vocabulary	AAS model / JSON, RDF, SHACL, SPARQL	Using ontological modeling and semantic technologies.
[18]	Generating AAS from engineering data	Collect data models into AutomationML and map attributes into the AAS format	Engineering data / AutomationML	Semantics are specified as roles and relations between objects in AutomationML
[19]	Interoperability in manufacturing systems for exposing information	Map XML elements to the AddressSpace of an OPC UA	AAS model elements into OPC UA node / XML	Standard semantics in OPC UA model
Category 2: Applying NLP methods, specifically embedding language models				
[2]	Data migration into AAS	Using embedding language model to generate mappings	Technical properties / textual data	Semantic embedding with language models, ECLASS dictionary
[20][21] [22]	Semantic interoperability in heterogeneous AAS	Using trained embedding language model to generate mappings	Technical properties / knowledge graph, textual data	Semantic embedding with language models, ECLASS dictionary
[23]	Map technical operational data for building monitoring	Using trained embedding language to automatically map heterogenous data points	Technical properties / Information model from various communication protocols	Semantic embedding with language models, ECLASS dictionary

2) Semantic Enhancement of AAS

This emphasizes enhancing data communication through identifiable semantic descriptions of data. It involves assigning clear semantic identifiers from standardized dictionaries or adding detailed descriptions to data elements. These approaches connect to the concept of 'semantic interoperability', which seeks to transcend the limitations posed by varying data formats and representation styles through identifying the semantic equivalency of data.

III. Theoretical Foundation of the Work

We generalize the ideas from these research works and form a principal proposition of "semantic communication":

> *Effective communication between two systems can be achieved provided that the data being communicated is clearly defined and disambiguously interpreted.*

Rule-based methods adhere to this proposition, as they rely on predefined, explicit mappings for translation. These mappings eliminate ambiguity by clearly specifying how data from one model should be converted to another. However, the drawback of this approach is indeed having these rigorous rules in place. Firstly, this mapping is fixed and constrained by a source model and a target model, making it difficult to adapt to new or evolving data models, these lead to high cost in enumerating and managing of these rules (**inflexibility**). Secondly, there are often nuanced differences between elements of two models, and a rigorous mapping may force subtle changes and lead to semantic shift. This result in information distortion, and an evident and intuitive example is that rule-based translation often falls short in accurately translating in NLP tasks (**distortion**).

The driving idea of this work is to depart from the rule-based transformation model, engage the idea of utilizing semantics, and explore a novel conceptual approach to translate information from one source system to a target system. The theoretical framework of this approach is outlined in Figure 2**Error! Reference source not found.**, which emphasizes capturing the meaning of the data being communicated and encapsulates it into an atomic semantic unit called a "semantic node" as illustrated in Figure 2 (B).

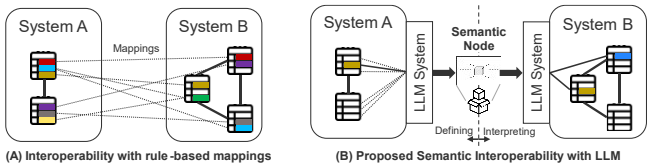

Figure 2 Interoperability between two systems with different modeling

This implies that representation form and data format become of less concern, the disparities of diverse domain-specific languages or model specification can be overcome by semantic interpretation capability of LLM, thus achieving semantic interoperability. Following this principle, in this work, the data in AAS are enriched with meta-data that contribute to articulating the semantic meaning in this intermediate unit called "semantic node". This semantic node contains several text elements used to characterize the semantics of the communicated data: "Name", "Value", "Conceptual definition", "Affordance", "Value type", "Unit", "Source description". The concrete specification is detailed in Section V.A "Semantic Node".

IV. Background in LLM Technology

Building on the proposed abstract proposition of semantic communication, there remains a concrete question: How can the "data being communicated" be "clearly defined" and "disambiguously interpreted"? To address this, we first provide a summary of the background of LLM technology and introduce the system design. We aim to conclusively answer this question by the end of this section.

A. Large language models interpret semantics

In 2017, as OpenAI's scientist report in [24] their discovery of "sentiment neuron" – a single semantic unit in a neural network that precisely predicts customers' sentiment in review texts and only response to a specific meaning. This finding encourages further use of training objectives such as masked-language-modeling and next-token-predictions to learn high-quality concept representations from texts. Following this, the invention of transformer architecture [Ref.], its integration into large language models, and the scaling-up of LLMs have led to further new interesting insights: in a LLM, the basic syntactic rules are learned in lower layer while high-level semantic concepts are learned in higher layer [25]; Moreover, 'poly-semantic neurons', which handle multiple unrelated concepts, enable the reuse of neurons in processing semantic through superposition, whereas 'mono-semantic neurons', representing singular meanings, enhance concept differentiation for precise detection and reasoning [26][27]. These neurons form elemental units of LLM's processing "circuits"[27]. Furthermore, by investigating the correlation between neuron activation and output texts, semantic neurons can be identified[28]. These works provide evidence that LLMs possess an inherent structure enabling them to effectively interpret semantics[1].

B. Embedding LLM and generative LLM

The evolution of Large Language Models has led to the emergence of two distinct types: embedding LLMs and generative LLMs. **Embedding LLMs** transform text into high-dimensional vector spaces, which can be used for indexing. Semantic relationships can be determined through mathematical operations. For example, semantic similarities between two sentences can be identified using embedding LLM, making them suitable for applications like search engines and recommendation systems. On the other hand, **generative LLMs** specialize in text production. They mostly operate on "causal language modeling" method, where each new token is predicted based on the preceding sequence, a process also known as "auto-regression" or "next-token prediction". This method is particularly effective for content generation, producing contextually coherent text.

[1] For a visual example: https://openaipublic.blob.core.windows.net/neuron-explainer/neuron-viewer/index.html

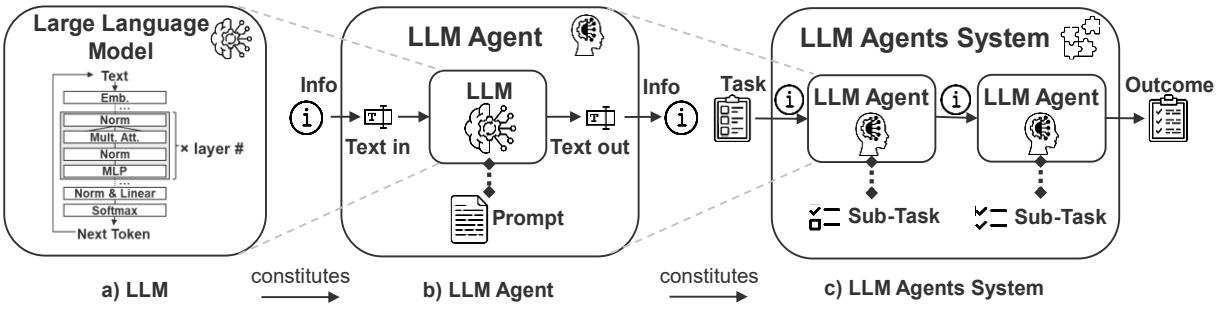

Figure 3 From single large language model to collaborative LLM agents system

C. *Prompting of causal generative LLM*

A causal generative LLM operates by focusing on the important preceding tokens, predicting the next token based on probabilistic estimation. From a high-level intuition, a prompt can be viewed as an incomplete text sequence, providing an anchoring context for generating a continuation. The model's response is shaped by how these prompts trigger the LLM's learned patterns, guiding the generation process towards a specific result. This allows for a dynamic interaction with the LLM, where the content generation can be steered through careful prompt design in desired directions to elicit the learned patterns and knowledge in LLM.

D. *LLM Agents System*

Conventionally, *agent-oriented system design* refers to a framework used in software engineering for building complex systems that consist of multiple agents that interact with each other to perform tasks and achieve goals [Ref.]. For LLM applications, this design methodology is especially useful for overcoming challenging tasks by applying task decomposition and modular solution strategies ("divide and conquer")

The primary advantage of LLM-agent system is its capability to simplify a complex task by breaking it down into smaller, manageable sub-tasks, each assigned to a specialized agent. A unique aspect of LLM-agents is that their behavior can be defined by the prompt in natural language. In the prompt, defining a specific role is a strategic way to guide and constrain their behavior. In our previous paper [29], we defined a unified structure for specifying prompt for LLM-agent. The method to create a LLM agents system is illustrated in Figure 3.

E. *Retrieval-Augmented Generation (RAG)*

As the name indicates, Retrieval-Augmented Generation (RAG) combines the generative capabilities of LLMs with a retrieval mechanism that queries helpful information from an external knowledge base, such as a database or a vast text corpus. This approach dynamically incorporates retrieved information into the generation process (cf. Figure 4). As a result, RAG enhances the LLM's ability to handle tasks that demand task-specific knowledge beyond what the model has previously learned. This integration allows for a more precise and useful response generation, especially in scenarios where up-to-date or specialized knowledge is required.

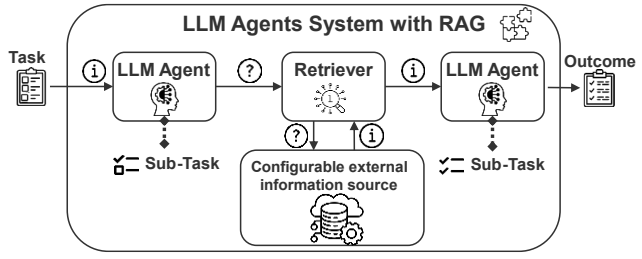

Figure 4 Integration of Retrieval-Augmented Generation (RAG) in LLM agents system

F. ***Conclusion: why use LLM for semantic processing***

Leveraging LLM is a promising approach for ensuring "data being communicated" is "clearly defined" and "disambiguously interpreted". Firstly, LLMs are trained to interpret semantics from extensive training on diverse texts and generate language that aligns with human understanding, and this capability is evidenced by the identification of specialized semantic neurons [28] and reasoning circuit [27]. Secondly, the nuanced interpretation of texts with model internal semantic neuron structures and the ability to distinguish between various meanings enable LLMs to disambiguate terms and phrases effectively. Furthermore, the application of techniques such as text embedding for capturing semantic relationships and generative next-token-prediction for text production allows for precise articulation of concepts in text. Finally, by designing LLM-agent systems and incorporating the mechanism of Retrieval-Augmented Generation (RAG), an information processing pipeline can be created for task-specific applications, i.e., the automation of AAS-instance model creation from raw textual data.

V. DESIGN AND METHODOLOGY[2]

To handle the semantics of a given piece of information, we propose a data structure unit called "semantic node", designed to encapsulate various dimensions of meaning. Building on this foundation, we develop the use case and detail the system design.

A. *"Semantic Node"*

The proposed "semantic node" is visualized in Figure 9 and specified in Table 2. The motivation for this is two folds: Firstly, text makes meaning. We want to create an atomic unit that materializes the semantic information that serves to our semantic communication proposition (c.f. Section III), wherein each text field contributes to defining an aspect of the semantics; Secondly, this data structure facilitates software implementation, and can be independently extracted, modified, evaluated, and converted through the information processing pipeline.

[2] A visualized demo of web-application can be accessed here.

Table 2 Specification of the elements in conceptualized data structure "Semantic Node"

Semantic node	
Name	A concise and specific title assigned to the semantic node. It should be concise yet descriptive enough to convey the essence of the feature at a glance.
Conceptual definition	A definition of what the semantic node represents.
Usage of data (Affordance)	Describes the potential usage and application of the semantic node.
Value	The actual data that is described by the semantic node.
Value type *optional*	The type of data used for representing the actual data. (by default: String)
Unit *optional*	The measurement unit associated with the value. (if applicable)
Source description *optional*	This is an extended explanation that situates the semantic node within the specific context of its source, making the semantic node contextually traceable. (require extra information input)

The semantic node is an atomic unit to capture the meaning of a piece of information. This structure is processed by a system constituted of LLM-agents, which synthesizes its elements to enrich the node with specific meanings. The most semantically rich information is considered to be the name, the conceptual definition and the use of data (affordance). The value shall be extracted from the source texts to ensure the nodes' authenticity. Furthermore, the structure includes three optional fields: a source description that enriches background context but requires additional data input; and the value type and unit are not indispensable, because they contribute less to defining the meaning and can often be inferred from other elements (entailment).

This semantic node is a conceptualized structure that serves our theoretical frameworks. In the next sections, we use this constructed semantic node to design the software application of AAS generation software powered by LLM agents.

B. Use Case description

The designed LLM-based system serves two primary purposes: firstly, **for system users**, it facilitates the creation of AAS instances by automatically generating data models that describe the technical properties of automation components, thereby reducing manual effort in data migration. Secondly, **for research purposes**, the LLM produces semantic descriptions for technical data in text, as specified in "semantic node". These generated texts are not only integral to the AAS for conveying technical information but also serve as evidence for evaluation. By examining these generated texts, we assess the LLM-system's ability to accurately interpret specialized technical concepts and gauge the overall performance of the LLM in processing and interpreting the semantics of technical data. This dual functionality underscores the system's utility in enhancing semantic interoperability in digital twin implementations and providing a measurable framework to evaluate semantic interpretation capabilities.

Figure 5 depicts the functions of "AASbyLLM" and the

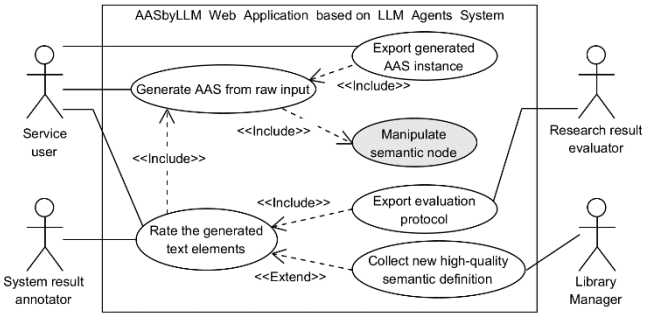

Figure 5 Use case diagram of the AASbyLLM application

interacting stakeholders in this work. The software provides a web application for users to test the thoroughness of our system and assist us (annotators and evaluators) during the evaluation process. Additionally, the validated high-quality semantic definition generated by the LLM-system can be collected by a library manager and used for enrichment of a dictionary (e.g., ECLASS).

C. System Design

The system diagram presents an overview of the components and the processing pathways involved in the handling of semantic nodes. Its primary objective is to automate the creation of a structured AAS instance model. The process begins with user interaction via a user interface, where raw data comprising technical properties is inputted. This data then undergoes semantic enrichment and manipulation by the LLM agents in the form of semantic node, eventually producing an AAS model instance based on AAS standard specifications (cf. Figure 6).

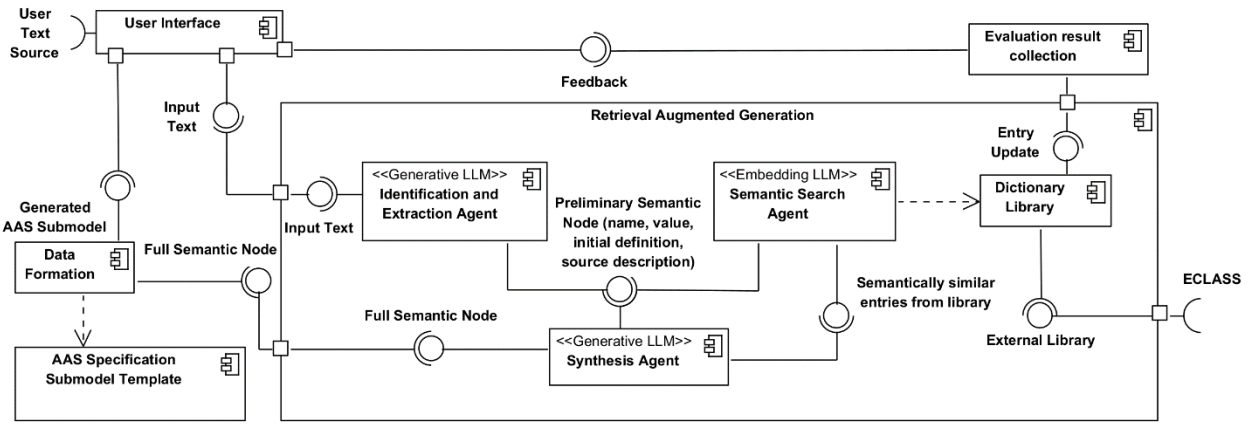

Figure 6 System components diagram of software "AASbyLLM" for AAS instance model generation with LLM agents system

User Interface: The user interface serves as the starting point and the user initializes the generation process by submitting the raw input textual data. This input may range from structured data such as spreadsheets or JSON-files to unstructured text such as copy-pasted paragraphs from a technical manual or text from a table of technical properties. These textual inputs will equally be treated as textual data.

Retrieval-Augmented Generation (RAG): After initial input, the RAG sub-system group is engaged, which combines the generative LLMs with information retrieval mechanism from an external dictionary library. This RAG subsystem consists of several LLM-agents.

- Identification and Extraction Agent: In the RAG sub-system. An LLM-agent is designed to identify and extract the name, the value, and an initial definition for a semantic node from the input text. This LLM processes the given input text and initially creates a name, definition, and contextual description for each semantic node as output, enriching the raw data with semantic details in a data structure.

- Semantic Search Agent: Following identification and extraction, this system component performs a semantic search using an embedding LLM to find semantically similar entries in the ECLASS dictionary. The search mechanism is based on our previous work [Ref.], where a vectorized embedding index, called "semantic fingerprint", is created for comparison between queried text and each ECLASS dictionary entry. The result is a list of retrieved similar definition entries from ECLASS dictionary.

- Synthesis Agent: This step incorporates the results from the semantic search into the generation process. An LLM-agent is prompted to generate a judgment of the relevance of the retrieved entries, accompanied by a short reason in text. The purpose of this step is two folds: firstly, semantic search is based on relationship of semantic similarity, which is a typical proxy metric for search but suboptimal for determining precise relevance. Inappropriate results shall be filtered out; Secondly, in this step, the generated judgement and reason serve as intermediate textual material for considering more nuanced relationships during the whole process. By instructing the LLM to judge and reason for each search result, the LLM generates more precise semantic node. After synthesis, a complete semantic node is created based on RAG, ready for AAS model creation. Details about this process are illustrated in our web demo under tab "LLM reasoning details" [GitHub link].

AAS Data Formation: The system then proceeds to form the AAS model by moving the data from the synthesized semantic nodes into an AAS JSON template. The system leverages established AAS framework to store the generated texts field from semantic node for technical properties of automation components. Specifically, it fills property name in "IdShort" and extracted value in "value", and other fields in "ConceptDescription". Furthermore, we recognize the importance of using identifiable semantics to identify the meaning of data properties and assign a unique "SemanticID" for each newly generated concept definition. This process ensures that the final AAS instance model is in the correct format and adheres to the AAS standard specification, as shown in Figure 7.

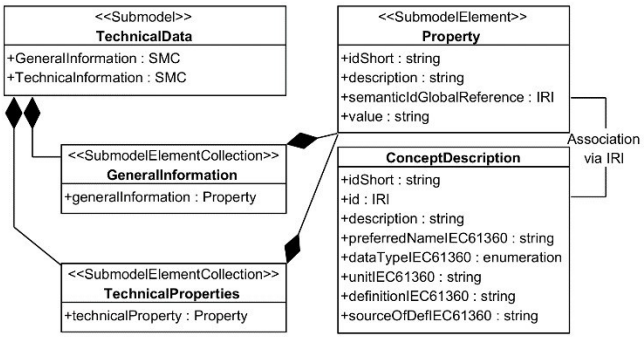

Figure 7 The AAS model elements generated and filled by the implemented software AASbyLLM

User Interface Feedback: The formed AAS model is then presented back to the user via the UI. Users can inspect the generated AAS model, assessing its accuracy and quality, and download the created AAS instance model. Meanwhile, the system collects the user rating.

Potential Dictionary Library Enhancement: It would be infeasible for a dictionary to cover all possible meanings in the world. For this specific use case, where the ECLASS-entries in some cases may not be perfectly aligned with current properties, more accurate definitions of data property can be generated by the system. Once verified by human experts, these refined definitions can be used for library enrichment.

D. Prompt design

The LLM-agent is the elemental function component in the designed system, and it is defined and realized with a carefully written prompt. We designed a structured prompt template for instructing the behavior of generative LLM-agents. This prompt template is introduced and explained in our previous work based on insights of NLP-research. The essential constitutes and key elements of this structured prompt are introduced in Figure 8. More concrete and detailed examples are released in [GitHub-Repo].

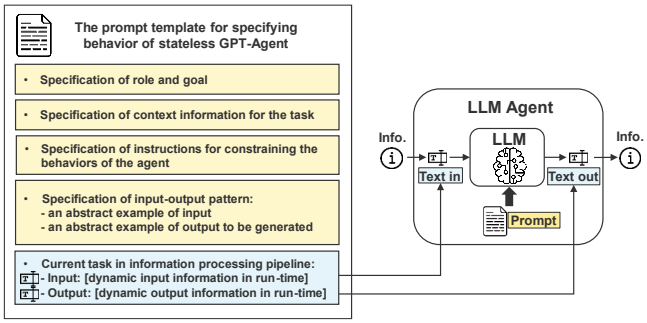

Figure 8 The proposed prompt design for creating LLM Agent

E. Model selection

We selected four models for comparison, chosen for their similar size and comparable performance. These include the proprietary commercial model *GPT-3.5* and three open-source models deployable on local GPU servers: "*Llama_2_70B_HF*", "*Mixtral_8x7B_Instruct_v0.1*", and "*WizardLM_70B_v1.0*".

"Llama_2_70B_HF" is an autoregressive language model and undergoes supervised fine-tuning (SFT) and

reinforcement learning with human feedback (RLHF) post-training to align with human preferences for helpfulness. It serves as the baseline model for our comparison of open-source models.

"Mixtral_8x7B_Instruct" [] is a model further developed from "Llama_2_7B" based on the Sparse Mixture of Experts (SMoE) architecture and fine-tuned for following instructions. It is an example of models that utilize innovative architectural designs to improve performance, enabling it to discern complex patterns and deliver more accurate predictions.

"WizardLM_70B_v1.0" is a model further fine-tuned from Llama. Its specialty lies in using synthetical complex instructions (Evol-instruct []) to train a base model solving more difficult tasks. This model exemplifies how further training on a curated dataset can enhance a model. This method is promising because it utilizes automatically generated synthetical data for scalable training of LLMs. This implies that the large language models can train themselves to become more performant.

These models are used to power the designed system. By having a comparison basis and evaluating the outcome, we can gain insights into how the model's capability and system design can have an impact on downstream task results, and how effective the AAS-generation tool is.

Evaluation

Evaluating the meaning expressed by language can be a challenge due to the unlimited combination of texts in natural language, which can result in an immeasurable number of possible expressions. To address this, we developed a human evaluation benchmark and a web-based software to support the evaluation under various configurations. We also formulated proxy metrics to comprehensively assess the system's effectiveness. The evaluation is categorized into three sections: 1) an end-to-end evaluation for practical usability, 2) a comparative analysis of capability of 4 different LLMs, and 3) an ablation study to investigate the importance of integrating external knowledge through Retrieval-Augmented Generation (RAG), with or without utilizing the external ECLASS library.

The evaluation is fundamentally linked to how well humans perceive the generated texts to be **error-free** and **informative**. Though the evaluation leans on human perception, it yields quantifiable results based on statistical principle of large number, indicating the effectiveness of our methodology (pass rate, helpfulness score, element-level rating scores, and inaccuracy rate). Ultimately, the utility of the proposed Asset Administration Shell (AAS) generation system is assessed based on the extent to which it reduces human effort, highlighting the practicality of our AAS generation solution.

The evaluation process is demonstrated with web-application hosted here [Ref.].

F. Evaluation Setups

The evaluation assesses the ability of the designed LLM-system to interpret technical concepts within an automation system. Technical properties representing the concepts were drawn from **20 automation components**, achieving diverse coverage by including 5 sensors, 5 actuators, 5 controllers, and 5 connectivity components. The text results produced by the system provide the primary evidence for its assessment.

In the comparative study, **8 design variations** (4 models × 2 mechanisms) of the system were examined. These variations involved four distinct models that have representative characteristics, each tested in two mechanism configurations: one with the integration of the external ECLASS library for Retrieval-Augmented Generation (RAG) and one without it.

7 master's students with engineering backgrounds with above-average GPA were chosen as annotators to closely resemble the user knowledge profiles. The annotators' individual knowledge level and preference are confounding factors in this experiment. To mitigate potential bias, we randomly allocated the 20 groups of technical properties among 7 annotators and examinate the statistical results calculated from large sample size.

G. Evaluation Execution with Software Support

The human evaluators are first tasked with reading the technical properties from data sheet of an automation device and making binary statement whether they comprehend the meaning of these technical properties. Then, they determine whether there are errors in the generated 5 data elements ("name", "value", "conceptual definition", "affordance", "value unit"), any **inaccuracy** in texts or **contradiction** between their understanding and the generated elements is

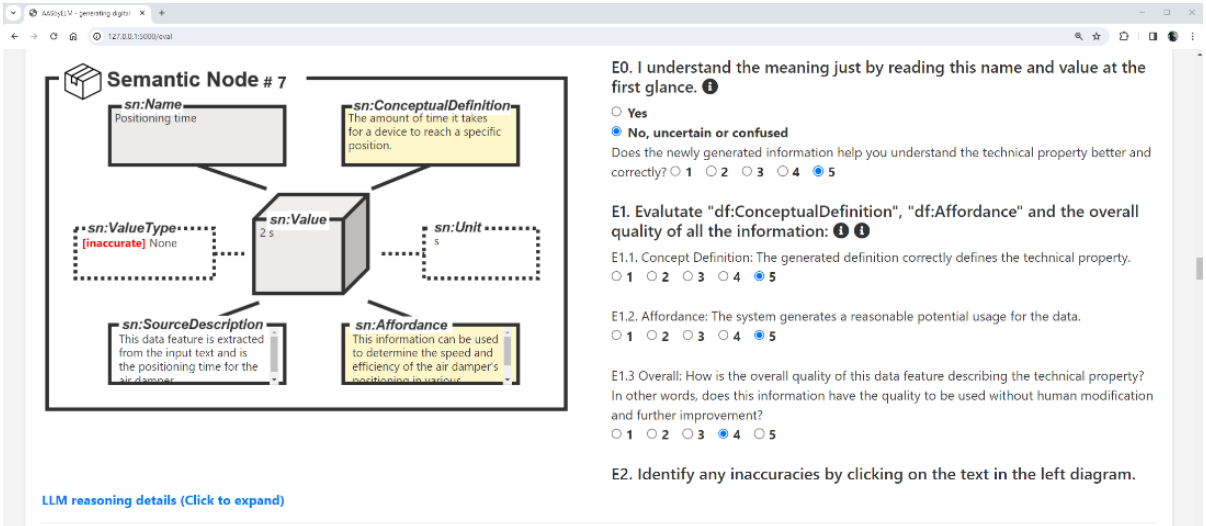

Figure 9 Screenshot of the application UI used for supporting the annotating and evaluating process

determined as an error. Then, they are tasked to rate the text quality of the two most semantic-rich elements on a scale 1 to 5: "**conceptual definition**" and "**use of data (affordance)**". Additionally, they evaluate whether the generated texts help them understand the clear meaning of the technical concept if they did not comprehend the meaning of the technical concept at the beginning ("**helpfulness**"). Finally, they determine whether the overall generated semantic node has the quality to be released without human modification and further improvement, this "**overall**" perceived quality is also on a scale 1 to 5. This evaluation process is visualized in Figure 9.

The evaluation relies on the statistical large number to yield valid results. During the evaluation, a total of 2,259 technical property samples across all the 20 selected automation components by using 4 models and 2 design mechanisms were assessed. Derived from making 10 distinct rating decisions for each individual technical property, this comprehensive assessment resulted in a collection of 22,590 total evaluation data points.

The details of the evaluation are released [here][3], including raw data, software web-application, evaluation raw results, calculation spreadsheets, and the analyzed results.

H. *End-to-End Evaluation*

For the end-to-end evaluation of the system designed to automate the generation of Asset Administration Shell (AAS) models using Large Language Models (LLMs), two distinct metrics are proposed: **the effective generation rate** (pass rate) and **helpfulness** (ability to provide implicit knowledge).

1) *Effective Generation Rate*

The end-to-end evaluation aims to measure the capability of the designed LLM-system to generate semantic information units, referred to as a "semantic node". This end-to-end evaluation metric is termed "pass rate" or "percentage of effective generation", which measures if the generated result is **error-free** and **informative**. For a sample to achieve the "passed" status:

$$Passed(S) = NoInaccurateText(S) \wedge (DefinitionRating(S)==5) \wedge (AffordanceRating(S)==5) \wedge (OverallRating(S)==5)$$

The pass rate is a proxy metric for effectiveness of the system. indicating the readiness of generated information

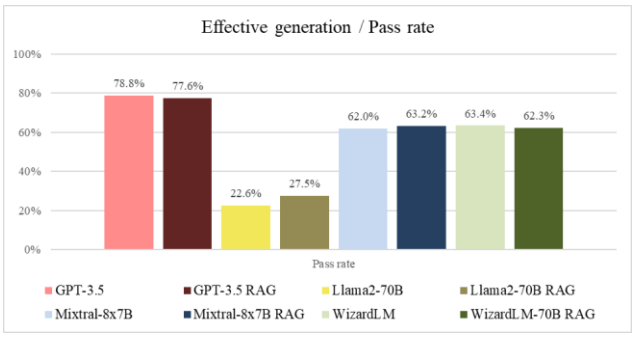

Figure 10 End-to-end pass rate of the generated instance model under different system configurations, a passed case means that the generated semantic node of a technical property is ready for release without the need for further human modifications.

models for practical application without the need for further human modifications.

2) *Helpfulness Score*

In the technical domain, original texts often assume that the reader possesses prior knowledge of specialized concepts, which can lead to confusion and misunderstandings. This issue of "**implicit knowledge assumption**" becomes prominent when the presented data is inherently complex or incomplete, relying on a foundation of technical knowledge of an agent (machine/human) to correctly interpret the data's meaning.

During human evaluation, the annotator first assesses whether the name and value of the technical property alone provide a clear sense making. If not, the system then reveals the generated texts for review. The evaluator then scores the overall helpfulness of these elements in semantic node. Based on their effectiveness in clarifying the meaning and enhancing understanding[4], the annotator rates on a scale from 1 to 5.

The helpfulness proxy metric evaluates the system's ability to provide this implicit knowledge, i.e., to generate texts that aid human users in comprehending the clear meaning of technical properties beyond just name and value. This helpfulness proxy metric is presented in Figure 11, together with other detailed metrics (overall quality, definition text quality, affordance text quality)

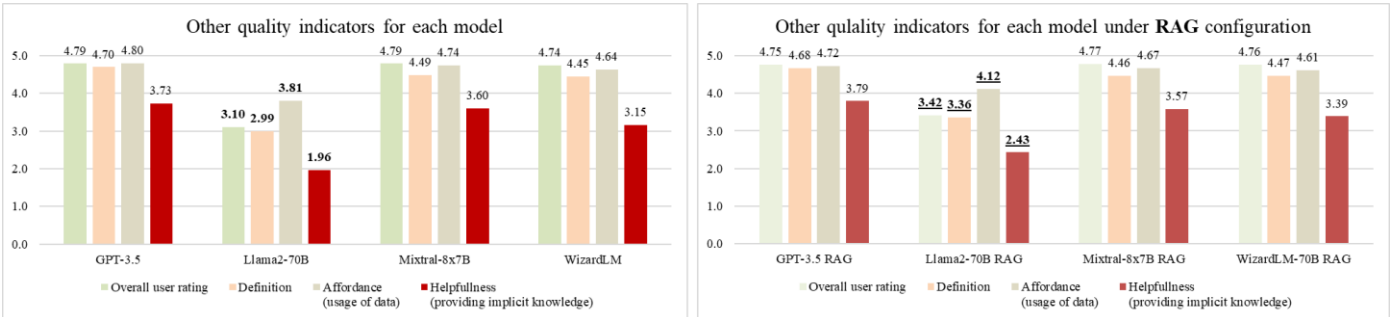

Figure 11 Comparative system performance across different configurations with and without RAG mechanism. Each group represents the performance of a system powered by a specific model, with each color indicating a proxy metric for a particular quality indicator. The comparison between the two diagrams demonstrates the effectiveness of the RAG mechanism.

[3] GitHub Link: AASbyLLM

[4] Specifically, the annotator is prompt to answer the following question: "On a scale of 1 to 5, does the newly generated information help you understand the technical qproperty better and correctly?"

$$Helpfulness\ Score = \frac{\sum Rating\ of\ each\ confusing\ case}{\#\ confusing\ cases}$$

This proxy metric reflects the **knowledge capability** of an LLM-powered system, specifically, the capability to determine and produce textual data aligned to specialized concepts. From a communications perspective, it indicates that even if the sender makes implicit assumption and simplify the data being communicated to merely a name and value (not loss-free), recipients can still accurately interpret the intended semantics, as long as the recipient has the knowledge to determine the semantics. Additionally, this metric highlights the LLM's ability to add informative explanation and help humans in understanding specialized technical data in scenarios of **non-loss-free communication**.

3) Detailed view on inaccurate generation

Figure X details the inaccurate generation ratio. The Llama2 based model cannot reliably generate the meaning and usage of the data, as indicated by the "definition_inaccurate" and "affordance_inaccurate" bars. While the other models rarely generate incorrect texts.

The inaccurate generation ratio in general correlates with the previous evaluation results. RAG only has significant effects for Llama-2 model.

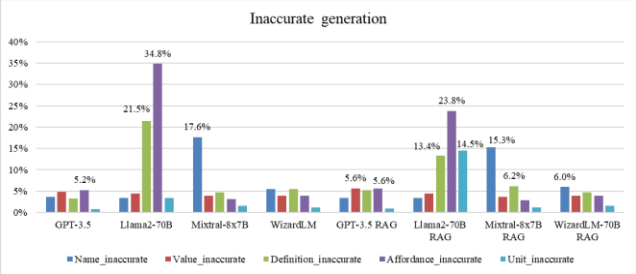

Figure 12 Comparative analysis of inaccurate generation rates by different LLMs with and without RAG

I. *Ablation study of RAG mechanism*

Retrieval augmented generation is seen as a promising mechanism to enhance the performance of LLM-system by dynamically incorporating queried data from external knowledge base during the text generation process. To evaluate the RAG mechanism's effect, we conducted a controlled experiment comparing two configurations of our LLM system: one with RAG based on ECLASS-dictionary data definitions, and another where RAG was disabled. The impact of the Retrieval-Augmented Generation (RAG) mechanism on the overall performance of each model varies, revealing distinct patterns. The diagram in Figure 13 visually demonstrates this effect.

1) RAG has significant effect for weaker model

LLAMA-2, which initially had the lowest effective generation rate, shows significant improvement with the introduction of RAG. We calculate the statistical significance with unequal variances t-test (Welch's t-test) for each metric based on two configurations (with RAG or without RAG) with a significance threshold of 0.05. All metrics (helpfulness, overall rating, generated definition rating, affordance rating) indicate significant improvement. The t-test result is released on the GitHub Repo[5], together with other detailed calculation processes and results during the statistical analysis in a spreadsheet.

2) RAG has <u>no</u> significant effect for stronger models

Surprisingly, for the other three stronger models (GPT3.5, Mixtral, WizardLM), the experiment results indicate no significant effect of RAG on enhancing or degrading the quality of the generated texts on all the metrics. This observation suggests that the stronger models may already possess the necessary knowledge to excel in this particular task, making additional information from the RAG mechanism redundant. An explanation is that these models have effectively learned the relevant technical concepts during their training phase, indicating that a saturation point has been reached where RAG's contribution of new knowledge does not further enhance model performance. It can also be explained by the robustness of these models for not being affected by similar but not fully relevant information.

3) Implications for using RAG

An overall intuition that can be drawn from this evaluation is that RAG can enhance the performance of weaker models, but it may not surpass the performance of fine-tuned stronger models.

This dynamic can be metaphorically characterized as "**cheat sheet effect**" of RAG, where weaker language model can be improved with external help, while better model does not need to.

While RAG can guide the style and direction of the output by incorporating text from external dictionaries, the semantic similarity search component may also introduce noise by fetching **similar yet irrelevant** information from the external database, affecting the final output's relevance and accuracy. Furthermore, another drawback of the RAG is its lower efficiency. The implemented RAG mode statistically consumes 4.2 times the processing duration, as it needs to process more integrated texts and more numbers of model invocations to generate the final response.

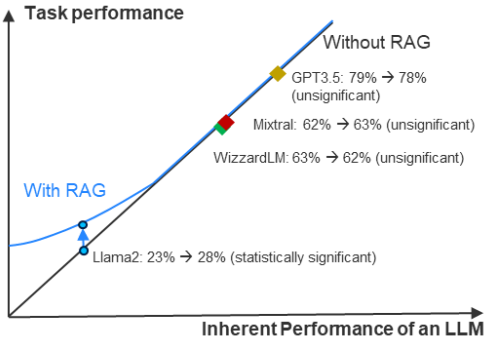

Figure 13 The graphical illustration of the hypothesis how RAG affect the performance of LLM system in technical concepts understanding based on our observed results. The lines are drawn based on assumed interpolation and extrapolation. Without RAG, a LLM only uses its inherent knowledge to execute task, task performance shall be equal to the performance of the LLM (the 1:1 line); with RAG, significant boosting effect has been observed for weaker model Llama2 in our experiment, while no significant effect has been observed for stronger models.

[5] GitHub Link: AASbyLLM

From these findings and the assumption that these stronger models should have learned the conceptual meaning of technic specific concepts, we can formulate a hypothesis:

> *The effectiveness of the RAG mechanism is reliant upon its ability to introduce new, previously unknown knowledge to an LLM.*

This hypothesis underscores the importance of carefully considering the deployment of the RAG mechanism, particularly in determining when its application is beneficial. It suggests that the decision to employ RAG should be based on an assessment of the LLM's existing knowledge base and the specific requirements of the task at hand. Furthermore, this insight can guide the strategic development of RAG datasets, ensuring they are tailored to fill knowledge gaps in the LLMs, thereby maximizing the utility and efficiency of the model's performance on specific tasks.

VI. Discussion of Potential Impacts

In this section, we discuss the implications and potential impact of our work, exploring how it contributes to the research community and the broader field of industrial digitalization.

A. Streamlining Digital Transformation with LLMs

The use of LLM to automate AAS-instances generation provides technological catalysator for information exchange in manufacturing and digital twin technologies. This approach not only simplifies the creation of standardized digital representations for assets but also facilitates seamless data sharing and utilization across platforms.

B. Semantic Interoperability based on LLM "Adapter"

Semantic interoperability, the ability for different systems to understand and interpret data consistently, is crucial for seamless communication and integration across diverse technological platforms. In a broader sense, the LLMs are integrated into a data processing pipeline and act as an interpretive layer. LLMs enable a more intuitive and accurate exchange of information between disparate data sources and systems. This capacity to interpret, translate, and connect data across various domains and formats is key to achieving seamless interoperability and integration.

C. Enhancing Standards and Optimization through LLMs

Based on our experience during the experimenting, the static data definition in ECLASS alone often falls short in covering all the dynamic need for accurately annotating a given technical properties. LLMs emerge as a powerful solution to this challenge, offering the ability to dynamically determine semantics and generate meaningful definition of data.

Therefore, the relationship between AAS, ECLASS dictionary, and LLMs becomes synergistic. LLMs can facilitate the continuous improvement and adaptation of ECLASS standards, ensuring they remain relevant and comprehensive to cover a large number of possible technical concepts. The dynamic semantic interpretation and generation capability of LLMs is also crucial for simplifying deployment and maintaining utility of digital twins across various industrial applications, paving the way for a more interconnected and intelligent data exchange.

VII. Conclusion

This paper presents a comprehensive exploration into the automated generation of Asset Administration Shells using Large Language Models, with a focus on enhancing semantic interoperability within the framework of Industry 4.0. This approach not only facilitates the automated generation of AAS models to reduce human effort and implementation cost in creating AAS instances. By introducing a "semantic node" data structure to capture the essential semantics of a piece of information and developing a novel approach that leverages LLMs for the semantic translation of technical properties in AAS model, the system translates technical data into AAS models with semantic enhancement. Furthermore, the semantic definitions generated by the LLM system possess the capability to dynamically annotate information, and the resulting conceptual definition can be verified by human and be used to form a dictionary collection or enrich an external dictionary such as ECLASS.

The evaluation of various LLM configurations and the impact of the Retrieval-Augmented Generation (RAG) mechanism has yielded detailed insights into the deployment of LLM systems in this context. Our research demonstrates that LLMs can substantially streamline the AAS instance creation process, achieving up to a 78% pass rate in generating error-free and informative AAS model elements. This suggests that a significant portion of the effort typically required for crafting AAS models could be shifted towards more efficient validation processes. Open-source models deliver sufficient performance for this application and can be fairly used for commercial use. However, our comparative analysis indicates that the effectiveness of these models varies, depending on their acquired knowledge during training and the strategic integration of external semantic resources (RAG based on ECLASS dictionary). Based on our observations, we propose a new hypothesis regarding the "cheat-sheet effect of RAG" conjecture, which we plan to explore further. This paper serves as a proof of concept for using LLMs to automate the generation of AAS instances. The research result also lays the groundwork for further collaboration with research and industrial sectors to amplify our findings' value and practicality in the context of industrial digitalization and smart manufacturing.

References

[1] S. Bader *et al.*, "Details of the asset administration shell part 1 - the exchange of information between partners in the value chain of industrie 4.0 (Version 3.0 RC02)," 2022.

[2] Y. Xia, N. Jazdi, and M. Weyrich, "Automated generation of Asset Administration Shell: a transfer learning approach with neural language model and semantic fingerprints," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–4. doi: 10.1109/ETFA52439.2022.9921637.

[3] F. Ocker, C. Urban, B. Vogel-Heuser, and C. Diedrich, "Leveraging the Asset Administration Shell for Agent-Based Production Systems," *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 837–844, Jan. 2021, doi: 10.1016/J.IFACOL.2021.08.186.

[4] D. and K. C. and D. J. Schnauffer Georg and Görzig, "Asset Administration Shell for the Wiring Harness System," in *Advances in Automotive Production Technology – Towards Software-Defined Manufacturing and Resilient Supply Chains*, F. and A. C. and H. D. Kiefl Niklas and Wulle, Ed., Cham: Springer International Publishing, 2023, pp. 324–332.

[5] H. Eichelberger and C. Niederée, "Asset Administration Shells, Configuration, Code Generation: A power trio for Industry 4.0 Platforms," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–8. doi: 10.1109/ETFA54631.2023.10275339.

[6] T. Miny, M. Thies, U. Epple, and C. Diedrich, "Model Transformation for Asset Administration Shells," in *IECON 2020 The 46th Annual Conference*

of the IEEE Industrial Electronics Society, 2020, pp. 2207–2212. doi: 10.1109/IECON43393.2020.9254649.

[7] M. Platenius-Mohr, S. Malakuti, S. Grüner, J. Schmitt, and T. Goldschmidt, "File- and API-based interoperability of digital twins by model transformation: An IIoT case study using asset administration shell," *Future Generation Computer Systems*, vol. 113, pp. 94–105, 2020, doi: https://doi.org/10.1016/j.future.2020.07.004.

[8] M. Platenius-Mohr, S. Malakuti, S. Grüner, and T. Goldschmidt, "Interoperable Digital Twins in IIoT Systems by Transformation of Information Models: A Case Study with Asset Administration Shell," in *Proceedings of the 9th International Conference on the Internet of Things*, in IoT '19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3365871.3365873.

[9] J. Zhao *et al.*, "A Semi-Automatic Approach for Asset Administration Shell Creation from Heterogeneous Data," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3673–3679, 2023, doi: https://doi.org/10.1016/j.ifacol.2023.10.1532.

[10] L. M. V. Da Silva, A. Köcher, M. S. Gill, M. Weiss, and A. Fay, "Toward a Mapping of Capability and Skill Models using Asset Administration Shells and Ontologies," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–4. doi: 10.1109/ETFA54631.2023.10275459.

[11] Y. Huang, S. Dhouib, L. P. Medinacelli, and J. Malenfant, "Enabling Semantic Interoperability of Asset Administration Shells through an Ontology-Based Modeling Method," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, in MODELS '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 497–502. doi: 10.1145/3550356.3561606.

[12] C. Schmidt, F. Volz, L. Stojanovic, and G. Sutschet, "Increasing Interoperability between Digital Twin Standards and Specifications: Transformation of DTDL to AAS," *Sensors*, vol. 23, no. 18, 2023, doi: 10.3390/s23187742.

[13] N. Braunisch, M. Ristin-Kaufmann, R. Lehmann, M. Wollschlaeger, and H. W. van de Venn, "Generation of Digital Twins for Information Exchange Between Partners in the Industrie 4.0 Value Chain," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–6. doi: 10.1109/INDIN51400.2023.10218306.

[14] S. Rongen, N. Nikolova, and M. van der Pas, "Modelling with AAS and RDF in Industry 4.0," *Comput Ind*, vol. 148, p. 103910, 2023, doi: https://doi.org/10.1016/j.compind.2023.103910.

[15] Y. Huang, S. Dhouib, L. P. Medinacelli, and J. Malenfant, "Semantic Interoperability of Digital Twins: Ontology-based Capability Checking in AAS Modeling Framework," in *2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2023, pp. 1–8. doi: 10.1109/ICPS58381.2023.10128003.

[16] J. Fuchs, J. Schmidt, J. Franke, K. Rehman, M. Sauer, and S. Karnouskos, "I4.0-compliant integration of assets utilizing the Asset Administration Shell," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1243–1247. doi: 10.1109/ETFA.2019.8869255.

[17] T. and A. A. and S. A. L. and A. A. Moreno Tomás and Sobral, "Semantic Asset Administration Shell Towards a Cognitive Digital Twin," in *Flexible Automation and Intelligent Manufacturing: Establishing Bridges for More Sustainable Manufacturing Systems*, L. P. and S. J. C. and P. M. T. and P. C. M. A. Silva Francisco J. G. and Ferreira, Ed., Cham: Springer Nature Switzerland, 2024, pp. 679–686.

[18] A. Lüder, A.-K. Behnert, F. Rinker, and S. Biffl, "Generating Industry 4.0 Asset Administration Shells with Data from Engineering Data Logistics," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020, pp. 867–874. doi: 10.1109/ETFA46521.2020.9212149.

[19] S. Cavalieri and M. G. Salafia, "Insights into Mapping Solutions Based on OPC UA Information Model Applied to the Industry 4.0 Asset Administration Shell," *Computers*, vol. 9, no. 2, 2020, doi: 10.3390/computers9020028.

[20] J. Beermann, R. Benfer, M. Both, J. Müller, and C. Diedrich, "Comparison of Different Natural Language Processing Models to Achieve Semantic Interoperability of Heterogeneous Asset Administration Shells," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–6. doi: 10.1109/INDIN51400.2023.10218154.

[21] M. Both, J. Müller, and C. Diedrich, "Automated mapping of semantically heterogeneous I4.0 asset administration shells by methods of natural language processing," *At-Automatisierungstechnik*, vol. 69, no. 11, pp. 940–951, Nov. 2021, doi: 10.1515/AUTO-2021-0050/MACHINEREADABLECITATION/RIS.

[22] A. Cartus, M. Both, M. Nicolai, J. Müller, and C. Diedrich, "Interoperability of semantically heterogeneous digital twins through Natural Language Processing methods ," *CLIMA 2022 conference*, May 2022, doi: 10.34641/clima.2022.143.

[23] M. Both *et al.*, "Automated monitoring applications for existing buildings through natural language processing based semantic mapping of operational data and creation of digital twins," *Energy Build*, vol. 300, p. 113635, Dec. 2023, doi: 10.1016/J.ENBUILD.2023.113635.

[24] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to Generate Reviews and Discovering Sentiment," Apr. 2017, Accessed: Feb. 22, 2024. [Online]. Available: https://arxiv.org/abs/1704.01444v2

[25] A. Rogers, O. Kovaleva, and A. Rumshisky, "A Primer in BERTology: What We Know About How BERT Works," *Trans Assoc Comput Linguist*, vol. 8, pp. 842–866, 2020, doi: 10.1162/TACL_A_00349.

[26] W. Gurnee, N. Nanda, M. Pauly, K. Harvey, D. Troitskii, and D. Bertsimas, "Finding Neurons in a Haystack: Case Studies with Sparse Probing," May 2023, Accessed: Feb. 22, 2024. [Online]. Available: https://arxiv.org/abs/2305.01610v2

[27] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, "Zoom In: An Introduction to Circuits," *Distill*, vol. 5, no. 3, p. e00024.001, Mar. 2020, doi: 10.23915/DISTILL.00024.001.

[28] S. Bills *et al.*, "Language models can explain neurons in language models," *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2023.

[29] Y. Xia, M. Shenoy, N. Jazdi, and M. Weyrich, "Towards autonomous system: flexible modular production system enhanced with large language model agents," Apr. 2023, doi: 10.1109/ETFA54631.2023.10275362.