Model-Driven Integration of Domain Knowledge into Machine Learning Workflows: A Case for Multidisciplinary System Design

1st Neeraj Katiyar
Hitachi Energy Research
Montreal, Canada
neeraj.katiyar@hitachienergy.com

2nd Jhelum Chakravorty
Hitachi Energy Research
Montreal, Canada
jhelum.chakravorty@hitachienergy.com

***Abstract*—Complex multidisciplinary energy systems, such as gas turbines, and power systems involve several interrelated subsystems, each designed by special engineering teams with dedicated domain expertise. The rapid adoption of machine learning (ML) in the design and manufacturing of such systems introduces several software engineering challenges. An important challenge is how to incorporate engineering knowledge from domain experts into a machine learning workflow in a systematic and (semi-)automated way. This paper presents a vision towards a model-driven approach to address this challenge by capturing domain knowledge using knowledge graphs. Using gas turbines as a use case, we propose a high-level architecture that supports the iterative evaluation of ML models through automated performance reporting enriched with domain insights.**

***Index Terms*—Artificial intelligence, Software engineering, knowledge engineering, learning systems, machine learning, knowledge graph, meta modeling, model-driven software engineering.**

I. Introduction

Gas turbines are the commonly used type of internal combustion engine used for power generation. The design and development of such engines usually involve expertise from mechanical engineers for various subsystems such as the secondary air system, turbine, compression, combustion, and fan. Hence, this results in a considerable amount of engineering knowledge that various teams within the project could exploit. Figure 1 gives a brief overview of information flow between subsystems in aero-derivative gas turbines.

With the rapid innovation in artificial intelligence, industries have started integrating machine learning (ML) workflows in multidisciplinary systems' design and space exploration. In big research organizations, ML practitioners are responsible for the proof of concept (PoC) development of surrogate models for various components of such complex multidisciplinary systems. Due to the lack of a standardized process for domain-knowledge gathering, repetitive discussions are inevitable among domain experts and ML practitioners throughout the ML workflow for various purposes such as model requirement understanding, conforming to system-specific constraints, respecting operating conditions, and performance metrics selection. Therefore, optimization of ML models and evaluation of performance can be a very time-consuming process.

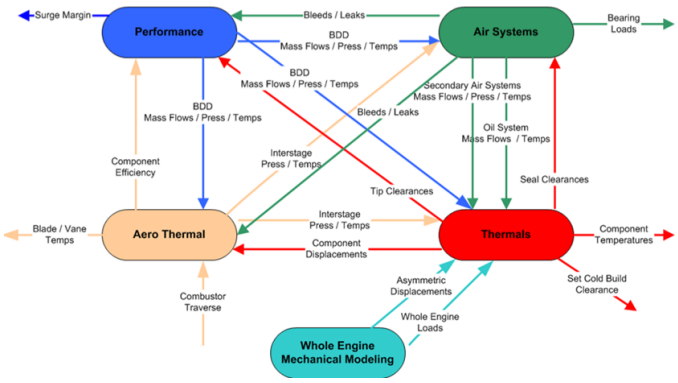

Fig. 1. Inter-relatedness of engine subsystems and models. Arrows highlight information passed between the subsystems [1].

Since model-based software engineering (MBSE) techniques have been applied successfully to many large-scale industrial applications ([2]), our research hypothesis is that a model-driven approach could effectively aid software engineering challenges in integrating the ML workflow for the design and space exploration of such multidisciplinary systems. Since gas turbines reflect an ideal multidisciplinary complex system, in this vision paper we have proposed our vision conceptually applied to the gas turbines as a use case.

We see two key challenges in the systematic and automated integration of the ML workflow in gas turbine design exploration, which could be improved via knowledge graph: *domain knowledge* and *performance reporting* as summarized below:

- *Domain knowledge gathering*: In multidisciplinary systems like gas turbines, domain knowledge is often soiled within specialized teams—mechanical, thermal, control systems, and materials engineering-each using their own tools, terminologies, and assumptions. This knowledge is typically embedded in design documents, simulation models, spreadsheets, and even informal conversations. Capturing and structuring this knowledge for use in ML workflows is a major challenge. ML engineers often lack the deep domain expertise required to interpret this

information correctly, leading to frequent back-and-forth with domain experts. This not only slows down the development cycle but also increases the risk of misinterpretation or loss of critical insights. A systematic approach to gathering and formalizing domain knowledge—such as through knowledge graphs—can help bridge this gap by making expert knowledge machine-readable, reusable, and traceable across the design process.

- *Performance reporting:* ML engineers often face significant friction when evaluating model performance in the context of complex engineering systems. Unlike standard ML benchmarks, performance metrics in gas turbine design are highly domain-specific, often derived from physical laws, safety constraints, or long-term operational goals. These metrics are not always readily available or formally defined—they may reside in expert intuition, legacy documents, or proprietary simulation tools. As a result, ML practitioners must engage in repeated, time-consuming discussions with domain experts to understand which metrics matter, how to compute them, and how to interpret results. This manual process not only delays iteration cycles but also hampers reproducibility and traceability. Automating performance reporting through a structured, knowledge-driven approach can streamline this process, enabling faster, more reliable integration of ML into engineering workflows.

This paper addresses in further detail several relevant use cases where knowledge graphs can be leveraged for systematically capturing the key concepts and generation of domain-focused performance reports.

II. Research Questions

To address the software engineering (SE) challenges identified in Section III, we define two research questions (RQs), each with specific objectives and corresponding conceptualized solutions proposed in this paper.

RQ1

How can we exploit modeling to standardize the process of knowledge acquisition for ML projects in a large, multi-disciplinary industrial setting?

This RQ is motivated with the challenge of "Ineffective Use of Engineering Knowledge" discussed in Section Section III, where domain knowledge is fragmented across teams and tools.

Obj 1.1: Analyze current practices in domain knowledge acquisition to inform the design of a structured modeling approach.

Obj 1.2 Propose a meta-model that formalizes requirements, engineering insights, and data characteristics to support traceable and reusable knowledge integration.

To achieve Obj 1.1 and Obj 1.2, we analyzed current ML workflows in multidisciplinary industrial settings (see figure 2) —particularly in gas turbine design—to identify key pain points in domain knowledge acquisition. Also, We introduce a UML-based meta-model (figure 5) and knowledge graph architecture (figure 3) that captures domain-specific requirements and insights in a structured, machine-readable format. This supports consistent knowledge acquisition and reuse across ML projects.

RQ2

How we can overcome identified SE challenges when integrating ML workflow in large scale system design?

RQ2 targets the challenge of "Performance Evaluation and Reporting" (Section III), where ML practitioners face friction in aligning evaluation with engineering goals.

Obj 2.1: Define a workflow that aligns ML performance evaluation with domain-specific constraints and metrics.

Obj 2.2 Conceptualize a semi-automated reporting mechanism to reduce iteration cycles and manual effort.

To address the above objectives, we proposed a model-driven ML workflow (see figure 4) that integrates domain knowledge acquisition with performance evaluation. The workflow is designed to ensure that evaluation metrics are not generic but are instead derived from domain-specific requirements captured during the knowledge acquisition phase. We also conceptualized a semi-automated reporting tool that leverages the knowledge graph to generate domain-aligned performance reports (figure 4, point 3). While still at the architectural level, the proposed tool aims to reduce manual effort, support versioning, and streamline feedback loops between ML practitioners and domain experts

III. Challenges

The integration of ML into the design of complex, multidisciplinary systems presents unique SE challenges. In this section, we categorize these challenges into two major themes as follows, Each of these categories reflects a critical bottleneck in the current ML workflow, where the lack of structured processes and automation hinders scalability, traceability, and collaboration between ML practitioners and domain experts.

- *Ineffective Use Of Engineering Knowledge*

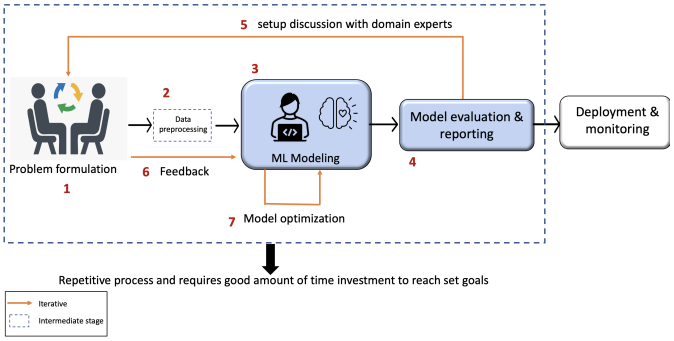

Fig. 2. Existing workflow for ML development adopted by industries

In large-scale industrial settings, engineering knowledge is often distributed across multiple teams and embedded in diverse formats—ranging from simulation models and spreadsheets to informal discussions and legacy documentation. This knowledge is typically tacit, highly contextual, and not readily accessible to ML practitioners. As a result, ML workflows often rely on generic assumptions or incomplete information, leading to suboptimal model performance and misalignment with engineering goals.

For example, in the context of gas turbine design, understanding the thermodynamic behavior of subsystems or the implications of material fatigue under varying load conditions requires deep domain expertise. Without a structured mechanism to capture and reuse this knowledge, ML engineers are forced to repeatedly consult domain experts, which is time-consuming and error-prone. This inefficiency not only slows down development cycles but also limits the reproducibility and scalability of ML solutions.

A model-driven approach—such as using knowledge graphs to formalize and interlink domain concepts—can help address this challenge by making engineering knowledge machine-readable, queryable, and reusable across projects.

- *Performance Evaluation and Reporting* The goal of performance evaluation in ML-driven engineering systems is multifaceted. A model must not only generalize well to unseen inputs but also conform to environmental constraints, handle edge cases gracefully, and demonstrate robustness under varying conditions. In safety-critical domains like aero-derivative gas turbines, these requirements are even more stringent. A model that performs well on average may still be unacceptable if it fails under rare but high-risk scenarios.

  For instance, over-prediction or under-prediction of inlet parameters—critical components that regulate airflow into the turbine—can have severe consequences. In such cases, minimizing average error (e.g., mean squared error or mean absolute error) is insufficient. Instead, domain-specific metrics that prioritize the avoidance of catastrophic outliers are essential. This highlights the need for performance evaluation to reflect business priorities and safety constraints, not just statistical accuracy.

  As shown in the Figure 2 ML practitioners in industries with multidisciplinary setting are responsible for selecting evaluation metrics, generating plots, and documenting results—often without clear or stable requirements. This process typically unfolds in three manual steps:

  – *Metric selection and result plotting:* In the absence of a structured knowledge base, practitioners default to standard ML metrics, which may not align with business or safety goals.
  – *Documentation:* Without proper versioning tools, performance results are documented manually, making traceability and reproducibility difficult.
  – *Evaluation and feedback:* It involves repeated discussions to interpret results and refine metrics, consuming significant time and efforts.

  This repetitive and time-intensive process underscores the need for a (semi-)automated system that can generate domain-specific performance statistics, support versioning, and enable efficient tracking with minimal manual intervention.

IV. Related Work

This section reviews prior work across three key areas relevant to our research as follows:

1) *Modeling for knowledge Representation*

Model-driven engineering (MDE) and knowledge graphs are increasingly used to formalize domain knowledge in complex systems. Recent work by [3] emphasizes the role of digital twins and semantic modeling in bridging the gap between domain expertise and data-driven methods. Knowledge graphs, in particular, have shown promise in representing multidisciplinary relationships and constraints in engineering systems in the survey report by [4]. Formal modeling approaches, such as domain-specific languages (DSLs), have also been proposed to reduce ambiguity and improve system safety [5]. For instance, [6] introduce a DSL for describing ML datasets, focusing on structure, provenance, and social concerns-complementing our data view that captures deployment and ML data characteristics

Similarly, Neo4j has been explored by [7] as a graph database alternative to traditional relational database management system (RDBMS) for its scalability and suitability in representing highly connected engineering data. MIT researchers [8] have also developed natural language interfaces for querying graph databases, enabling non-experts to interact with structured knowledge bases.

Additionally, [9] presents a UML-based model-driven framework for domain-specific adaptation of time series forecasting pipelines. Validated through academic and industrial case studies, this work demonstrates the practical benefits of integrating domain knowledge into ML

workflows and reinforces the importance of structured modeling in multidisciplinary domains.

2) *AutoML and ML Workflow Platforms:* AutoML platforms such as Amazon SageMaker, Google AutoML, and Azure ML have significantly streamlined the ML development lifecycle by automating tasks like data preprocessing, model selection, and hyperparameter tuning. However, these platforms are typically designed for general-purpose applications and often lack the flexibility to incorporate domain-specific constraints or performance metrics.
   [10] provide a comprehensive review of ML applications in industrial settings and emphasize that while AutoML tools are promising, their effectiveness is limited in domains where safety, interpretability, and domain alignment are critical. These tools often assume static problem definitions and do not support iterative refinement based on evolving engineering requirements.
   This gap has led to calls for more customizable and domain-aware ML workflows that can integrate expert-defined metrics, constraints, and feedback loops. Such workflows would not only improve model relevance but also enhance trust and adoption in industrial environments.
3) *ML in Industrial Systems:* ML is increasingly applied in industrial systems for predictive maintenance ([11]), anomaly detection, and design optimization ([1]). [12] identify multiple challenges (23 in total) in deploying ML at scale in industrial settings, including adaptability, scalability, and safety. These challenges are particularly relevant in multidisciplinary systems, where ML must align with strict engineering and safety requirements. The integration of ML into such systems requires not only technical robustness but also traceability and explainability. The use of knowledge graphs and structured modeling can help bridge this gap by aligning ML outputs with domain-specific goals.

V. Solution

To address the challenges outlined in Section III, we propose a model-driven workflow that integrates domain knowledge acquisition with (semi-)automated performance reporting. This workflow is illustrated in Figure 4, which outlines the key stages from knowledge gathering to model evaluation and feedback. The approach is designed to streamline collaboration between ML practitioners and domain experts, reduce iteration cycles, and improve traceability across the ML life-cycle.

The solution is structured around two core components:

1) **Knowledge Acquisition**–capturing domain-specific requirements, data insights, and engineering constraints using a structured meta-model. To support this, we use Unified Modeling Language (UML) to represent various views of the system. UML is widely adopted in both academia and industry for modeling complex software systems due to its versatility in design, analysis, and documentation tasks ([13]).

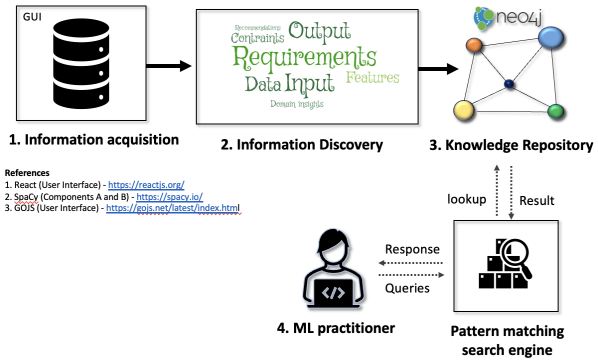

Fig. 3. Workflow for knowledge acquisition

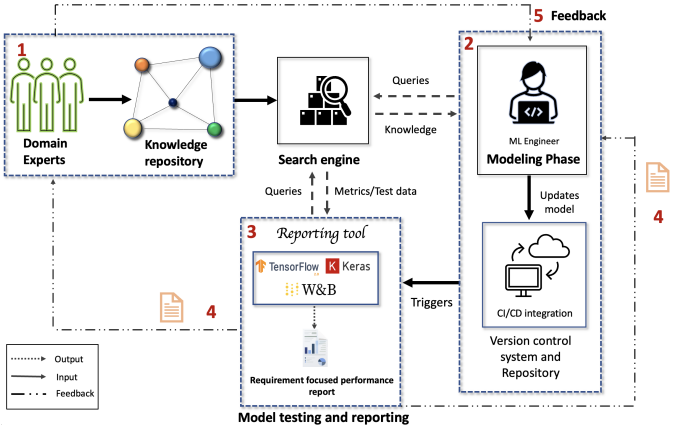

Fig. 4. Proposed ML workflow. **1**: Knowledge acquisition, **2**: ML Development, **3**: Reporting, **4**: Feedback loop with experts, **5**: Evaluation.

2) **Semi-automated Reporting** – leveraging the knowledge graph to generate performance reports aligned with domain-specific metrics and business goals.

Below we discuss the core components in detail:

1) **Knowledge Acquisition through Knowledge Graph**
   Figure 3 illustrates the proposed workflow for knowledge acquisition in the context of integrating domain expertise into ML workflows. It outlines the structured process of gathering, organizing, and formalizing domain knowledge.
   To achieve this workflow,we formalize the domain of our envisioned knowledge graph in a meta-model that shows the required concepts to capture for the effective use of domain knowledge and build a knowledge repository. This meta-model serves as the foundation for capturing requirements, engineering insights, data characteristics, and project evolution in a structured and standardized manner. This approach is inspired with prior work by [10] and [14] which has shown that formal modeling can significantly reduce ambiguity and improve traceability in complex engineering systems.
   The meta-model, illustrated in figure 5, is organized into three distinct views and one extended view (version) to track the artifacts. The first captures project

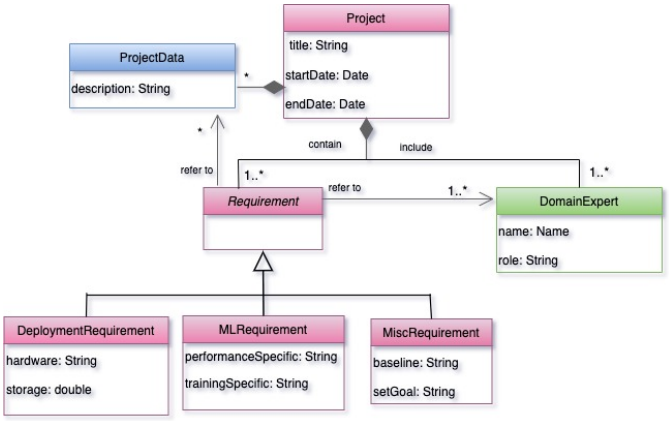

Fig. 5. Meta-model & Requirement view

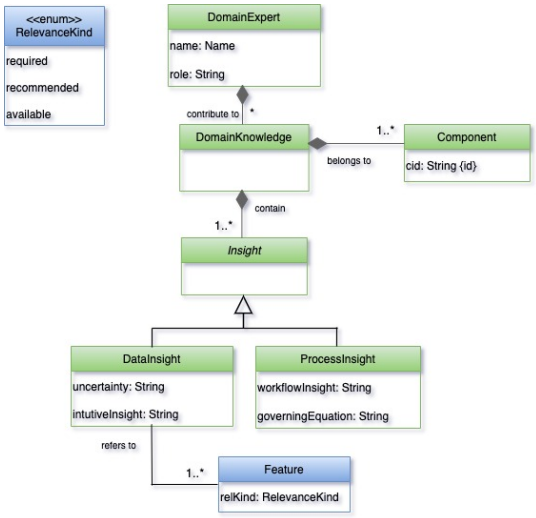

Fig. 6. Knowledge view

requirements, including *DeploymentRequirement*, *MLRequirement*, and *MiscRequirement*. The second focuses on engineering knowledge that can be formalized. The third addresses project data, detailing key information relevant to both deployment and training contexts. The final view tracks the evolution of requirements and data over time. Each of these views is discussed in detail below.

a) *Requirement view:* It helps capture the foundational needs of a project, ensuring that ML models are aligned with engineering goals and deployment realities. As shown in figure 5 the requirement class captures three essential categories of information as follows:
   i) DeploymentRequirement: This class captures all constraints and expectations related to deploying the ML model in a real-world industrial setting. It includes operational constraints, integration needs, performance expectations, and hardware/software dependencies.
   ii) MLRequirement: It focuses on the technical and algorithmic aspects of the ML model itself. It includes model objectives, data requirements, training constraints, evaluation metrics beyond standard ML metrics like accuracy or MSE, and Interpretability & Explainability.
   iii) MiscRequirement: This category captures additional project-specific or organizational requirements that don't fall neatly into the other two categories. It may include: regulatory compliance, documentation needs, collaboration protocols, and Security & Privacy requirements.

b) *Knowledge view:*
   The primary objective of this view is to capture the concepts of *DataInsight* and *ProcessInsight* as depicted in figure 6.
   The *DataInsight* class encapsulates critical information related to the data used in the project. This includes aspects such as data uncertainty (e.g., limited feature availability) and intuitive insights (e.g., interrelationships among features). It references the *Feature* class to contextualize these insights.
   Domain experts may contribute to one or more domains (e.g., machine learning, mechanical engineering), and these domains can span various system components. For instance, combustion and compression are treated as distinct components, each potentially requiring specialized domain knowledge.

c) *Data view:* In most of the projects related to multidisciplinary systems, the deployment data abbreviations are different from ML deployment data. Hence, this view depicted in figure 7 captures this information within *DeploymentData* and *MLData* concepts. *DeploymentData* captures insights such as `featureList` (e.g. list of features specific to the deployment of the model in production) and `deployment-realization` (records the mapping of feature labels with labels being used in deployment). MLData concept is for understanding labels (for the features), keyIdentifier (in gas turbine design, a unique io key is being used to map input and output; hence capturing these io keys helps in mapping labels and data cleaning) and feature list (this list includes the available features for training purpose). MLData class can include multiple DataSet (e.g. in case of multiple data sources). Furthermore, each DataSet is composed of many Feature which captures the feature value and its kind (required, recommended, or available). The *Feature* class is associated with both input and output roles with *DataSet* under an XOR constraint, indicating that a feature can serve as either an input or output, but not both simultaneously.

d) *Versioning view:* Understanding progressive modi-

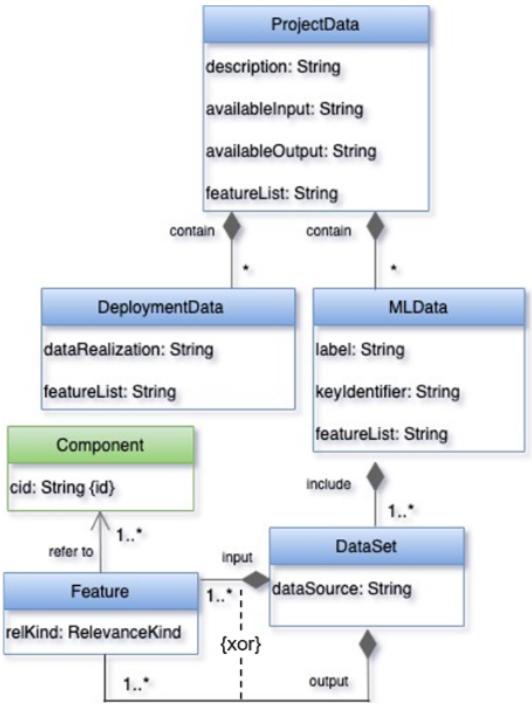

Fig. 7. Data view

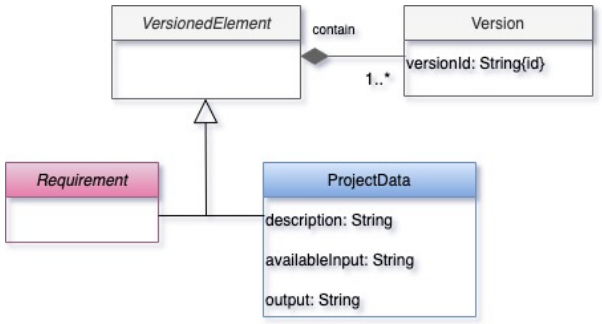

Fig. 8. Versioning view

fication of requirements and data within the project helps the team in various ways (e.g. tracking the evolution of model assumptions, comparing performance across different data versions, and ensuring reproducibility of results). Hence as shown in figure 8 *VersionedElement* is generalized to *ProjectData* and *Requirements* classes to captures the evolution of project data and various requirements as the project proceeds. *Version* concept is introduced to capture the `versionId` of each element within the composing concepts.

2) **(Semi-) Automated Report Generation**
As discussed in the earlier section, improvements in model performance beyond a certain threshold often yield diminishing business value. In multidisciplinary domains such as aero-derivative systems, iterative discussions with domain experts are essential to align model evaluation with system-specific requirements. Recent advancements in ML automation—such as AutoML, Amazon SageMaker, DVC, and W&B—have inspired the integration of CI/CD practices into industrial ML workflows.
We propose leveraging the knowledge graph, introduced earlier in this section, to automate the selection of domain-specific metrics and generate performance reports. These reports not only evaluate the current model but also compare it with previously deployed versions, enabling traceable and informed decision-making. Figure 4 illustrates this high-level workflow.
To support this, we propose to develop a **report generation package tool**—a modular set of Python libraries designed to automate performance evaluation. This tool enables ML practitioners to avoid repetitive coding and empowers non-ML teams to participate in performance analysis.
The tool operates through a three-step process:

a) *Metrics selection:* This step involves extracting metrics from the knowledge graph using a graph pattern matching search engine, updating the trained model and the test data in the repository. A user could use the command line or graphical user interface (GUI) to provide inputs.
b) *Result generation and conversion into jupyter-notebook:* The next step involves invoking required performance libraries (inbuilt within the generation package tool)—further branching concepts would be used to keep track of the performance of previous models for further comparison. The output of this step is the documented report in a jupyter-notebook. W&B and DVC frameworks, discussed in the related work section, would be used to document the results in jupyter-notebook format.
c) *Result evaluation:* The last step triggers an auto email routine to share the link of the jupyter report across project managers and engineers for further analysis and investigation. This automation would avoid turnaround time in setting up a synchronized meeting, which often takes up many days to schedule.

VI. Conclusion And Future Work

Domain knowledge is the cornerstone of effective machine learning workflows in complex engineering systems. However, the current lack of structured mechanisms to capture and integrate this knowledge often results in fragmented communication, repeated manual efforts, and suboptimal model alignment with real-world constraints. This paper presented a vision for a model-driven approach that leverages knowledge graphs to formalize domain expertise and automate performance evaluation.

Our proposed solution addresses two key challenges: (1) the need for systematic knowledge acquisition from domain

experts, and (2) the automation of performance reporting to reduce turnaround time and improve traceability. By introducing a meta-model that captures requirements, engineering insights, data characteristics, and versioning, we provide a structured foundation for building reusable and traceable ML workflows. The accompanying report generation tool, integrated with CI/CD practices, further streamlines the evaluation process and democratizes access to performance insights across teams.

Looking ahead, future work will focus on implementing and validating this framework in real-world industrial environments with a multidisciplinary use cases such as gas turbines and power systems. This will include implementing the meta-model in a knowledge graph platform (e.g., Neo4j), applying it to capture domain knowledge from existing engineering documentation, and integrating it with an ML pipeline for surrogate model development. We will assess the effectiveness of the approach based on metrics such as reduction in iteration cycles, completeness of captured knowledge, and accuracy of performance reporting.

REFERENCES

[1] S. Pilarski, M. Staniszewski, F. Villeneuve, and D. Varro, "On artificial intelligence for simulation and design space exploration in gas turbine design," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion*, 2019, pp. 170–174.

[2] J. M. Borky and T. H. Bradley, *Effective model-based systems engineering*. Springer, 2018.

[3] T. A. McDermott, M. R. Blackburn, and P. A. Beling, "Artificial intelligence and future of systems engineering," in *Systems Engineering and Artificial Intelligence*. Cham: Springer International Publishing, 2021, pp. 47–59, ISBN: 978-3-030-77283-3.

[4] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne, *Knowledge graphs: Opportunities and challenges*, 2023. arXiv: 2303.13948 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2303.13948.

[5] Y. Liu and J.-M. Bruel, "Modeling and verification of natural language requirements based on states and modes," *Form. Asp. Comput.*, vol. 36, no. 2, Jun. 2024, ISSN: 0934-5043. DOI: 10.1145/3640822. [Online]. Available: https://doi.org/10.1145/3640822.

[6] J. Giner-Miguelez, A. Gómez, and J. Cabot, "A domain-specific language for describing machine learning datasets," *Journal of Computer Languages*, vol. 76, p. 101 209, 2023, ISSN: 2590-1184. DOI: https://doi.org/10.1016/j.cola.2023.101209. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590118423000199.

[7] F. M. Santos López and E. G. Santos De La Cruz, "Literature review about neo4j graph database as a feasible alternative for replacing rdbms," *Industrial Data*, pp. 135–139, 2015.

[8] C. Sun, "A natural language interface for querying graph databases," 2018.

[9] N. Katiyar, *A Model-Driven Framework for Domain-Specific Adaptation of Time Series Forecasting Pipeline*. McGill University (Canada), 2023.

[10] M. Bertolini, D. Mezzogori, M. Neroni, and F. Zammori, "Machine learning for industrial applications: A comprehensive literature review," vol. 175, p. 114 820, 2021, ISSN: 0957-4174.

[11] A. E. H. Gabsi, "Integrating artificial intelligence in industry 4.0: Insights, challenges, and future prospects–a literature review," *Annals of Operations Research*, May 2024. DOI: 10.1007/s10479-024-06012-6.

[12] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson, "Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions," vol. 127, p. 106 368, 2020, ISSN: 0950-5849.

[13] F. Ciccozzi, I. Malavolta, and B. Selic, "Execution of uml models: A systematic review of research and practice," *Softw. Syst. Model.*, vol. 18, no. 3, pp. 2313–2360, Jun. 2019, ISSN: 1619-1366. DOI: 10.1007/s10270-018-0675-4. [Online]. Available: https://doi.org/10.1007/s10270-018-0675-4.

[14] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Extracting domain models from natural-language requirements: Approach and industrial evaluation," ser. MODELS '16, ACM, 2016, ISBN: 9781450343213.