Ontology-supported AI Model and Dataset Management

Jan Novacek*, Ali Ahari*†, Tobias Müller*†, Sebastian Reiter*,
Alexander Viehl*, Oliver Bringmann*†
*FZI Research Center for Information Technology
Haid-und-Neu-Str. 10-14, 76131 Karlsruhe
†University of Tübingen
Sand 14, 72076 Tübingen

***Abstract*—Recently, there has been a great deal of research into improving AI methods and their application. The main focus is on tracking progress, enabling transparent comparisons, and fostering a more profound understanding of AI. In that process, different organizations generate and use plenty of assets that need to be tracked, traced and managed. Moreover, it is important to discover assets relevant for the task at hand. This paper presents research aiming to contribute to answering the question of what is required to exchange and manage AI models and related assets effectively without semantic gaps in an industrial context. We introduce a platform for AI model exchange, which facilitates the usage, exchange, and analysis of AI models and datasets. The platform incorporates an ontology that can foster a more profound common understanding of what is required in these tasks and help tackle the issues mentioned above. Finally, we elucidate the utility of the platform through the illustration of a use case in the context of real-time critical systems.**

***Index Terms*—Artificial Intelligence, Semantic Web, Linked Data, Ontology, Semantic Annotation**

I. Introduction

The widespread application and sheer number of existing Artificial Intelligence (AI) models demand discovering the right assets and building trust in the end-user using the models. By the time of this writing, the data collection and development of AI models is often separated from the final users of the models. However, distributed and shared use can only be achieved through a clear understanding of the functional scope of AI models. This requires a characterization of AI models, which includes basic descriptions of the models and basic information about the data, quality, and context.

In the automotive industry, for instance, AI models and related assets are transferred between different tiers along the supply chain. A predominant issue in this regard is a lack of structured metadata of AI models with explicit semantics and a common understanding of such a metadata format. Popular AI communities, like *Hugging Face*[1], for example, host numerous AI models and datasets. Information about these resources is provided by semi-structured descriptions, mainly consisting of natural language text of differing kinds of content and amounts. While some but not all models have structured metadata, the lack of them hinders the shared use and development of AI models. To tackle these issues, this work aims to answer the following research question:

> What is required to develop and use an AI model collaboratively in an industrial context?

Current state-of-the-art machine learning platforms such as *MLflow*[2], *H2O*[3] and *Ray*[4] have means to add metadata to models. However, the metadata specifications lack interoperability. This problem becomes even more apparent when models need

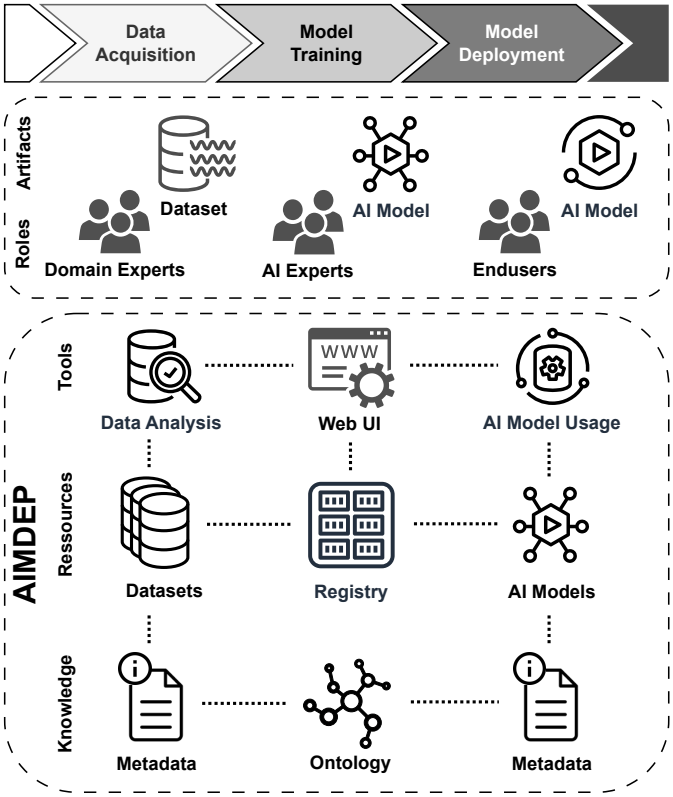

Fig. 1: Platform tools, resources, and knowledge for different machine learning development phases

[1]https://huggingface.co

[2]https://mlflow.org
[3]https://h2o.ai
[4]https://www.ray.io

This is the accepted manuscript of: J. Novacek, A. Ahari, T. Müller, S. Reiter, A. Viehl, and O. Bringmann, "Ontology-Supported AI Model and Dataset Management," in 2024 IEEE 22nd International Conference on Industrial Informatics (INDIN), pp. 1–6, 2024. The version of record is available at https://doi.org/10.1109/INDIN58382.2024.10774524.

to be exchanged among different companies due to differing metadata content or formats.

In this paper, we present the *AI Model and Dataset Exchange Platform (AIMDEP)*, a platform for collaborative AI model development and use, see Fig. 1. This platform incorporates the *AI Model and Dataset Exchange Ontology (AIMDEO)*, an ontology for the description of machine learning assets, focusing on the expression of metadata of AI models and datasets. The practicability thereof is further demonstrated by a use case.

The main contributions of this paper are:

- A platform for collaborative AI development and use
- An ontology for the description of AI models and datasets
- Use case showing the applicability of the approach

This paper is structured as follows: Section II describes identified related work regarding AI asset management as well as AI asset characterization. Section III describes the *AIMDEP*, followed by a description of the ontology in Section IV. In Section V, we evaluate the approach with a use case that shows the applicability and benefits of this approach regarding AI model and dataset usage. Finally, Section VI concludes the paper.

II. Related Work

This section describes related work and points out differences and similarities to the presented approach. Besides AI asset management, AI asset characterization is covered.

A. AI Asset Management

The authors in [1] assessed various tools and platforms for machine learning asset management. Of all the options, only MLFlow supports asset registry and exchange without requiring data sharing with third parties. However, it does not support ontologies for the description of metadata of assets. *MLEM*[5] is an open-source tool that provides a standard interface for deploying machine learning models and offers a model registry. However, its model registry is very limited and does not collect metadata. Hugging Face is a hub for machine learning models and datasets and offers an interface for deploying the models.

B. AI Asset Characterization

Recent work from Blagec et al. introduced the *Intelligence Task Ontology and Knowledge Graph (ITO)* [2] which aims primarily to study scientific research but can also be used to annotate and organize information in the AI domain. While the ITO has the purpose of describing AI tasks, the AIMDEO ontology presented in this paper is meant to support the exchange of AI models and datasets. Moreover, in contrast to the ITO, the AIMDEO contains concepts for describing the provenance and kind of AI model as well as parameters of AI models and datasets. Additionally, the ontology presented here allows specifying model evaluation metrics. Both, the ITO and the AIMDEO support the specification of intended tasks and subtasks of AI models and datasets.

Idowu et al. presented the *Experiment Management Meta-Model (EMMM)* [3], which is a metamodel of asset types and their relationships common to the management of machine learning experiments. Besides modeling central concepts such as *models*, *parameters*, or *dependencies*, this metamodel enables the specification of arbitrary metadata for any asset. In addition to the asset structures and their relationships, the metamodel considers version control structures for machine learning as well as traditional assets. While the metamodel covers central concepts, element metadata is captured using arbitrary key-value mappings, which lacks a common format and explicit semantics.

III. Exchange Platform

With the advancement of AI in the recent decade, a vast number of AI assets have been produced within various companies and organizations. However, these assets generally lack any additional metadata and are either not discoverable or inaccessible by others. There is also a discrepancy between the environment in which an asset is created and the environment in which other users will use it, which can lead to various compatibility issues and hinder the usage of assets. To address the abovementioned issues, we have developed an *AI Model and Dataset Exchange Platform (AIMDEP)*, which provides a central registry to make AI models and datasets visible and accessible to all users. The AIMDEP uses the AIMDEO to specify the assets and captures additional metadata for each asset accordingly to make them more interpretable. Moreover, AIMDEP supports the export of AI model metadata according to the AIMDEO in the form of micro-ontologies. Finally, the online deployment integrated into the AIMDEP provides a standard platform-independent interface to deploy the assets.

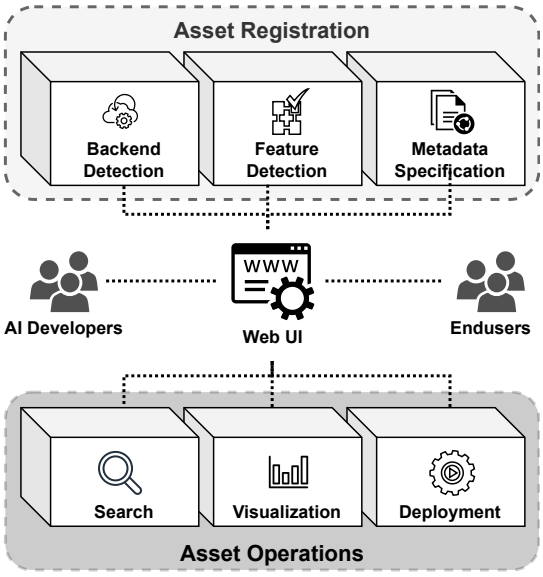

Fig. 2: Core features of the platform

[5]https://mlem.ai

Fig. 2 shows an overview of the core AIMDEP features regarding asset registration and operations. The platform employs a client-server architecture, with the server and the and the Web User Interface (UI) implemented using the Django framework. Additionally, custom clients can interact with the server through the provided Representational State Transfer (REST) Application Programming Interface (API). Leveraging Django's Model-View-Template (MVT) architecture, any updates to the ontology in the AIMDEO can seamlessly apply to the model component of the server without disrupting its overall functionality. This design ensures the platform's flexibility for accommodating future ontology updates. The communication between the server and the clients is encrypted, and the access management policy is implemented on the server side to restrict access to assets to privileged users. The models and datasets go through a register process to be added to the AIMDEO-based database. The AIMDEP supports multiple operations on the registered assets. For both datasets and models, it offers a semantic search functionality and the download of assets for offline deployment. The datasets can be visualized to gain more insight into the data using Plotly [4], and the models can be deployed online and evaluated using the interface provided by MLEM [5]. Next, we discuss each part in more detail.

A. Asset Registration

The registration of models and datasets starts by uploading the physical asset files to the platform. The AIMDEP attempts to identify the back-end required to access and deploy the asset. This includes identifying the data handlers for the datasets and the framework needed to evaluate and deploy the model. The AIMDEP supports various common dataset formats, including Comma-separated Values (CSV), Excel, JSON, and Parquet, and offers support for models exported by the most popular machine learning frameworks, including *Scikit-Learn*[6], *TensorFlow*[7], and *PyTorch*[8]. The user can edit the selected framework if required. Any custom dataset and model can also be registered without specifying the framework. However, the platform would be unable to visualize the dataset or deploy the model later, since the platform requires the framework to interact with the assets.

Next, models and datasets' input and output features are defined semi-automatically. The platform tries to detect the features, but the user should verify and add further metadata for each feature. Each feature is then stored as a *Parameter* based on AIMDEO. The user should also add the configuration parameters of models at this stage. Finally, the user adds the additional metadata following the AIMDEO, which includes, for instance, adding metrics to the model. The asset can then be registered in the central database of the platform for later access and operations.

[6]https://scikit-learn.org/
[7]https://www.tensorflow.org
[8]https://pytorch.org

B. Asset Operations

The AIMDEP supports multiple asset operations: For datasets, it offers interactive visualization in the form of statistical tables, various plots, and feature analyses (only for numerical datasets) to identify the importance of each feature. This offers a better insight into the data to train and evaluate efficient models later. The visualizations are rendered by a *Plotly*[9] engine on the server side. Since the visualization of the whole dataset can be very time-consuming for massive datasets, the AIMDEP can limit the visualization to a random subset of the dataset with a given size.

Online deployment is offered for the registered models with a supported framework. It enables a simple and platform-independent interface for the inference of the models. The inference of a model can be performed either through the Representational State Transfer (REST) Application Programming Interface (API) or through a graphical interface based on *Gradio*[10], which is generated based on the task and subtask of the model and its input and output features.

The AIMDEP supports the search and download of assets for both models and datasets. The search is powered by *OpenSearch*[11] and utilizes the metadata captured for each asset to perform a search and returns the most relevant assets for the query. Assets can also be downloaded for offline use or integration in custom workflows. Next to the original files, additional metadata captured for the asset is also provided as micro-ontologies containing the attributes and their values. Fig. 4 shows an example output of an export of AI model and corresponding dataset metadata from AIMDEP referring to the AIMDEO.

IV. Ontology

The *AI Model and Dataset Exchange Ontology (AIMDEO)* captures essential concepts that are part of collaborative AI model development and use. The ontology is expressed as an Web Ontology Language (OWL) [4] ontology. Overall metrics of the ontology are shown in Tab. 1.

Metric	Amount
Axiom count	414
Logical axioms count	178
Declaration axioms count	105
Class count	47
Object property count	29
Data property count	17
Individual count	5
Annotation property count	18
DL expressivity	SHOIQ(D)

Tab. 1: Ontology metrics

A. Collaboration Scenarios

We identified different collaboration scenarios to get a more profound understanding of concepts required to capture all

[9]https://plotly.com
[10]https://www.gradio.app
[11]https://opensearch.org

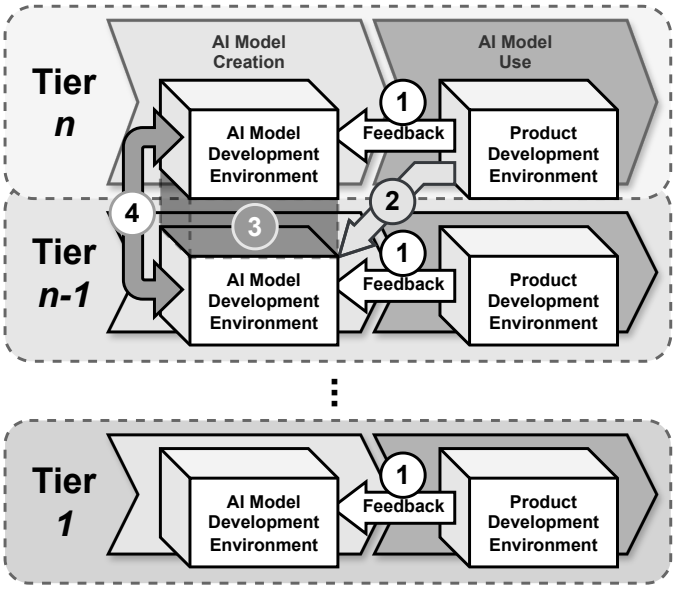

Fig. 3: AI model collaboration scenarios

information necessary in collaborative AI model development and use. Fig. 3 illustrates the scenarios described in the remainder of this section.

1) Internal Use: AI models and datasets remain with the creator. AI models are used from within the AI development environment and, for example, within a tier in the automotive industry. Results are then stored in a product development environment, and feedback is returned to the AI development environment, see ①.

2) Shared Data: If the data cannot be captured internally, an exchange of data between the supply chain participants is an option. In this scenario, data from the development database can be used in the AI development environment, see ②. This form of collaboration requires description and well-defined interfaces for data access.

3) Shared Models: Another form of cooperation is sharing trained models. In contrast to the former cooperation form, no data from the product development environment is shared. Instead, AI models are shared, see ③.

4) Model Services: The fourth and last cooperation scenario is based on the deployment and utilization of services. Instead of exchanging an AI model directly, users send corresponding requests to a service, that delegates to the AI model internally, see ④.

B. Ontology Concepts

Creating the ontology requires identification of relevant concepts and their relationships. These are then modeled in the ontology as corresponding OWL classes and properties. Considering the usage scenarios, identified essential concepts are shown in Tab. 2.

Concept	Description
General metadata	
Author	Provenance information
Description	Natural language description
Name	Name of an AI model or dataset
Framework	Back-end used to execute an AI model
Asset metadata	
Dataset	Data used for training and testing
Data source	Location of a dataset
Training-/test-split	Splitting of a dataset
Data points	Size of a dataset
AI Model characterization	
Task / sub-task	Intended purpose of an AI model
Input	Input of an AI model
Output	Output of an AI model
Parameter	Parameter of an AI model
Quality criterion	Quality criterion of an AI model
Score	Score of an AI model regarding a metric

Tab. 2: Essential concepts of the ontology

V. Evaluation

This section evaluates the proposed approach with the help of a collaboration scenario where typical stakeholders, like domain experts, AI experts, and end users, work interdependently and interdisciplinary together. The use case covers the development and usage of an AI model that facilitates the prediction of memory access time, which is essential in predicting software timing in safety-critical applications [5].

The dataset is created by a domain expert or, in our case, a hardware developer. The domain expert describes his inherent knowledge using our proposed ontology AIMDEO. In our use case, this covers cache (memory hierarchy) characteristics, such as *replacement strategy* and *size*. The dataset is registered on the AIMDEP, easing the *Dataset Exchange*. The AI expert can access the dataset and the knowledge specified by the domain expert. This supports the AI expert in designing a suitable AI model.

The developed model is registered for the *AI Model Exchange*, similar to the data set's registration. The AI expert provides additional information about the model, which is then annotated according to the ontology for the end user. An end user, in our case a software developer, is looking for an AI model to predict software timing. The developer looks for a suitable memory configuration for his or her time-sensitive software for embedded systems. He or she can use the keyword search provided by the platform to find an appropriate model on the platform. The found model can then be used to analyze his software with the memory configuration, as described using the ontology, thus assessing the purchase of test hardware.

The following sections outline this exchange with the mentioned use case.

A. Dataset Exchange

The hardware developer publishes the dataset he has created using the proposed platform AIMDEP. The platform recognizes all features semi-automatically when the dataset is uploaded, making it easier to describe them. One advantage of automatically recognizing all features of supported data formats is reducing susceptibility to errors, e.g., by forgetting

features. In combination with the platform's input mask, the user is also actively supported when entering information, such as a description and the value range. In the subsequent phase, metadata about the dataset and configuration parameters, which in this instance comprise information regarding the memory configuration, such as the *replacement strategy* or the *memory size*, can be specified. This information is then saved using the ontology AIMDEO and exported as OWL files upon the download of the dataset. The representation of the dataset in OWL can be observed in Fig. 4, lines 53-61, with further details available in Section IV. A search function enables AI experts on the platform to locate the dataset easily and quickly. The user does not have to download the dataset manually to analyze it; instead, they can use the analysis functions provided by the platform. This allows for the rapid analysis, comparison, and selection of a suitable dataset, which can then be downloaded together with the description of the dataset by the ontology. This process can also be accomplished via a REST API, enabling easy integration with other tools.

B. AI Model Exchange

Once the AI expert has developed a suitable AI model, it can be published on the platform. Additional information is added to the model using the ontology to enable the end user to quickly and easily assess the model's suitability for the specific application. Firstly, the input and output parameters are described and named. For this purpose, the name, a description, the data type, and optionally, a valid value range and the distribution are specified for each feature; as an example, the information for the AI model of the use case can be seen in Fig. 4, lines 4-18. In addition to this information, the uploader can specify configuration parameters according to the scheme in subsection V-A. With the evaluation information, end users can also assess the quality of the model more efficiently and better, such as using the quality metric and the breakdown of the dataset for training and testing. In addition to this information, meta-information about the model, such as the framework used, a description of the model, and the publication date, is provided. Another essential piece of information is the dataset specification used to assess better the model's suitability for the end users' data. The end user, in our case, the software developer, can find the model using the platform's built-in search function, which reduces the effort to find an AI model. By linking the dataset, the user can also find out what format and quality the training data for the model provided was in and whether their data matches it. The platform provides execution of the model for different frameworks; see Section III, which is directly done via the platform. The fast execution makes trying out the model for end users' data possible without setting up a suitable runtime environment. This lowers the hurdle for using and experimenting with an AI model. If the model meets the end user's criteria, they can download the model easily together with the ontology description and use it in their workflow.

```
1  @prefix ns: <http://.../aimodel/> .
2  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
3
4  <http://.../aimodel/metric/model0> a ns:AIModel ;
5      ns:hasName "RF Instruction Cache-Line Access Classifier" ;
6      ns:hasDescription "AI Model to classify single cache-line accesses
7                         to the instruction cache as hit or miss." ;
8      ns:hasAIModelAuthor <http://.../aimodel/dataset/dataset0/author0> ;
9      ns:hasAIModelParameter <http://.../aimodel/parameter/parameter0>,
10                            <http://.../aimodel/parameter/parameter1> ;
11     ns:hasAIModelInput <http://.../aimodel/input/input0> ;
12     ns:hasAIModelOutput <http://.../aimodel/output/output0> ;
13     ns:hasAIModelDataset <http://.../aimodel/dataset/dataset0> ;
14     ns:hasAIModelMetric <http://.../aimodel/metric/metric0> ;
15     ns:hasFramework "scikit-learn" ;
16     ns:hasTask "https://identifiers.org/ito:ITO_42033" ;
17     ns:hasSubtask "https://identifiers.org/ito:ITO_00498" ;
18     ns:hasVersion "1.0" .
19
20 <http://.../aimodel/metric/metric0> a ns:AIModelMetric ;
21     ns:hasName "average precision" ;
22     ns:hasDescription "Model evaluation metric" ;
23     ns:hasScore "0.9794" ;
24     ns:hasUnit "dimensionless" .
25
26 <http://.../aimodel/parameter/parameter0> a ns:AIModelParameter ;
27     ns:hasName "Replacement-Strategy" ;
28     ns:hasDescription "Replacement-Strategy of the used Instruction Cache" ;
29     ns:hasUnit "string" ;
30     ns:hasScore "LRU" .
31
32 <http://.../aimodel/parameter/parameter1> a ns:AIModelParameter ;
33     ns:hasName "Cache-Size" ;
34     ns:hasDescription "Size of the used Instruction Cache" ;
35     ns:hasUnit "Byte" ;
36     ns:hasScore "2048" .
37
38 <http://.../aimodel/input/input0> a ns:AIModelInput ;
39     ns:hasName "set" ;
40     ns:hasDescription "Set of the cache-line" ;
41     ns:hasUnit "uint8" .
42
43 <http://.../aimodel/output/output0> a ns:AIModelOutput ;
44     ns:hasName "miss" ;
45     ns:hasDescription "Category, if the access to the cache-line was
46                        a hit (false) or a miss (true)" ;
47     ns:hasUnit "bool" .
48
49 <http://.../aimodel/dataset/dataset0/source0> a ns:AIModelDatasetSource ;
50     ns:hasName "Cache-accesses database location" .
51     ns:hasLocation "file://.../datasets/0/hbu9vRD.csv" ;
52
53 <http://.../aimodel/dataset/dataset0> a ns:AIModelDataset ;
54     ns:hasAIModelDatasetAuthor <http://.../aimodel/dataset/dataset0/author1> ;
55     ns:hasAIModelDatasetSource <http://.../aimodel/dataset/dataset0/source0> ;
56     ns:hasDescription "Contains records of instruction-cache-accesses
57                        for different generated complex pseudo-programs." ;
58     ns:hasName "Complex Programs Instruction Cache-Lines Set Sorted" ;
59     ns:hasTask "https://identifiers.org/ito:ITO_42033" ;
60     ns:hasSubtask "https://identifiers.org/ito:ITO_00498" ;
61     ns:hasVersion "3.0" .
```

Fig. 4: Micro-ontology describing AI model metadata

VI. Conclusion

We used the AIMDEP successfully to collaboratively develop AI models. However, its application is not limited to that. It is possible to use either the platform or the ontology separately. The platform eases collaborative AI model development while the ontology can help to ensure that required information is available, and how this information can be provided. The metadata specification and export functionality of the platform can help in integrating different tools required in the AI model development process.

Future work could, for instance, investigate on extending EMMM to create a certain degree of compatibility. The generic key-value pair metadata associated with the resource types in the metamodel could be used to store AIMDEO metadata. Creating further interoperability between AIMDEO and ITO might also be worth investigating.

Acknowledgment

This paper is funded by the BMWi within the project progressivKI (grant number 19A21006M).

References

[1] S. Idowu, D. Strüber, and T. Berger, "Asset management in machine learning: State-of-research and state-of-practice," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–35, 2022.

[2] K. Blagec, A. Barbosa-Silva, S. Ott, and M. Samwald, "A curated, ontology-based, large-scale knowledge graph of artificial intelligence tasks and benchmarks," *Scientific Data*, vol. 9, no. 1, p. 322, 2022.

[3] S. Idowu, D. Strüber, and T. Berger, "EMMM: A unified meta-model for tracking machine learning experiments," in *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2022, pp. 48–55.

[4] G. Antoniou and F. Van Harmelen, "Web ontology language: Owl," in *Handbook on ontologies*. Springer, 2004, pp. 67–92.

[5] S. Ottlik, C. Gerum, A. Viehl, W. Rosenstiel, and O. Bringmann, "Context-sensitive timing automata for fast source level simulation," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 512–517.