DATCloud: A Model-Driven Framework for Multi-Layered Data-Intensive Architectures

Moamin Abughazala*†, Henry Muccini*

* DISIM Department, University of L'Aquila, L'Aquila, Italy
{moamin.abughazala1, henry.muccini}@univaq.it

† Department of Information Technology, An Najah N. University, Nablus, Palestine
m.abughazaleh@najah.edu

***Abstract*—The complexity of multi-layered, data-intensive systems demands frameworks that ensure flexibility, scalability, and efficiency. DATCloud is a model-driven framework designed to facilitate the modeling, validation, and refinement of multi-layered architectures, addressing scalability, modularity, and real-world requirements. By adhering to ISO/IEC/IEEE 42010 standards, DATCloud leverages structural and behavioral meta-models and graphical domain-specific languages (DSLs) to enhance reusability and stakeholder communication. Initial validation through the VASARI system at the Uffizi Gallery demonstrates a 40% reduction in modeling time and a 32% improvement in flexibility compared to manual methods. While effective, DATCloud is a work in progress, with plans to integrate advanced code generation, simulation tools, and domain-specific extensions to further enhance its capabilities for applications in healthcare, smart cities, and other data-intensive domains.**

***Index Terms*—Modeling Data Architecture, multi-layered architectures, Data-Intensive applications**

I. Introduction

The increasing complexity of data-intensive systems, driven by the rapid expansion of Internet of Things (IoT) devices, has introduced significant challenges for system modeling. Modern architectures must address scalability, modularity, and data management across multi-layered cloud, fog, and edge systems [1]. However, existing frameworks often lack robust modeling capabilities, resulting in inefficiencies when designing, adapting, and managing these architectures [2].

To address these challenges, this paper introduces **DATCloud** a model-driven framework designed to simplify and enhance the modeling of multi-layered, data-intensive systems. Unlike traditional methods DATCloud models and combines data systems across cloud, fog, and edge layers. DATCloud uses structural and behavioral meta-models to represent the logical architecture of systems, including components, data flows, and workflows. These meta-models provide architects with the tools needed to define, validate, and refine system designs efficiently. The framework further incorporates graphical domain-specific languages (DSLs), offering an intuitive interface for creating models that ensure consistency and compliance with architectural standards such as ISO/IEC/IEEE 42010 [3].

A key focus of DATCloud is improving the efficiency and flexibility of the modeling process. By supporting templates,and modular design, the framework reduces the effort required for iterative development and simplifies communication between stakeholders. These capabilities enable architects to handle complex architectures while minimizing manual overhead.

Research, such as Monitor-IoT [4], SimulatIoT [5], Silva et al. [6], and CHESSIoT [7], has highlighted the potential of multilayer monitoring architectures for IoT systems. These studies cover software, hardware, physical, and deployment aspects, but reveal a notable gap in the *Data View*.

The rest of this paper is structured as follows: Section II reviews related work, highlighting existing frameworks and their limitations in addressing the challenges of modeling multi-layered, data-intensive systems. Section III describes the design of DATCloud, detailing its structural and behavioral meta-models, core functionalities, and adherence to ISO/IEC/IEEE 42010 standards. Section IV presents the case study and discussion of initial results for DATCloud, focusing on metrics for modeling time and flexibility. It highlights stakeholder insights and demonstrates DATCloud's practical effectiveness in managing visitor flow and architectural flexibility through its application in the VASARI system at the Uffizi Gallery. Finally, Section V outlines future directions for enhancing the framework, including advanced code generation and domain-specific extensions, and concludes with a summary of its contributions.

II. Related Work

The rapid growth of IoT systems and data-intensive applications has prompted the development of various frameworks and methodologies for modeling multi-layered architectures. While significant progress has been made, some gaps still exist, particularly in the **data view** and in **integrating data-intensive applications across the cloud, fog, and edge layers**. This section reviews the key contributions in these areas and highlights the gaps that DATCloud aims to address.

A. Modeling IoT Multi-Layer Architectures

Efficient architectures are essential for IoT systems to process distributed data across edge, fog, and cloud layers. Various frameworks have been proposed to model and manage these complex systems effectively.

- Bauer et al. [2] introduced an IoT Reference Architecture emphasizing the roles of cloud, fog, and edge layers. While this model outlines task allocation, it lacks a concrete framework for modeling workflows and data interactions between these layers.
- Taivalsaari and Mikkonen [8] focus on IoT systems' lifecycle, emphasizing the interplay between edge devices and the cloud. However, their approach primarily addresses operational challenges rather than architectural modeling.
- Ihirwe et al. [7] proposed CHESSIoT, a model-driven engineering (MDE) framework for IoT system design. This framework leverages domain-specific languages (DSLs) to model and verify IoT architectures. While effective for IoT-specific use cases, CHESSIoT does not generalize to multi-layered systems involving complex data workflows.

While current frameworks recognize the hierarchical structure of IoT systems, they mainly concentrate on system behavior or task allocation. They do not adequately address the **data view**, which creates a gap in *how data is modeled, stored, and processed across multi-layered architectures*. Additionally, most frameworks are specific to certain domains and lack the flexibility needed for wider applications.

B. Modeling Data-Intensive Applications

Data-intensive applications are defined by their need to manage, process, and analyze large volumes of data across distributed systems. Effective modeling of such systems requires frameworks that prioritize data workflows, scalability, and adaptability.

- Kleppmann [9] explores the challenges of building scalable, distributed systems for handling data pipelines and real-time analytics. While this work provides a thorough understanding of data-intensive systems, it focuses on implementation rather than architectural modeling.
- Abughazala et al. [10] [11] [12] proposed a Framework, one of the few works explicitly addressing modeling data-intensive workflows. The Framework introduces meta-models to describe how data flows between components. Its focus on industrial data workflows limits its relevance to general multi-layered architectures, including cloud, fog, and edge systems.
- Raj [13] introduced a conceptual model for designing a data pipeline consisting of two primary components: nodes and connectors. The nodes serve as the core abstract data units, while the connectors facilitate data transmission and communication between these nodes.
- Borelli [14] proposed a classification framework for key software components and their interrelationships to model software architectures tailored for specific IoT applications. These components are presented as abstract representations.
- Erraissi [15], [16] developed a meta-model encompassing data sources, ingestion layers, and a Big Data visualization layer to provide a structured approach to managing and visualizing Big Data workflows.

Current frameworks for data-intensive applications **lack a unified approach to modeling data workflows across cloud, fog, and edge layers**. Instead of providing holistic integration, they often focus on isolated aspects. This gap highlights the need for a scalable and modular framework, such as DATCloud, to effectively tackle these complexities and ensure practical applicability in real-world scenarios.

III. Framework Design

This section introduces DATCloud's framework design, focusing on two key components: the **structural meta-model**, which defines the architecture of multi-layered systems, and the **behavioral meta-model**, which captures workflows and interactions. Together with graphical DSLs, these meta-models enable scalable, modular, and standards-compliant modeling for data-intensive systems.

A. Structural Meta-Model

Using the structural meta-model, users can establish architecture across cloud, fog, and edge layers, model nodes and connections, choose data storage types like NoSQL or NewSQL, and select communication protocols such as MQTT or REST. They can also illustrate data formats and data flow. Figure 1 illustrates the structural concepts of the Data Architecture Modeling Language (DAML) meta-model, showcasing the primary components and their interrelations within data-intensive architectures. The DAML structural meta-model defines the physical and logical structure of data systems, emphasizing their nodes, formats, storage, and connections. The key elements are:

- *Data Architecture*: Represents the overarching framework that integrates multiple DataNodes and their interactions, encapsulating the architecture of the data system.
- *DataNode*: The central element of the model, representing a computational or storage unit responsible for handling data operations [17]. Attributes include:
  - NodeName: The identifier of the DataNode.
  - Description: A description of the DataNode's role.
  - Type: Specifies the kind of node, such as Server, Gateway, or Device.
- *ProcessingType*: Categorizes the nature of data handling within a DataNode, such as Batch processing (Processes data in large, pre-defined batches), Real-Time processing (Processes data continuously as it streams).
- *Data Location*: Defines the physical or virtual storage location of the data, which could be Cloud, Fog, or Edge systems.

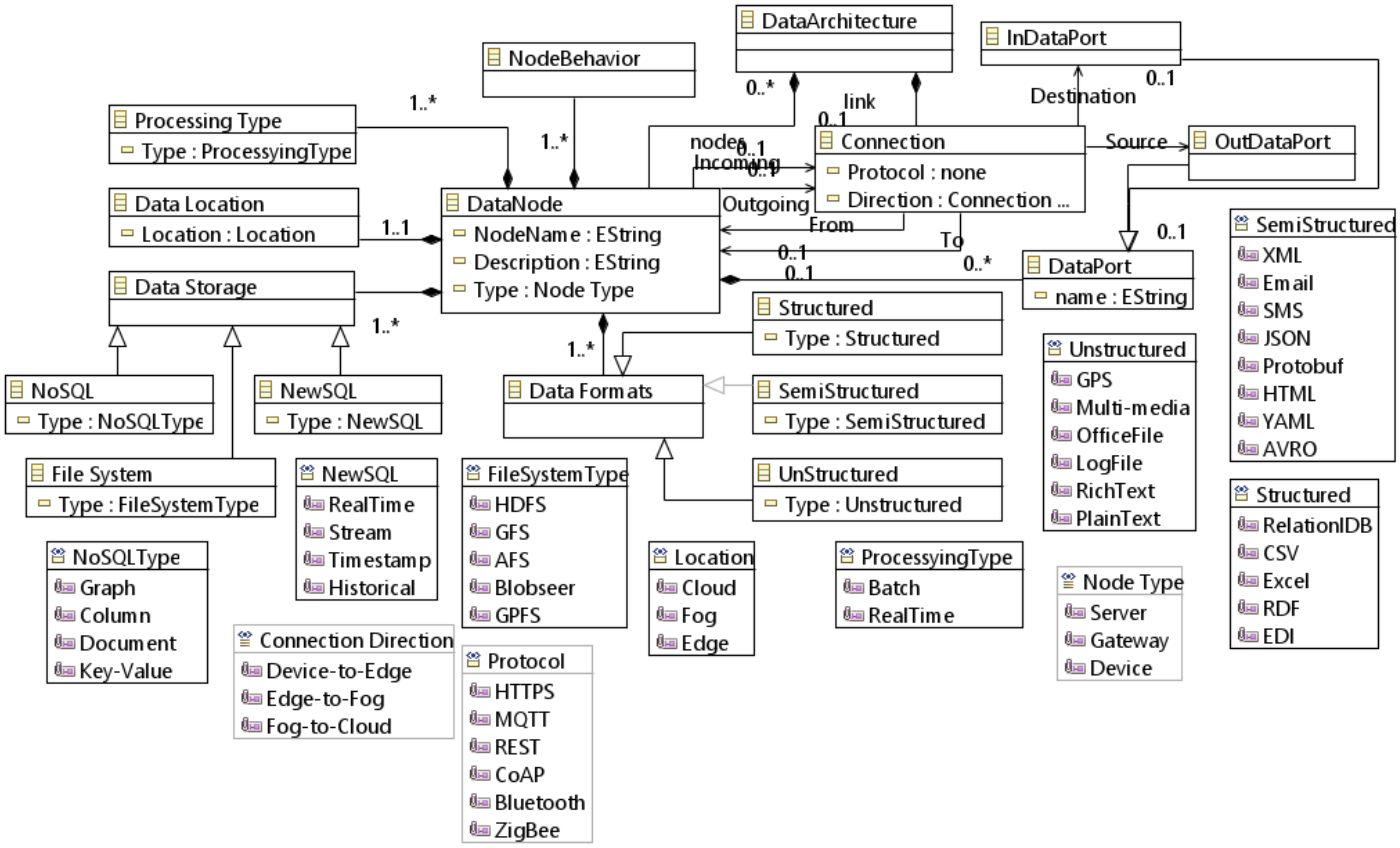

Fig. 1. Meta-model: structural concepts

- *Data Storage*: Describes the storage solutions utilized in the architecture, classified as:
  - NoSQL: Includes databases such as Graph, Column, Document, and Key-Value stores.
  - NewSQL: Represents scalable, modern relational databases that support real-time, historical, or stream data.
  - File System: Includes distributed and scalable storage types like HDFS, GFS, Blobseer, and GPFS.
- *Data Formats*: Specifies the types of data handled by the DataNode, categorized into:
  Structured: Relational databases, CSV, Excel, RDF, EDI. Semi-Structured: XML, Email, JSON, YAML, AVRO, and Protobuf. Unstructured: GPS data, multimedia files, logs, office files, and plain text.
- *DataPort*: Represents the interfaces (input/output ports) used for data exchange between nodes:
  InDataPort: For receiving data. OutDataPort: For sending processed data.
- *Connection*: Describes the links between DataPorts, enabling data transfer within the system. These are categorized by direction (Incoming and Outgoing) and protocol (e.g., HTTPS, MQTT, REST).
- *NodeBehavior*: Specifies the functional operations performed by a DataNode, including its responses to actions and events.
- *Incoming/Outgoing Connections*: Define the flow direction of data within the architecture, crucial for understanding dependencies and workflows.

This structural meta-model provides a clear and detailed representation of the components, storage methods, data formats, and interactions within a data-intensive system, forming the basis for cloud data architecture modeling.

B. Behavioral Meta-model

DATCloud allows end-users to define system component behavior using a graphical DSL. The behavioral meta-model supports workflows for: Data Ingestion, Data Processing, Data Output. Figure 2 depicts the Behavioral Meta-model of the Cloud Data Architecture Modeling Language (Cloud-DAML), focusing on the internal activities and workflows within DataNodes. This meta-model outlines how data is processed, analyzed, and consumed, detailing the roles of actions, events, and interactions. The key components are:

- *NodeBehavior*: Represents the internal behavior of a DataNode, encapsulating the logic, rules, and sequences governing data handling.

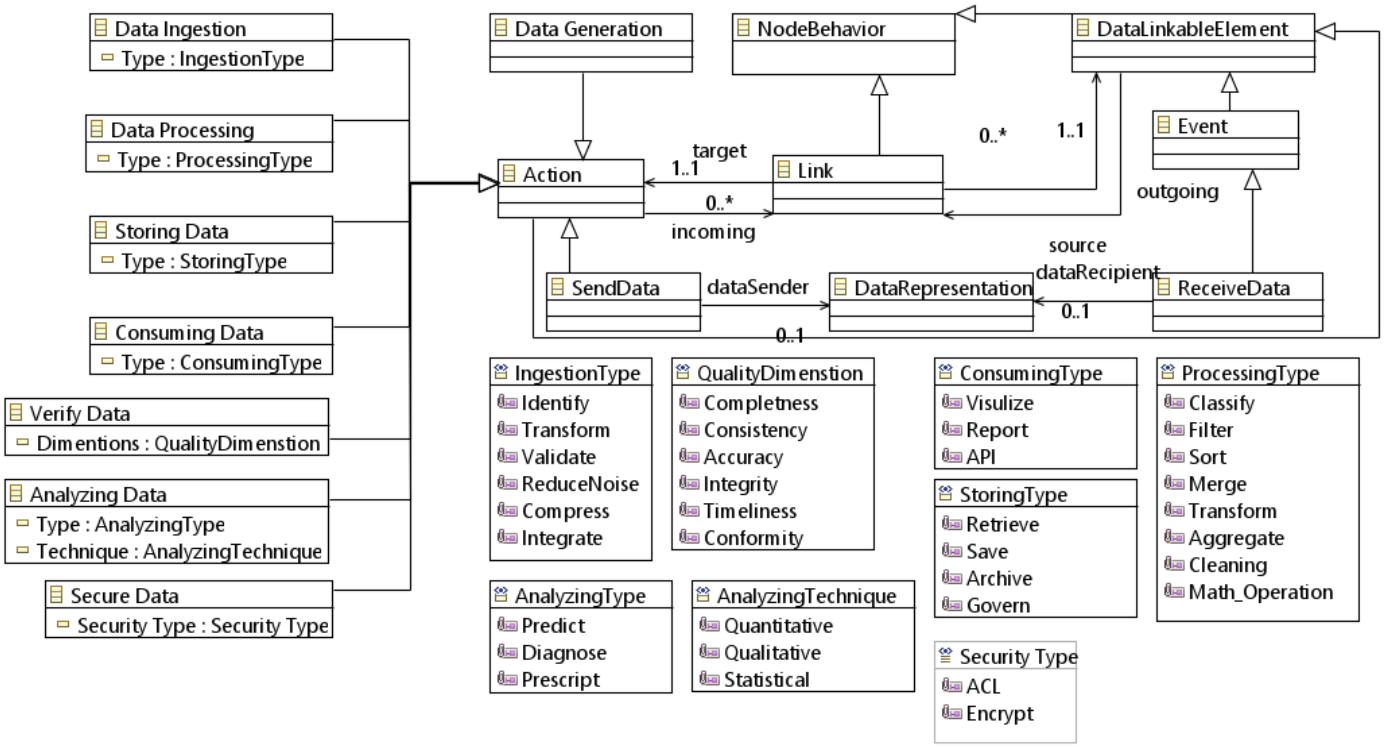

Fig. 2. Meta-model: behavioral concepts

- *Elements*: These include actions and events that define the step-by-step data processing tasks and their triggers.
- *DataPorts*:
  - InDataPort: Serves as an entry point for data into a node.
  - OutDataPort: Functions as an exit point for processed data.
- *Actions*: Core behavioral elements representing atomic tasks executed within a node. Actions can be triggered by events or due to previous steps in the workflow. Examples include:
  - Data Generation: Creation or sourcing of new data.
  - Data Ingestion: Data transfer to a staging area for processing.
  - Data Processing: Transformations and computations (e.g., filtering, aggregation, analysis).
  - Storing Data: Saving processed data to databases or data lakes.
  - Analyzing Data: Performing in-depth analysis to generate insights.
  - Consuming Data: Utilizing processed data for visualization, reporting, or APIs.
  - Verifying Data: Ensuring data quality by checking completeness, accuracy, and consistency.
  - Secure Data: Applying security measures such as ACLs (Access Control List) and encryption to protect data during processing and transfer.
- *Events*:
  Triggered by external stimuli or preceding actions within the system. Example: *ReceiveData*: Handles incoming data via an InDataPort, initiating the processing workflow.
- *Connections*: Links: Define logical pathways and dependencies between actions and events, specifying the sequence of operations and data flow.

This behavioral meta-model complements the structural model by detailing the internal workflows and interactions of DataNodes, providing a comprehensive framework for modeling data lifecycle within cloud Data Architecture.

C. Adherence to ISO/IEC/IEEE 42010 Standards

In accordance with the ISO/IEC/IEEE 42010 standard [3], DATCloud offers multiple architectural views and system components are effectively described by their workflows through structural and behavioral models. Their structural and behavioral meta-models are used to guarantee the clear representations of both the logical and physical architectures and are consistent and complete. To improve stakeholder communication, DATCloud has adopted the use of graphical domain specific languages (DSLs) that generate easily understandable visual models to support the interface between technical and non-technical stakeholders. It also has automated validation to ensure that models are coherent, exhaustive and that they meet stakeholder's needs. Its modular and scalable meta-models enable it to be adaptable for iterative development and changing needs.

D. How To Use DATCloud

DATCloud defines the architecture using the structural meta-model, specifying nodes, data flows, storage types, and communication protocols. The behavioral meta-model captures workflows within and between nodes, such as data ingestion and processing. Pre-defined templates and reusable components streamline the process, while automated tools validate the model for consistency and resolve dependencies. Iterative refinements ensure alignment with system requirements. DATCloud supports scalability, enabling quick updates to workflows or architecture, and minimizes manual effort through its intuitive interface and automation.

IV. Discussion of Initial Results

This section examines DATCloud's application to the VASARI system at the Uffizi Gallery, validating its effectiveness in real-world scenarios. Metrics such as modeling time and flexibility were evaluated using participant logs and feedback. The results highlight DATCloud's ability to streamline workflows, adapt to changes, and address challenges in multi-layered, data-intensive architectures.

A. CASE STUDY

The VASARI system has been deployed at the Uffizi Gallery to control the flow of visitors and has been used to validate the effectiveness of DATCloud. This case study is a model of architecture with IoT sensors at the edge layer, fog nodes for processing, and cloud for analytics. The structural meta-model of the DATCloud modeled the relations of these components, and the behavioral meta-model defined the workflows for visitor tracking and queue management. Figure 3 illustrates how DATCloud models the structure of the Uffizi Gallery. It shows the IoT sensors, the data processing at the fog layer, and the centralized cloud analytics platform. The diagram highlights the flow of visitor data from the sensors to analytics, showcasing DATCloud's flexibility and modularity in facilitating these processes.

B. Initial Results

To assess the performance of DATCloud in the VASARI system, two key metrics were evaluated: Modeling Time and Flexibility in Architectural Design. Table I demonstrates the significant reduction in modeling time achieved using DATCloud, while Table II outlines the framework's ability to improve flexibility when adapting to new requirements. For example, the use of pre-defined templates reduced workflow definition time by 50%, while modular design allowed for faster addition of architectural layers.

C. Metrics and Methodology

The results in Tables I and II were derived from structured time logs maintained by the participants during the case study. Participants included system architects, data analysts, and museum staff responsible for managing the VASARI system. Each participant logged their time spent on specific modeling tasks, such as defining workflows, validating the system, and refining models, using both manual methods and DATCloud. The average values for modeling time (T_{base} and T_{DAT}) and flexibility (E_{base} and E_{DAT}) were calculated to provide a comprehensive evaluation of the framework's impact.

- **Modeling Time (Table I)**: The total time required for workflow definition, system validation, and model refinement was significantly reduced when using DATCloud, with an overall time savings of 40%.

$$\text{Time Savings } (\%) = \frac{T_{\text{base}} - T_{\text{DAT}}}{T_{\text{base}}} \times 100$$

TABLE I
TIME SAVINGS ACHIEVED USING DATCLOUD

Task	T_{base} (hrs)	T_{DAT} (hrs)	T_{Saving} (%)
Workflow Definition	40	20	50%
System Validation	30	20	33%
Model Refinement	30	20	33%
Total	**100**	**60**	**40%**

- **Flexibility (Table II)**: Improvements in flexibility were assessed based on the time and effort needed to modify workflows, add architectural layers (e.g., fog and edge layers), and reuse templates. DATCloud improved overall adaptability by 32%, with the most notable gains observed in adding fog and edge layers.

$$\text{Flexibility Improvement } (\%) = \frac{E_{\text{base}} - E_{\text{DAT}}}{E_{\text{base}}} \times 100$$

TABLE II
FLEXIBILITY IMPROVEMENT ACHIEVED USING DATCLOUD

Task	E_{base} (hrs)	E_{DAT} (hrs)	$T_{\text{Improvement}}$ (%)
Add Fog Layer	15	8	47%
Add Edge Layer	20	12	40%
Modify Workflow	12	7	42%
Reuse Templates	Not Applicable	5	Significant
Total	**47**	**32**	**32%**

D. Stakeholder Feedback

Stakeholders provided valuable feedback during semi-structured interviews and observational studies conducted as part of the VASARI case study. Training sessions and hands-on workshops ensured that all participants were familiar with DATCloud's features. Stakeholders noted that the framework's intuitive design, pre-defined templates, and automated tools significantly streamlined the modeling process. They also highlighted the framework's ability to adapt to evolving requirements as a critical advantage. Key suggestions included enhancing visualization tools for complex workflows, expanding the library of domain-specific templates, and improving onboarding materials. In response, DATCloud introduced a more detailed template library, a refined graphical interface

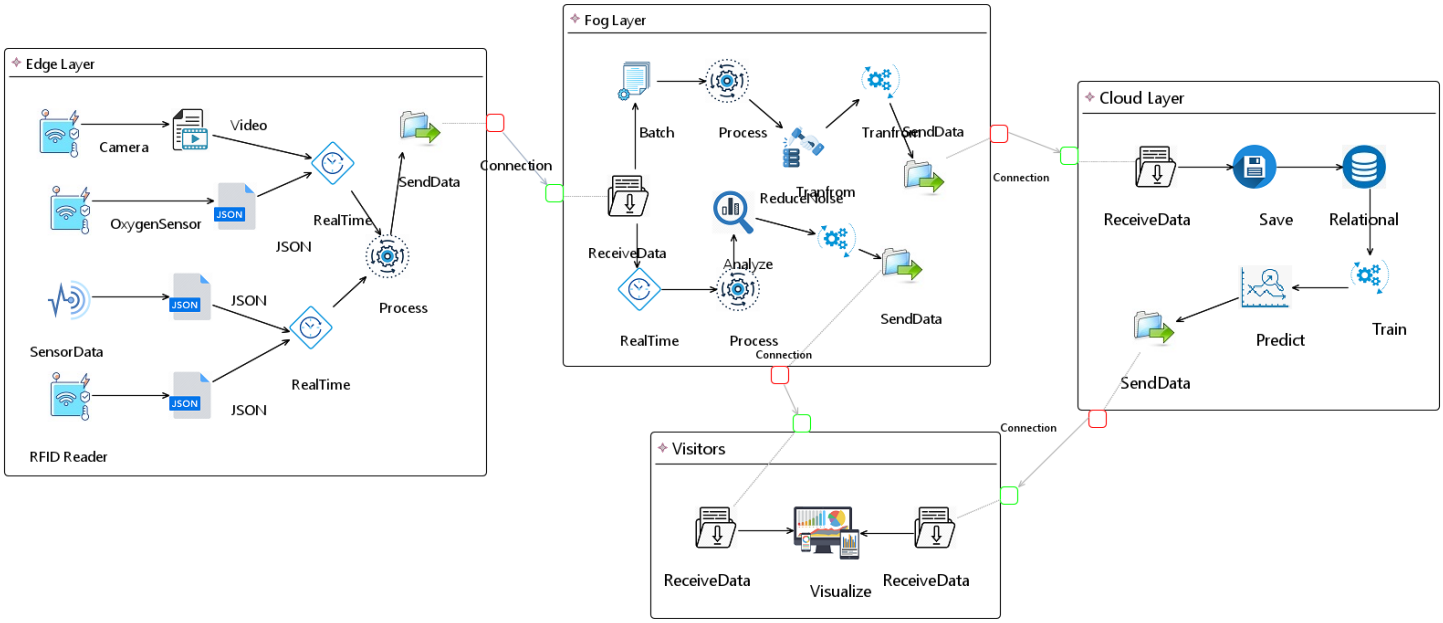

Fig. 3. A DATCloud Application for Uffizi Gallery

with collapsible nodes, and interactive training modules to support new users.

V. Conclusion and Future Work

DATCloud is a model-driven framework that simplifies multi-layered, data-intensive architecture modeling, achieving a 40% reduction in modeling time and 32% flexibility improvement in the VASARI system. It leverages structural and behavioral meta-models, adhering to ISO/IEC/IEEE 42010 standards to streamline design and enhance stakeholder communication. Future work includes advanced code generation, simulation tools, and domain-specific applications to enhance its scalability and versatility, making it ideal for real-world implementations in areas like healthcare and smart cities.

References

[1] Antero Taivalsaari and Tommi Mikkonen. On the development of iot systems. In *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 13–19, 2018.

[2] Martin Bauer, Mathieu Boussard, Nicola Bui, Jourik De Loof, Carsten Magerkurth, Stefan Meissner, Andreas Nettsträter, Julinda Stefa, Matthias Thoma, and Joachim W Walewski. Iot reference architecture. *enabling things to talk: designing IoT solutions with the IoT architectural reference model*, pages 163–211, 2013.

[3] ISO/IEC/IEEE. ISO/IEC/IEEE 42010:2011 Systems and software engineering – Architecture description, 2011.

[4] Lenin Erazo-Garzón, Priscila Cedillo, Gustavo Rossi, and José Moyano. A domain-specific language for modeling iot system architectures that support monitoring. *IEEE Access*, 10:61639–61665, 2022.

[5] José A. Barriga, Pedro J. Clemente, Encarna Sosa-Sánchez, and Álvaro E. Prieto. Simulateiot: Domain specific language to design, code generation and execute iot simulation environments. *IEEE Access*, 9:92531–92552, 2021.

[6] Ivanovitch Silva, Rafael Leandro, Daniel Macedo, and Luiz Affonso Guedes. A dependability evaluation tool for the internet of things. *Computers & Electrical Engineering*, 39(7):2005–2018, 2013.

[7] Felicien Ihirwe, Davide Di Ruscio, Simone Gianfranceschi, and Alfonso Pierantonio. Chessiot: A model-driven approach for engineering multi-layered iot systems. *Journal of Computer Languages*, 78:101254, 2024.

[8] Antero Taivalsaari and Tommi Mikkonen. On the development of iot systems. In *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 13–19. IEEE, 2018.

[9] Martin Kleppmann. *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. " O'Reilly Media, Inc.", 2017.

[10] Moamin Abughazala, Henry Muccini, and Mohammad Sharaf. Architecture description framework for data-intensive applications. In *2023 Fourth International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pages 99–106. IEEE, 2023.

[11] Moamin Abughazala, Henry Muccini, and Mohammad Sharaf. Dat: Data architecture modeling tool for data-driven applications. In *European Conference on Software Architecture*, pages 90–101. Springer, 2022.

[12] Moamin Abughazala and Henry Muccini. Modeling data analytics architecture for iot applications using dat. In *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*, pages 284–291, 2023.

[13] Aiswarya Raj, Jan Bosch, Helena Holmström Olsson, and Tian J Wang. Modelling data pipelines. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 13–20. IEEE, 2020.

[14] Fabrizio Borelli, Gabriela Biondi, Flávio Horita, and Carlos Kamienski. Architectural software patterns for the development of iot smart applications. *arXiv preprint arXiv:2003.04781*, 2020.

[15] Allae Erraissi and Abdessamad Belangour. Data sources and ingestion big data layers: meta-modeling of key concepts and features. *International Journal of Engineering and Technology*, 7(4):3607–3612, 2018.

[16] Allae Erraissi, Banane Mouad, and Abdessamad Belangour. A big data visualization layer meta-model proposition. In *2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, pages 1–5. IEEE, 2019.

[17] C. Szyperski. *Component Software. Beyond Object Oriented Programming*. Addison Wesley, 1998.