

MDE-Based Graphical Tool for Modeling Data Provenance According to the W3C PROV Standard

Marcos Alves Vieira^{1,2}^a and Sergio T. Carvalho²^b

¹Instituto Federal Goiano - IF Goiano, Brazil

²Universidade Federal de Goiás - UFG, Brazil

Keywords: Data Provenance, W3C PROV, Graphical Modeling, Metamodel, PROV-N, Modeling Tool, MDE.

Abstract: The rise of the Internet of Things (IoT) and ubiquitous computing has led to a significant increase in data volumes, necessitating robust management. Data provenance is crucial for ensuring data reliability, integrity, and quality, tracking the origins, transformations, and movements of data. The W3C PROV standard, with syntaxes like PROV-N, provides textual and graphical representations for expressing and storing data provenance. However, despite its importance, there is a lack of user-friendly graphical tools for developers, particularly in IoT and ubiquitous computing. This paper addresses this gap by introducing an innovative graphical tool that enables the creation of user-friendly graphical data provenance models adhering to the W3C PROV standard. The tool offers an intuitive interface for developers, simplifying the process of obtaining PROV-N code from the generated provenance graph. We demonstrate the tool's versatility across diverse domains, emphasizing its role in bridging the gap in graphical provenance modeling. The paper outlines the Model-Driven Engineering (MDE) methodology used in the tool development, and introduces its underlying Ecore metamodel aligned with the PROV data model (PROV-DM). Evaluation results of the metamodel are presented, and potential applications of the tool are discussed, emphasizing its contribution to enhancing provenance-aware applications.

1 INTRODUCTION

The Internet of Things (IoT) and ubiquitous computing have revolutionized our digital interactions, generating vast data volumes requiring effective management. Data provenance, tracking data origins, transformations, and movements, is critical for ensuring reliability, integrity, and quality in these domains (Hu et al., 2020; Herschel et al., 2017).


The W3C PROV standard, encompassing the PROV-DM data model, offers concrete syntaxes for representing, serializing, and storing data provenance (Moreau et al., 2013a). PROV-N provides textual representation, while a graphical layout follows standard conventions in a directed graph.


When instrumenting software for data provenance, developers must model captured object information and determine the appropriate application flow time. For instance, in an IoT-based security system, tracking conditions triggering emergency alarms from each sensor may be relevant.

Data provenance can also be obtained through the PROV-Template approach (Moreau et al., 2018), involving mining relevant data directly from databases or log files. This bypasses the need for explicit tracking or recording of provenance information during activity execution.

In IoT and ubiquitous computing contexts, data provenance's significance is paramount. Regardless of the capture method, there is a need to model the process for an accurate representation of data and their interrelationships. However, to the best of the author's knowledge, there are no tools available for developers to create desired provenance models for software capture in a graphical manner. This gap intensifies due to data ubiquity and complexity in IoT and ubiquitous computing scenarios. Developers typically adopt notations like PROV-N from the W3C PROV model, yet a lack of graphical tools means developers must convert static PROV graphs into PROV-N code without interactivity. Modifications in the PROV-N model mandate regenerating the corresponding PROV graph and vice versa.

This paper introduces a novel graphical tool addressing this gap, facilitating user-friendly creation

^a <https://orcid.org/0000-0002-5498-5015>

^b <https://orcid.org/0000-0002-4191-5173>

of data provenance models compliant with the W3C PROV standard. The tool provides a direct, intuitive, and interactive graphical interface for developers in provenance modeling. Additionally, the tool streamlines obtaining PROV-N code from the generated graphical model.

The remainder of this paper is structured as follows: Section 2 explores the theoretical and technological underpinnings employed in this study; Section 3 discuss related works and highlights the contribution of our proposal; Section 4 presents the foundational Ecore metamodel representing the PROV data model, followed by the W3C PROV-compliant graphical provenance modeling tool built upon this metamodel; Section 5 showcases the tool’s application in modeling data provenance across diverse domains; finally, Section 6 provides concluding remarks.

2 BACKGROUND

2.1 Data Provenance

Provenance refers to information detailing the production process of a product, be it a data entity or a physical object (Herschel et al., 2017). The W3C Provenance Working Group¹ defines provenance as “information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability, or trustworthiness.” The term originates from the Latin *prōvenīre*, meaning “coming from.”

Data provenance is crucial in various application areas (Glavic, 2021; Pérez et al., 2018): in open information systems, it aids in determining data origins and identifying responsible entities; in science applications, it provides insights into research result derivation, ensuring transparency and reproducibility; in news contexts, it is vital for verifying blog and news item origins, enhancing credibility and trustworthiness; in legal domains, provenance influences licensing, attribution, and privacy information management. In IoT, data provenance spans supply chains, health monitoring, digital forensics, and intelligent IoT services (Hu et al., 2020). Across these domains, provenance ensures accountability, authenticity, and informed decision-making (Herschel et al., 2017).

2.2 W3C PROV Family of Documents

The PROV Family of Documents, introduced by the W3C Provenance Working Group, encompasses a

comprehensive framework that includes a data model, serializations, and supplementary definitions. It establishes standardized representations and interoperable mechanisms for seamless sharing of provenance data across platforms.

In W3C PROV, provenance is represented by three central figures: *Entity*, *Agent*, and *Activity*, connected by various relationships designed for assertions about the past, such as *wasGeneratedBy*, *used*, *wasAssociatedWith*, and *wasAttributedTo*. These relationships are defined in the Provenance Data Model (PROV-DM) (Moreau et al., 2013a), the conceptual model underlying the W3C PROV family of specifications. The PROV standard also introduces PROV notation (PROV-N) (Moreau et al., 2013b) for concise, human-readable PROV instances and serializations, with the aim of simplifying the translation of the PROV data model into concrete syntax.

Entities, activities, and agents can be graphically represented using W3C PROV conventions², forming a provenance graph. This graphical representation is further exemplified in Figure 1, depicting the process of writing a journalistic article with reliance on a government-provided database. In this example, journalist Bob, represented as an agent, was assigned the entity *employment-article-v1.html*, indicating his responsibility for creating the article. The activity *writeArticle* demonstrates that the article was written with associations *wasGeneratedBy* and *used*, signifying the usage of *oesm11st.zip* as a base entity. The graphical representation and PROV-N notation code were developed under the proposed graphical data provenance modeling tool, detailed in Section 3.

2.3 Model-Driven Engineering

Model-Driven Engineering (MDE) is a software development approach that revolves around creating, manipulating, and utilizing models as the central artifacts throughout the development lifecycle (Bucchiarone et al., 2020).

Models are typically constructed using Domain-Specific Modeling Languages (DSMLs), which, in turn, are defined by a metamodel (López-Fernández et al., 2015). DSMLs provide specialized languages tailored to specific domains, allowing for the creation of models that capture the essential concepts and relationships within that domain. In this sense, a metamodel is “a model that defines the structure of a modeling language” (Rodrigues da Silva, 2015) and serves as a formal description of the language’s syntax, semantics, and constraints (Bruel et al., 2018).

¹<https://www.w3.org/2011/prov/>

²<https://www.w3.org/2011/prov/wiki/Diagrams>

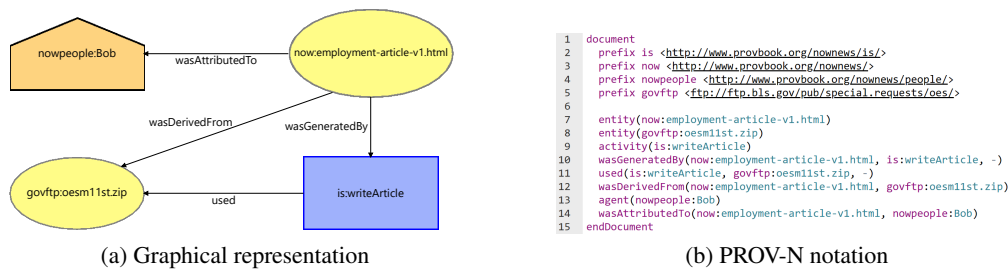


Figure 1: Example of a provenance record represented in W3C PROV. Adapted from: (Moreau and Groth, 2013).

2.4 Eclipse Modeling Framework

The Eclipse Modeling Framework (EMF) (Steinberg et al., 2008) is a robust modeling framework built on top of the Eclipse Integrated Development Environment (IDE). EMF offers a range of features for creating, editing, and validating models and metamodels. One of its key capabilities is generating code from models, allowing developers to create Java implementations that correspond to the metamodel’s classes. This way, each metaclass in the metamodel can be mapped to a corresponding Java class, enabling the instantiation of these classes to create models that adhere to the metamodel.

In EMF, metamodels are created using the Ecore meta-metamodel, which serves as the central language for defining metamodels within EMF (Steinberg et al., 2008). Ecore itself is based on the MOF (Meta Object Facility) meta-metamodel.

2.5 Eclipse Sirius

Eclipse Sirius (Madiot and Paganelli, 2015) is an Eclipse project that empowers the creation of customized graphical modeling workbenches. This allows architects to develop domain-specific modelers tailored to their unique business domains. Utilizing the Eclipse Modeling stack, particularly the Eclipse Modeling Framework (EMF), Sirius provides architects with a platform to declaratively specify their desired modelers using a description model. This specification encompasses crucial aspects, including domain element representation, visual styling information, and behavioral characteristics. Sirius offers a comprehensive environment for creating these specifications, resulting in description models that seamlessly function as Eclipse plug-ins. Notably, Sirius supports dynamic and incremental development, enabling architects to make real-time modifications to modeler definitions, ensuring continuous adaptability and responsiveness to the evolving requirements and challenges of dynamic business environments.

2.6 Eclipse Acceleo

Eclipse Acceleo³ is an open-source, template-based source code generation technology by the Eclipse Foundation. It enables the creation of custom code generators with comprehensive integration into the Eclipse IDE, featuring syntax highlighting, real-time error detection, quick fixes, and dedicated views for code generation design patterns and navigation within the code generator. Acceleo supports incremental generation, allowing manual modifications to generated code while preserving changes during subsequent regenerations by defining protected areas for safe modifications. The technology also incorporates the Acceleo Query Language (AQL) for navigation and querying of Ecore (meta)models, offering advanced features such as robust validation and support for calling methods defined through Java services.

3 RELATED WORK

ProvToolbox (Moreau, 2016) is a Java library developed by the W3C Provenance Working Group. It creates Java representations of PROV-DM and facilitates conversion between Resource Description Framework (RDF), PROV-XML, PROV-N, and PROV-JSON. Offering command-line utilities, APIs, and graphical interfaces, ProvToolbox supports diverse provenance-related tasks, including parsing and serializing documents, consistency checks, visualizations, and integration into existing software systems. The toolbox aims to enhance the adoption of provenance across domains, empowering users to effectively manage and leverage provenance information for tasks such as data analysis, reproducibility, accountability, and transparency.

PROV Python (Huynh, 2020) is a Python package designed for efficient handling of provenance information. Centered on data provenance and analytics, it allows users to create W3C PROV documents

³<https://eclipse.dev/acceleo/>

in Python, export them to formats such as PROV-N and PROV-JSON, and visualize them graphically. The package also facilitates interaction with ProvStore (Huynh and Moreau, 2015), a cloud-based provenance repository.

While the tools discussed ensure compliance with the W3C PROV standard and support provenance recording and conversion between different formats, they lack direct graphical representation of provenance and the ability to convert from a graphical format to textual formats like PROV-N. Currently, no available tool allows for graphical modeling of provenance. The tool presented in the forthcoming section aims to address this gap by offering intuitive and interactive graphical modeling of provenance under the W3C PROV standard. This tool streamlines recording and extraction provenance from software systems. Once graphically modeled, it is possible to export the provenance record directly in PROV-N notation.

4 GRAPHICAL PROVENANCE MODELING TOOL

Provenance modeling, akin to modeling other software aspects, enables the visualization of dependencies and relationships within data manipulated by the software. Graphical provenance modeling is especially beneficial, providing system developers with an interactive and visual approach to handling provenance models. Following model design, developers can generate corresponding PROV-N codes and instrument the software to record provenance based on established models and specified granularity preferences, including decisions about what provenance data to store, where within the application, and at what frequency.

The development of the graphical data provenance modeling tool followed a systematic three-stage approach: (1) converting the PROV-DM data model into an Ecore metamodel, named Ecore4PROV-DM; (2) constructing a graphical modeling tool based on the developed metamodel; and (3) creating model-to-text (M2T) transformation templates to convert provenance models built in the tool into PROV-N code instances. The subsequent subsections provide detailed insights into each of these stages.

4.1 Ecore4PROV-DM Metamodel

In the first stage of the graphical data provenance modeling tool development, we transformed the PROV data model into a cohesive Ecore metamodel, serving as the foundational framework for subsequent

modeling activities. This conversion closely adhered to the descriptions of W3C PROV components as outlined in (Moreau et al., 2013a), ensuring alignment with established standards and guidelines.

The resulting metamodel, named Ecore4PROV-DM and depicted in Figure 2, was constructed in EMF to represent the PROV data model. The metamodel's semantics are defined by relationships between metaclasses, with multiplicities represented by edges and their numbering. The metaclasses in blue constitute the metamodel's core: *Entity*, *Activity*, and *Agent*.

PROV-DM's creators categorized its model elements into six components, each serving specific purposes (Moreau et al., 2013a). The Ecore4PROV-DM metamodel follows this component-based approach:

1. Entities and Activities: define relationships between an *Entity* and an *Activity*. This component is represented in Ecore4PROV-DM in yellow.
2. Derivations: addresses derivations of entities from other entities and subtypes of derivation. Represented in Ecore4PROV-DM in red.
3. Agents, Responsibility, Influence: addresses agents and the relationships that represent their responsibility and influence over an entity or activity. Represented in Ecore4PROV-DM in green.
4. Bundles: refers to a mechanism for supporting the provenance of provenance. Represented in Ecore4PROV-DM in purple.
5. Alternatives: allows expressing alternatives or specializations of entities. Contains the *Entity* metaclass and two binary associations, which are self-relationships of the *Entity* metaclass: *alternateOf* and *specializationOf*.
6. Collections: concerns the notion of collection, which is an entity that has some members. The members are, in turn, entities and therefore their provenance can be expressed. This component is represented in Ecore4PROV-DM in orange.

The *Attributes* and *AttributeValue* metaclasses, depicted in grey in Figure 2, are used to instantiate the attribute-value pairs that most of the other Ecore4PROV-DM metaclasses can have.

4.1.1 Ecore4PROV-DM Metamodel Evaluation

A team of metamodeling experts, following the Metamodel Quality Requirements and Evaluation (MQuARE) framework, as proposed by (Kudo et al., 2020), evaluated the completeness, correctness and usability of Ecore4PROV-DM in comparison to the PROV data model (PROV-DM). MQuARE outlines 19 Metamodel Quality Requirements (MQRs) associated

with 23 quality measures, guiding metamodel evaluations.

The evaluation involved presenting evaluators with two foundational documents providing context for the metamodel domain, including an introduction to data provenance concepts, a brief outline of PROV-DM, and comprehensive insights into the implementation of Ecore4PROV-DM within the Eclipse Modeling Framework (EMF). After reviewing these documents and the Ecore4PROV-DM class diagram, the experts completed a questionnaire covering personal details, educational background, knowledge of data provenance and metamodeling, and specific inquiries about Ecore4PROV-DM's completeness and accuracy concerning its alignment with the PROV data model.

The evaluation questionnaire aimed to assess three Metamodel Quality Requirements (MQRs) outlined by the MQuaRE framework. Specifically, it focused on MQR02, evaluating the metamodel's completeness by assessing its coverage of concepts specified in the PROV-DM data model. Additionally, it addressed MQR03, examining the correctness of the metamodel to ensure accurate representation of PROV-DM concepts. Finally, MQR07, a usability requirement, measured the proportion of metamodel concepts that are easily identifiable to users. These MQRs were chosen for their alignment with research objectives, fa-

cilitating the evaluation of the metamodel's accuracy, comprehensiveness, and user-friendliness in correctly representing PROV-DM concepts.

Fifteen experts, all holding degrees in Computer Science, participated in the evaluation process. Of these, one had an undergraduate degree, while the rest held master's degrees, PhDs, or were pursuing a PhD. Regarding prior knowledge, 73.3% were unfamiliar with data provenance, 20% lacked knowledge of Model-Driven Engineering (MDE), and 26.7% were not acquainted with the Eclipse Modeling Framework (EMF). Regarding the support documents, 93.4% of respondents found the document on the fundamentals of data provenance and PROV-DM easily understandable, and 80% expressed clarity in comprehending the document explaining the implementation of Ecore4PROV-DM in EMF. The remaining evaluators affirmed that both documents were sufficiently clear, with none indicating difficulty in comprehension.

The evaluation results demonstrate unanimous agreement among the participating experts regarding the seamless alignment of Ecore4PROV-DM with the PROV-DM specification (MQR02 and MQR03). However, one specialist expressed reservations concerning the modeling of the "End" concept (*wasEndedBy*) within Component 1, and another specialist raised concerns about the implementation of Compo-

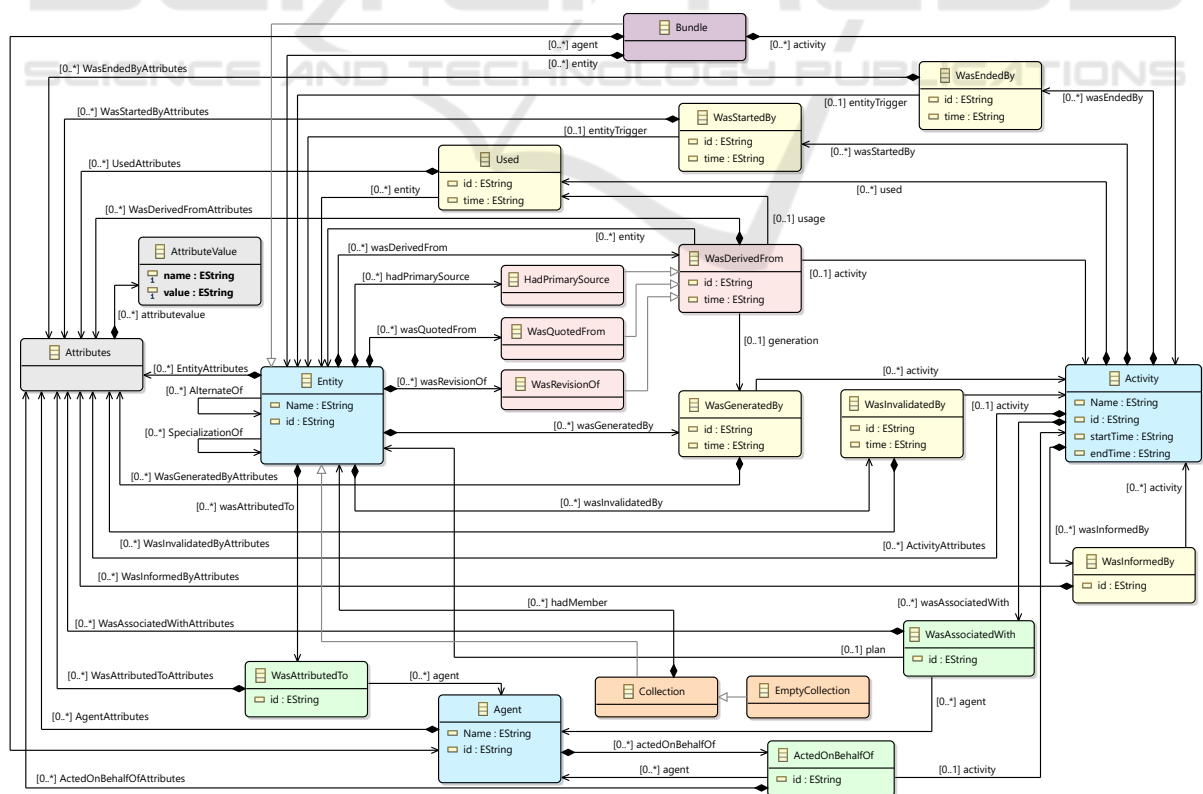


Figure 2: Ecore4PROV-DM: Ecore metamodel representing the PROV-DM Data Model.

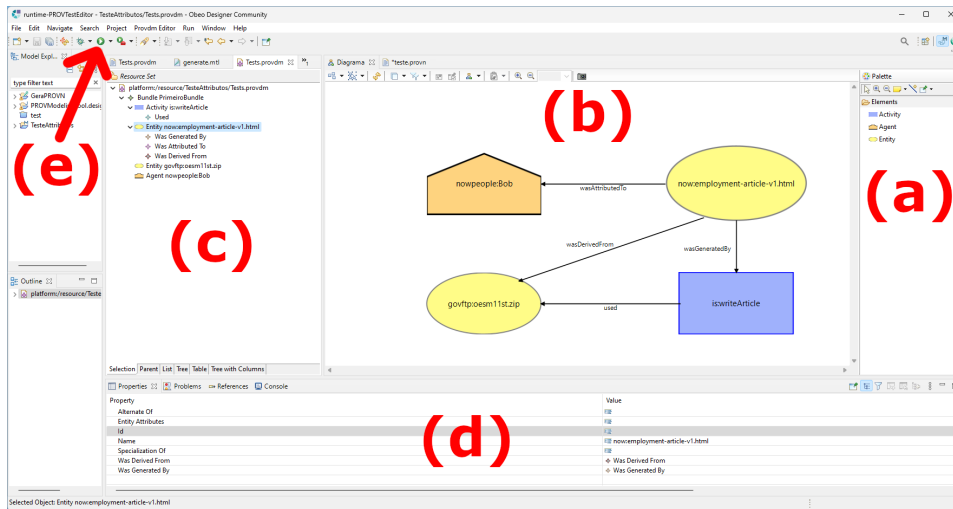


Figure 3: MDE-based graphical tool for modeling data provenance according to the W3C PROV standard.

ment 4 (*Bundles*).

In the MQR07 context, evaluators highlighted that the “Specialization” (*specializationOf*) concept in Component 5, while correctly modeled, is less apparent in the metamodel, with 33.3% finding it not easily identifiable. Similarly, the “Member” (*hadMember*) concept in Component 6 and the “Alternative” (*alternateOf*) concept in Component 5 were identified as not evident by 26.6% of specialists. However, the majority of the twenty-two concepts across the six PROV-DM components were deemed clearly evident in Ecore4PROV-DM by most evaluators. A comprehensive report detailing the outcomes of this evaluation is subject to publication in a forthcoming paper.

4.2 Development of the Graphical Data Provenance Modeling Tool

The second stage focused on constructing the graphical modeling tool, utilizing the Ecore4PROV-DM metamodel from the preceding stage. The concrete syntax of the metamodel was implemented in Eclipse Sirius, using the Obeo Designer⁴ tool, which was chosen for its seamless integration with EMF, Aceleo, and Sirius, along with its capacity to switch perspectives during modeling. This choice facilitated the creation of an intuitive and visually expressive modeling environment. To ensure adherence to the W3C PROV standard, style definitions were carefully integrated, representing entities as yellow oval nodes, activities as blue rectangles, and agents as orange “pentagon houses.” The relationships between these elements were depicted as oriented edges with labels denoting the relationship type.

⁴<https://www.obeodesigner.com/>

4.3 Model-to-Text Transformation

In the third stage, templates were developed within Eclipse Aceleo for all the elements that make up the six components of the PROV data model, in order to allow the provenance model developed in the graphical modelling tool to be converted into PROV notation (PROV-N) code. The language syntax of each element followed the PROV-DM documentation guidelines (Moreau and Groth, 2013). This stage enabled the synthesis of a comprehensive provenance model, leveraging the PROV data model as the underlying foundation while embracing the expressive power of the PROV-N notation.

Figure 3 provides an overview of the graphical provenance modeling tool. The Element Palette (a) on the right allows the selection and addition of W3C PROV standard elements (Activity, Agent, and Entity) to the graphical model by dragging them to the Diagram area (b). Clicking on an element in the Diagram area or using the Tree Editor (c) enables editing of its Properties (d) and the addition or modification of relationships with other model elements. The Execute icon (e) triggers the execution of the modeling tool, generating corresponding PROV-N code that represents the graphical model. This code encapsulates the model’s relationships, attributes, and structure, providing a textual representation of the provenance information encoded in the graphical representation.

5 MODELING DATA PROVENANCE GRAPHICALLY

In this section, we demonstrate the use of the graphical provenance modeling tool. After modeling each example, the tool automatically generated the respective PROV-N code, which was subsequently manually validated for correctness, ensuring that the tool is generating valid PROV-N code. For validation, we used the PROV document validation service provided by the Open Provenance website⁵. Due to space limitations, we will refrain from presenting the PROV-N codes generated by the tool.

5.1 Image Acquisition via Sensors

The first example demonstrates the graphical modeling of a provenance model that is publicly available on ProvStore⁶, a cloud-based provenance repository.

This particular provenance model is associated with the Urban Observatory project at Newcastle University in the United Kingdom. The Observatory monitors various urban indicators through streams of active sensors and CCTV cameras. Due to space constraints, we have only modeled a portion of this provenance model within the tool. Specifically, we have focused on the bundle that specifies the provenance data collected during the image acquisition process from sensors. This model includes two entities, *var:sensor* and *var:image*, along with one activity, *var:acquisition*. Each of these model components has its set of properties and values, detailing the data collection process. Figure 4 illustrates the graphical provenance model.

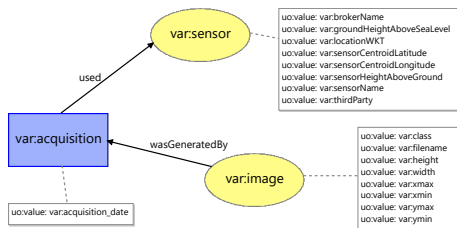


Figure 4: Graphical provenance model depicting an image acquisition process using a sensor.

5.2 Graphical Modeling of Provenance Templates

A provenance template (PROV-Template) (Moreau et al., 2018) is a PROV document that delineates the

⁵<https://openprovenance.org/service/validator.html>

⁶<https://openprovenance.org/store/documents/497>

desired provenance to be generated. Within a provenance template, variables are employed as placeholders for values. Consequently, a provenance template serves as a declarative specification of the intended provenance to be produced by an application. A collection of bindings establishes associations between variables and corresponding values. The PROV-Template expansion algorithm, when provided with a template and a set of bindings, produces a provenance document in which variables have been substituted with their corresponding values.

Hence, the first step in this methodology involves designing a “provenance model” that describes the structure of the provenance that is intended to be generated or captured. This task is usually done by drafting the provenance model on paper and then translating the model manually into PROV-N notation. However, this process is both antiquated and susceptible to errors. By employing the graphical provenance modeling tool introduced in this paper, it becomes possible to interactively design the provenance model graphically. Once the model is finalized, generating its PROV-N code is a straightforward task. This code is then employed by the expansion algorithm, in conjunction with the “binding” file and the designated database serving as the source of provenance data.

Figure 5 illustrates the provenance model presented in (Moreau, 2017), describing an arithmetic operation, for example, the sum of two values resulting in a third value. Thus, the arithmetic operation type is denoted by the activity *var:operation*, which uses two input values, represented by the entities *var:consumed1* and *var:consumed2*, producing a result represented by the entity *var:produced*, which is triggered by an *var:agent*.

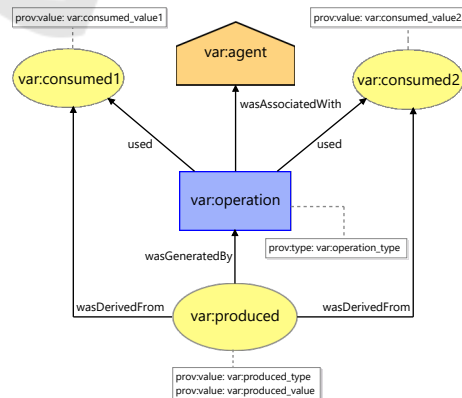


Figure 5: Graphical provenance model depicting the PROV-Template for an arithmetic operation.

6 CONCLUDING REMARKS

This paper presented a graphical tool for data provenance modeling based on the W3C PROV standard. The development involved three key stages: converting the PROV data model (PROV-DM) into an Ecore metamodel, constructing the graphical tool with Eclipse Sirius, and utilizing Eclipse Acceleo for model-to-text transformations to generate PROV-N code. The Ecore4PROV-DM metamodel underwent evaluation by fifteen metamodeling experts, resulting in consensus on its alignment with PROV-DM.

The graphical tool addresses a significant gap by providing a user-friendly interface for creating expressive data provenance models, enhancing visualization, and comprehension of provenance relationships. It allows exporting PROV-N code from the graphical model, ensuring interoperability and compatibility with existing provenance representations. The tool facilitates seamless integration of data provenance into systems, fostering the adoption of provenance-aware applications and extending its utility to diverse applications, including provenance information extraction from databases using the PROV-Template approach.

While demonstrating the tool's use in certain scenarios, further evaluation is crucial for future research. Usability studies and expert feedback would enhance functionality, user experience, and effectiveness. Investigating the tool's integration with other provenance management systems and assessing performance in real-world scenarios will contribute to its practical applicability.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support provided by the Instituto Federal Goiano in facilitating and funding this research.

REFERENCES

Bruel, J.-M., Combemale, B., Guerra, E., Jézéquel, J.-M., Kienzle, J., de Lara, J., Mussbacher, G., Syriani, E., and Vangheluwe, H. (2018). Model transformation reuse across metamodels. In Rensink, A. and Sánchez Cuadrado, J., editors, *Theory and Practice of Model Transformation*, pages 92–109, Cham. Springer International Publishing.

Bucchiarone, A., Cabot, J., Paige, R. F., and Pierantonio, A. (2020). Grand challenges in model-driven engineering: an analysis of the state of the research. *Software and Systems Modeling*, 19(1):5–13.

Glavic, B. (2021). Data provenance. *Foundations and Trends® in Databases*, 9(3-4):209–441.

Herschel, M., Diestelkämper, R., and Ben Lahmar, H. (2017). A survey on provenance: What for? What form? What from? *The VLDB Journal*, 26(6):881–906.

Hu, R., Yan, Z., Ding, W., and Yang, L. T. (2020). A survey on data provenance in IoT. *World Wide Web*, 23(2):1441–1463.

Huynh, T. D. (2020). PROV Python - A library for W3C Provenance Data Model supporting PROV-JSON, PROV-XML and PROV-O (RDF). Available online: <https://pypi.org/project/prov/>.

Huynh, T. D. and Moreau, L. (2015). ProvStore: A Public Provenance Repository. In Ludäscher, B. and Plale, B., editors, *Provenance and Annotation of Data and Processes*, pages 275–277, Cham. Springer International Publishing.

Kudo, T. N., Bulcão-Neto, R. F., and Vincenzi, A. M. R. (2020). Metamodel Quality Requirements and Evaluation (MQaRE). Technical report, Departamento de Computação, UFScar, São Carlos-SP, Brazil. v 2.0.

López-Fernández, J. J., Cuadrado, J. S., Guerra, E., and de Lara, J. (2015). Example-driven meta-model development. *Software & Systems Modeling*, 14(4):1323–1347.

Madiot, F. and Paganelli, M. (2015). Eclipse sirius demonstration. *P&D@ MoDELS*, 1554:9–11.

Moreau, L. (2016). ProvToolbox - Java library to create and convert W3C PROV data model representations. Available online: <https://lucmoreau.github.io/ProvToolbox/>.

Moreau, L. (2017). PROV-Template: A Quick Start. Available online: <https://lucmoreau.wordpress.com/2017/03/30/prov-template-a-quick-start>.

Moreau, L., Batlajery, B. V., Huynh, T. D., Michaelides, D., and Packer, H. (2018). A templating system to generate provenance. *IEEE Transactions on Software Engineering*, 44(2):103–121.

Moreau, L. and Groth, P. (2013). *Provenance: An Introduction to PROV*. Springer International Publishing.

Moreau, L., Missier, P., Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Lebo, G. K. T., McCusker, J., Miles, S., Myers, J., and Sahoo, S. (2013a). PROV-DM: The PROV Data Model. Available online: <https://www.w3.org/TR/prov-dm/>.

Moreau, L., Missier, P., Cheney, J., and Soiland-Reyes, S. (2013b). PROV-N: The Provenance Notation. Available online: <https://www.w3.org/TR/prov-n/>.

Pérez, B., Rubio, J., and Sáenz-Adán, C. (2018). A systematic review of provenance systems. *Knowledge and Information Systems*, 57(3):495–543.

Rodrigues da Silva, A. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139–155.

Steinberg, D., Budinsky, F., Merks, E., and Paternostro, M. (2008). *EMF: Eclipse Modeling Framework*. Pearson Education.