Design and Implementation of a Diagrammatic Tool for Creating RDF graphs

Anca Chiș-Rațiu[1] and Robert Andrei Buchmann[2]

Business Informatics Research Center, Faculty of Economics and Business Administration, Babeș-Bolyai University, Cluj Napoca, Romania

[1]achisratiu@yahoo.com,[2]robert.buchmann@econ.ubbcluj.ro

Abstract. Graph databases are the next generation of relational databases, as they allow convenient retrieval of complex network structures and employ graph theory to store and navigate relationships. The resulting data models are simpler and more expressive than those produced using traditional relational databases. However, there is a shortage of tools for creating such graphs in an intuitive visual way. Metamodeling can help in this respect, as it enables the agile customization of tools for diagrammatic knowledge representation and editing, under constraints imposed through metamodels.

Our goal is to present a modeling tool that was customized to support the creation of Resource Description Framework (RDF) data graphs, by integrating notions of Conceptual Modeling and the Agile Modeling Method Engineering (AMME) framework, using ADOxx as an instance environment to obtain a usable prototype - a starting point for the future evolution an Open Models Laboratory project. The proposed modeling tool is inspired by the model-driven code generation paradigm, as an ADOxx script was developed to generate, out of diagrammatic structures, RDF graphs written in the N-triples serialization syntax. The paper highlights benefits of the proposed modeling tool as this successfully resolves fundamental user-oriented issues regarding the easy production of knowledge graphs guided by the Linked Enterprise Data paradigm and supported by a Conceptual Modeling "look and feel".

Keywords: Resource Description Framework, Conceptual modeling, Enterprise knowledge graphs

1 Introduction

Web 2.0 established a large information space where people share valuable data and contribute human-readable knowledge in various fields - see Wikipedia, a free encyclopedia project, where many individuals are motivated to contribute by adding and revising content. The next stage, Web 3.0, has encouraged a similar production scale and scope for machine-readable knowledge to be made available to Linked Enterprise Data projects [1], where experts share and retrieve large-scale connected RDF graphs [2] that store flexible relational data enriched with domain-specific rules or schemata.

Linked Data is about information creation coupled with information sharing, where documentation, construction and distribution processes are equal in terms of importance.

In order to support enterprise knowledge creation, conceptual modeling methods can be employed - not only for data modeling purposes, but also for capturing domain-specific abstractions, business entities and their relationships. Modeling methods are deployed as tools that enable the description of a system based on instantiated concepts defined on a metamodel level, and can be enhanced by mechanisms that are relevant for the production or transformation of knowledge graphs.

The goal of this paper is to present a conceptual modeling tool that was customized with the help of a metamodeling methodology to allow users to easily construct RDF knowledge graphs by visual means. The model-driven code generation paradigm inspired the work, in the sense that the tool generates knowledge encoded in the machine-readable N-triples syntax [3], readily available for uploading and managing it in an RDF database management system. This started as a student dissertation project that opens the path towards establishing a research project fit to participate in the Open Models Laboratory (OMiLAB) ecosystem [4][5].

Code generation is acknowledged as a key benefit of conceptual modeling, supporting the development of software systems in the sense that development time becomes consistently shorter. As examples, Modeliosoft tool products [6] provide model-driven code generation for development languages like Java (from UML classes), C++ and SQL (generated from ERD diagrams). Process automation is made possible by generating BPEL or XPDL process descriptions from BPMN diagrams. However, there is a shortage of tools that generate RDF serializations from visual models of their corresponding graphs – i.e., RDF visual tools typically provide a visualization of already created RDF graphs, instead of providing a graphical way to build such views and then generate the corresponding machine-readable structures. In the first draft of our prototype, the target syntax of choice is N-triples (supported by any RDF management systems). In the future this can be extended towards a richer knowledge editor, to be distributed for open use within the OMiLAB global network.

The motivation for creating a new modeling tool for creating RDF data graphs is that nowadays we have an increased utilization of enterprise knowledge, but the means of creating that knowledge with an optimal learning curve for non-technical users are limited. RDF graphs should be created in an easy way, not more complicated than filling data in the cells of SQL tables, and not necessarily conforming a preimposed schema since RDF graphs can be schemaless databases – i.e., instance data can be created separately from the schema; or, the schema is employed for semantic annotation purposes, to support reasoning rather than validation. Linked Enterprise Data is commonly lifted or derived through adapters (e.g., D2RQ [7]) from legacy non-graph data sources. When data is created from scratch, graph creation is often blended with ontology engineering in the same tool (form-based or text-based), following the traditional relational database creation process – first schema, then instance data.

We aim to minimize the effort of RDF data creation by developing a diagrammatic tool with conceptual modeling "look and feel", potentially evolving towards a flexible

knowledge base editor (with the possibility to add domain-specific dynamic notation and other metamodeling-powered features). In the current draft, the tool only allows simple graph editing, limited annotation of nodes and the generation of machine-readable serialization. The key novelty is that this is built on an open access meta-modeling platform, ADOxx [8], to ensure the evolution of the tool according to future academic exploitation goals – e.g., teaching semantic technology with user-friendly tool support.

The requirements selected for this tool originate in teaching goals – i.e., to provide an intuitive RDF editing toolkit to novices, focusing on a visual building process as a replacement for the traditional text-based or form-based building. Additionally, a meta-requirement imposed the need to have an agilely evolving tool that can incorporate additional functionality and domain-specific annotations towards the goal of enabling enterprise knowledge creation – hence the adopted metamodeling approach.

The remainder of the paper is further structured as follows: Section 2 introduces background on the enablers - RDF, AMME and the ADOxx metamodeling platform. Section 3 provides methodological considerations in relation to Design Science, as a guide for this effort. Section 4 provides comments on related works and proposed benefits. Section 5 presents the modeling method conceptualization providing insights about syntax, semantics, the serialization mechanism and the modeling procedure. The paper ends with conclusions and an outlook to future developments.

2 Background on Technological Enablers

Enterprises are willing to adopt the Linked Enterprise Data concept [1] due to its benefits regarding semantic interoperability and connectivity of data originating in legacy silos. However, this also comes with a need of easily building connected data, requiring no more effort than when filling data tables in traditional databases. Conceptual modeling can help in this respect, if we see it through the lens of agility – i.e., conceptual modeling tools that do not necessarily follow established methods (ER, UML etc.) but are instead customized for specific requirements. In our case, these requirements originate in the need to capture RDF semantics with (i) minimal notation, (ii) a Conceptual Modeling "look and feel", and (iii) mechanisms for generating machine-readable RDF from diagrammatic graphs.

Agile development accepts change and even expects it, therefore the proposed tool also highlights characteristics of Agile Modeling Method Engineering (AMME) as a key methodology for customizing modeling tools for a diversity of purposes. According to [9] and [10], AMME provides a conceptualization method that repurposes agility principles established in software engineering, and ensures that the necessary semantics are captured in relation to modeling needs. This framework has also been applied in the community-oriented research environment of OMiLAB, in the creation of a multitude of tools – see BEE-UP (an educational project for teaching Model-Driven Software Engineering and Business Process Management topics) [11], or the ComVantage method (a research-oriented project addressing Knowledge Management and Enterprise Architecture Management concerns) [12]. Both mentioned tools

allow the lifting of RDF graphs from modeling languages (e.g., UML, the domain-specific ComVantage language) – however those graphs are limited to the semantics prescribed by the supported languages. In our case, AMME is employed to tailor a minimal modeling language directly for the RDF semantics, without any intermediate abstraction layer.

2.1 Resource Description Framework (RDF)

The standard technology for representing and sharing semantic information is the Resource Description Framework (RDF), where "resources" can be anything including documents, people, objects or abstract concepts. In particular, RDF is used to publish and interlink data on the Web or to represent knowledge in knowledge management applications. RDF employs a graph-based data model, which is significantly different than the earlier interoperability standards such as XML (based on DOM and hierarchical data structures). Data graphs are more flexible than DOM trees because the queries are more powerful and flexible as they can navigate a graph in any direction along arbitrary chains of relationships. RDF databases are considered NoSQL databases since they are queried with other means than SQL – the standard language for this is SPARQL [13]. Consequently, RDF graphs are the main data model serving the Linked Data paradigm. They are also related to the concept of Smart Data, if reasoning and rule systems are deployed on top of RDF graphs.

The RDF data model is based on small units called statements, describing resources in the form of triples of resource identifiers (URIs):

<Subject> <Predicate> <Object> .

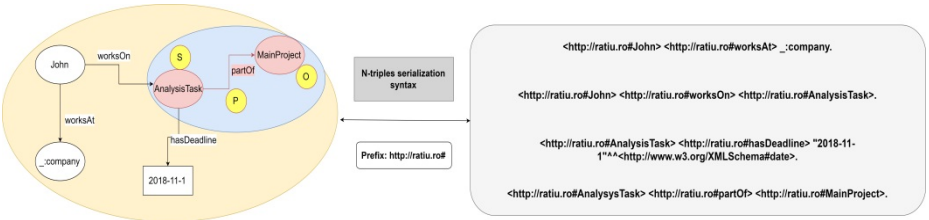

Fig. 1 Resource Description Framework (RDF)

As exemplified in Figure 1, the triples can be visualized as a connected graph, consisting of nodes and arcs. The subjects and objects of the triples are the nodes and the predicates form the arcs – however, predicates themselves can also be described thus becoming subjects (nodes) in other graphs. Resources are of two types: addressable, easy to access on the Web through a URL (files, e-mail accounts, Web services, HTML paragraphs) and non-addressable, representing abstract or concrete things that can have a representation on the Web but they generally exist independently of the Internet (concrete persons, places, attributes, verbs, concepts). This second category rely on a universal identification scheme - URIs which become the terms used to write RDF statements and at the same time act like identifiers (similar to ISBNs for

books, but also indicating provenance as an HTTP domain). Most of the terms used in RDF statements are URIs or URLs, but there are some exceptions: the object can be anonymous/blank, without identity or provenance, or a literal representing a simple data value of some type - integer, boolean etc. Subjects can also be anonymous/blank, while the predicate is always an URI. RDF offers multiple syntaxes for writing such graphs in machine-readable form - the most important ones are Turtle, RDF/XML and N-triples. The paper's proposed modeling tool is programmed to automatically generate in N-triples the RDF data graphs that are built by visual means, as enabled by a metamodeling platform (ADOxx) and methodology (AMME).

2.2 Agile Modeling Method Engineering (AMME)

Agile software development started as a global movement promoting a different way of thinking than traditional approaches such as waterfall, encouraging evolutionary development, continuous improvement and adaptability. In order to support agile development, to transfer agility principles also to the world of conceptual modeling and to motivate the relevance of agile modeling methods, the framework of AMME was created. The following paragraph summarizes its role, as described in [10].

AMME is guided by modeling requirements which can evolve as richer semantics become necessary for design-time and run-time modeling use cases, or are affected by change requests as users become more familiar with modeling possibilities. The outcome of AMME is a modeling tool tailored for a customized diagrammatic language, including any relevant functionality that can be built on models created with that language. The OMiLAB environment facilitates the deployment of AMME, fosters a community of researchers having a common understanding of model value and of the concept of modeling method as introduced by [14]. Also, the NEMO summer school event facilitates knowledge transfer between academics, practitioners and educators – during this, the principles of AMME are taught in dedicated training sessions [15].

AMME has been applied to several OMiLAB projects - some of them are presented in [16]. In our case the key benefit for adopting AMME was related to the evolving nature of the tool, incorporating gradual understanding of RDF semantics during a student project, and, on the other hand, facilitating an evolution roadmap towards a rich knowledge editing environment that has the potential to become a future OMiLAB project.

2.3 The ADOxx metamodeling platform

ADOxx [8] is a metamodeling platform employed as a fast prototyping environment for AMME to allow method engineers to develop their modeling method and to iteratively loop through a conceptualization lifecycle for incremental enrichment of the modeling prototype. A meta-metamodel is built in ADOxx to allow the implementation of a modeling language notation, grammar and vocabulary. In our work, these are subordinated to the goal of creating RDF graphs by visual means. In addition, an internal scripting language allows the implementation of model-based functionality,

which in our case is the generation of N-triples serializations from the visual structures.

In ADOxx diagrammatic elements are described by their conceptual schemata defining machine-readable properties to be exposed to the modeler for editing. The schemata will generate annotation sheets supporting various ways of describing or linking model elements - they are repurposed in the work at hand to derive RDF descriptions. A formal description of the ADOxx metamodeling approach was presented in [17].

3 Methodological Considerations

Design Science is a solution-oriented methodology focusing on the design and investigation of artifacts developed to address a particular problem. For example, the design of an enterprise modeling method aligned to business goals is subordinated to Design Science, as opposed to observational science which focuses on the utilization of such methods. A modeling method becomes usable when deployed as a modeling tool – however, both a method and a tool can be considered Design Science artifacts (addressing requirements on different levels of abstraction and reusability). More generally, artifacts can be models, methods, constructs, design theories. All these categories are knowledge-containing - the knowledge may cover design logic, conceptualization cycles, tool usage procedures etc. In our case, the artifact is a modeling tool whose development steps have deployed the building blocks of the "modeling method" notion published in [14]: syntax and semantics, notation (minimal), mechanisms, modeling procedure. These building blocks will establish the structure of Section 5 which details the technical aspects of the implementation.

The proposed tool can be considered the deployment of a modeling method – however, due to the origin of this tool in a student project guided as a learning experience, the development process was as follows: we initiated this effort as a simple ADOxx implementation exercise (triggered by the requirement to support basic RDF editing), followed by an abstraction exercise (i.e., a reflection on how the results illustrate the abstract notion of modeling method). For the future, we aim to formalize and extend this modeling method (as recommended by AMME), then to revisit the tool implementation coming from an enriched understanding of the modeling method concept and finally to evolve the tool according to the richer method specifications.

The project aims to address the shortage of visual RDF editing tools by providing an open use and open source tool for this purpose. In the engineering cycle inspired by Design Science, the first step was the problem investigation, followed by the treatment design step (the interaction between artifact and context – a university setting where RDF is being taught as a knowledge representation technique). The next step, the design validation, takes place whenever changes are being made in artifact – currently this mostly took the form of basic tool testing. In addition, the treatment validation also involved the expert opinion of professors supervising the project. The simulation method has been used as the prototype can be easily applied in a simulated context for producing RDF graphs given some enterprise description.

4 Related Work and Proposed Benefits

A variety of tools have been developed during recent times to support a visual management of knowledge representations, including knowledge graphs, ontologies or diagrammatic models. TopBraid Composer [18] combines semantic modeling capabilities with data conversion features, in order to build Semantic Web and Linked Data applications. MS Visio [19] provides the ability of creating easy and intuitive diagrams using shapes and templates. An open-source ontology editor and framework for building intelligent systems, Protege [20], is already supported by a strong community of both academics and practitioners – it enables users to build ontology-based solutions, being adopted in areas such as biomedicine, e-commerce etc. However, Protege does not provide direct conceptual model-driven means of visually creating RDF graphs (the user cannot add new nodes and properties to an empty view), the tool only allows creating graphs by using editing forms (subordinated to ontologies) and afterwards the generation of graph visualizations with the help of plug-ins such as OntoViz [21].

Callimachus [22] is a data editing platform and provides an easy way to create custom forms for data entry that end up in RDF graphs. Also, RDF Studio [23] is an environment for editing, browsing and visualizing RDF and ontologies, however it does not employ an agile conceptual modeling foundation to allow for free extension and evolution. DotNetRDF [24] is a Windows GUI toolset that includes a text editor for RDF, providing syntax highlighting, validation and auto-completion. These are replaced in our case by a visual editing tool where correctness is controlled through metamodeling declarations (domains, ranges, cardinalities) and a script that produces correct code from the diagrammatic representation. A tool that is closer to our goal is OWLGrEd [25], one of the first to provide graphical interaction for editing ontologies.

The proposed modeling tool is based on an evolving modeling method, driven by changing requirements that originated in a student project. This required the application of AMME in building a flexible modeling tool which can also evolve as the RDF specification evolves, or as additional features are required. Flexibility is enabled by employing metamodeling and ADOxx, where both the visualization layer and the code generation layer can be further enriched. In the current iteration the proposed tool is limited to instance-level graphs and limited schema support, therefore it does not extend to the scope covered by e.g., OWLGrEd. However, its implementation relying on OMiLAB resources (AMME, ADOxx) is open to a flexible evolution, and to an investigation of mapping the formal semantics of RDF and OWL to underlying metamodeling structures that facilitate the tool implementation.

5 Method Conceptualization

5.1 Syntactic and Semantic Specification (Metamodel)

The modeling tool presented in this paper allows the visual construction of RDF data graphs. The syntax in RDF is the set of rules and principles that govern the structure of RDF statements, as composed of the three terms: a subject, a predicate (or property, relation, attribute) and an object. Based on these, a modeling symbol has been attached to each machine-readable concept in the ADOxx Development Toolkit.

Figure 2 describes the concepts that have explicit machine-readable semantics based on the minimum graph-based RDF semantics. The root class, D_construct, provides the predefined ADOxx skeleton for all concepts in modeling tool, any concept inherits from it several built-in properties allowing the definition of graphical notation and the conceptional schema of attributes for editable annotations/properties. The class Node was created for inheritance purposes, as this is not visible to users. The other subclasses represent the typical term types in RDF statements (URI/URL, Simple/Literal values, Anonymous/Blank, RDF Class). URI allows to set the prefix, whereas Literal values can get a data type and a language code as annotations. All of these terms are contained in Graphs, which is a class created for aggregation purposes (see the graphic container in Figure 4).

Modeling relations manifest as visual connectors, inside a model type. The relations created in the ADOxx Development Toolkit are highlighted in Figure 3, to capture some of the standard relations that occur in RDF graphs (rdf:type, rdfs:subClassOf) as well as visual connectors that become predicates.

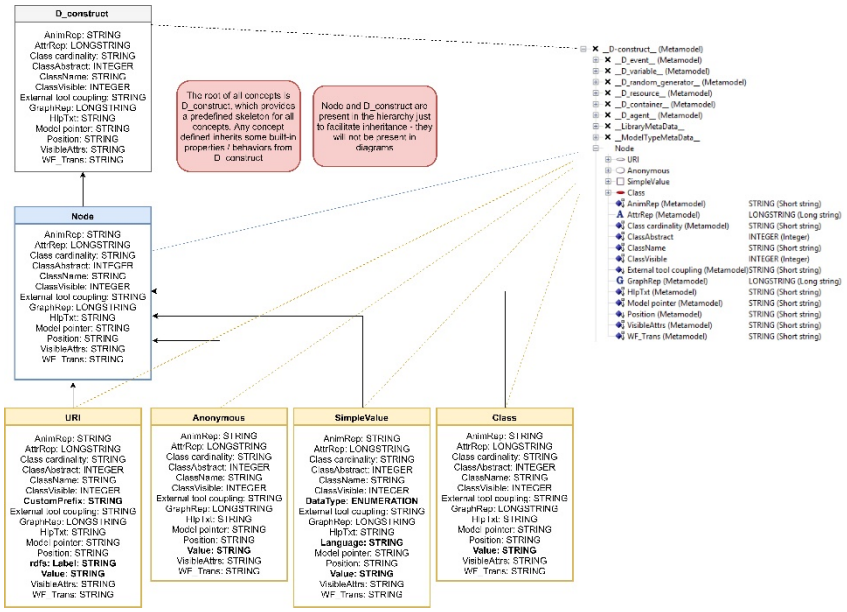

Fig. 2. Concepts having explicit machine-readable semantics

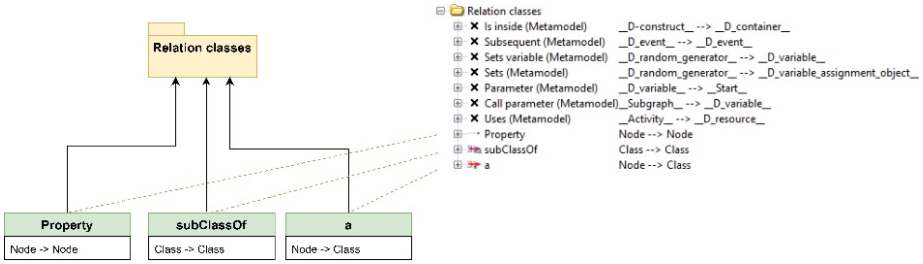

Fig. 3. Relation classes proposed in the modeling tool

5.2 Model Serialization Mechanism

The proposed modeling tool is based on the model-driven code generation paradigm as an ADOxx script was developed to generate in N-triples syntax a serialization of the diagrammatic graphs. In the ADOxx Development Toolkit a menu item was created to trigger the script and generate the serialization code. Also, a function was created for storing internal IDs of modeling objects according to every connector that is created in diagrammatic models, which form the statements created graphically by the user by drawing the property relation between the subject and object nodes.

Figure 4 presents an example of a model and also the usage flow: the user creates visual data graphs – for each GUI event object IDs are stored in a text file; afterwards, this file is converted by a user-triggered menu item in the N-triples syntax according to the types of various nodes.

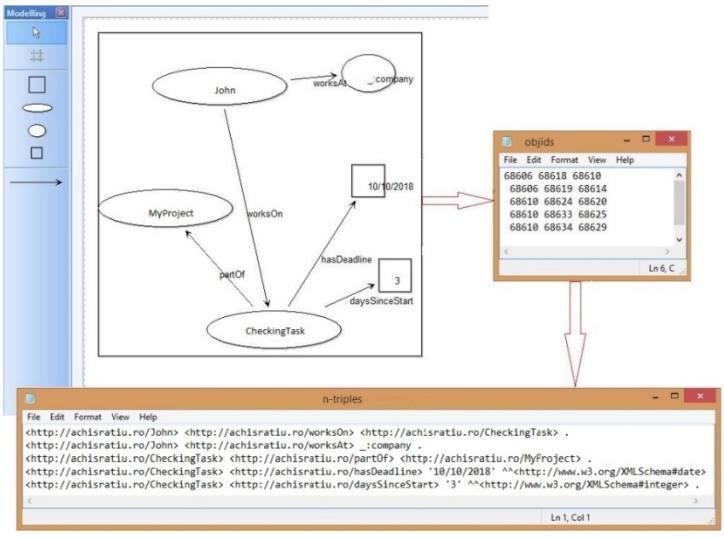

Fig. 4. Example of enterprise data graph created in the modeling tool

The ADOxx script (see a code sample in Figure 5) for generating serializations in the N-triples syntax comprises the following procedures:

- Procedure "GetGraphPrefix" searches graphs in the active model and obtains their ids. For each graph found, it searches for the attributes and the value is stored for using it further on nodes, to form the URIs;
- Procedures "TransformURI", "TransformAnonymous", "TransformSimpleValue", as the IDs of nodes have been sent to transform from object ID to the specific forms prescribed by the N-triples syntax: <URI> for nodes of type URI, underscore-prefixed identifiers for Anonymous, and typed values for literal nodes. For URIs, the graph-level prefix is added;
- Procedures "TransformType", "TransformSubClassOf", the IDs of visual connectors are transformed according to their prescribed types (rdf:type, rdfs:subClassOf) or according to their URI.

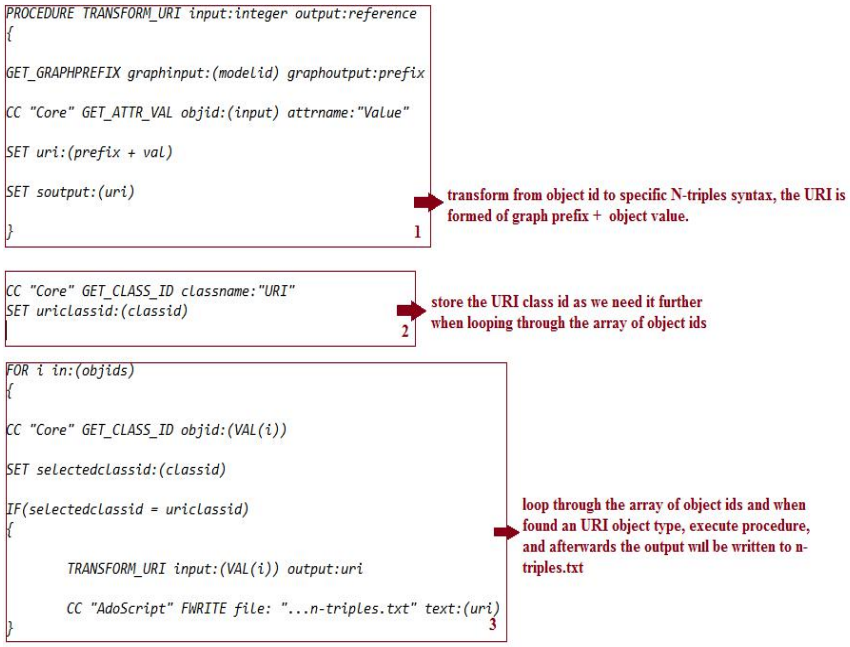

Fig. 5. ADOxx script sample for generating N-triples

5.3 Modeling procedure

As suggested before, the modeling procedure is determined by how different steps of the code generation were scripted: IDs are collected through GUI interaction events and written in a text file during the graphical creation of connectors. Then, the user decides when to convert this collection of IDs into N-triples according to the syntactic rules of N-triples.

Validation checks are also applied to ensure correctness: the metamodel-level constraints allow the user to create graphs to contain any subjects, predicates and objects as long as the statements are correctly and conforming to RDF principles. In other words, it prevents errors such as: a simple value cannot be a subject, a "subClassOf" connector can only connect "Classes" etc. Another aspect that requires sanity checks is the graph structure: the statements must be part of some graph, and also this graph must have a previously defined prefix/namespace attached to it. This opens the path to further extend the tool towards a more refined distinction between named graphs and simple graphs.

6 Conclusions and Outlook

The paper contributes to research related to the implementation of modeling tools that support users in expressing graph-like knowledge conforming the Resource Description Framework. RDF data graphs have a high degree of flexibility in data representation, which can be easily integrated on a metamodeling level to support the creation of enterprise knowledge graphs by visual means. The modeling tool described in this paper can have impact in the Linked Enterprise Data area - however current evaluations did not involve real stakeholders. The tool was only used in a teaching environment and in a testing context. An RDF creation productivity framework is under development to support the compared evaluation of such tools.

The proposed modeling tool showcases the agility enabled by AMME in creating purposeful tools that do not deploy a traditional modeling language, but rather a diagrammatic method for producing technology-specific code. This can be included in the paradigm of Technology-Specific Modeling Languages, a flavor of Domain-Specific Modeling Languages as introduced in [26].

In future work, additional mechanisms will be added to allow integration with ontologies that will also be created with diagrammatic means (see [27]). The ontology level is now only rudimentary supported by the attachment of types to graph nodes.

7 References

1. Wood, D.: Linking Enterprise Data, Springer (2010)
2. Resource Description Framework (RDF), https://www.w3.org/RDF/, last accessed 2018/08/26.
3. RDF 1.1 N-Triples, https://www.w3.org/TR/n-triples/, last accessed 2018/08/26.
4. OMiLAB Global Network, http://www.omilab.org/psm/home, last accessed 2018/08/26.
5. Bork, D., Miron, E. T. "OMiLAB - An open innovation community for modeling method engineering", Proceedings of ICMIE 2017, Niculescu Publishing, p. 64-77 (2017).
6. MODELIO, https://www.modeliosoft.com/en/, last accessed 2018/08/26.
7. D2RQ, http://d2rq.org/, last accessed 2018/08/26.
8. ADOxx.org, https://www.adoxx.org/, last accessed 2018/08/26.
9. Karagiannis, D.: Agile modelling method engineering. In: Proceedings of PCI 2015, pp. 5-10, ACM, New York (2015)

10. Karagiannis, D.: Conceptual Modelling Methods: The AMME Agile Engineering Approach, LNBIP 273, pp 3-19, Springer, Berlin Heidelberg (2018)
11. Bee-Up, http://austria.omilab.org/psm/content/bee-up/info, last accessed 2018/08/26.
12. Buchmann, R.A., Karagiannis, D., Visic, N.: Requirements definition for domain-specific modeling languages: the Comvantage case. In: Proceedings of BIR 2013, LNBIP 158, pp. 19-33, Springer, Berlin Heidelberg (2013)
13. SPARQL Query Language for RDF, https://www.w3.org/TR/rdf-sparql-query/, last accessed 2018/08/26.
14. Karagiannis, D., Kuhn, H.: Metamodelling Platforms. In: Proceedings of EC-Web 2002, LNCS 2455, pp 182. Springer, Berlin Heidelberg (2002)
15. Next-Generation Enterprise Modelling in the Age of Internet of Things, http://nemo.omilab.org/2018/, last accessed 2018/08/27.
16. Karagiannis, D., Mayr, H. C., Mylopoulos, J. (Eds.), Domain-Specific Conceptual Modeling, Springer (2016)
17. Fill, H. G., Redmond, T., Karagiannis, D.: Formalizing Meta Models with FDMM: The ADOxx Case, Proceedings of ICEIS 2012, LNBIP 141, pp 429-451, Springer, Berlin Heidelberg (2018)
18. TopBraid Composer Maestro Edition, https://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/, last accessed 2018/08/26.
19. Visio - Microsoft Store, https://www.microsoft.com/en-us/store/collections/visio, last accessed 2018/08/26.
20. Protege, https://protege.stanford.edu/, last accessed 2018/08/26.
21. OntoViz, https://protegewiki.stanford.edu/wiki/OntoViz, last accessed 2018/08/26.
22. Callimachus, http://callimachusproject.org/, last accessed 2018/08/26.
23. RDFStudio, http://www.linkeddatatools.com/rdf-studio, last accessed 2018/08/26.
24. DotNetRDF, http://www.dotnetrdf.org/, last accessed 2018/08/26.
25. OWLGrEd, http://owlgred.lumii.lv/, last accessed 2018/08/26.
26. Harkai, A., Cinpoeru, M., Buchmann, R. A.: The "What" Facet of the Zachman Framework – A Linked Data-Driven Interpretation, In: CAiSE 2018: Advanced Information Systems Engineering Workshops, pp. 197-208, Springer (2018)
27. Fill, H. G.: SeMFIS: A Flexible Engineering Platform for Semantic Annotations of Concep-tual Models, Semantic Web 8 (5): 747-763 (2016).