Towards a modeling method for low-code knowledge graph building

Andrei Chiş[1], Robert Andrei Buchmann[1] and Ana-Maria Ghiran[1]

[1] *OMiLAB@UBB-FSEGA, Faculty of Economics and Business Administration, Babeş-Bolyai University, Cluj-Napoca, Romania*

Abstract

Context and motivation. Knowledge graphs (KGs) provide an effective means for representing and managing both knowledge and data in a uniform format. KG editors have traditionally taken the form of logic-based or slot-based ontology/schema editors obscuring the graph structure or limiting the graph perspective to visualizations of KGs created by other means.
Question/problem. KG implementation is tied to RDF serializations that citizen users might have difficulties handling. This poses a blockade on citizen-level adoption of knowledge capture means as the majority of stakeholders lack technical skills. There's a shortage of open tools enabling a diagrammatic user experience, particularly needed by educators and those who must build KG exemplars.
Principal ideas/result. The paper presents an early version of a modeling tool for developing KGs in a visual environment, providing a drag-and-drop user experience combined with tabular editing and features enabled by means of metamodeling.
Contribution. The proposed modeling tool advocates the repurposing of metamodeling practices towards establishing flexible low-code KG building environments to enable more citizen-centric knowledge capture methods based on diagramming.

Keywords
Low code knowledge graphs, RDF, Agile modeling method engineering, Conceptual modeling, ADOxx

1. Introduction

Even though knowledge graphs are not a new technology their citizen-level adoption has been deterred due to a series of factors such as technological readiness (limited number of market-ready products for many years). Building ontologies and graphs involves non-trivial skills regarding e.g. RDF serialization formats [1] such Turtle, TriG or N-Triples. Users must be proficient with knowledge representation before any attempt to capture even the simplest KG fragment – but most citizen users attempting to capture networks of associations are only familiar with mind mapping and similar visual means of note taking. With our work we try to find a middle ground between the two, by repurposing our experience with DSML (domain-specific modeling language) engineering towards developing a visual language for not only designing, but actually building graph-like visual structures that can be exported as RDF fragments for both the TBOX and ABOX levels.

Currently, the ongoing trend in the business and technological worlds of automating as much as possible with minimal code input/handling led to the apparition of "citizen development" and "citizen automation" [2], paving the way for "citizen knowledge acquisition" [3] as a new paradigm for sourcing and employing relevant organizational information from stakeholders. In order to democratize stakeholder access to Semantic Web-based knowledge acquisition, this paper presents early results from a project developing the GRAPHxx modeling tool, which aims to help users create ontologies and knowledge graphs in a visual, frictionless manner, hiding from users most parts of

Companion Proceedings of the 16th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling and the 13th Enterprise Design and Engineering Working Conference, November 28 – December 1, 2023, Vienna, Austria
andrei.chis@econ.ubbcluj.ro (A. Chiş); robert.buchmann@econ.ubbcluj.ro (R. A. Buchmann); anamaria.ghiran@econ.ubbcluj.ro (A. M. Ghiran)
0000-0003-0173-7250 (A. Chiş); 0000-0002-7385-1610 (R. A. Buchmann); 0000-0001-7890-9386 (A. M. Ghiran)

OWL's [4] technicalities and formalism. Even the more experienced knowledge graph developers such as KG engineers, educators or academics can benefit from a tool that simplifies the process of creating and managing graph artifacts, thus reducing the effort of knowledge capture, graph development or of building didactic examples that can be edited by visual means. Therefore, GRAPHxx acts as a visual-first, drag and drop development platform for KGs. Other capabilities offered by GRAPHxx are a Turtle format export and direct loading of exported graphs to a selected GraphDB [5] server for querying and operationalization.

Numerous visual tools or visual plug-ins for KG tools are available, such as Gra.fo [6], OWLGrEd [7], Protégé [8] or Graffoo [9], which is a visual syntax for graphical modeling and visualization of ontologies, used in conjunction with yEd [10]. Another artifact in this field is represented by GRAPHOL [11] - a fully graphical language inspired by UML, having the purpose of modeling OWL 2 ontologies. Moreover, Chowlk [12] presents another approach being positioned as an UML to OWL ontology converter. These artifacts are primarily focused on the ontological level of knowledge representation, offering no or little support for visually populating knowledge graphs.

Methodologically, the work presented here is the result of an initial iteration of the Design Science Research (DSR) cycle [13] (as an artifact-oriented research methodology) and of Agile Modeling Method Engineering (AMME) [14] adopted as prototyping methodology.

The remainder of the paper is structured as follows: Section 2 summarizes some motivating tool requirements. Section 3 presents design decisions and implementation details, while Section 4 provides more information on the evaluation approach. Section 5 concludes the paper, discussing several limitations of GRAPHxx.

2. Prospective users and tool requirements

By interacting with KG educators and students who must create and showcase KG examples we collected a series of usability requirements, which are the main focus of the new tool compared to other tools who are either only visualizers, or are closely aligned to the OWL formal concepts rather than the knowledge acquisition experience.

Usability requirements refer primarily to specific criteria and features which allow users to accomplish the main tasks when it comes to visually developing and editing KGs – our motivating requirements are summarized below:

- to have the ability to separate schema and data graph elements in separate diagrams;
- to provide a frictionless experience, suitable for less advanced users need to create knowledge graph fragments through a diagramming experience (mostly) agnostic from the RDF technological specificity;
- to intuitively handle patterns such as blank nodes, lists and prefixes;
- to integrate both tabular editing and visual editing of graph elements and relations, providing convenient switching between them, as well as domain-specific icon customization to achieve semantic transparency as advocated by [15].

The main user categories of the modeling tool are constituted by:

1. those who are used to the diagramming user experience of modeling tools or mind mapping tools but are otherwise *non-technical or with a limited RDF technological background* – that need to structure organizational knowledge in a machine-readable format;
2. *knowledge graph engineers* – who would benefit from a visual modeling tool that allows them to develop ontologies but also concrete knowledge graphs or graph fragments in a faster manner than writing RDF code by hand;

3. *educators* – who need means of presenting quickly drawn KGs to students and novices during IS/CS courses and seminars, sometimes long before delving into the more advanced RDF/OWL jargon (prefixes, containers, axioms etc.).

3. Design decisions and implementation details

In order to assess the feasibility of using the modeling tool for KG development and knowledge capture on a real world scenario, a use-case derived from a higher education institution was formulated. It follows the creation of a graph fragment alongside its ontological schema, being populated with instances pertaining to the organizational environment of the university where the authors of the paper perform their teaching and research activities. The institution in cause was subject to several digitalization strategies over the course of last years, exploring the adoption of knowledge graphs in order to store relevant information.

Concretely, the developed KG was populated with entities related to administrative units such as departments or faculties, roles involved in teaching and research activities, persons who accomplish such roles and several data nodes. Information about entities and data nodes was obtained from different organizational sources, mostly being tied to legacy databases. Alongside entities, relationships were added in order to show the dependencies between organizational units, roles and staff. Each entity, data node or relationship available in the knowledge graph has a series of attributes that were set in order to increase the level of information provided. The knowledge graph obtained in this manner offers semantic querying capabilities (see Section 4) which can prove to be an important means of decision support for management departments. The development of larger scale knowledge graphs containing data about the whole academic organizational context could prove to be of significant help for higher management purpose and traceability of value chains involved in teaching, research and academic development activities.

Agile Modeling Method Engineering (AMME) [14] is an established approach to engineering and evaluating modeling methods. A modeling method comprises a modeling language (typically a DSML), operating procedures and model-driven mechanisms/artifacts.

ADOxx [16] is the most used metamodeling platform of choice for AMME, due to its agile metamodel deployment and capabilities of scripting model-driven artifacts, which have been employed in projects of diverse domain-specificities [17] and is the core technological enabler of OMiLAB's digital innovation environment [18].

The ADOxx-based metamodel was tailored to incorporate the RDF specificity as the "domain" of a DSML, however avoiding as much as possible specific OWL jargon, offering a friendlier environment for novices in the semantic web technologies field.

GRAPHxxModellingNode, SchemaNode and ContentNode act as superclasses for the concepts available in their respective modeltypes. GRAPHxx achieves separation of concepts in 2 different model types: *Graph Schema* and *Graph Content.*

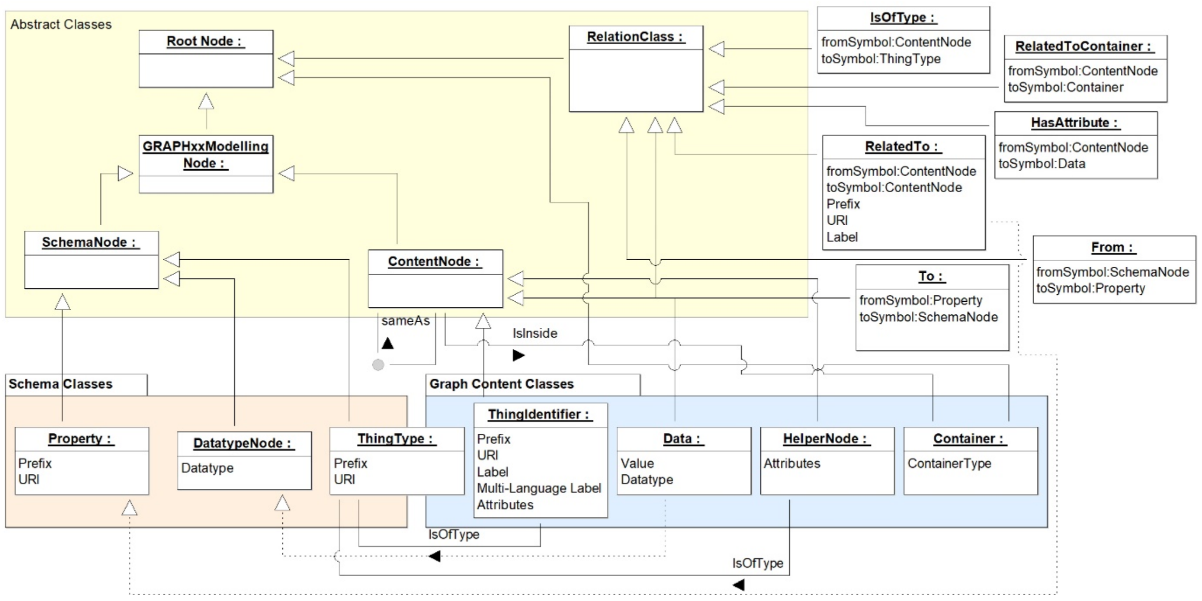


Figure 1: Proposed metamodel of GRAPHxx Modeling Tool

In *Graph Content* diagrams the primary concepts are *ThingIdentifier* – an individual that can act as a subject or object in RDF terminology, being of a *ThingType*; *Data* – for storing literal values, *HelperNode* – anonymous nodes in RDF terminology and *Container* – which can represent ordered or unordered lists by setting the *ContainerType* attribute. On the relation classes side there are *RelatedTo* – being a concrete representation of *Property* from *Graph Schema* diagrams, *IsOfType*, *HasAttribute* – for linking *ThingIdentifiers* (instances) to specific *Data* and *RelatedToContainer* – for linking things to Containers. The relation *sameAs* comes in the form of a hyperlink supported by ADOxx to navigate between graph fragments about the same thing (owl:sameAs).

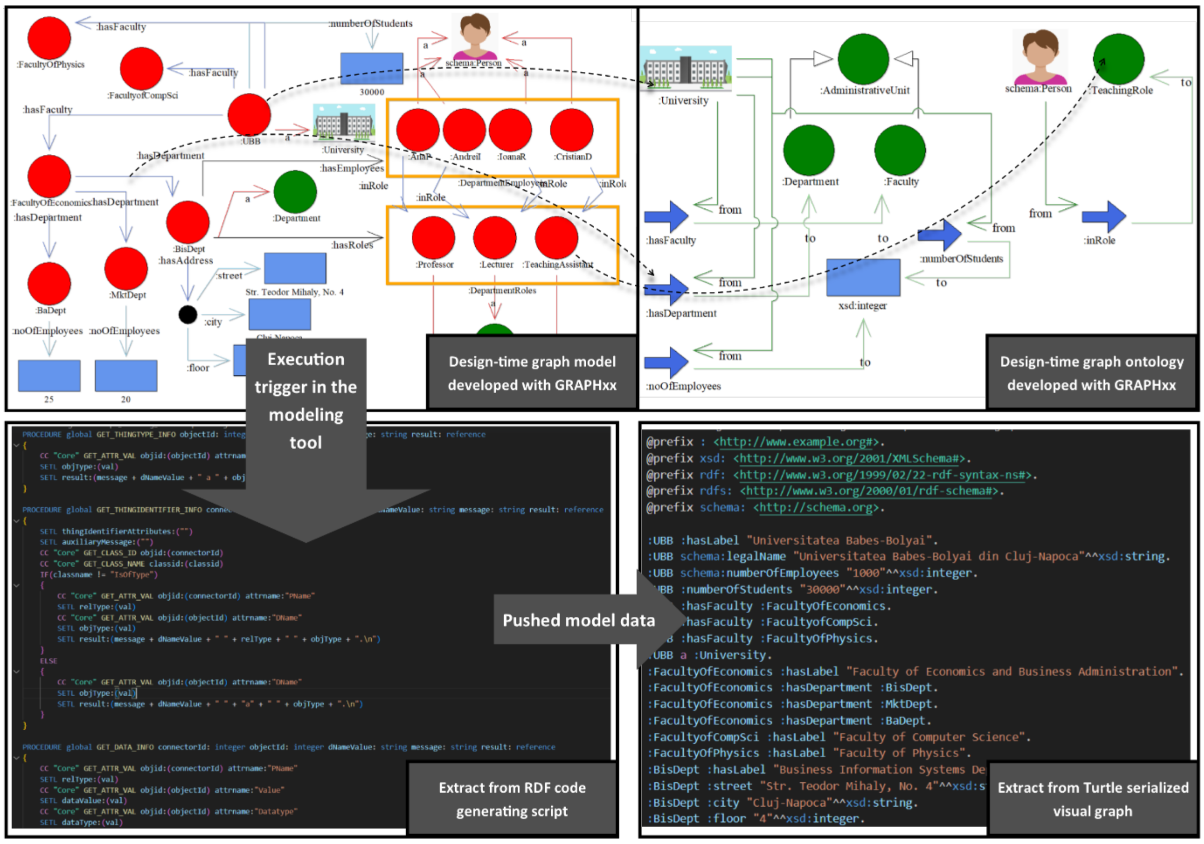
The *Graph Schema* diagrams are used for building ontologies and use constructs such as Property (predicates from RDF terminology), *DatatypeNode*, *ThingType* (classes that are instantiated in *Graph Content* diagrams).

On the relations side the most used are *IsOfType* (a non-OWL-specific representation for rdf:type), the *From* (trivialized from rdfs:domain) and the *To* (trivialized from rdfs:range).

There are two model level attributes for storing prefixes and for choosing whether prefixes should be listed in the diagram or obscured.

Two main functionalities - graph serialization as Turtle file and serialized graph loading to GraphDB server instance - were implemented with the help of AdoScript [19] and Python.

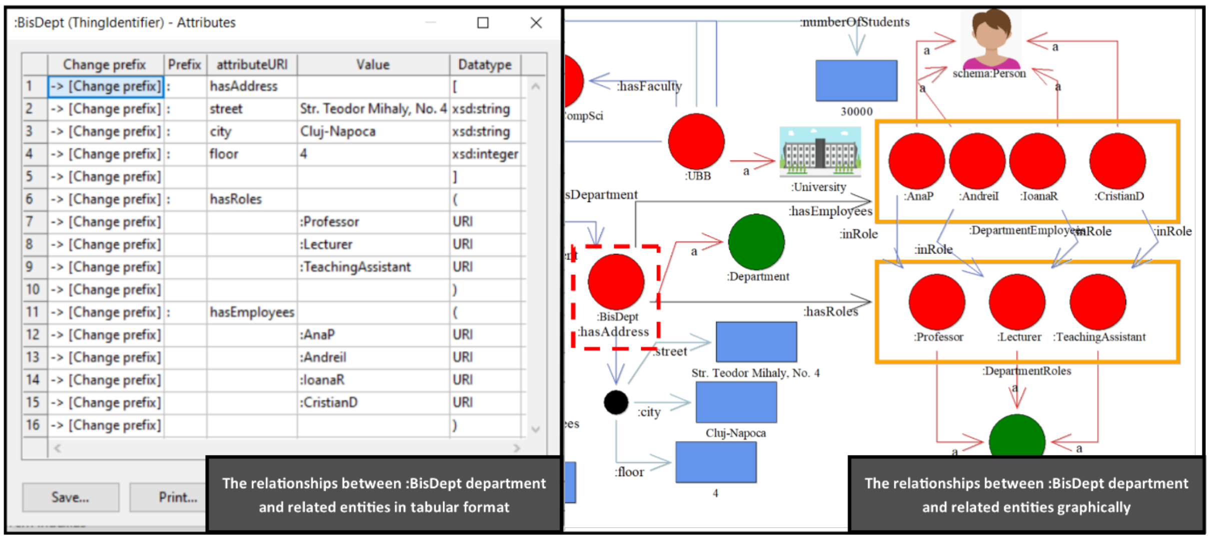
Figure 2 suggests this diagram-to-RDF code generation functionality. Both content and schema (pertaining to an RDF description of some university resources) are visible and written in a Turtle file - instances on the left, the RDF schema on the right and the option of adding isOfType relationships either as hyperlinks between the two or as visual arrows (if there is not risk of visual cluttering), or of switching between the two (e.g. first link ontology classes to instances, then visualize them in the instance-level diagram also). Other functionalities of the modeling tool include prefix changing over multiple diagram elements, URI generation from entity Labels and finally the upload to a dedicated GraphDB triple store, directly from the modeling environment.



```
PROCEDURE global GET_THINGTYPE_INFO objectId: integ                    e: string result: reference
{
    CC "Core" GET_ATTR_VAL objid:(objectId) attrnam
    SETL objType:(val)
    SETL result:(message + dNameValue + " a " + obj
}

PROCEDURE global GET_THINGIDENTIFIER_INFO conne                    nameValue: string message: string result: reference
{
    SETL thingIdentifierAttributes:("")
    SETL auxiliaryMessage:("")
    CC "Core" GET_CLASS_ID objid:(connectorId)
    CC "Core" GET_CLASS_NAME classid:(classid)
    IF(classname != "IsOfType")
    {
        CC "Core" GET_ATTR_VAL objid:(connectorId) attrname:"PName"
        SETL relType:(val)
        CC "Core" GET_ATTR_VAL objid:(objectId) attrname:"DName"
        SETL objType:(val)
        SETL result:(message + dNameValue + " " + relType + " " + objType + ".\n")
    }
    ELSE
    {
        CC "Core" GET_ATTR_VAL objid:(objectId) attrname:"DName"
        SETL objType:(val)
        SETL result:(message + dNameValue + " " + "a" + " " + objType + ".\n")
    }
}

PROCEDURE global GET_DATA_INFO connectorId: integer objectId: integer dNameValue: string message: string result: reference
{
    CC "Core" GET_ATTR_VAL objid:(connectorId) attrname:"PName"
    SETL relType:(val)
    CC "Core" GET_ATTR_VAL objid:(objectId) attrname:"Value"
    SETL dataValue:(val)
    CC "Core" GET_ATTR_VAL objid:(objectId) attrname:"Datatype"
    SETL dataType:(val)
```

```
@prefix : <http://www.example.org#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix schema: <http://schema.org>.

:UBB :hasLabel "Universitatea Babes-Bolyai".
:UBB schema:legalName "Universitatea Babes-Bolyai din Cluj-Napoca"^^xsd:string.
:UBB schema:numberOfEmployees "1000"^^xsd:integer.
UBB :numberOfStudents "30000"^^xsd:integer.
    :hasFaculty :FacultyOfEconomics.
    hasFaculty :FacultyofCompSci.
  B :hasFaculty :FacultyOfPhysics.
UBB a :University.
:FacultyOfEconomics :hasLabel "Faculty of Economics and Business Administration".
:FacultyOfEconomics :hasDepartment :BisDept.
:FacultyOfEconomics :hasDepartment :MktDept.
:FacultyOfEconomics :hasDepartment :BaDept.
:FacultyofCompSci :hasLabel "Faculty of Computer Science".
:FacultyOfPhysics :hasLabel "Faculty of Physics".
:BisDept :hasLabel "Business Information Systems De
:BisDept :street "Str. Teodor Mihaly, No. 4"^^xsd:s
:BisDept :city "Cluj-Napoca"^^xsd:string.
:BisDept :floor "4"^^xsd:integer.
```

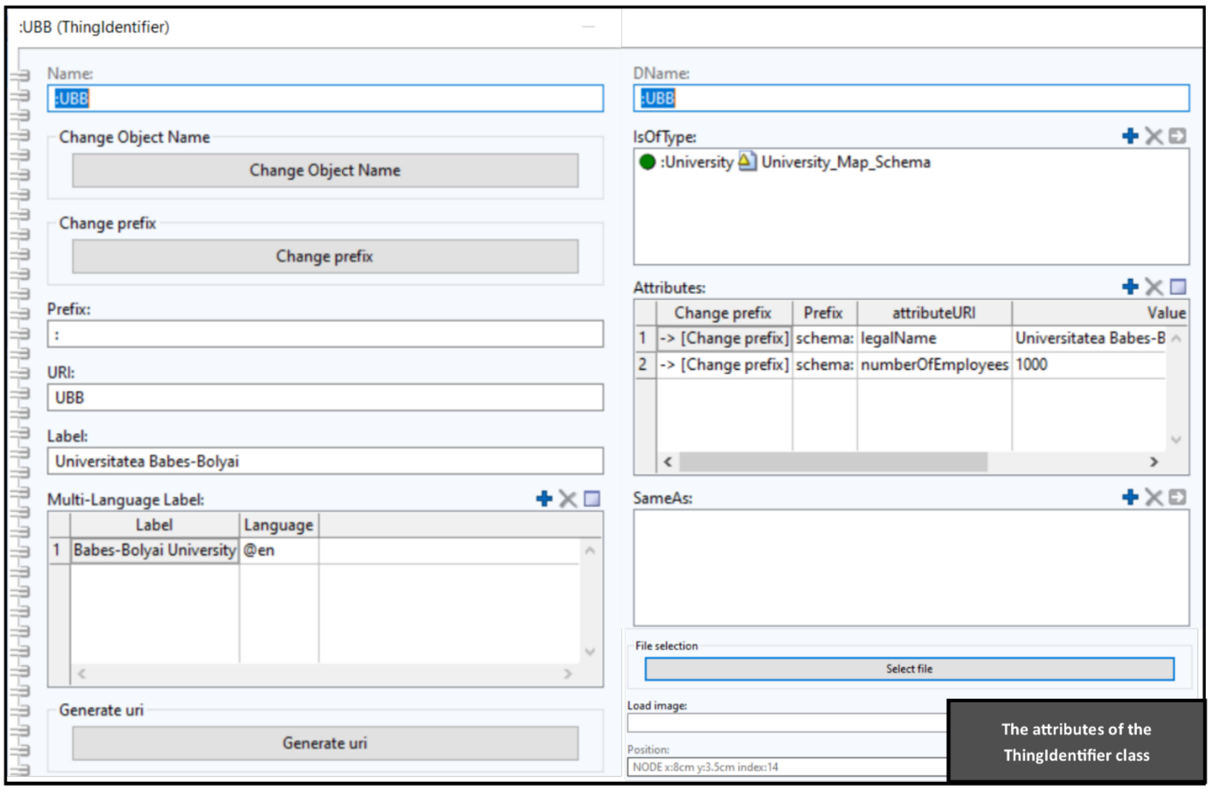
Figure 2: KG exemplar converted to Turtle via Ado-scripting


:BisDept (ThingIdentifier) - Attributes
Save...
Print...
The relationships between :BisDept department and related entities in tabular format
The relationships between :BisDept department and related entities graphically
:numberOfStudents
:hasFaculty
30000
schema:Person
:UBB
:University
:hasEmployees
:AnaP
:AndreiI
:IoanaR
:CristianD
:DepartmentEmployeeinRole
:inRole
:Department
:BisDept
:hasAddress
:hasRoles
:Professor
:Lecturer
:TeachingAssistant
:DepartmentRoles
:street
Str. Teodor Mihaly, No. 4
:city
Cluj-Napoca
:floor
4

	Change prefix	Prefix	attributeURI	Value	Datatype
1	-> [Change prefix]	:	hasAddress		[
2	-> [Change prefix]	:	street	Str. Teodor Mihaly, No. 4	xsd:string
3	-> [Change prefix]	:	city	Cluj-Napoca	xsd:string
4	-> [Change prefix]	:	floor	4	xsd:integer
5	-> [Change prefix]				]
6	-> [Change prefix]	:	hasRoles		(
7	-> [Change prefix]			:Professor	URI
8	-> [Change prefix]			:Lecturer	URI
9	-> [Change prefix]			:TeachingAssistant	URI
10	-> [Change prefix]				)
11	-> [Change prefix]	:	hasEmployees		(
12	-> [Change prefix]			:AnaP	URI
13	-> [Change prefix]			:AndreiI	URI
14	-> [Change prefix]			:IoanaR	URI
15	-> [Change prefix]			:CristianD	URI
16	-> [Change prefix]				)

Figure 3: Patterns of handling RDF predicates: tabular on the left and diagrammatic on the right

Figure 3 showcases two patterns of handling RDF properties. The modeling tool offers more control to end users regarding how much visual knowledge has to be incorporated in a certain diagram, the tabular format being a viable alternative in cases where visual cluttering becomes an issue. They can be edited separately and converted between tabular and visual representations, just like the type links mentioned before - actually this script-driven switching between different user perspectives on the graph content is one key motivator behind GraphXX.


:UBB (ThingIdentifier)

Name: :UBB

Change Object Name

Change prefix

Prefix: :

URI: UBB

Label: Universitatea Babes-Bolyai

Multi-Language Label:

	Label	Language
1	Babes-Bolyai University	@en

Generate uri

DName: :UBB

IsOfType: ● :University University_Map_Schema

Attributes:

	Change prefix	Prefix	attributeURI	Value
1	-> [Change prefix]	schema:	legalName	Universitatea Babes-B
2	-> [Change prefix]	schema:	numberOfEmployees	1000

SameAs:

File selection: Select file

Load image:

Position: NODE x:8cm y:3.5cm index:14

Figure 4: Modeling attributes of a Thing

Figure 4 shows the way in which attributes can be set for ThingIdentifiers in the modeling tool. The same idea applies for all the other modeling concepts that have attributes which can be modified by users. In the case of ThingIdentifiers, the combination between File selection and Load image attributes allows users to load custom graphical representations at design time, bringing the KG content closer to a mind mapping or "boxes-and-arrows" diagramming experience, achieving semantic transparency [15], thus making diagrams friendlier for novices and teaching purposes.

4. Evaluation approach

The initial Design Science iteration is still on-going, with only early evaluation steps and further planning available at the time of this reporting. According to the Design Science artifact evaluation criteria taxonomy [20] the following criteria are targeted:

A. *Evolution ("learning capability")* which translates to gradually capturing richer conceptualization following feedback received from users, thanks to the AMME methodology. AMME's iterative nature and the usage of ADOxx fast-prototyping environment allows this. Currently, the state of the presented modeling tool shows progress made after the first iteration - initial feedback includes the requirement to raise the abstraction level above RDF to graph databases in general;

B. *Consistency with technology ("harnessing existing technology")* translates to the requirement of interoperability, already accomplished but open to providing more diverse export formats;

C. *Completeness* achieved through responding to a set of semantic traceability requirements, by harnessing AQL [21], the ADOxx embedded metamodel-driven query engine. It allows parsing intermodel hyperlinks to extract information related to a multi-diagram graph structure, chains of relationships within a diagram, chains of links between different diagrams, tabular properties, e.g.:

… retrieve the class at ontology level for the Professor element in the Business Information Systems Department container:

(<"ThingIdentifier">[?"Label" = "Business Information Systems Department"]->"RelatedToList"[?"Name" = ":DepartmentRoles"])<-"Is inside"[?"Label" = "Professor"]-->"IsOfType";

… retrieve all the names of the employees of the Business Information Systems Department:

(<"ThingIdentifier">[?"Label" = "Business Information Systems Department"]->"RelatedToList"[?"Name" = ":DepartmentEmployees"])<-"Is inside"[?"Label" != ""]

These represent two examples of university top-management questions which could be answered by running AQL queries, thus achieving Semantic traceability through diagrammatic hyperlinks.

The article of Siau and Rossi [22] presents several different approaches when it comes to evaluating modeling tools. In relation to our developed artifact the research plan includes:

- *Feature comparison* with other commercial tools for developing knowledge graphs, like TopBraid Composer [23] or PoolParty [24]. Typically, most tools are aimed for OWL experts or are limited to the TBOX level, offering low support for OWL-agnostic users who are however capable of structuring a description in a "box-and-arrow" sketching approach;
- *Metrics*, which relies on running a series of experiments for determining times needed for graph and ontology development, measuring the performance of GRAPHxx in serializing small to medium developed graphs or graph fragments as Turtle compliant files against the classical approach of writing RDF code by hand in Turtle format, as seen in Table 1;
- *Survey*, performed by running a series of modeling experiments with a group of students, subsequently responding to a series of questionnaires about the modeling experience with GRAPHxx.

Table 1
Comparison of serialization times

No. of triples	Graph design and serialization times using GRAPHxx (s)	Writing equivalent Turtle by hand (s)
10	155.45	178.54
20	385.89	438.37
30	464.14	530.25

5. Limitations and outlook

In its current state, GRAPHxx is limited to Turtle serialization and is tied to building knowledge graphs following the RDF standard, which remains to be developed to also cover labelled property graphs from non-RDF servers [25]. Other serialization formats will be explored as part of future iterations in the modeling tool development cycle, as well as the potential brought by bridging Large Language Models (LLMs) with the modeling tool in an effort of providing automated graph description or querying in natural language directly from the modeling tool.

In a longer term, we aim to integrate this with enterprise modeling languages where a notion of data or knowledge objects is already available and could be drilled down to detailed linked data assets in KG-driven enterprise architectures or business processes.

Acknowledgements

The present work has received financial support through the project: Integrated system for automating business processes using artificial intelligence, POC/163/1/3/121075 - a Project cofinanced by the European Regional Development Fund (ERDF) through the Competitiveness Operational Programme 2014-2020.

References

[1] W3C, The Resource Description Framework (RDF). URL: https://www.w3.org/RDF/. Last accessed on 25.09.2023.

[2] L. V. Herm, C. Janiesch, A. Helm, F. Imgrund, A. Hofmann, A. Winkelmann, A Framework for Implementing Robotic Process Automation Projects, Information Systems and e-Business Management, Springer, 21 (2022): 1-35. doi:10.1007/s10257-022-00553-8.

[3] A. Voelz, C. Muck, D. M. Amlashi, D. Karagiannis, Citizen-centric Design of Consumable Services for Smart Cities, Digital Government: Research and Practice, Association for Computing Machinery, 4.3 (2023): 1 – 18. doi:10.1145/3597420.

[4] W3C, The Web Ontology Language (OWL) Specification. URL: https://www.w3.org/OWL/. Last accessed on 24.09.2023.

[5] Ontotext, The GraphDB Semantic Graph Database. URL: https://graphdb.ontotext.com/. Last accessed on 26.09.2023.

[6] Data.world, The Gra.fo Knowledge Graph Designer. URL: https://gra.fo/. Last accessed on 25.09.2023.

[7] IMCS University of Latvia, The OWLGrEd Ontology Editor. URL: http://owlgred.lumii.lv/. Last accessed on 30.09.2023.

[8] Stanford University, The PROTÉGÉ Ontology Editor. URL: https://protege.stanford.edu/. Last accessed on 30.09.2023.

[9] R. Falco, A. Gangemi, S. Peroni, D. Shotton, F. Vitali, Modelling OWL Ontologies with Graffoo, in: The Semantic Web: ESWC 2014 Satellite Events, ESWC 2014, Lecture Notes in Computer Science, Springer, Cham, 2014, 8798. doi:10.1007/978-3-319-11955-7_42.

[10] yWorks GmbH, The yEd graph editor. URL: https://www.yworks.com/products/yed#. Last accessed on 30.09.2023.

[11] D. Lembo, V. Santarelli, D. F. Savo, G. De Giacomo, Graphol: A Graphical Language for Ontology Modeling Equivalent to OWL 2, Future Internet, 14.3 (2022): 78. doi:10.3390/fi14030078.

[12] S. Chávez-Feria, R. García-Castro, M. Poveda-Villalón, Chowlk: from UML-Based Ontology Conceptualizations to OWL, in: The Semantic Web: ESWC 2022, Lecture Notes in Computer Science, Springer, Cham, 2022, 13261. doi:10.1007/978-3-031-06981-9_20.

[13] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-662-43839-8.

[14] D. Karagiannis, Agile Modeling Method Engineering, in: Proceedings of the 19th Panhellenic Conference on Informatics, Association for Computing Machinery, 2015, pp. 5–10. doi:10.1145/2801948.2802040.

[15] D. Moody, The "Physics" of Notations: Towards a Scientific Basis for Constructing Visual Notations in Software Engineering, IEEE Transactions on Software Engineering, 35.5 (2009): 756–779. doi:10.1109/TSE.2009.67.

[16] BOC GmbH, The ADOxx Metamodeling Plaform. URL: https://www.adoxx.org/live/home. Last accessed on 01.10.2023.

[17] D. Karagiannis, M. Lee, K. Hinkelmann, W. Utz (Eds.), Domain-Specific Conceptual Modeling. Springer, Cham, 2022. doi:10.1007/978-3-030-93547-4.

[18] D. Karagiannis, R. A. Buchmann, X. Boucher, S. Cavalieri, A. Florea, D. Kiritsis, M. Lee, OMiLAB: A Smart Innovation Environment for Digital Engineers, in: Proceedings of PRO-VE 2020, Springer, 2020, pp. 273-282. doi:10.1007/978-3-030-62412-5_23.

[19] BOC GmbH, The AdoScript Scripting Language. URL: https://www.adoxx.org/live/adoscript-documentation. Last accessed on 28.09.2023.

[20] N. Prat, I. Comyn-Wattiau, J. Akoka, Artifact Evaluation in Information Systems Design-Science Research – A Holistic View, in: Proceedings of PACIS 2014, AIS, 2014, 23. URL: https://aisel.aisnet.org/pacis2014/23.

[21] BOC GmbH, The AQL Query Language. URL: https://www.adoxx.org/live/adoxx-query-language-aql. Last accessed on 30.09.2023.

[22] K. Siau, M. Rossi, Evaluation of Information Modeling Methods - A Review, in: Proceedings of the HICSS 1998, IEEE, 1998, 5, pp. 314-322. doi:10.1109/HICSS.1998.648327.

[23] TopQuadrant, The TopBraid EDG. URL: https://www.topquadrant.com/taxonomy-and-ontology-management. Last accessed on 26.09.2023.

[24] Semantic Web Company, The PoolParty Semantic Suite. URL: https://www.poolparty.biz/. Last accessed on 27.09.2023.

[25] R. Angles, The Property Graph Database Model, in: Alberto Mendelzon Workshop on Foundations of Data Management, CEUR-WS, 2100, 26, 2018. URL: https://ceur-ws.org/Vol-2100/paper26.pdf.