Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 60 (2015) 150 – 159

Procedia Computer Science

19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

Pattern-based transformation of diagrammatic conceptual models for semantic enrichment in the Web of Data

Robert Andrei Buchmann[a] *, Dimitris Karagiannis [b] †

[a]*Business Information Systems Dpt., Faculty of Economic Sciences and Business Administration, University Babeş Bolyai, str. T. Mihali 58-60, 400591, Cluj Napoca, Romania*

[b]*Knowledge Engineering Research Group, Faculty of Computer Science, University of Vienna, Waehringerstr. 29, 1090, Vienna, Austria*

Abstract

The goal of this paper is to bridge the paradigms of Linked Data and Conceptual Modeling, which have been developed from quite distinct concerns, although certain opportunities may stimulate the evolution of the Web towards a new type of knowledge space driven by diagrammatic models. To this end, the work at hand investigates structural patterns in a multitude of modeling languages accumulated over time within the Open Model Initiative Laboratory and defines for each pattern transformation rules that produce graph-based model serializations, thus enabling the processing of diagrammatic models using the Web of Data technological space and practices.

Peer-review under responsibility of KES International

Keywords: Web of Data, Conceptual Modeling, Linked Data, Graph databases

1. Introduction

The work at hand proposes a convergence between the paradigms of Linked Data[1] and Conceptual Modeling, by exploiting the underlying graph nature of diagrammatic conceptual models in order to expose such models to browsing, processing and retrieval across the so-called "Web of Data". This graph nature is typically enabled as a meta-

* Corresponding author. Tel.: +40.264 41 86 52; fax: +40.264 41 25 70.
E-mail address: robert.buchmann@econ.ubbcluj.ro
† Corresponding author. Tel.: +43-1-4277-7891; fax: +43-1-4277-878910.
E-mail address: dk@dke.univie.ac.at

Peer-review under responsibility of KES International
doi:10.1016/j.procs.2015.08.114

structure by meta-modeling platforms, although such platforms rely traditionally on relational databases for storage of the model information. This implies the presence of a multitude of normalized tables or NULL values (due to the irregular use of properties in model elements) and insufficient openness for the information described in models. Alternatively, graph databases allow a more straightforward abstraction and internal representation. The Linked Data paradigm adds to this standards and best practices for publishing, remote querying and federation across the Web.

The concrete result of our effort comprises: (a) a set of recurring structural patterns identified in a corpus of modeling languages accumulated over time in the Open Model Initiative Laboratory[2]; (b) a mechanism for converting such modeling patterns to Linked Open Data (thus enabling the management of models with tools from the technological space underlying the Web of Data). The result was validated with a proof-of-concept implementation successfully employed in the ComVantage FP7 EU project[3] to make model information available to run-time systems (mobile apps)[4].

The overall envisioned benefit is that *run-time information* (e.g. legacy execution data lifted in the Web of Data via existing adapters – e.g. D2RQ[5]) can be semantically linked and enriched with *design-time information* (knowledge externalized in various types of diagrams), thus enabling richer constraints and federation possibilities even in the absence of ontologies. This vision also raises arguments in the favor of non-standard modeling methods[8], which have been largely shadowed by the popularity of standards such as UML[6] or BPMN[7]. Modeling methods may be tailored to expose, within the Web of Data, richer semantics driven by requirements coming either (a) directly from modeling stakeholders, or (b) indirectly via run-time applications that must consume model information.

The remainder of the paper is organized as follows: Section 2 sets the motivation, positions the work and discusses some related works. Section 3 catalogues recurring patterns and pattern variants identified in diagrammatic modeling languages and recommends for each of them a Linked Data structure. Section 4 showcases the proof-of-concept implementation and its key features. The paper ends with a conclusive SWOT analysis, to highlight some limitations as further work concerns.

2. Related work

The goal of semantic enrichment has been addressed[9,10] largely through traditional semantic lifting/matching techniques, often involving the setup of metadata or alignment ontologies[11] or statistical knowledge discovery. While focused on the processing of text and tagged content, the literature neglects the availability of semantics in visual form, namely in diagrammatic models (mostly considered to serve human communication or code generation tasks in software engineering). There are cases where semantic enrichment of models is addressed[12] as annotation of models, while we employ model semantics to extend run-time data. We also aim to stimulate an interdisciplinary convergence between conceptual modeling and the Semantic Web. Traditionally this convergence manifested at design-time, by checking the ontological consistency of language grammars[13]. The work at hand is rather run-time focused, by having conceptual models providing complementary semantics to run-time vocabularies/ontologies.

Model-driven development has been largely subordinated by software engineering communities to the goal of code generation, which established the dictum "modeling is programming"[14]. The work at hand highlights the fact that diagrammatic models can also serve, even at run-time, other purposes than what is advocated by the "modeling as programming" vision – a wide range of demonstrations and roadmaps have been discussed in the Models@runtime seminars and publications[15]. On one hand, we have the "modeling is configuration" approach, where models act as "control panels" for some run-time functionality, via some XML configuration serialization (e.g. BPEL[16]). On the other hand, the "modeling as knowledge representation" approach treats models as knowledge structures (typically conceptual graphs) governed by a terminological box which can be derived from the language metamodel. This vision is also inspired by the fact that business process modeling has been traditionally seen as means of knowledge externalization in knowledge management and knowledge-based systems[17].

Relative to the research challenges raised in the Models@runtime seminars with respect to business process management[18], the work at hand provides a semantics-oriented instantiation for the aspects of "causal connections" (by enabling semantic linking between run-time data and models) and "reasoning" (by enabling the use of query-time graph transformations as production rules).

3. Research challenges, motivation and enablers

In the context of the Models@runtime paradigm, we approach several specific challenges: *(a) how can the knowledge that is expressed in diagrammatic form be exposed to run-time systems in a way that preserves diagram semantics; (b) how can the diagrammatic model information be linked within the Web of Data, so that it can benefit from standard means for querying, federation and rule-based reasoning?*

In the Web of Data technological space, graph-based data models (RDF[19]) improve on the hierarchical (XML-based) and relational models, as they provide a structure that is more flexible, easier to query and federate, and therefore more naturally fit to the networked nature of the Web itself. Fig. 1 motivates the hereby proposed convergence within a frame adapted from the Meta-Object Facility framework[20] . On the left side, a fragment of a business process model (a mock-up of the ADONIS[21] notation) is isolated as a "statement" expressed in a logographic language (the business process modeling language). On the right side, a simple Linked Data record is expressed as a statement written in Turtle[22] (one of the numerous graph serialization syntaxes for Linked Data graphs). Both the diagram fragment and the Turtle statement relate to a meta level of abstraction where the terminology/vocabulary of the language is defined. These in turn are instances of the higher level meta-metamodel. Therefore **metamodeling** is a key enabler for the work at hand.

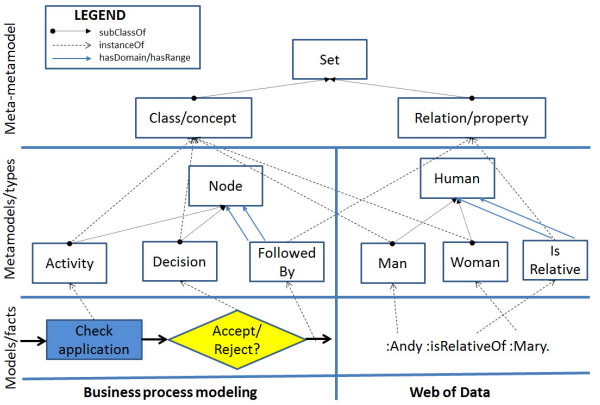

Fig. 1. A convergence between diagrammatic modeling and Web of Data across abstraction layers

The metamodeling paradigm relies on the existence of meta-metamodels that establish the primitive constructs necessary to describe language terminologies (metamodels). Various technologies (ADOxx[23], MetaEdit+[24] etc.) work with different meta-metamodels (ADOxx, GOPPRR[25], ECore[26] etc.) however, as the analysis of Kern et al.[27] shows, they are reducible to the MOF top abstraction. The meta-metamodel underlying the Linked Data paradigm takes a similar approach of establishing some key primitives – *resource*, *property* - and some useful top-level specialization (individual resources and class resources, data properties and object properties). A decomposition concept is available in both worlds – the *model type* (diagram type) is a partition of the language metamodel; the *named graph* is a partition of a Linked Data graph and can be itself described as an abstract resource.

In the next section graph data structures are designated for a catalogue of patterns recurring in popular modeling languages. These are the basis for the diagram conversion proof-of-concept (in figures full URIs will be omitted for readability).

4. Design decisions

Based on an investigation of more than a dozen modeling languages (with modeling tools available within the Open Model Initiative Laboratory[2]) including standards (UML, BPMN), commercial ones (EPC, ADONIS BPMS) and domain-specific ones (ComVantage[28]), an extensive set of modeling patterns have been collected and mapped to Linked Data structures. A mechanism for model-to-Linked Data transformation was implemented[28] based on such

mappings, treating each of these patterns as a transformation rule – the input is the internal representation of the pattern in the meta-modeling platform (ADOxx, in our implementation) and the output is a platform-independent RDF serialization (in the TriG syntax[29]) of the graph corresponding to each pattern (to be depicted in the figures throughout this section). The implementation also includes a direct upload mechanism via the Sesame HTTP protocol[30], to streamline the transfer of models between the modeling environment and a model repository.

The primary requirement for Linked Data is that most data points (except for blank nodes and literals) take the form of universal resource identifiers (URIs), as well as the possibility to refer / link to any identified entities through RDF statements. This implies that all relevant elements of a diagram must have such URIs when exported in the Web. This can be accomplished by concatenating a namespace (specified by the modeler) with the internal identifiers of model elements. However, additional customization must also allow a modeler to link the model information to external resources (e.g. existing Linked Data, models that have been previously exported). This aspect will be tackled in Section 3.5.

With respect to the transformation granularity, patterns must be devised to generate relevant graph data structures from the information present in diagrams and their associated property sets (e.g. annotations). Each diagram becomes a named graph isolating the information contained within the diagram, together with possible hyperlinks to other related diagrams, as illustrated by the minimal example in Fig. 2.

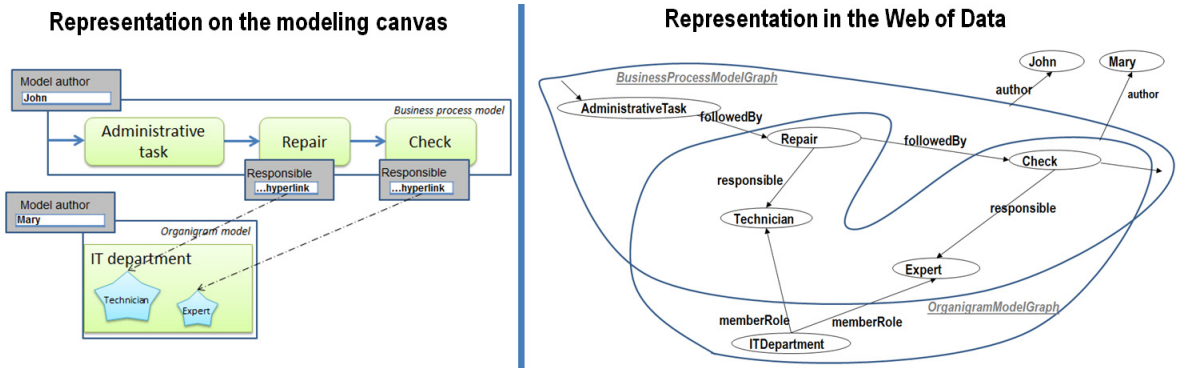

Fig. 2. Generated graph structure for inter-linked diagrams

4.1. Design decisions for modeling relations

A multitude of visual manifestations must be considered for the notion of "modeling relation". For a 2D drawing canvas, this multitude may be reduced to several recurring patterns shown in Fig. 3.

Pattern1. Visual connectors with attributes. Each occurrence of such a connector must be distinguished through its properties (e.g. the condition of a sequence arrow), hence an n-ary relation must be introduced.

Pattern2. Visual connectors without attributes may be mapped directly to RDF predicates.

Pattern3. Relations implied by positions are relations that are visually communicated by their relative position. In the simplest case it can be a containment relationship (e.g. activities within the swimlane of a business process model).

Pattern4. Relations implied by functionality. The typical functionality that communicates the presence of some relation are hyperlinks for cross-model navigation. They can be straightforwardly mapped as in Fig. 3, or if they are characterized by some properties they can become n-ary relations (to be discussed in Section 3.3).

4.2. Design decisions for complex properties

Complex properties are typically present in the form of tables assigned to various model elements. The way tables should be converted to data graphs is dependent on the meaning intended by the metamodel designer, who can communicate this to the conversion mechanism by applying certain naming conventions (a non-ambiguous automat-

ed detection of the metamodeler's intended interpretation of tables cannot be guaranteed). Therefore a vocabulary of naming conventions (highlighted with an asterisk in the figures) must be introduced to distinguish between the patterns further discussed. In all examples we assume to have a modeling object representing a mobile device (PhoneX) - an element of some model used to collect app requirements by modeling means (for extensive examples of such models see the domain-specific modeling method ComVantage[28], where parts of the language are dedicated to such a scenario).

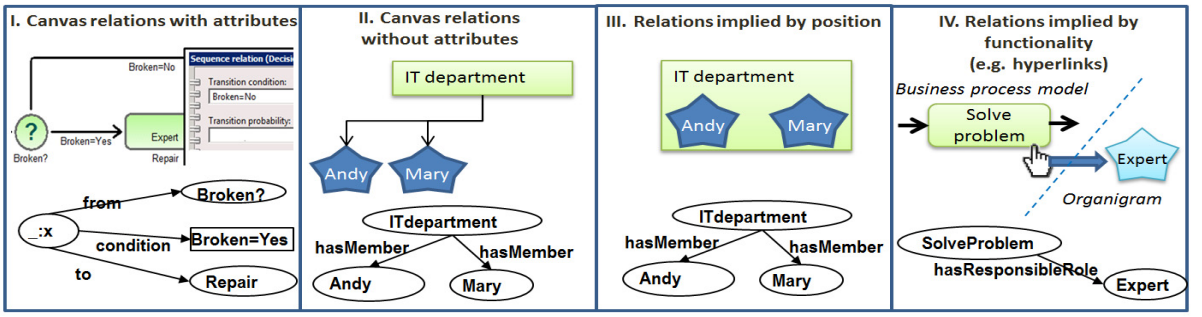

Fig. 3. Modeling relation patterns and their corresponding graph data structures

Pattern1. The statement collector. The tabular attribute is used to collect directly Linked Data statements, preferably properties of the current modeling element (PhoneX). Variations of this pattern (Fig. 4) may also allow to collect statements where the current element is a statement object, or even statements unrelated to the modeling element. Another proposed variation includes hyperlinks to elements of some other model (here, an organigram that includes the roles who require the mobile device support).

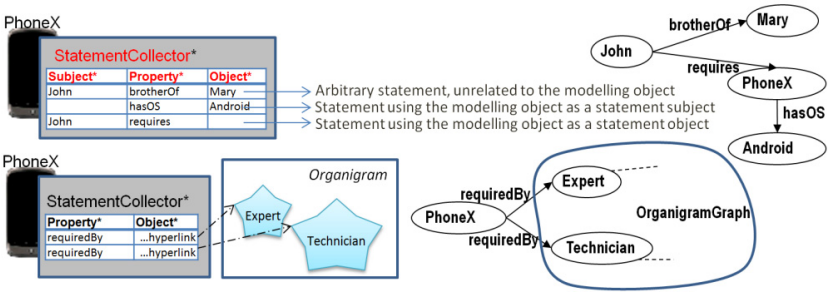

Fig. 4. The statement collector pattern

Pattern2. The non-canvas object. In the spirit of the entity-relationship model, the presence of a primary key (ID) in the table should indicate that each row of the table can be interpreted as an instance object, while the rest of the fields are its properties. It can be interpreted that such a table contains modeling objects that do not have a visualization (notation) on the canvas (in Fig. 5 each record represents an app that can be run on PhoneX). If the ID field name is omitted, objects should be considered anonymous, thus generating blank nodes. If the *_inv* suffix is present in the tabular attribute name, the relation between the table records and the selected element (PhoneX) should be inverted.

Pattern3. The n-ary relation. If the table contains hyperlinks to elements in other models, the structure may be interpreted as an n-ary relation (Fig. 6) whose participants are (a) the current modeling element, (b) the modeling elements targeted by the hyperlinks and (c) any other property values present in the table. The inversion of the RDF predicate direction may also be applied here just like in the previous case, for individual fields or for the table as a whole. If an ID field is present, then the pattern converges with Pattern 2, since any n-ary relation may be considered a non-canvas object that is described by its multiple binary relations to elements from other models.

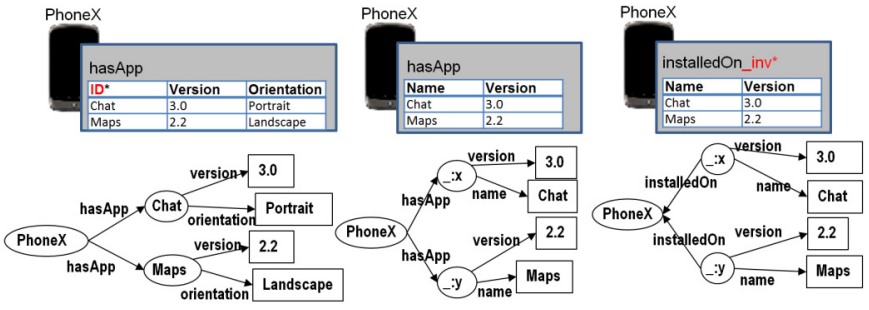

Fig. 5. The non-canvas object pattern

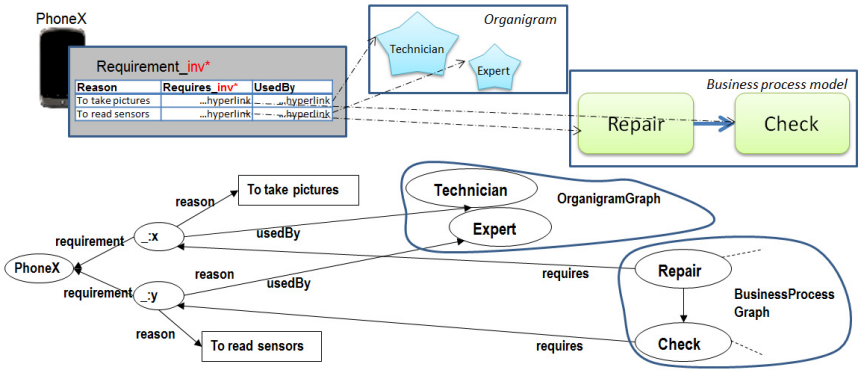

Fig. 6. The n-ary relation pattern

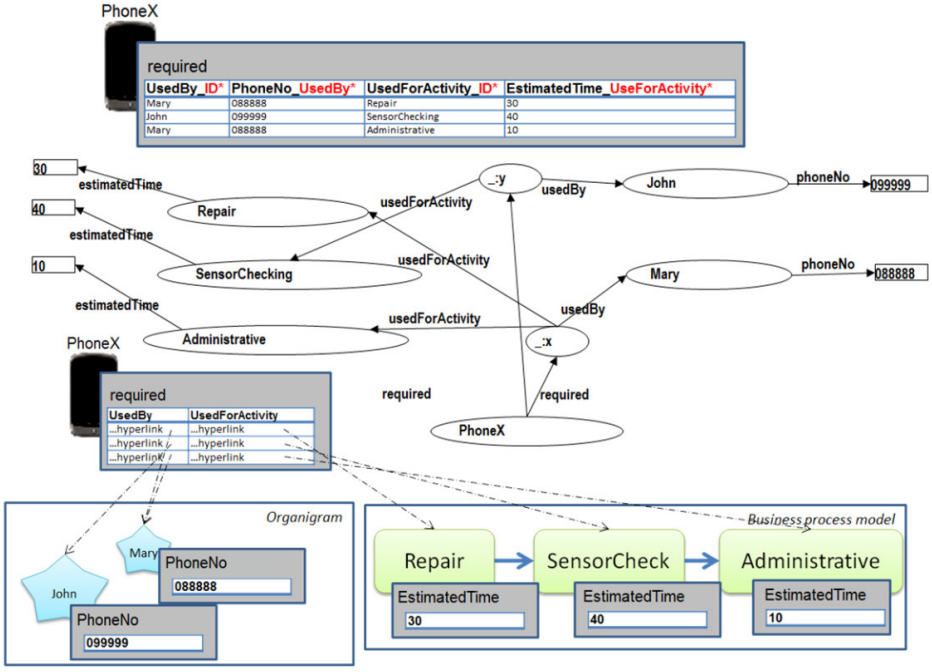

Fig. 7. Defusing the normalization anti-pattern

Pattern4. The normalization (anti)-pattern. In this case the table includes significant complexity in the sense of functional dependencies that should be defused through a database normalization approach. Naming conventions are necessary to indicate which are the entity IDs, and which are their specific attributes. The pattern should be avoided since typically it is a symptom of bad modeling language design – a modeler should generally work on the drawing canvas more than on filling complex data tables (property sets are typically employed in diagrams as annotations or to store data required for some modeling functionality - e.g. simulation, aggregation). The normalization anti-pattern illustrated in Fig. 7 should be defused and reduced to Pattern 3 by redesigning the language in the sense of introducing two new model types to isolate some properties and avoid redundancy (the bottom side of the figure).

Pattern5. Ordered variants. All the patterns discussed here may have ordered variants, which can be indicated by the presence of an Index property. A typical treatment for this, relative to Pattern2, is shown in Fig. 8. However, other approaches to ordering may be taken – the RDF vocabulary supports the use of sequence containers (rdf:Seq) or list-like collections (rdf:List), each of them having an impact on the complexity of read and update queries.

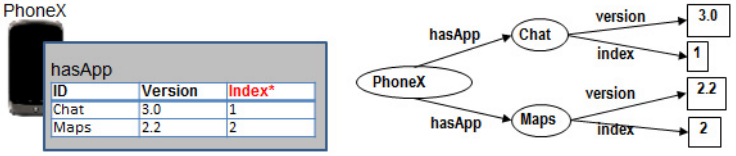

Fig. 8. Ordered variant of the non-canvas object pattern

4.3. Design decisions for types

A straightforward typing schema for the model elements should be derived from the metamodel that governs the modeling language. Meta-modeling platforms typically enable the management and storage of the metamodel separately from the model information. The transformation mechanism described here must access the metamodel information to generate classes and properties, thus producing a vocabulary (terminology) to anchor future queries. On the other side, some taxonomical refinement should also be enabled at model level. This relies on a dedicated property where the type may be specified (as Fig. 9 illustrates) by allowing the modelers to design their own taxonomies and freely link instances to them. Multiple models can be linked to a reusable vocabulary-model, both inside the modeling tool and outside (across the Web of Data). The taxonomy diagram should be assimilated as a specialization of the metamodel-level concept (here, "Activity" from some custom-notation "activity diagram").

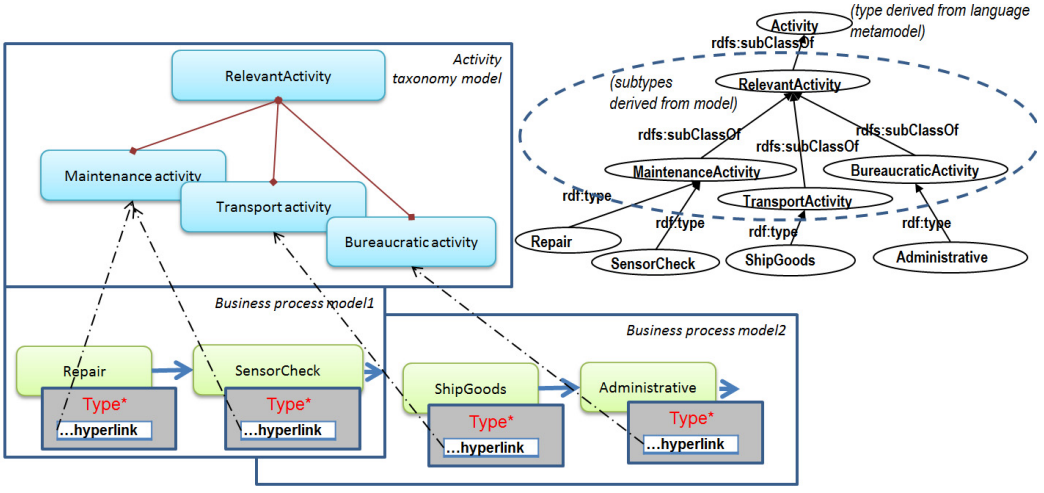

Fig. 9. Merging customizable model-level types with metamodel's prescribed types

4.4. Linking diagrammatic models to the Web of Data

A linking mechanism must be deployed to allow modelers to setup links to existing data or existing models that have been already converted. For **modeling objects** this can be enabled in the metamodel by prescribing a "preferred identifier" whose value will override the URI typically generated by the conversion mechanism. Also, any string properties whose values comply to the URI schema should be interpreted as external resources linked to the current modeling element. Automation may improve the productivity of linking. Specifically, modeling elements can be generated with the preferred identifier prefilled from some external source (a file, the result of a query against a Linked Data server, an import of an existing serialized model skeleton). Alternatively, linking may be performed outside the modeling tool, with ontology alignment techniques (*owl:sameAs* statements to indicate equivalence between modeling elements and existing resources, maintained manually or generated with some similarity detection tool, e.g. Silk[31]).

For **classes, properties, hyperlinks, modeling relations** having a preferred identifier at model level is prone to inconsistencies (multiple occurrence of a connector should get the same identifier), therefore some automated means must ensure consistent assignment of preferred identifiers. For this purpose we recommend that the model-to-data conversion tool allows the execution of transformation rules (based on SPARQL CONSTRUCT or DELETE/INSERT queries[32]) to include the preferred identifiers in the exported graph. Of course, this can also be accomplished by the metamodeler, but the power to control such vocabulary-level links should be passed to the modeler.

5. Validation by implementation: a proof-of-concept

Fig. 10 showcases the user interface of a proof-of-concept implementation for the ADOxx meta-modeling platform, based on the hereby discussed patterns. The tool has been implemented as a prototype for the ComVantage FP7 EU project, where model information was exposed in a Web of Data to support richer queries and model-based reasoning on the side of run-time components. The tool UI is organized in three tabs: (from left-to-right in Fig. 10): (a) transformation of the metamodel; (b) transformation of models; (c) remote model query client (against a Linked Data endpoint). The numbered items in the figure correspond to the elements and features enumerated as follows:

1. The Open/Save buttons provide, by means of file format selection, the key functionality: transformation from the internal format of the meta-modeling platform (ADOxx for this prototype) to the targeted named graph serialization or vice versa (if an import to the modeling tool is necessary);
2. The metamodel visualization area provides means for consulting the RDF triples that are generated for the metamodel information, as well as means for filtering the information (e.g. if the modeler does not want to export all the properties of the model elements);
3. For each metamodel element, some key properties (e.g. type, superclass, label) may be quickly consulted in this area;
4. A selected property may be designated to override the generated URI of model elements for the purpose of linking to external resources (hence equivalent to owl:sameAs); another property may be designated to be equivalent to rdf:type (hence allowing the modeler to attach types at model level);
5. A Sesame triplestore may be selected for RESTful upload or download of the model information (the REST URL and parameters are configured in window highlighted by item 13);
6. Two buttons trigger the upload or download through Sesame's REST protocol;
7. Local queries may be run to perform transformations on the metamodel graph via query-time production rules (CONSTRUCT) whose results are merged or substracted from the current state of the graph;
8. A button triggers the local query execution;
9. The open/save buttons are also present in the Model Management area, with the same role of performing the transformation, but this time strictly for the model information;
10. A navigation view is also available for the model elements, where their generated URIs may be consulted;
11. Some key properties may be consulted for each model element;
12. The upload-download functionality to a Sesame server is also available for the model graphs;

13. A dedicated window provides means of configuring the connection to the Sesame server (URI components and credentials necessary in the REST calls);
14. Remote queries against the currently selected Sesame server may be tested;
15. This area displays query results and performance for the tested remote query.

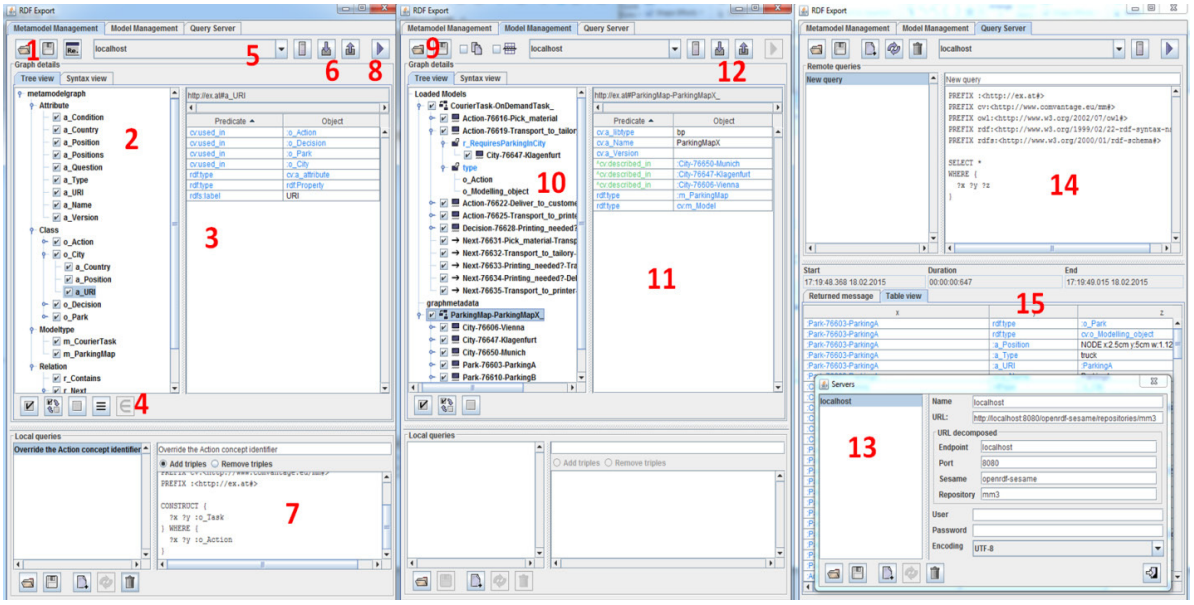

Fig. 10. Screenshots of the model-to-Linked Data converter prototype

6. SWOT analysis and conclusions

6.1. SWOT analysis

The limitations and achievements of this work are highlighted by the following SWOT analysis: **Strengths:** Conceptual models developed on domain-specific modeling methods provide complementary execution-time information. Models can be edited independently from data to influence system behavior and the model repository thus becomes a knowledge base that can be managed with tools and protocols from the Web of Data technological space. **Weaknesses:** The paper does not discuss issues pertaining to the management, versioning and maintenance of model information exposed as Linked Data. For our demonstrator all changes are handled by PUT requests (thus for every change in the modeling tool, the entire model is be re-uploaded). A mechanism for finer granularity updates is envisioned, to translate each type of CRUD operation between the modeling tool and the model repository. **Opportunities:** The actionable benefits of the hereby proposed approach may facilitate the transition to a "Web of semantically-enriched data", a model-driven knowledge space that may act as an intermediate step between the Linked Data paradigm and an actionable, pragmatic Semantic Web approach. **Threats:** The uptake of the Semantic Web paradigm is still slow, as other approaches are available to support data publishing and openness (e.g. OData[33]). However the general principles of the Web of Data, particularly the focus on a federated graph of data, are shared as a guiding roadmap.

6.2. Final conclusions and future plans

This paper advocates the exposure of diagrammatic model semantics within the Web of Data, towards the benefit of evolving it into a "Web of Models and Data" where knowledge captured in diagrams enriches the semantics of

Linked Data beyond its originating data schema. Various graph data structure patterns have been devised for modeling patterns identified in a corpus of modeling languages available within the Open Model Initiative Laboratory. The future plans for this work are determined by the weaknesses highlighted in the SWOT analysis: the emphasis will be placed on refining the granularity of graph updates in relation to model editing operations.

References

1. Heath T, Bizer C, Linked Data: Evolving the Web into a Global Data Space. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 2011, 1:1, 1-136. Morgan & Claypool.
2. The Open Model Initiative Laboratory. http://omilab.org.
3. The ComVantage FP7 EU Project. http://comvantage.eu.
4. Ziegler J, Graube M, Pfeffer J, Urbas L. Beyond app-chaining - mobile app orchestration for efficient model driven software generation. In: *Proceedings of the 17th IEEE International Conference on Emerging Technologies and Factory Automation*. IEEE; 2012, p. 1-8.
5. D2RQ – Accessing Relational Databases as Virtual RDF Graphs, http://d2rq.org/.
6. Object Management Group, UML resource page, http://www.uml.org/.
7. Object Management Group, BPMN specification, http://www.bpmn.org/.
8. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Bauknecht K, Tjoa AM, Quirchmayr G (eds.) *Proceedings of the Third International Conference EC-Web 2002 – DEXA 2002*, LNCS 2455, Springer; 2002, p. 451-464.
9. Costa R, Lima C. An architecture to support semantic enrichment of knowledge sources in collaborative engineering projects. In: Fred A, Dietz J L G, Liu K, Filipe J (eds.) *Communications in Computer and Information Scienc*e vol. 272, Springer, 2013, p. 276-289.
10. Hinze A, Heese R, Schlege A, Luczak-Roesch M. User-defined semantic enrichment of full text documents. In: Zaphiris P, Buchanan G, Rasmussen E, Loizides F (eds.) *Theory and Practice of digital Libraries*, LNCS 7489, Springer, 2012, p. 209-214.
11. Euzenat J, Shvaiko P, *Ontology Matching*. Springer, 2013.
12. Lin Y, Soelvberg A, Goal annotation of process models for semantic enrichment of process knowledge. In: Krogstie J, Opdahl A, Sindre G (eds.) *Proceedings of CAISE 2007*, LNCS 4495, Springer, 2007, p. 355-369.
13. Guizzardi G, Ferreira Pires L, van Sinderen M. An Ontology-Based Approach for Evaluating the Domain Appropriateness and Comprehensibility Appropriateness of Modeling Languages. In Briand L, Williams C (eds.) *Proceedings of MoDELS 2005*, LNCS 3713, Springer; 2005. p. 691-705.
14. Aquino N, Vanderdonckt J, Panach J I, Pastor O. Conceptual modelling of interaction. In: Embley D, Thalheim B (eds.) *Handbook of conceptual modeling: theory, practice and research challenges*. Springer; 2011, p. 335-355.
15. Bencomo N, France R, Cheng BHC, Aßmann U, *Models@run.time*. LNCS 8378, Springer, 2014.
16. OASIS, BPEL Specification, https://www.oasis-open.org/committees/wsbpel/.
17. Maier R, *Knowledge Management Systems*, Springer; 2004.
18. Redlich D, Blair G, Rashid A, Molka T, Gilani W. Research Challenges for Business Process Models at Run-time. In: Bencomo N, France R, Cheng BHC, Aßmann U (eds.) *Models@run.time*. LNCS 8378, Springer; 2014. p. 208-236.
19. W3C, Resource Description Framework, http://www.w3.org/RDF/.
20. Object Management Group, MetaObject Facility Homepage, http://www.omg.org/mof/.
21. BOC-Group, ADONIS Community Edition, http://www.adonis-community.com/.
22. W3C, The Turtle syntax, http://www.w3.org/TR/turtle/.
23. BOC-Group, ADOxx Metamodelling Platform, http://www.adoxx.org/live/.
24. Tolvanen JP, Kelly, S. MetaEdit+: defining and using integrated domain-specific modeling languages. In: *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, ACM; 2009, p. 819–820.
25. Kelly S, Tolvanen JP. *Domain-Specific Modeling: Enabling Full Code Generation*. John Wiley & Son Inc.; 2008.
26. Budinsky F, Steinberg D, Merks E, Ellersick R, Grose TJ. *Eclipse Modeling Framework*. The Eclipse Series. Addison Wesley, 2004.
27. Kern H, Hummel A, Kuhne S, Towards a comparative analysis of meta-metamodels. In: *The 11th Workshop on Domain-Specific Modeling*, Portland, USA, 2011. http://www.dsmforum.org/events/DSM11/Papers/kern.pdf.
28. Open Model Initiative Laboratory, ComVantage modelling prototype and resources. http://www.omilab.org/web/comvantage/home.
29. W3C, The TriG syntax, http://www.w3.org/TR/trig/.
30. The Sesame HTTP Protocol, http://rdf4j.org/sesame/2.7/docs/system.docbook?view#The_Sesame_REST_HTTP_Protocol
31. Link Discovery with SILK. http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/.
32. W3C, The SPARQL 1.1 Query Language, http://www.w3.org/TR/sparql11-overview/
33. OASIS, Open Data Protocol, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=odata.