Information Systems 56 (2016) 174–197

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Linked Open Models: Extending Linked Open Data with conceptual model information

Dimitris Karagiannis [a,*,1], Robert Andrei Buchmann [b,*,2]

[a] University of Vienna, Faculty of Computer Science, Knowledge Engineering Research Group, Währinger str. 29, A-1090 Vienna, Austria
[b] Babes-Bolyai University, Faculty of Economics and Business Administration, Business Information Systems Department, Str. T. Mihali 58-60, 400591 Cluj-Napoca, Romania

ARTICLE INFO

Article history:
Received 30 April 2014
Received in revised form
27 September 2015
Accepted 4 October 2015
Available online 22 October 2015

Keywords:
Linked Open Data
Linked Open Models
Conceptual modeling
Metamodeling
Model query

ABSTRACT

As the uptake of the Semantic Web vision has been relatively slow, a strategy based on pragmatic steps is being deployed in order to setup enablers and to stimulate acceptance. "Linked Open Data" refers to one of these early steps, benefiting from an available technological space (RDF, HTTP). The paper proposes "Linked Open Models" as a possible additional step, whose aim is to enable users to externalize knowledge in the form of diagrammatic models – a type of content that is human-readable, as well as linkable in the way promoted by the Linked Data paradigm. Consequently, diagrams become user-generated content that semantically enriches Linked Data, thus allowing richer constraints or connections in queries. The vision emerged from the context and use cases provided by the ComVantage FP7 research project, where linking benefits for conceptual diagrammatic models have been investigated. However the paper also discusses the vision's degree of generality, beyond the scope of the exemplary project use cases. Feasibility was demonstrated with a vocabulary and a prototype mechanism for exposing the models created with a hybrid, domain-specific modeling method in a Linked Data-driven collaboration environment.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

The slow adoption of the Semantic Web vision [1] has motivated researchers in defining and setting up concepts or technologies acting as enablers and facilitators, focused on pragmatic solutions and rooted in available technology, necessary to support a gradual advancement over the gap between the current state of affairs and the vision. While research results from artificial intelligence and ontology engineering are available as proof-of-concepts for the far end of the vision, it became obvious that the building blocks on which the Web 2.0 was developed (e.g., AJAX, XML/JSON, content management systems) were lacking certain ingredients required by the Semantic Web vision.

The Linked Data paradigm [2] emerged as such an intermediate step, with the aim of establishing both an instance data test-bed and an enabling environment for the future Semantic Web. The paradigm reduces the role of "ontologies" to the function of inducing structure to Linked Datasets (as "vocabularies"), and this is also how the term "ontology" will be used throughout this paper (unless indicated otherwise). With Linked Data, emphasis is placed on structuring, distributing and querying data in ways that are inspired by the "semantic networks" approach from artificial intelligence, grafted on the networked nature of the Web. The paradigm aims to induce a network effect to data linking, not unlike the one that enabled the growth of Web 1.0 (based on document

* Corresponding authors.
E-mail addresses: dk@dke.univie.ac.at (D. Karagiannis), robert.buchmann@econ.ubbcluj.ro (R.A. Buchmann).
[1] Tel.: +43 1 4277 78910.
[2] Tel.: +40 264 418652.

http://dx.doi.org/10.1016/j.is.2015.10.001
0306-4379/© 2015 Elsevier Ltd. All rights reserved.

linking) or Web 2.0 (based on social and content linking). We extend this view with an additional layer – "Linked Models" – by employing Linked Data principles and techniques for content of different nature and granularity – *conceptual (diagrammatic) models*.

The Linked Data paradigm aims to educate the Web developers towards new practices for information structuring, with an impact comparable to the one that made possible the rise of the relational databases. Indeed, "thinking in tables" had the advantage of being well defined at formalism level (relational algebra), well promoted by tools and well assimilated (on the presentation layer) by the end-users who produce content (data). Filling up, sorting, filtering tables is not anymore a matter of technical skills, it is an intuitive way in which most users expect to interact with their data records, regardless of purpose. A possible analogy, intended to bring the argument closer to the work at hand, relies on a statement made in the panel discussions of the Open Model Initiative workshop of 2012 by John Mylopoulos: "…[in the future] conceptual modeling will be taught in the elementary school" [3]. The statement assumes a level of education and tool support that enables end-users to structure knowledge without feeling that they're actually making a content structuring effort. It is a motivational assumption for the work at hand that such enablers may come from the field of conceptual modeling.

The goal of this paper is to communicate the vision of "Linked Open Models", with an instantiation in the case of the ComVantage EU research project [4], where models are exported using an RDF [5] vocabulary in order to facilitate inter-model linking, model-to-data linking, model transformation and sharing. The paper is structured as follows: Section 2 states the problem and positions its research goals relative to the context of the ComVantage research project. Section 3 describes a minimal but representative running example, detailing the mechanism of exposing diagrammatic conceptual models as Linked Data. Section 4 analyses both foreseen and confirmed benefits on some exemplary cases, and continues with a discussion on the level of generality and reusability beyond the project context. Section 5 indicates related works. The paper concludes with a contribution summary against the baseline given by the related works, provides a SWOT evaluation based on the existing proof-of-concept and formulates the final takeaway messages.

2. Problem context and statement

2.1. Motivation

Modeling languages are commonly involved in model-driven software engineering or automated programming [6], as "means to an end", typically aiming for code generation. We, through the Open Model Initiative [7], advocate a broader scope for conceptual modeling, as means of *knowledge externalization/representation* for the purpose of communication and understanding [8]. For example, the practice of Business Process Modeling has developed in time as a successful decision support or knowledge management approach [9,10] even without the benefit of workflow automation. Machine-readability is just a particular type of "understanding" and, in this respect, the work at hand enables model understanding and sharing within a Linked Open Data environment, with the potential of enabling *diagram-awareness* in information systems at run-time (a possible generalization of the process-awareness concept [11]).

Just as Web 2.0 enabled user content production through simple forms and templates provided by content management systems, the work at hand promotes diagrammatic conceptual modeling as a way of inducing richer structure and semantics for Linked Data in a user-oriented way, for those who are agnostic of ontology engineering but are familiar with diagrammatic modeling. The challenge of knowledge acquisition for the Semantic Web can thus be met on a more domain-focused level (compared to generic ontology editors like Protégé [12]), possibly sacrificing inference capabilities (or rather compensating them with query-time model transformations, as a use case will later suggest).

We add to the motivation the research challenges stated in the FInES Roadmap [13]. FInES is a research community and project cluster focused on investigating the potential of enterprise systems that will leverage the benefits of the "Future Internet". A particular subset of their research challenges are grouped in the so-called *Knowledge Dimension*: *RC1. The Unified Digital Enterprise; RC2. Linked Open Knowledge; RC3. Complex Systems Modeling.* Conceptual modeling can have significant impact in advancing this dimension, and the Linked Open Models vision discussed in this paper is a pragmatic proposal in this respect, just as Linked Open Data is a pragmatic enabler for the Semantic Web. In relation to the FInES challenges, modeling languages can provide an entry point to the *Linked Open Knowledge* and linking capabilities are required to glue together the digital image of the enterprise across different types of models.

2.2. Framework

The work at hand relies on (a) *methodological enablers*: the metamodeling framework of the Open Model Initiative Laboratory [7] and the notion of "modeling method" as defined in [14]; (b) *technological enablers*: a metamodeling platform (ADOxx® [15]) on which such a method can be implemented in the form of a modeling tool, as well as the technological space of the Linked Data paradigm employed for model serialization.

According to [14], a *modeling method* comprises the following building blocks:

(1) A *modeling language* describes the set of modeling constructs, including their custom notation (how they look), grammar (how they can be visually connected) and semantics (property sets and relations prescribed by a metamodel). The modeling language can be partitioned in *model types* addressing different facets or abstraction layers of the system under study. This partitioning can be a usability feature (a top-down decomposition approach to avoid visual cluttering in diagrams) or a

consequence of hybridization (a bottom-up strategy employed to interconnect modeling language fragments). In any case, the different types of models can be bridged by functional hyperlinks that enable cross-model navigation. The work at hand exploits this by semantically lifting such presentational links in a Linked Data environment.

(2) A *modeling procedure* defines the steps that must be taken by modelers towards some goal (i.e., the precedence of creating different types of models to make linking possible). The work at hand adds some guidelines on how models should be prepared for linking.

(3) *Mechanisms and algorithms* cover functionality for model processing, with various purposes-visualization, transformation, evaluation etc. In this respect, the work at hand proposes a feature for exposing content created with a hybrid modeling method to a Linked Data environment, and suggests additional functionality that can benefit from this.

2.3. Project context

The context and case studies for the work at hand have been provided by the ComVantage research project (deliverables and details on the consortium are available at [4]). A brief overview is provided here to facilitate the understanding of the use cases to be discussed in later sections:

The project aimed to define and deploy an IT architecture based on mobile apps consuming Linked Data, as support for business process execution in virtual enterprises [16]. The run-time architecture was complemented by a design-time component in the form of an enterprise modeling method (and tool – a modeling prototype is available at [17]). Key challenges included the bridging of design-time and run-time components to induce *model-awareness in the mobile apps*, and the solution described in the work at hand was adopted to achieve this. An overview of the hybrid modeling method is provided here, suggesting the different model types available for describing different *facets* of a "ComVantage enterprise" [4, deliverable D312]:

- The *Motivators facet*. Based on a flavor of the feature modeling approach [18], motivators are typically described as abstract value structures that can represent products, services or a mix of them (e.g., when dealing with product servitization). Due to the application areas of the project, some domain-specific motivators are also supported (e.g., defect models that trigger corresponding maintenance processes – see [19]);
- The *Participants facet*. This covers different kinds of resources that might be required to perform tasks on different levels of abstraction. In a decompositional and taxonomical manner, one can describe *liable entities* (e.g., business roles, business partners, individual roles or employees), *concrete assets* (e.g., mobile apps, Linked Data endpoints) or *abstract assets* (e.g., skills, mobile app capabilities, data entities);
- The *Tasks facet*. This captures the actual work to be performed, from two perspectives – *procedures* (modeled as control flows) or *interactions* (between the resources described in the *Participants* facet).

Vertically, the method provides several layers expressing different modeling *scopes* that might be of interest to the modeler:

- The *business scope* is the most abstract one, concerned only with the business model (described by a flavor of the e3 value language [20]) and its high level participants (business roles, market segments);
- The *supply chain scope* describes high level production processes or service delivery processes and their participants (business roles, concrete business partners and their capabilities – i.e., the value they provide);
- The *enterprise scope* is concerned with internal processes of an enterprise (typically subprocesses of the previous "scope"), allocated to responsible individuals (roles or employees from an organigram) or capabilities (skills);
- The *app management scope* describes the required orchestration of apps for a business process (an *app orchestration* is a generalization of the app-chaining technique, employed to prescribe app execution for various purposes, e.g., training new employees [21]);
- The *app requirements scope* is concerned with how a user should interact with the user interface and the data requirements for a mobile app – the "participants" here being UI elements and required data entities of an ER diagram.

Despite the complexity, the different model types allow the modelers to focus on the facets they are interested in, thus avoiding a "take all or leave all" approach. A metamodel describes how the model types are related to each other through relations of weaker semantics (e.g., part-of) or richer semantics (e.g., a process activity requires a certain skill). These relations take the functional form of hyperlinks for navigation across models (e.g., from an activity in a business process model to the responsible role in an organigram, from a business partner to the value they are capable of providing).

Examples of such links are indicated in Fig. 1, between several models in the *supply chain scope* (a production process – the "tasks", the corresponding product structure – the "motivator", and the organizational map – the business "participants"). Additional detail on how certain models types evolved and are semantically linked are available in the iterative method specification in [4, deliverables D312, D822] and the guidelines documentation for the modeling prototype [17]. The use case discussion from Section 4.1 will give insight to some model types relevant to the discussed examples; otherwise, the project's domain-specificity will not be detailed here, since the scope of the paper is more general and the proposed mechanism is intended to be reusable beyond the project needs.

2.4. Problem definition and requirements

We define here the problem in more general terms than suggested in the previous section by the project context. The assumption is that companies acting in a collaborative environment have the ability to lift and share with collaborators Linked Data from their legacy systems.

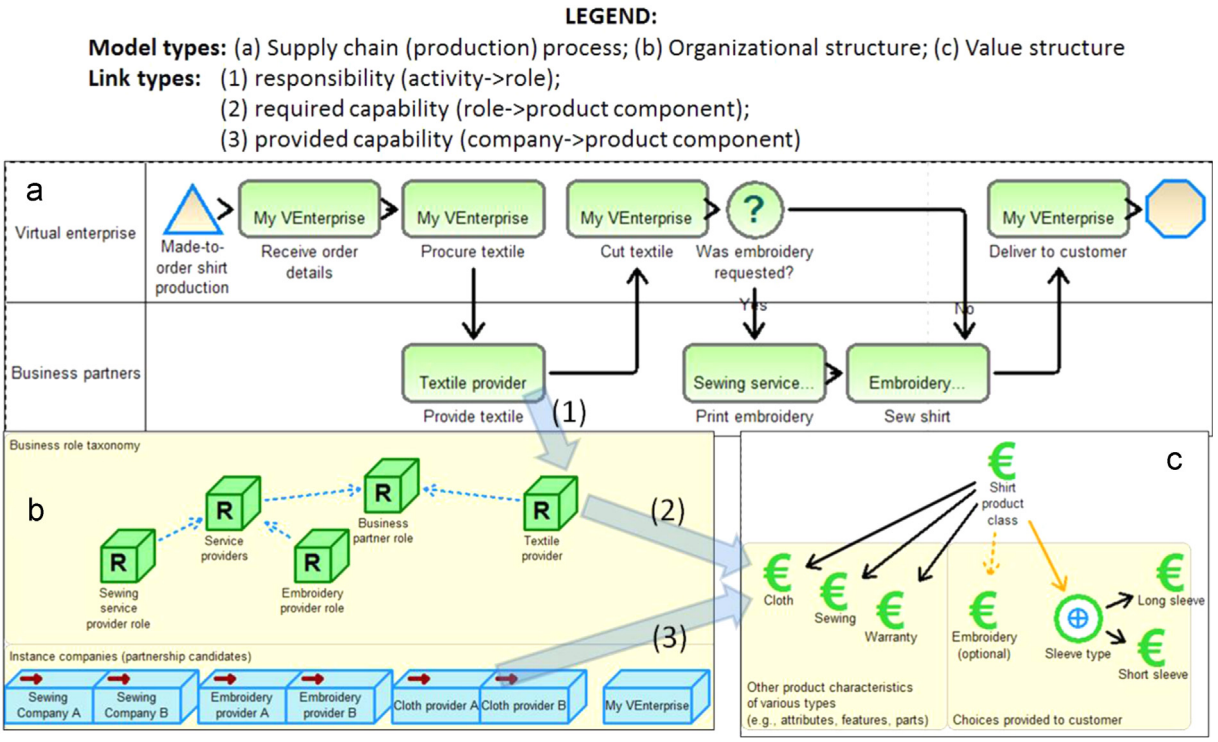

Fig. 1. ComVantage model samples in the Supply chain scope.

Table 1
Requirements for Linked Open Models.

Compliance level	Requirements for Linked Open Data [2]	Requirements for modeling tools creating Linked Open Models	*As-Is* situation: the ADONIS [24] tool
1-star	Can be retrieved on the web, with Open License	Tool must be able to export models on the Web, with Open License	PNG/HTML/RTF exports are available for documentation purposes. No direct Web upload is provided from the tool
2-stars	Machine-readable	Exported models must be machine-readable	Proprietary ADL export is available for model interchange purposes
3-stars	Non-proprietary format	Exported models must be in a non-proprietary format	XML export is also available as a non-proprietary format to complement ADL
4-stars	Uses RDF standards	a. Exported models must be expressed with an RDF vocabulary b. HTTP URIs should identify all model elements; c. Direct HTTP upload to a server URI of choice via a standard SPARQL protocol should be provided in the tool.	Not available, subject of the work at hand.
5-stars	It is linked RDF	Models should be linkable outside the modeling tool, in a way that enables queries over multiple models.	To support inter-model navigation, hyperlinks can be created in the modeling tool.

For knowledge management purposes, they rely on diagrammatic modeling to describe process-centric supply chains and workflows in relation to their business context (motivators, requirements, available resources). Consequently, run-time data and diagrams are stored and managed traditionally by different systems. In this context, certain requirements are raised: (a) run-time systems must be sensitive to the knowledge expressed in diagrammatic form; (b) partner companies need to link models created with different installations of the modeling tool, or even different modeling tools that use different notations for the same conceptual structure; (c) legacy data is rigidly structured according to its originating system and lacks relations between disparate data sources; such relations are, however, available in diagrammatic models and can significantly enhance querying capabilities in the absence of an integrative ontology (or as a complement). Additional assumptions are provided on a use case basis, in Section 4.1.

We propose the Linked Open Models as an additional layer that can be built upon the foundations established by Linked Open Data, towards shaping up the "knowledge

dimension" envisioned by FInES. The technological space to be employed is based on RDF [5] (for the data model and format), SPARQL [22] (for queries) and HTTP-based protocols [23] (for remote querying). In order to align this vision, in Table 1 we take into consideration the key requirements that define "5-stars quality" Linked Open Data [2] and establish, by analogy, similar requirements for Linked Open Models. On the last column these are compared to the familiar case of the ADONIS business process management tool [24].

3. Linked Open Models

3.1. Running example

A diagram serialization mechanism is proposed by the work at hand, based on a multi-layered RDF vocabulary comprising a meta-metaconcepts (fixed by the vocabulary), metaconcepts (derived from the modeling language alphabet) and instance resources (generated from user-created diagrams). The proof-of-concept was implemented for the ComVantage modeling tool, which is built on the ADOxx metamodeling platform. A detailed description of the exported data structure is provided here for a minimal running example that showcases two inter-linked models of different types.

Fig. 2 presents a minimal business process model linked to a minimal organizational structure model (from the *enterprise scope* of the ComVantage modeling method). The boxes between the two diagram canvases are property sheets ("editable attributes") that define the semantics for each model element (as prescribed by the language metamodel) and can be edited by the user outside the modeling canvas (transparent arrows suggest the order of interactions in the user interface). These boxes also allow the definition of references across models. References become hyperlinks when anchors are included on notation level, thus enabling navigability and can be themselves described by their own attributes (in a tabular form). In Fig. 2 the links indicate on a semantic level the assignment of an *Expert* role (provided by an *IT Department*) to two activities in a maintenance process model. Editable attributes can also be defined for visual connectors (e.g., the transition conditions outgoing from the "Broken" decision).

The business process model notation is customized (for project-specific purposes), but its semantics comply to typical control flow languages (a flavor of the ADONIS notation [24]). However the notation is irrelevant to the Linked Open Models, since they only expose the semantic network captured through diagrammatic models. Further considerations on visualization will be provided in Section 4.2.

The hypergraph resulting from this minimal model set is described graphically in Fig. 3 and formally in the next section. We prefer this rather informal graphical representation of the hypergraph to highlight the different types of edges and nodes, and to emphasize the key elements of the employed RDF vocabulary. The current implementation exposes such graphs in the syntax of choice for "named graphs" (Nquads [25], TriG [26], TriX [27]).

3.2. Formal descriptions

The different types of nodes and edges in the graph presented in Fig. 3 are derived from patterns detected in the diagram elements visible in Fig. 2. Since the RDF predicates are not directly translatable to graph-theoretic edges, we need to employ a 4-uniform hypergraph formalism which is

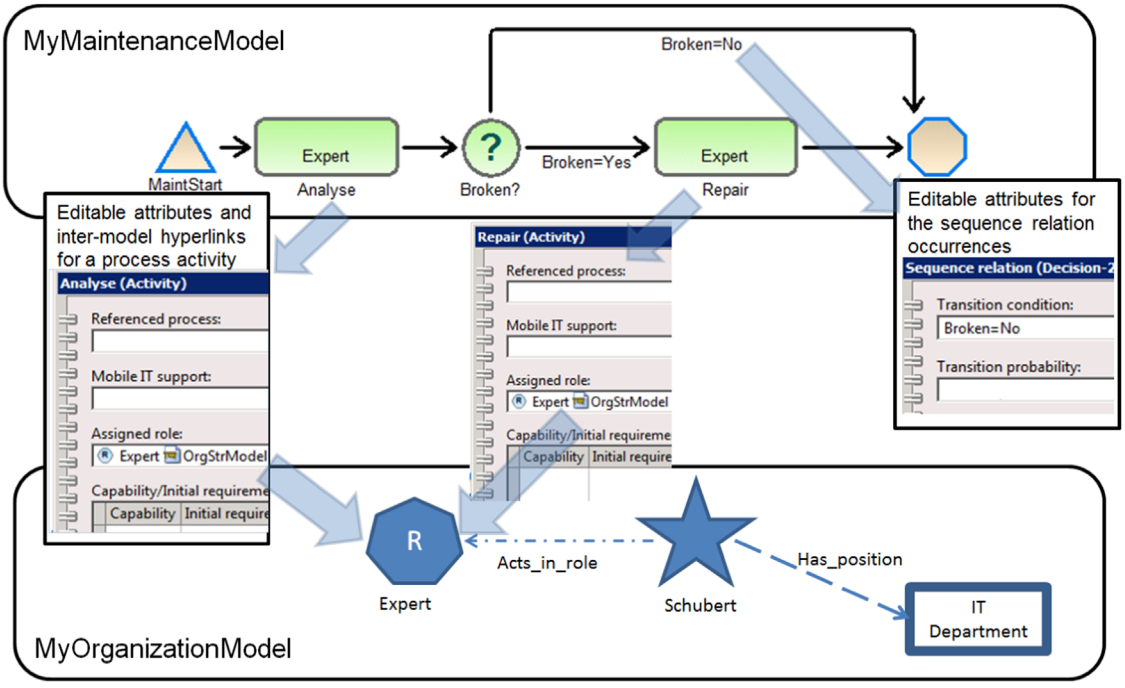

Fig. 2. A business process model with hyperlinks to an organizational model: a diagrammatic view.

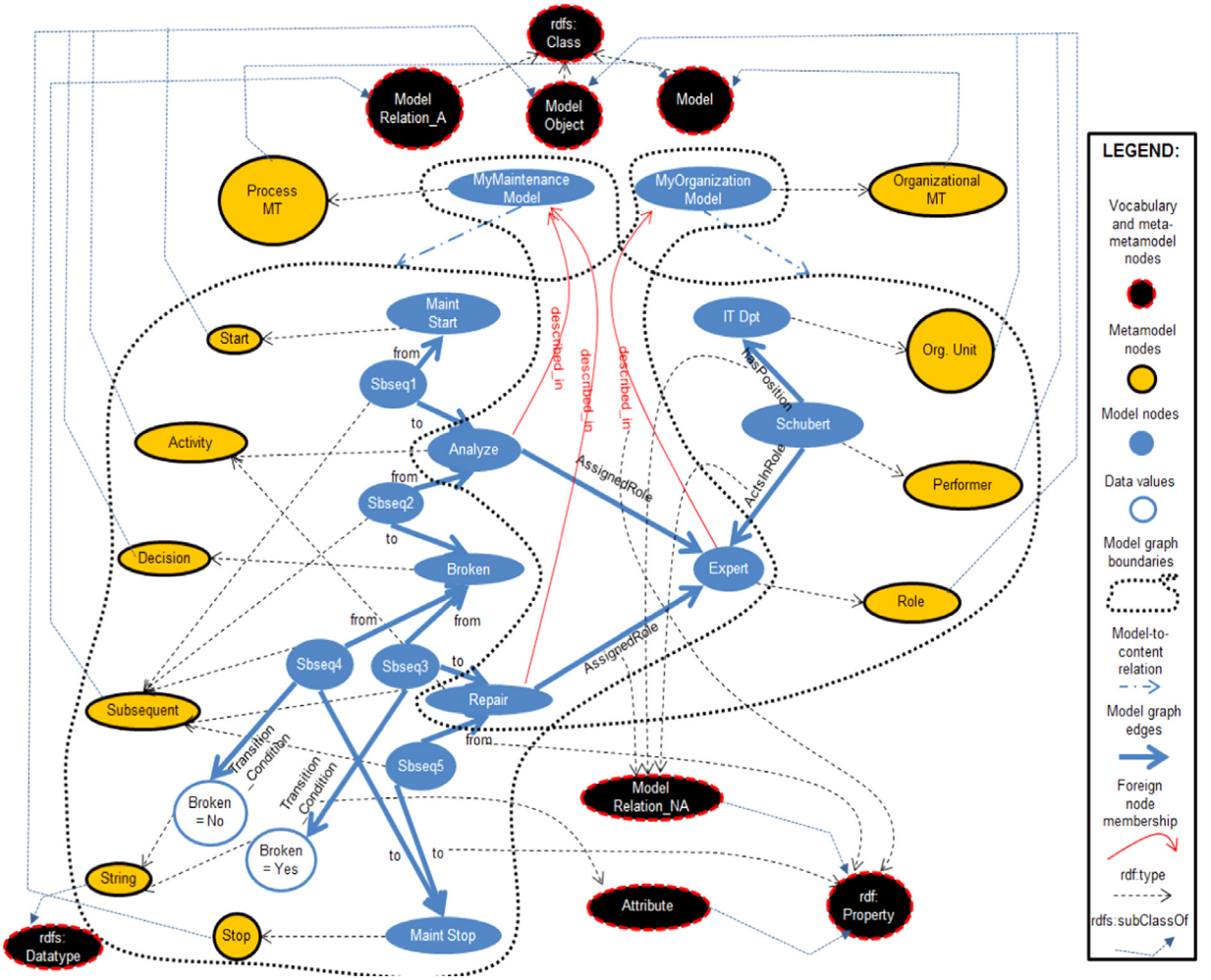

Fig. 3. The Linked Data hypergraph generated from the diagrams in Fig. 2.

fit to express N-quads, since it also treats named graphs and predicates as nodes (each quad becomes a 4-arity relation/hyperedge) [28]. The following definitions will cover the different types of nodes and edges visible in Fig. 3.

Definition 1. . A *Linked Model Base* is a set of inter-linked models (usually exported at the same time), including their links, corresponding metamodel and meta^{2}model elements. The resulted hypergraph comprises four types of nodes (N) and ordered hyperedges (E) representing the exported N-quads (n is the number of models):

$$N = MMM \cup MM \cup MMD \cup \bigcup_{i=1}^{n} MD_i$$

$$E = MMME \cup MME \cup MMDE \cup \bigcup_{i=1}^{n} MDE_i$$

The meaning of each component is further described:

Definition 2. . A *Linked Model Vocabulary* is a hypergraph comprising the metamodel-independent resources. It is, however, dependent on the meta^{2}model of the metamodeling platform (M0 abstraction level in the MOF architecture [29]) – for the ComVantage research project, it is based on the ADOxx metamodeling platform and its formalism, retaining only features of high generality for which mappings can be easily identified in other platforms. This hypergraph has the following components:

1) *MMM*, the set of nodes established at vocabulary level, including the primitive RDF resources, the meta^{2}model level constructs, and the contexts to be used for qualifying all triples of the Linked Model Base.

2) *MMME*, the set of hyperedges declaring *MMM* nodes as being of a primitive type or a subclass of a primitive. These hyperedges are qualified by a dedicated context, *cv:Mmg* (the meta-metagraph name):

$$MMM = Primitives \cup Vocab \cup Contexts$$

$$Primitives = \left\{ \begin{array}{c} rdf{:}Class, rdf{:}Property, rdfs{:}Datatype, rdf{:}List, \\ rdf{:}type, rdfs{:}subClassOf \end{array} \right\}$$

$$Vocab = \left\{ \begin{array}{c} cv{:}Model, cv{:}ModelObject, cv{:}ModelRelation\_A, \\ cv{:}ModelRelation\_NA, cv{:}Attribute, cv{:}from, \\ cv{:}to, cv{:}contains, cv{:}described\_in \end{array} \right\}$$

$$Contexts = \{cv{:}Mmg, cv{:}Mg, cv{:}Mdg\}$$

$$MMME = \left\{ \begin{array}{c} (s, p, o, cv{:}Mmg) \mid \\ s \in MMM, \\ p \in \{rdf{:}type, rdfs{:}subClassOf\}, \\ o \in Primitives \end{array} \right\}$$

Since the Linked Model Vocabulary is not model-dependent, it should not be re-exported with every model. It is fixed for a metamodeling platform and it was designed to be easily adopted by common metamodeling platforms that rely on a graph-based representation of relational models. Its basic constructs are: a class of all models-*cv:Model*; a class of all modeling objects (visual constructs) – *cv:ModelObject*; a class of all modeling relations accepting editable modeling attributes – *cv:ModelRelation_A*; a class of all modeling relations without attributes – *cv:ModelRelation_NA*; a class of all modeling attributes (editable in the property sheets) – *cv:Attribute*; two properties relating a modeling relation to its originating and ending modeling objects (thus capturing model syntax) – *cv:from* and *cv:to*; a property relating visual containers to their contents (therefore lacking a visual connector) – *cv:contains*; and a property relating a "foreign" modeling object (the target of a hyperlink) to its originating model – *cv:described_in*.

Thus, MMME edges describe the bridging between the proposed vocabulary and the RDF primitive resources. Examples of MMME hyperedges are as follows (their graph identifier is omitted):

cv:from rdf:type rdf:Property (it declares that the "from" edges, which relate a modeling relation to its originating modeling object, is an RDF property).

cv:ModelRelation_A rdf:type rdfs:Class (it declares the class of all modeling relations that accept attributes).

The namespace has been assigned to the ComVantage research project, although namespaces and versioning can be defined for each metamodeling platform, if a direct adoption of the vocabulary proposed here is not favored.

Definition 3. . A *Linked Metamodel* is a hypergraph comprising the elements defined by the metamodel of a modeling language (M1 abstraction level in MOF). For the ComVantage research project, it captures the syntactical elements of the modeling language, partly depicted in the example discussed here:

1. *MM,* the set of nodes derived from the metamodel (types of modeling nodes, of modeling edges, data types, model types, modeling attributes) and those *MMM* nodes required to link the metamodel to the Linked Model Vocabulary (*MMM′*). The bridging from metamodel elements to the Linked Model Vocabulary is also depicted in Fig. 4.
2. *MME,* the set of hyperedges declaring the type or subsuming *MM* nodes to *MMM* nodes. These are qualified by a dedicated context, *cv:Mg* (the metagraph name).

$$MM = ModelT \cup NodeT \cup EdgeT \cup DataT \cup Attrs \cup MMM'$$

$$ModelT = \{cv{:}ProcessMT, cv{:}OrganizationalMT\}$$

$$NodeT = \left\{ \begin{array}{c} cv{:}Start, cv{:}Activity, cv{:}Decision, cv{:}Stop, \\ cv{:}Subsequent, \\ cv{:}Role, cv{:}Performer, cv{:}OrgUnit \end{array} \right\}$$

$$EdgeT = \{cv{:}Assigned\_units, cv{:}Acts\_in\_role, cv{:}Has\_position\}$$

$$Attrs = cv{:}Transition\_condition$$

$$DataT = \{cv{:}String\}$$

$$MME = \left\{ \begin{array}{c} (s, p, o, cv{:}Mg) \mid \\ s \in MM, o \in MMM, p \in \{rdf{:}type, rdfs{:}subClassOf\} \end{array} \right\}$$

These hyperedges describe the bridging between language-specific concepts (derived from the language metamodel) and the generic Linked Model Vocabulary. Examples follow (context is omitted):

cv:ProcessMT rdfs:subClassOf cv:Model (specific to the discussed example, it declares that any process model is a model; a similar declaration is exported for the organizational model type).

cv:Activity rdfs:subClassOf cv:ModelObject (specific to the discussed example, it declares that any activity is a modeling object; other examples from the process model are the decision nodes, the starting and ending points etc. while the organizational models have, similarly, their own concept set, including roles, organizational units).

cv:Subsequent rdfs:subClassOf cv:ModelRelation_A (specific to the discussed example, it declares that any "subsequent" visual connector is a modeling relation that accepts attributes – the metamodel defines them as having a transition condition for some occurrences, therefore they must be treated as relations of a higher arity, and not as RDF binary predicates).

cv:Assigned_units rdf:type cv:ModelRelation_NA (specific to the discussed example, it declares any activity-to-role assignment hyperlink as being a modeling relation with no attributes; the same class covers also visual connectors lacking editable properties, such as the role-performer assignment in this example).

cv:Transition_condition rdf:type cv:Attribute (specific to the discussed example, it declares the transition condition as a modeling attribute, usually editable in the property sheets provided by the modeling tool).

Since the Linked Metamodel is not model-dependent, it should not be re-exported with every model. It is fixed for a modeling method/modeling tool implementation and can be provided to the community that adopts the tool. For example, the ComVantage modeling tool is deployed in a community-driven environment called "Open Model Initiative Laboratory" [7], which hosts metamodeling projects addressing concerns of various domain-specific communities. The constructs of the Linked Metamodel are RDF classes and properties directly derived from the metamodel of the modeling language. For brevity, the formal descriptions of *MM* only covers the resources relevant to the discussed example. For a complete

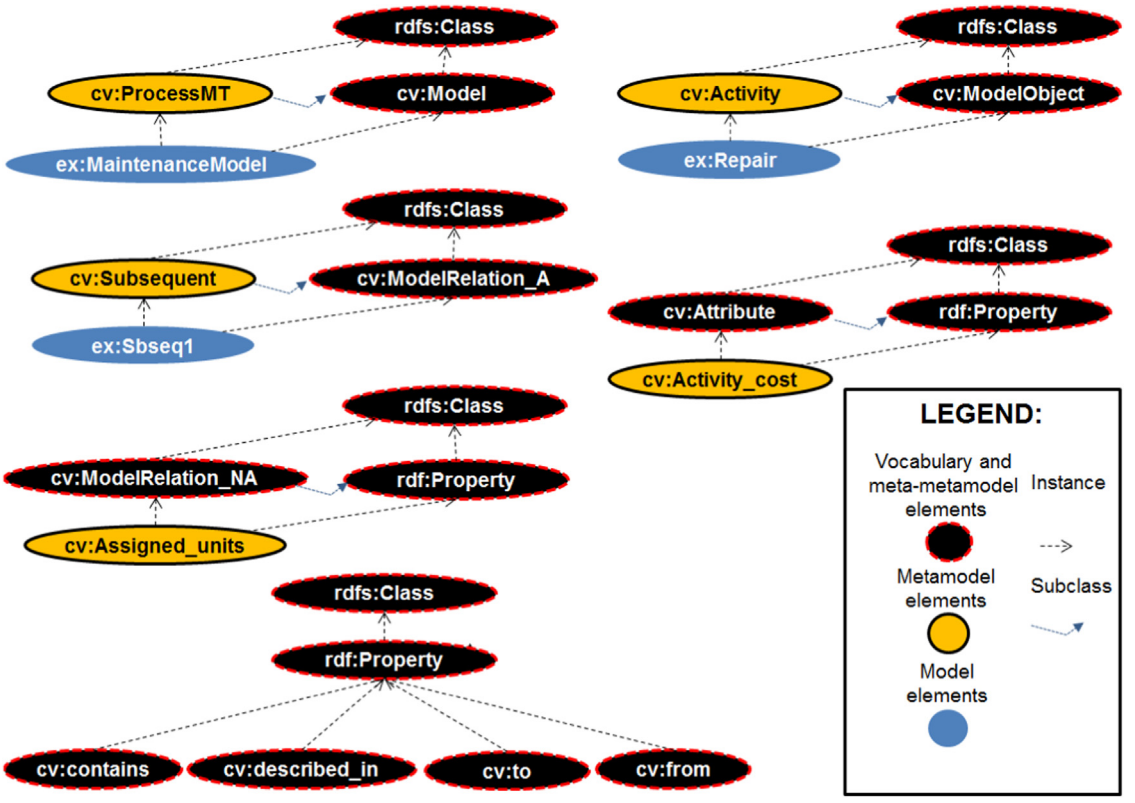

Fig. 4. Constructs of the Linked Model Vocabulary.

enumeration, the metamodel itself should be consulted [4, deliverables D312, D822].

It is important to remark that no metamodel constraint definition is exported, only the metamodel structure (types to be instantiated in models) since only these are necessary to answer SPARQL queries. Therefore, no validation is to be performed in the Linked Data version of the models – this is consistent with the non-prescriptive and Open World nature of Linked Data. Model consistency should therefore be enforced and guaranteed in the modeling tool acting as a "constraining gateway" to a Linked Open Models cloud. The metamodel should guarantee that all parties using the same modeling tool will be compliant to the same Linked Model Vocabulary and the same Linked Metamodel, with the same namespace. A model repository is further necessary to raise awareness of existing models and to facilitate linking, but this aspect is not in the paper scope.

Definition 4. . A *Linked Model Cluster* is a hypergraph comprising the metadata, structure and content of interlinked models, as defined by the modeler. For the ComVantage research project, it captures models created with the ComVantage modeling method. The components of this hypergraph are:

1. *MMD*, the set of nodes indicating model types, values of model metadata (not present in this example) and the subset of *MM* necessary to link the model to the Linked Metamodel (*MM′*, containing RDF primitives and types).
2. *MMDE*, the set of hyperedges having models as subjects, declaring their types (from *ModelT*) and any metadata attached by the modeler on model level (from *Values*, a set that will be structurally described further on).
3. MD_i is the set of nodes representing the elements of model i.
4. MDE_i is the set of hyperedges "gluing" together the contents of model i, using the model itself as a context/named graph (highlighted by model boundaries in Fig. 3).

Regarding the metadata component, we have, in the general case (the meaning of *Values* and *CustomMetadata* will be discussed further on, in the context of the model contents):

$$MMD = Models \cup CustomMetadata \cup MM'$$

$$MMDE = \left\{ \begin{array}{c} (s, rdf{:}type, o, cv{:}Mdg) | \\ s \in MMD, \\ o \in ModelT \end{array} \right\} \cup \left\{ \begin{array}{c} (s, p, o, cv{:}Mdg) | \\ s \in MMD, \\ p \in Attrs, \\ o \in Values \end{array} \right\}$$

$$\cup \left\{ \begin{array}{c} (s, p, o, cv{:}Mdg) | \\ s \in MMD \end{array} \right\}$$

For the discussed example, we only have two models and their type assignments:

$$Models = ex{:}MyMaintenanceModel, ex{:}MyOrganizationModel$$

$$MMDE = \left\{ \begin{array}{c} (ex{:}MyMaintenanceModel, rdf{:}type, cv{:}ProcessMT, cv{:}Mdg), \\ (ex{:}MyOrganizationModel, rdf{:}type, cv{:}OrganizationalMT, cv{:}Mdg) \end{array} \right\}$$

Regarding the model content component, we have in the general case:

$$MD_i = Objs_i \cup ForeignObjs_i \cup Relations_i \cup Values_i \cup Custom_i \cup MM'$$

$$Values_i = SimpleValues_i \cup Blanks_i \cup Fields_i$$

From the formal descriptions it can be observed that the RDF nodes representing model-specific content are the modeling objects (*Objs*), the objects to/from which hyperlinks exist (*ForeignObjs*), the modeling relations (*Relations*), the attribute values (*Values*), some arbitrary nodes (*Custom*) and those metamodel nodes necessary to link the model to the higher abstraction layers (types, RDF primitives).

The *Custom* nodes allow the extension of each model and modeling object description with arbitrary RDF triples, independent of the metamodel, to enable a certain level of flexibility which will be emphasized later in Section 4.1. A designated modeling attribute called "Property collector" (assigned to all objects) takes the form of a table where (predicate,object) or (subject,predicate) pairs can be created, assuming for the selected object the position of RDF subject or object, respectively.

The *Values* set comprises nodes involved in describing attribute values. These can be *simple values* (assigned to types from *DataT* or a mapping of them on XML Schema types) or, as permitted by some metamodeling platforms, complex values captured in editable tables. Tables are converted according to the methodology commonly employed when producing RDF graphs from relational databases [30]: a table becomes an RDF list of records, and each record translates to a blank node (*Blanks*) acting as subject for triples whose predicates are the table columns (*Fields*) and whose objects are the actual data points (again *SimpleValues*, but also *ForeignObjs* if inter-model links are allowed in tables). An example of such a table-attribute is shown in Fig. 5, where the access control requirements are described for an "information resource" object (*Machine sensor values*) in terms of "allowed role" (the *Subject* field), "allowed action" (the *Action* field) and additional natural language descriptions (the description fields). Several such records can be assigned to the same object (only the first one is shown in the derived RDF code) and the records may contain hyperlinks to foreign objects (here, the roles from an organizational model). In the ComVantage project context, such resource objects are further linked as requirements to business process models. Another example of a complex attribute is the above mentioned "Property collector" – however, that one gets a special treatment, as it translates directly in RDF triples (taking its nodes directly from the table) rather than following this table-to-graph conversion.

The model contents are glued together by hyperedges describing both visual and nonvisual relations:

$$MDE_i = TypeAssgn_i \cup ValueAssgn_i \cup ModelingConnections_i \cup CustomProperties_i$$

$$TypeAssgn_i = \left\{ \begin{array}{c} (s, rdf{:}type, o, c_i) \mid \\ s \in Objs_i \cup Relations_i \cup Blanks_i, \\ o \in NodeT \cup EdgeT \cup rdf{:}List, \\ c_i \in MMD \end{array} \right\}$$

$$ValueAssgn_i = \left\{ \begin{array}{c} (s, p, o, c_i) \mid \\ s \in Objs_i \cup Relations_i, \\ p \in Attrs, \\ o \in SimpleValues_i \cup Blanks_i, \\ c_i \in MMD \end{array} \right\}$$

$$\cup \left\{ \begin{array}{c} (s, p, o, c_i) \mid \\ s \in Blanks_i, \\ p \in Fields_i, \\ o \in SimpleValues_i \cup ForeignObjs_i, \\ c_i \in MMD \end{array} \right\}$$

$$\cup \left\{ \begin{array}{c} (s, p, o, c_i) \mid \\ s \in Blanks_i, \\ p \in \{rdf{:}first, rdf{:}last\}, \\ o \in Blanks_i \cup rdf{:}nil, \\ c_i \in MMD \end{array} \right\}$$

$$ModelingConnections_i = \left\{ \begin{array}{c} (s, p, o, c_i) \mid \\ s \in Relations_i, \\ p \in \{cv{:}from, cv{:}to\}, \\ o \in Objs_i, \\ c_i \in MMD \end{array} \right\}$$

$$\cup \left\{ \begin{array}{c} (s, cv{:}described\_in, o, c_i) \mid \\ s \in ForeignObjs_i, \\ o \in MMD, \\ c_i \in MMD \end{array} \right\}$$

$$\cup \left\{ \begin{array}{c} (s, p, o, c_i) \mid \\ s \in Objs_i \cup ForeignObjs_i, \\ p \in EdgeT \cup \{cv{:}contains\}, \\ o \in Objs_i \cup ForeignObjs_i, \\ c_i \in MMD \end{array} \right\}$$

$$CustomProperties_i = \left\{ \begin{array}{c} (s, p, o, c_i) \mid \\ s \in Objs_i, \\ p, o \in Custom_i, \\ c_i \in MMD \end{array} \right\}$$

The following sets describe the actual elements for the running example (Fig. 2):

$$Objs_1 = \{ ex{:}MaintStart, ex{:}MaintStop, ex{:}Analyze, ex{:}Broken, ex{:}Repair \}$$

$$ForeignObjs_1 = \{ex{:}Expert\}$$

$$Relations_1 = \{ ex{:}Sbseq1, ex{:}Sbseq2, ex{:}Sbseq3, ex{:}Sbseq4, ex{:}Sbseq5 \}$$

$$Values_1 = \{'' \text{Broken} = \text{No}'', '' \text{Broken} = \text{Yes}''\}$$

$$Objs_2 = \{ex{:}ITDpt, ex{:}Schubert, ex{:}Expert\}$$

$$ForeignObjs_2 = \{ex{:}Analyze, ex{:}Repair\}$$

The relations, as drilled down in the formal components of *MDEi*, are:

- *the assignment of types* to modeling objects (from *Objs* to *NodeT*), to modeling relations (from *Relations* to *EdgeT*) and to blank nodes representing the complex table-attributes (of type *rdf:List*); the assignment of datatypes to simple values does not generate new triples, as RDF uses simple type annotations of literal values;
- *the assignment of values* to attributes (through *Attrs*) which may involve the generation of rdf:List structures to describe table-attributes, as already indicated;
- *the connectors* present in the model: a) connectors with editable attributes (*Relations* connecting two objects with the from/to pattern); b) relations without editable attributes (visible connectors, inter-model links and the *cv:contains* relation for visual containers-e.g., between a

$$MDE_1 = \left\{ \begin{array}{c} (ex{:}Sbseq1, cv{:}from, ex{:}MaintStart, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq1, cv{:}to, ex{:}Analyze, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq2, cv{:}from, ex{:}Analyze, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq2, cv{:}to, ex{:}Broken, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq3, cv{:}from, ex{:}Broken, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq3, cv{:}to, ex{:}Repair, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq4, cv{:}from, ex{:}Broken, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq4, cv{:}to, ex{:}MaintStop, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq5, cv{:}from, ex{:}Repair, ex{:}MyMaintenanceModel), \\ \left(ex{:}Sbseq3, cv{:}Transition\_Condition, ``Broken = Yes''\hat{cv}{:}String, ex{:}MyMaintenanceModel\right), \\ \left(ex{:}Sbseq4, cv{:}Transition\_Condition, ``Broken = No''''\hat{cv}{:}String, ex{:}MyMaintenanceModel\right), \\ (\boldsymbol{ex{:}Analyze}, \boldsymbol{cv{:}Assigned\_units}, \boldsymbol{ex{:}Expert}, \boldsymbol{ex{:}MyMaintenanceModel}), \\ (\boldsymbol{ex{:}Repair}, \boldsymbol{cv{:}Assigned\_units}, \boldsymbol{ex{:}Expert}, \boldsymbol{ex{:}MyMaintenanceModel}), \\ (ex{:}Expert, cv{:}described\_in, ex{:}MyOrganizationModel, ex{:}MyMaintenanceModel), \\ (ex{:}Expert, cv{:}described\_in, ex{:}MyOrganizationModel, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq1, rdf{:}type, cv{:}Subsequent, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq2, rdf{:}type, cv{:}Subsequent, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq3, rdf{:}type, cv{:}Subsequent, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq4, rdf{:}type, cv{:}Subsequent, ex{:}MyMaintenanceModel), \\ (ex{:}Sbseq5, rdf{:}type, cv{:}Subsequent, ex{:}MyMaintenanceModel), \\ (ex{:}Analyze, rdf{:}type, cv{:}Activity, ex{:}MyMaintenanceModel), \\ (ex{:}Repair, rdf{:}type, cv{:}Activity, ex{:}MyMaintenanceModel), \\ (ex{:}Broken, rdf{:}type, cv{:}Decision, ex{:}MyMaintenanceModel), \\ (ex{:}MaintStart, rdf{:}type, cv{:}Start, ex{:}MyMaintenanceModel), \\ (ex{:}MaintStop, rdf{:}type, cv{:}Stop, ex{:}MyMaintenanceModel), \end{array} \right\}$$

$$MDE_2 = \left\{ \begin{array}{c} (ex{:}Schubert, cv{:}Has\_position, ex{:}ITDpt, ex{:}MyOrganizationModel), \\ (ex{:}Schubert, cv{:}Acts\_in\_role, ex{:}Expert, ex{:}MyOrganizationModel), \\ (\boldsymbol{ex{:}Analyze}, \boldsymbol{cv{:}Assigned\_units}, \boldsymbol{ex{:}Expert}, \boldsymbol{ex{:}MyOrganizationModel}), \\ (\boldsymbol{ex{:}Repair}, \boldsymbol{cv{:}Assigned\_units}, \boldsymbol{ex{:}Expert}, \boldsymbol{ex{:}MyOrganizationModel}), \\ (ex{:}Analyze, cv{:}described\_in, ex{:}MyMaintenanceModel, ex{:}MyOrganizationModel), \\ (ex{:}Repair, cv{:}described\_in, ex{:}MyMaintenanceModel, ex{:}MyOrganizationModel), \\ (ex{:}Schubert, rdf{:}type, cv{:}Performer, ex{:}MyOrganizationModel), \\ (ex{:}ITDpt, rdf{:}type, cv{:}OrgUnit, ex{:}MyOrganizationModel), \\ (ex{:}Expert, rdf{:}type, cv{:}Role, ex{:}MyOrganizationModel) \end{array} \right\}$$

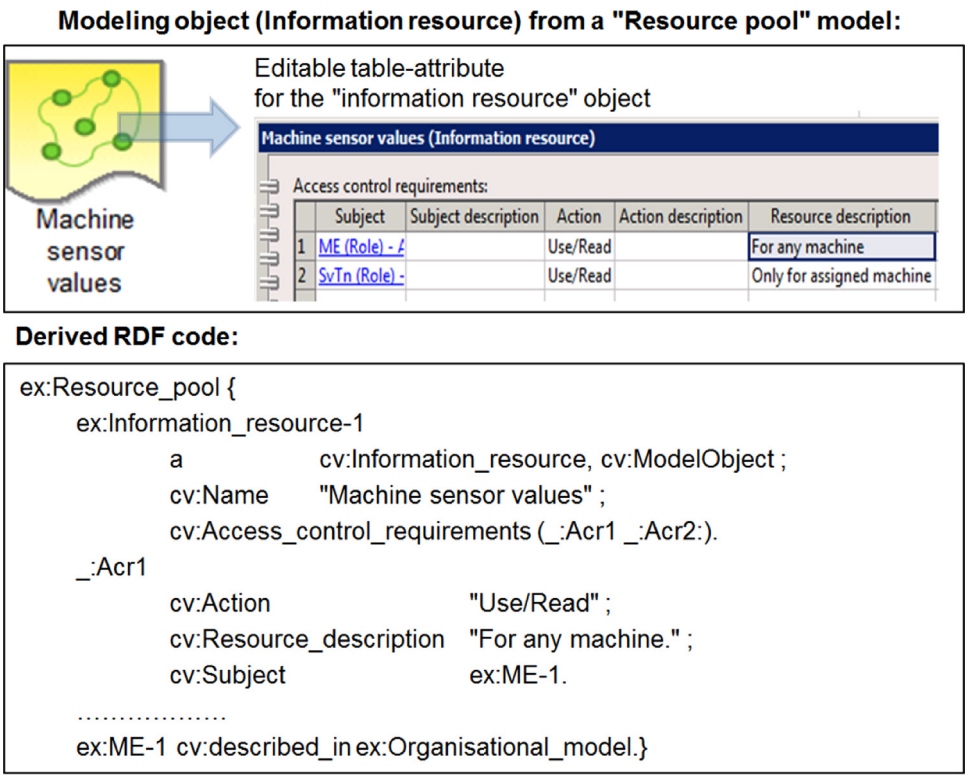

Fig. 5. RDF representation of a table-attribute.

process swimlane and the activities from that swimlane) and c) declarations of "model of origin" for foreign objects, through *cv:described_in*);

- *the custom properties* collected in the "Property collector" table-attributes.

It can be observed that the inter-model links (quads written in bold) are redundantly stored in both model graphs. This facilitates cross-model queries in both directions (e.g., to retrieve potential attributes of process activities starting from the assigned role and vice versa).

Some of the quads are explained here, from the context *ex:MyMaintenanceModel*:

ex:Sbseq2 cv:from ex:Broken; cv:to ex:Repair; cv:Transition_condition "Broken=yes"; rdf:type cv:Subsequent (an occurrence of a "subsequent arrow" treated as an n-ary relation that links two connected objects-here, a decision and an activity-and the "arrow attributes"-here, the transition condition; it is important to note that when relation attributes are irrelevant, a direct RDF relation can be derived at query time by applying a SPARQL CONSTRUCT on this from/to pattern).

ex:Expert cv:described_in ex: MyOrganizationModel (indicates that the current model has a link to a "foreign" object – *Expert*-which is native to the organizational model, thus all its properties should be retrieved from there).

ex:Repair cv:Assigned_units ex:Expert (the inter-model navigational capability of the modeling tool is captured as an RDF link; the chaining of links with the previously shown *cv:described_in* property allows for easily discerning links from visual connectors without attributes).

The namespace for the Linked Model Cluster should be edited by the modeler (prefixed here with *ex:*), to suggest the provenance of models and to assure URI uniqueness across multiple installations of the same modeling tool. The URIs of modeling elements are generated from their editable names and internal unique identifiers provided by the modeling tool (not visible here, for readability concerns). However, if the modeler is aware that a model element is reused (hence it is conceptually the same with one already provided by other modelers in a Linked Open Models cloud) the tool should provide the possibility of overriding the generated URI with a "sameAs link" (actually a subproperty of owl:sameAs) for each model and modeling construct. Use cases for this will be discussed in Section 4.1.

4. Implications

4.1. Use cases and benefits

In this section, several use cases are described, to highlight the envisioned benefits of serializing diagrams as Linked Open Models:

4.1.1. Model-to-model linking

Model sharing is a key requirement in some application domains. Frameworks from supply chain management (SCOR [31]) or virtual enterprise management [32] promote process visibility along the supply chain in order to facilitate a better collaboration between partners or an end-to-end process ownership approach. In SCOR, an

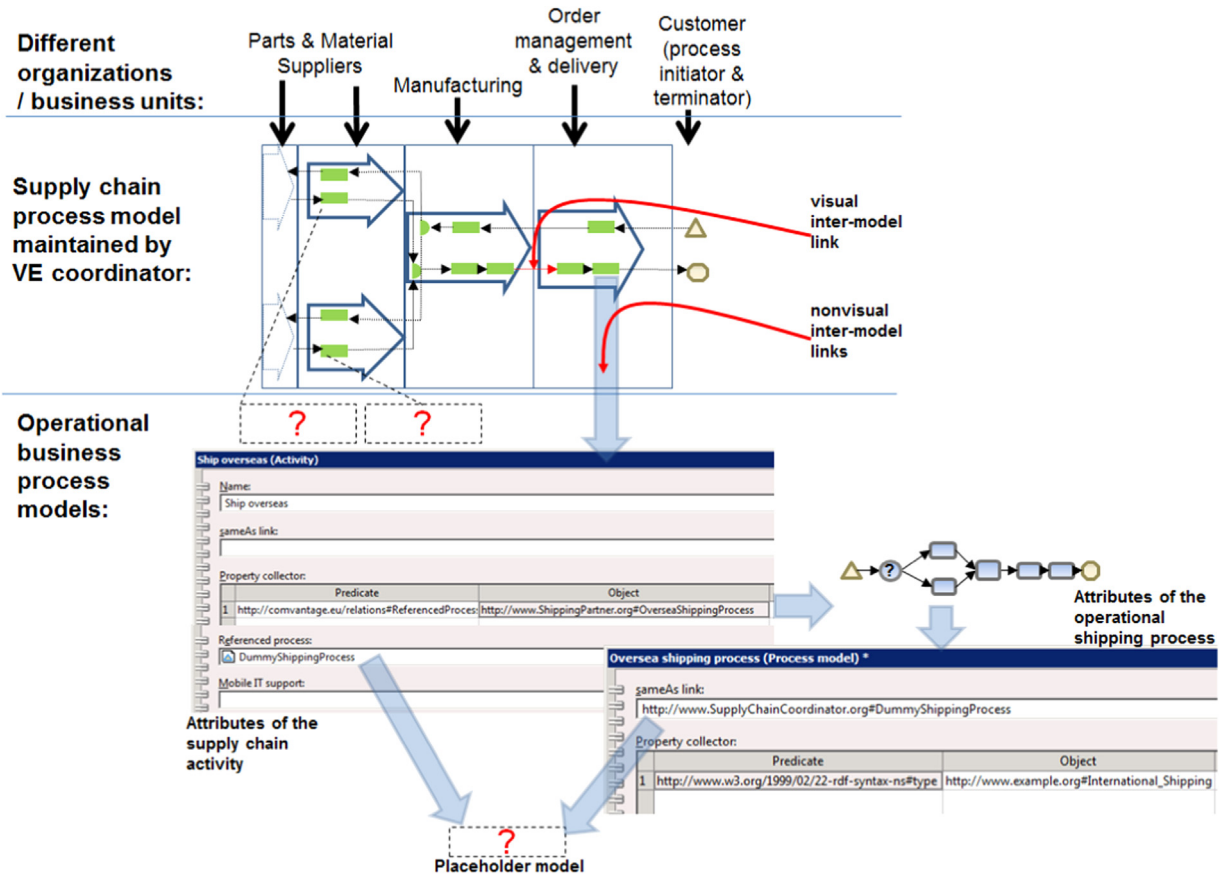

Fig. 6. Model linking in a supply chain modeling context.

organization can design a high level process of their supply chain as an orchestration of operational business processes and their responsible business units. Each operational business process is then detailed as a separate business process model, with its execution requirements. While the coordinating organization would be able to develop the high level supply chain process model, it might not have direct access to the lower level business process models of its supply chain partners (see Fig. 6). Or, a particular partner might not want to share a model immediately and fully, with all the attributes prescribed by the metamodel (e.g., internally estimated activity times and costs). The goal is to enable queries across models that are related, but have been created by the different organizations.

Fig. 6 depicts such a supply chain model mockup created by a virtual enterprise coordinator. Several vertical swimlanes represent different business roles, responsible for different supply chain steps. The organizations fulfilling those roles can (and must) further describe their operational business processes (according to the ComVantage modeling procedure, inspired by the SCOR approach). However, they may share these models with different strategies: some partners would share their full diagrammatic models; others would communicate the URI/URL where the model (with a filtered level of detail) was previously exported; finally, some would just delay the linking due to temporary unavailability of their models. In these scenarios, the nonvisual inter-model link from Fig. 6 (between the supply chain model and the operational business process model) can be created in multiple ways, depending on the situation:

- *Within the modeling tool*, if both the supply chain and the operational business process model are made available in diagrammatic form, in the same tool, and a hyperlink is created between them. The hyperlink between them becomes an RDF triple (*cv:ModelRelation_NA*) in the model graph, as previously suggested;
- *Between the modeling tool and the Linked Open Models cloud*, if the two models are created by different organizations, at different times, and cannot be open in the same modeling tool. In this scenario, the supply chain model will be linked to an empty placeholder process model (*DummyShippingProcess*) then, once the real one becomes available, the placeholder will be linked with the new model through its "sameAs link" attribute (available for all models and for all modeling objects). Another option is to use the property collector table, to create explicit RDF triples for the current modeling object. In Fig. 6 the property collector table is used twice: to link the *Ship overseas* supply chain activity to a (known) shipping process model (*Oversea shipping*

process), and to assign a custom type (*international shipping*, which is not prescribed by the metamodel) to the shipping process;
- *Directly in the Linked Data cloud*, through common data linking practices, as long as all URIs are known.

Obviously, the second and third approaches would bypass all constraints enforced through the modeling tool by the language syntax (e.g., domain, range, cardinality of relations). As long as the linking is performed within the modeling tool, there will be a guarantee of consistency, to the extent provided by the constraint-checking mechanisms of the modeling method. Otherwise, the Linked Open Models rely on the same "anyone can say anything about anything" principle [33] as the Semantic Web paradigm (and the Web as a whole), meaning that publishers have a default interest in assuring the provision of quality content and data, having at stake their reputation as a model source. Eventually, an import mechanism can be implemented to bring Linked Open Models back in a modeling tool, in order to perform validation checks against the tool-level constraints.

The URIs to be used for overriding/direct linking can be obtained in various ways: (a) exchanged directly between partners; (b) shared in a Sindice-type [34] repository of model URIs; (c) discovered by existing link discovery technology (SILK [35]); (d) discovered by queries on the partner organization's endpoint (all URIs have, as base, the domain URL of the organization producing them, the model types URIs are known from the shared Linked Metamodel and everyone can find models involving themselves as a modeling object – usually a swimlane, a business role etc.).

Finally, the visual inter-model link in Fig. 6 is actually a modeling relation (a visible connector/arrow) from the supply chain model. It will bridge queries across lower level (operational) models – e.g., for requesting estimated times from the upstream or downstream of the supply chain, relative to a selected supply chain step.

4.1.2. Model-to-data linking

This use case assumes that execution-time data is available and lifted semantically from legacy systems. Instruments for achieving this are available both at tool level (D2RQ [36]) and at methodological level (the RDB2RDF methodology for converting relational databases to Linked Data [30]).

For exemplification, we consider that the RDF data in Fig. 7 has been lifted from a relational database or log files. The left side graph contains human resources records, the right side contains activity costs recorded for multiple executions (namespaces are avoided for brevity).

We add the assumption that both the ERP/Logfile and the model use the same names for activities and

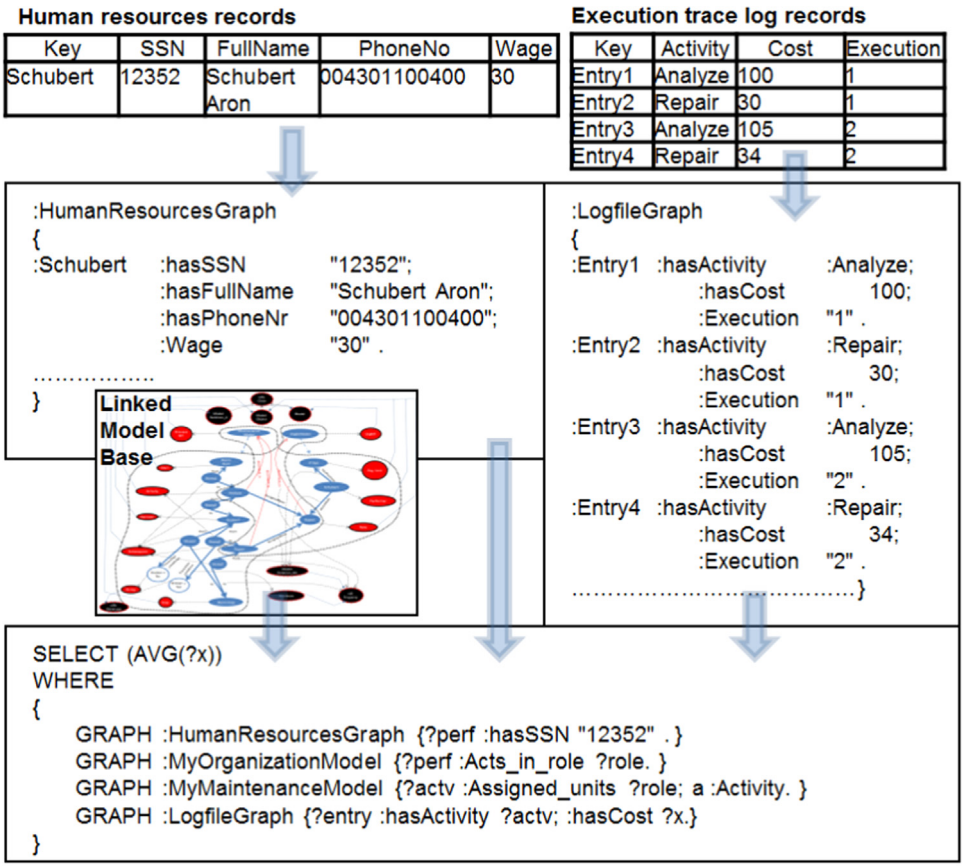

Fig. 7. Cross-querying of legacy data, execution data and Linked Open Models.

employees. We consider the assumption realistic, since both the legacy data structure and the models can be created in the same organization, having identifiers shared through an internal nomenclature or imported from an indexed repository. If not, an additional URI alignment ("sameAs" links) would have to be maintained, and mapping patterns would have to be included in the queries to make use of the alignment. Tools like SILK [35] derive such alignments using manually maintained linkage rules based on various similarity metrics.

When data is stored in the same RDF repository as the Linked Model Base, it becomes possible to formulate queries that retrieve (a) data based on relations that are described by the diagrammatic model (but not reflected in the legacy data schema); or (b) model elements, based on execution data constraints. The query in Fig. 7 belongs to the first category: it returns the average cost of activities assigned (through the model) to an employee, knowing its SSN. Hence, the Linked Models can expose relations that cannot be retrieved from the legacy data models, but are present in diagrammatic models, consequently gluing disparate data sources.

4.1.3. Rule-based model transformation

A key challenge in the ComVantage project was the deployment of orchestrated mobile apps [21] based on process-centric modeled requirements. The app orchestration engine takes input from models and deploys a flow of chained mobile apps for each particular user role, as derived from the modeled business processes. In other words, instead of orchestrating web services in the spirit proposed by BPEL [37], ComVantage orchestrates mobile apps by extending app chaining techniques [38] to a business process-driven approach.

In order to achieve this, app orchestration models are derived by employing graph transformation techniques on business process models linked to app requirements. This has been formalized through graph rewriting rules, with an algorithmic implementation available in the modeling prototype. The Linked Open Models approach opens new possibilities of externalizing this effort to a Linked Data cloud. Graph rewriting rules are, after all, production rules, and production rules on Linked Data can be expressed at query-time with graph generation queries (CONSTRUCT) or graph altering queries (INSERT/DELETE) using the SPARQL language. Thus, it is only natural for such model-to-model transformation mechanisms to be "delegated" to the Linked Data cloud, taking off some of the processing load from the client modeling tool.

Fig. 8 shows the iterative transformation steps necessary to derive the usage precedence flow of mobile apps (node labels are not meant to be readable, only the structural changes in the model, reflecting the sequence of graph transformations). The top-left panel shows the process model in the custom notation provided by the ComVantage modeling method: it contains a parallel (AND) split, a decision (XOR) split and some of the activities have app objects assigned to them as requirements (see legend also). The bottom right panel shows the final output, where only the app objects remain, with their precedence relations ("FollowedBy") derived from the business process control flow. The intermediate steps show the bypassing and then gradual removal of original process elements and relations, with each step being an actual graph rewriting operation.

By querying the Linked Data representation of the output model, an orchestration engine can deploy chained apps corresponding to the model nodes, to meet the needs of the business logic along the original process. It should be noted that the algorithm is generally reusable to any type of resources assigned to the input process model (to reflect, for example, the precedence of involving human resources, information resources etc.)

The work at hand does not have in scope further discussion on this mechanism, its use cases and graph rewriting formalism (the rules can be consulted in [4, deliverable D312]) but rather what the Linked Open Models brings to such a scenario. Production rules can be emulated with SPARQL queries to perform such a model transformation outside the modeling tool. We only provide here (in Table 2) a set of SPARQL graph editing queries that can perform the transformation described in Fig. 8. To keep the example easy to grasp, some simplifying assumptions are applied:

- It is assumed that a direct RDF predicate (cv:SR) is used between process elements. As previously discussed, the actual export mechanism generates a cv:from/cv:to pattern from those visual connectors, allowing for each occurrence of the visual connector to get its own properties. It is trivial to derive the direct cv:SR predicate from every occurrence of the cv:from/cv:to pattern, in order to obtain the structure on which these example queries run;
- The queries are also kept simple by avoiding (in most cases) type (class)-checking for the RDF nodes, since the modeling tool guarantees through its metamodel definition that, for example, the cv:MS relation only exists between a process element and an app object (the same applies to other relations).

For better understanding, the legend in Fig. 8 also contains the prefixed URIs and SPARQL variables used in the queries, mapped on their corresponding notations. Once the RDF representation of the orchestration model is produced, an orchestration engine can query it to deploy apps according to the flow dictated by the model.

4.1.4. Models as shared vocabularies

The structural flexibility envisioned for the Semantic Web can be transferred to Linked Open Models acting as *controlled open vocabularies*. Reference frameworks for specialized communities often define taxonomies that are, ultimately, shared vocabularies ready to be transferred in the Linked Data environment. In the context of the ComVantage research project, the SCOR framework is particularly relevant, as it defines a standard taxonomy of process categories and process identifiers to be used when structuring and modeling supply chains. Supply chain modeling tools sometimes try to comply to the SCOR framework by hardcoding the SCOR concepts as first class modeling objects [39]. However, such vocabularies also need to

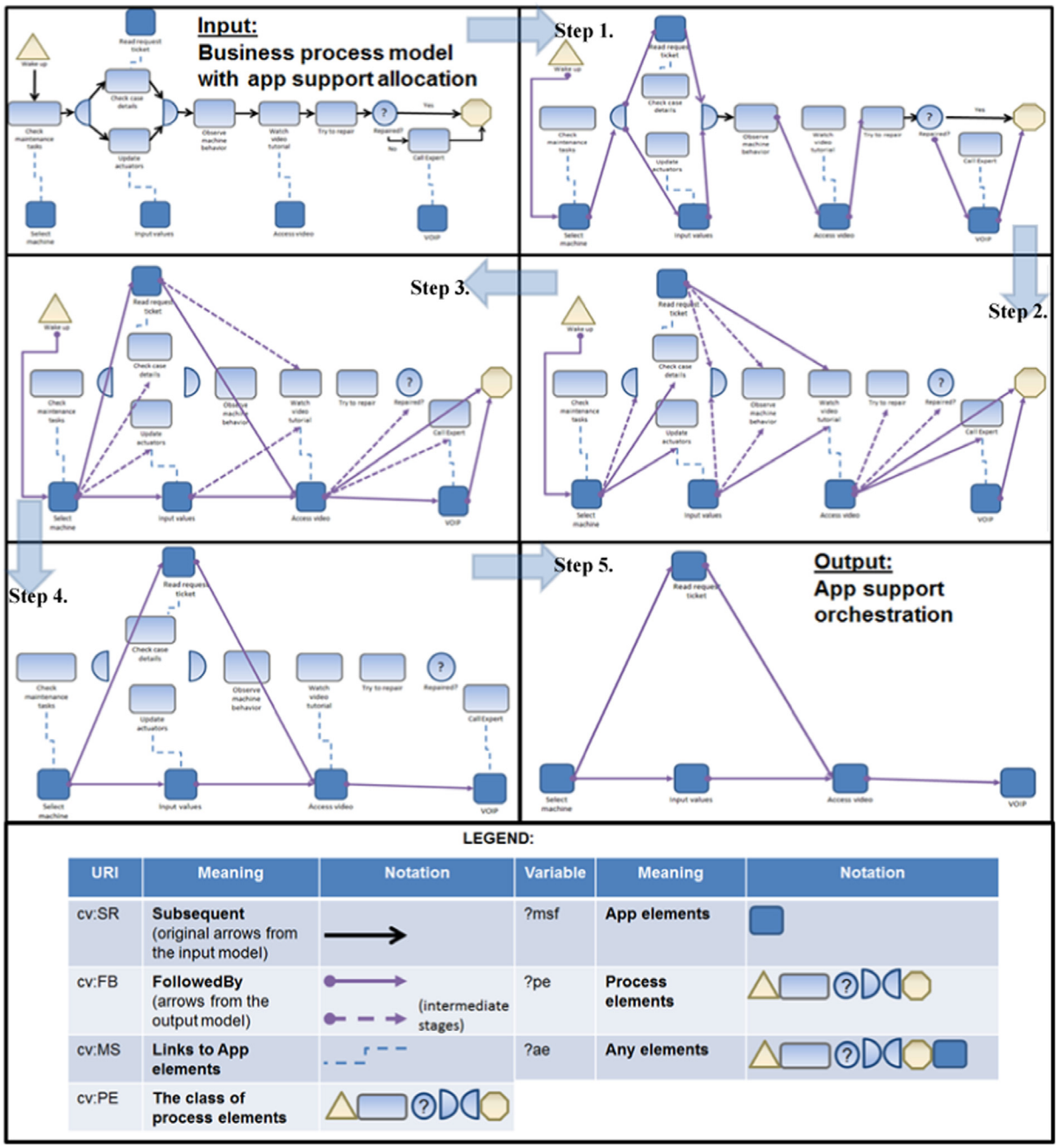

Fig. 8. The derivation of orchestration models from business process models: a diagrammatic view.

evolve. For nonstandard vocabularies, evolution has rather short iterations and is driven by change requests and refinements emerging from their user community. Even for standard taxonomies there is an evolution cycle that is controlled and versioned by some responsible body (the Supply Chain Council, in the SCOR case). In both situations, hard-coding controlled identifiers may prove too rigid, therefore a solution based on pool models is proposed in the context of the work at hand.

Pool models are visual repositories of modeling objects intended for reuse, possibly grouped in a hierarchy. *Reusing* can mean, on one hand, having in different models various objects that represent the same thing (for this, the previously discussed sameAs links can state their equivalence across models). However, it is often preferable (for usability reasons, but not only), to keep the reused objects in one place (the pool model) and to reference them from multiple diagrams. Depending on the meaning attributed to the reference link, one can achieve an actual "object reuse" across multiple models, or just multiple associations with the same target object. If the hyperlink signifies the *of type* relation, then the pool model acts as a

Table 2
The derivation of orchestration models from business process models: a SPARQL-based approach.

Step 1a.

```
DELETE {
?ae1 cv:SR ?pe2.
?ae1 cv:FB ?pe2.
}
INSERT {
?ae1 cv:FB ?msf3.
}
WHERE {
{ ?ae1 cv:SR ?pe2. }
UNION
{ ?ae1 cv:FB ?pe2. }
?pe2 cv:MS ?msf3.
}
```

Step 1b.

```
DELETE {
?pe1 cv:SR ?ae2.
?pe1 cv:FB ?ae2.
}
INSERT {
?msf3 cv:FB ?ae2.
}
WHERE {
{ ?pe1 cv:SR ?ae2. }
UNION
{ ?pe1 cv:FB ?ae2. }
?pe1 cv:MS ?msf3.
}
```

Step 2a.

```
DELETE {
?pe2 cv:SR ?ae4.
?pe2 cv:FB ?ae4.
}
INSERT {
?ae1 cv:FB ?ae4.
}
WHERE {
{ ?ae1 cv:SR ?pe2. }
UNION
{ ?ae1 cv:FB ?pe2. }
{ ?pe2 cv:SR ?ae3. }
UNION
{ ?pe2 cv:FB ?ae3. }
{ ?pe2 cv:SR ?ae4. }
UNION
{ ?pe2 cv:FB ?ae4. }
?pe2 rdf:type cv:PE.
FILTER (?ae3 != ?ae4).
}
```

Step 2b.

```
DELETE {
?ae2 cv:SR ?pe3.
?ae2 cv:FB ?pe3.
}
INSERT {
?ae2 cv:FB ?ae4.
}
WHERE {
{ ?ae1 cv:SR ?pe3. }
UNION
{ ?ae1 cv:FB ?pe3. }
{ ?ae2 cv:SR ?pe3. }
UNION
{ ?ae2 cv:FB ?pe3. }
{ ?pe3 cv:SR ?ae4. }
UNION
{ ?pe3 cv:FB ?ae4. }
?pe3 rdf:type cv:PE.
FILTER (?ae1 != ?ae2).
}
```

Step 3.

```
DELETE {
?ae1 cv:SR ?pe2.
?ae1 cv:FB ?pe2.
?pe2 cv:SR ?ae3.
?pe2 cv:FB ?ae3.
}
INSERT {
?ae1 cv:FB ?ae3.
}
WHERE {
{ ?ae1 cv:SR ?pe2. }
UNION
{ ?ae1 cv:FB ?pe2. }
{ ?pe2 cv:SR ?ae3. }
UNION
{ ?pe2 cv:FB ?ae3. }
?pe2 rdf:type cv:PE.
}
```

Step 4a.

```
DELETE {
?ae1 cv:SR ?pe2.
?ae1 cv:FB ?pe2.
}
WHERE {
{ ?ae1 cv:SR ?pe2. }
UNION
{ ?ae1 cv:FB ?pe2. }
?pe2 rdf:type cv:PE.
}
```

Step 4b.

```
DELETE {
?pe1 cv:SR ?ae2.
?pe1 cv:FB ?ae2.
}
WHERE {
{ ?pe1 cv:SR ?ae2. }
UNION
{ ?pe1 cv:FB ?ae2. }
?pe1 rdf:type cv:PE.
}
```

Step 5.

```
DELETE {
?x ?y ?pe1.
}
WHERE {
?x ?y ?pe1.
?pe1 rdf:type cv:PE.
}

DELETE {
?pe1 ?x ?y.
}
WHERE {
?pe1 ?x ?y.
?pe1 rdf:type cv:PE.
}
```

taxonomy of types, hence a controlled vocabulary for elements that are present in other diagrams.

Vocabulary extensions can be performed either by a) *creating new categories* (as modeling objects) in the pool models, or by b) *filling their property collector tables* (thus extending semantics on the fly, beyond the metamodel of the tool prescribes). Such pool models may be shared within their user community as controlled vocabularies, since each of their elements has its own URI. Linking would be performed as discussed before, either visually, in the imported pool model, or directly to its Linked Open Model version. In Fig. 9, two such pool models are highlighted:

a. *A common pool of apps* is reused to describe multiple app orchestration models (in the sense described in use case C.). This would not qualify as a "vocabulary", but rather as a catalog of resources (apps) to be reused in multiple models, through the "sameAs links" (other examples can be imagined from the supply chain contexts, such as reusing a catalog of candidate business partners in multiple interorganizational business processes). Property collector tables allow the unconstrained extension of the app descriptions, beyond what the metamodel prescribes, thus allowing the modeler to improvise a metadata vocabulary for experimentation purposes, without touching the language metamodel (and having the modeling tool reimplemented). Further on, each app is linked to more detailed app models (UI structure) and a set of capabilities (acting as requirements) collected in the second shared pool model:
b. *The capability pool* is closer to the notion of a "shared vocabulary", as it aims to collect and establish a taxonomy of capabilities that can describe an app. Each capability has its own generic property collector, enabling the creation of RDF links to other models, or new properties (without extending the modeling tool implementation).

Although these pool models are weakly constrained by the metamodel (and the property collector attributes not at all), they are particularly useful to communicate iterations of a shared vocabulary until it gets stabilized. Thus

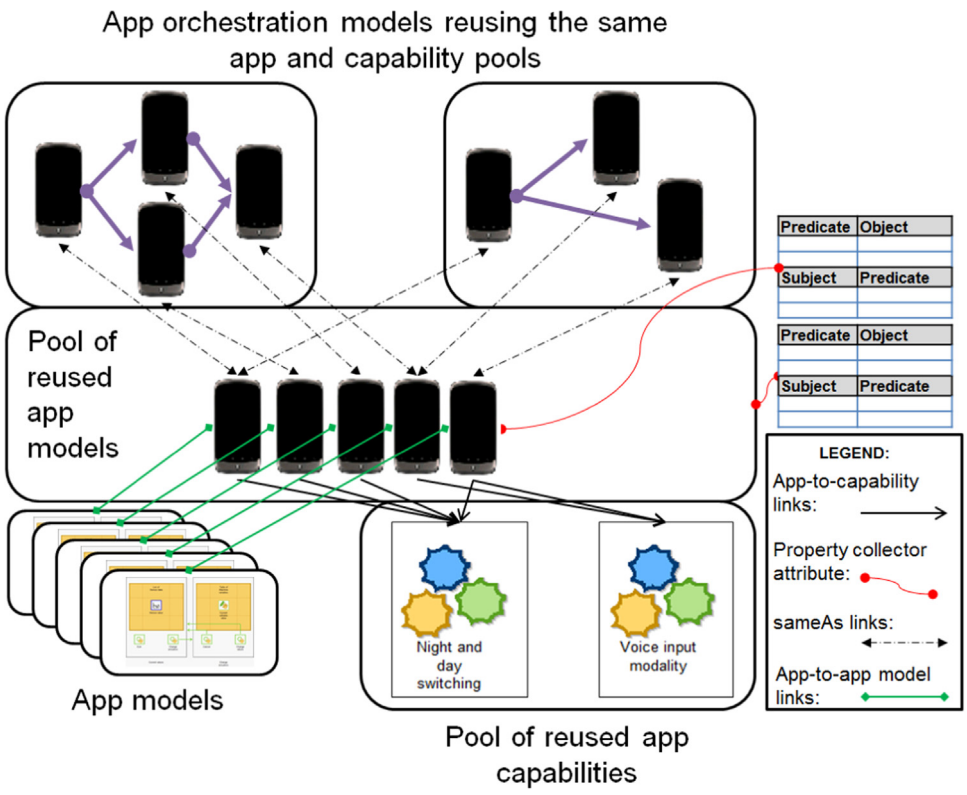

Fig. 9. Pool models employed as nonstandard vocabularies of reusable resources.

they will extend the metamodel in an ad-hoc way, as well as the sensitivity of model-aware run-time components. As pool models or property collectors are adopted and stabilized, they can be assimilated in the metamodel (as prescribed first order modeling citizens), but this is not mandatory: there is also value in just keeping them as shared vocabularies, possibly enhanced with some documenting annotations. Of course, with controlled shared vocabularies, there's always the challenge of managing evolution, versioning and preserving persistent linking. This is a general problem of Linked Data and ontology evolution (some specificity regarding Linked Open Models will be suggested in the final SWOT evaluation).

4.2. Discussion on the generality level

Both the modeling method and the model export mechanism presented here have been tailored on the domain specificity of the ComVantage research project. However, the designs are aimed to be reusable and extensible beyond the project goals. On one hand, the ComVantage domain is not narrow, but rather multi-faceted. Its modeling method supports the decomposition of a high-level business view across multiple levels of abstraction: from the top-level business model down to requirements that can be passed towards the run-time side. The discussed app orchestration can be generalized to any type of resource assigned to business process elements, to capture the usage precedence for that resource type. Delving further into the domain specific semantics of the model stack itself is not in the scope of this paper. We will rather discuss the Linked Open Models concept, as instantiated by the approach described here, and some directions for improving its global value:

4.2.1. Further relation specializations

The Linked Model Vocabulary relies on the object-relation dichotomy, to which common metamodeling platforms (investigated based on the overview provided by [40]) can be mapped. For example, MetaEdit+ [41,42] works with a similar dichotomy but calls its constructs "properties" and "non-properties", while Visio [43] deals with "properties" and "stencils". Various meta^{2}models add the primitives of "roles" and "ports", which can be expressed in relations of arity higher than 2, and these can be further decomposed in configurations of binary relation sets (as RDF does with reification or ternary relations). As [40] concludes, all the investigated meta^{2}models can, ultimately, be mapped to the core primitives of objects, relations and attributes, hence we chose to position the Linked Model Vocabulary on this level, with some minimal additions that are still highly reusable or extensible: the relation typing, the from/to pattern (allowing the possibility of having attributes in visual connectors, not only in modeling objects) and some special properties (for visual containment and foreign object membership).

The visual containment relation (*cv:contains*) is generated between containers and contained elements based on relative position and size for the visual grouping constructs (e.g., swimlanes in business process models). It can

be generalized to a "notationally-induced relation" property class that would include, for example: a *proximity* relation, generated whenever two objects are positioned close enough (e.g., enabling grouping without any visible containment); a *similarity* relation, whenever two objects have similar attribute values based on some chosen similarity metric (e.g., process activities involving the same role). Therefore, it would make sense to designate subclasses of *cv:ModelRelation_NA* for such relations that are derivable from the visual similarity or appearance of modeling elements.

The foreign object membership (*cv:described_in*) is not strictly necessary, as foreign objects can be retrieved with some relatively simple queries, by looking for objects that occur in multiple named graphs (the model graph where they have attributes is the "originating model", while for the other models they are "foreign"). However, this would be a "searching query", which means looking up multiple graphs and testing for absence of attributes (something that is undesirable in an Open World context). Since inter-model linking is the core benefit of Linked Open Models, and it is involved in the majority of use cases, storing explicitly such a relation greatly reduces querying effort. This is also the reason why the Linked Data Vocabulary does not specialize *cv:ModelRelation_NA* to distinguish hyperlinks from visual connectors. The presence of *cv:described_in* will detect when an RDF object is involved in a visible connector or when it is the target of a hyperlink (anyway, the distinction loses its value outside of the modeling tool, where notation and usability become irrelevant). Such a specialization can extend the Linked Model Vocabulary, if deemed relevant enough. For example, the version of the ADOxx metamodeling platform used in our experimentation only allows hyperlinks whose source anchor is a modeling object (and not relations, nor whole models). Platforms that support more flexible linking would benefit from further subclassing *cv:ModelRelation_NA* in this direction.

In any of these cases, extending the Linked Model Vocabulary for additional needs fits the Linked Data philosophy, as the schema itself is an RDF graph. Or, different variants of the Linked Model Vocabulary may coexist in a manner not unlike the coexistence of other schemata with overlapping semantics (e.g., SKOS [44] and RDF Schema [45]) – that is, if notions like stencil, nonproperty, port, role are considered relevant enough to be explicitly provided as specialized concepts to SPARQL queries.

4.2.2. Notation independence

Linked Open Models can be used to transfer model structure and contents between different compatible notations. The metamodeling framework promotes the concept of "modeling method engineering", for developing methods customized with domain-specificity and requirements for specialized communities. As a consequence, the core business process notation is non-standard, but its semantic backbone captures commonalities of swimlane-based control flow languages (see Fig. 2). The RDF serialization of the models is notation-agnostic, hence it can be imported/queried by other modeling tools to generate models in standard or preferred alternative notations. This can be done either (a) by a direct mapping, if the metamodel is similar and only the notation differs (canvas position attributes can also be exported to better support such a transfer); or (b) by graph transformations in the fashion suggested by the app orchestration use case from Section 4.1.3, if the input Linked Open Models have sufficient detail but cannot be mapped on the target notation.

4.2.3. Complex attribute values

In some modeling tools the model elements have complex attributes that can be edited in tabular forms. If attribute values of higher complexity (than bidimensional tables) are needed, the metamodeler should consider defusing this complexity by converting part of it in new entities to be added to the modeling language syntax (therefore to be used on the canvas rather than to be described in complex data forms). We have already presented the default way in which the ComVantage method exports tabular attributes (Fig. 5). In the general case, additional situations can be identified with respect to semantics, where the RDF representation can be made less cumbersome than an rdf:List structure, hence closer to the intended semantics and more intuitive for querying. Such situations are presented in Fig. 10. They rely on some naming conventions (in the leftmost column) used by the metamodeler to suggest the intended semantics, hence determining different output structures. The naming conventions will extend the Linked Model Vocabulary with new types of relations, to cover these cases.

The examples in Fig. 10 assume that there is a modeling object (on the canvas) called "source", whose property sheet contains a table-attribute visible in the second column. The attribute name (with a proposed naming convention to suggest intended use) is provided in the leftmost column.

- A table-attribute called "property_collector" allows the arbitrary generation of triples in the model graph, bypassing any metamodel constraints; the object having this attribute may take the role of subject or object in the generated triples, if the corresponding table cells are left empty. There is no relation or dependency between the URIs used in a property collector and other models;
- A table-attribute having the suffix "objcollector" will interpret each record from the table as a "non-canvas modeling object", and each field of the table as an attribute of such an object. Therefore, the table can be seen as a collection of objects left out from the visual language syntax, which can still be semantically defined and associated to the visual source object. Optionally, a key field can be added to the table, to serve as an identifier for such "non-canvas objects" (for reuse purposes). The table fields may contain literal values, but also hyperlinks to objects from other models, thus the non-canvas object can have both incoming and outgoing relations with canvas objects. In the rightmost column of Fig. 10 the non-canvas objects are exported as blank nodes;
- A table-attribute having the suffix "relcollector" will interpret each record from the table as a "non-canvas

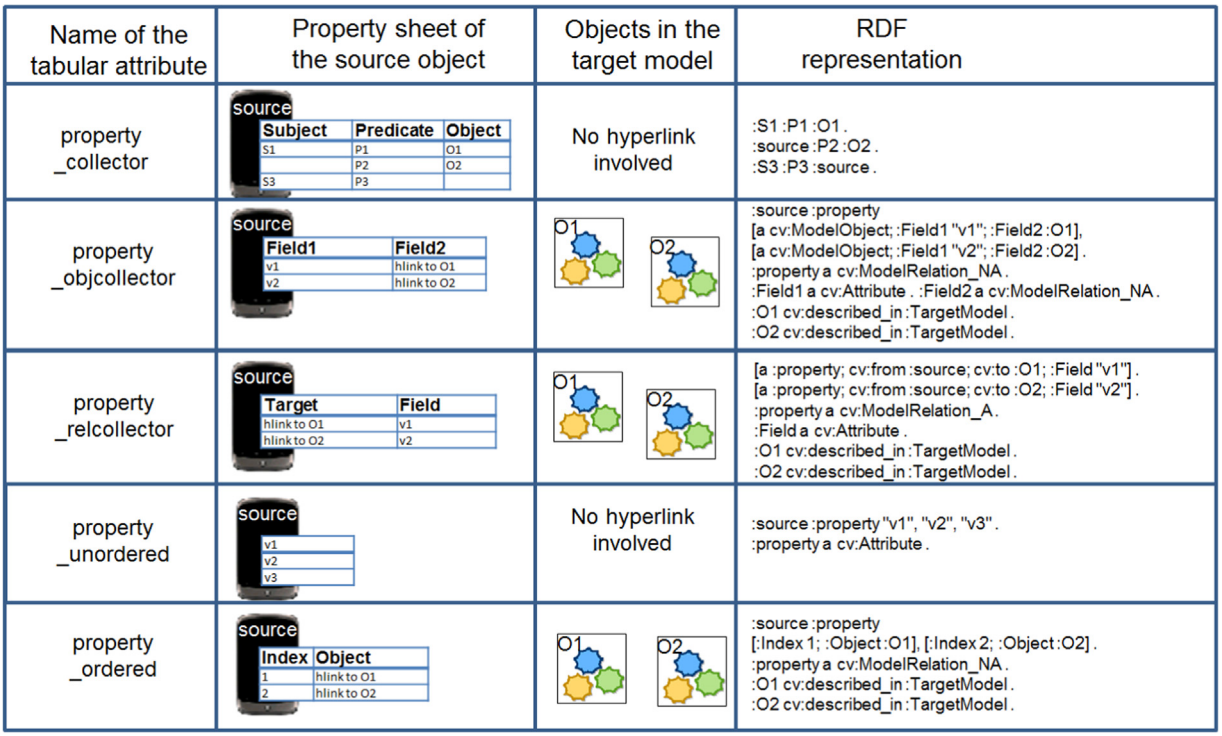

Fig. 10. RDF serializations for different complex attribute patterns.

modeling relation with attributes". One field in the table will indicate (by naming convention) the Target of the relation (hyperlink to some other modeling objects), while the other fields will become the attributes of the relation. In the export column it can be seen that the from/to pattern must be applied, since we have a modeling relation whose every occurrence should be distinguishable and enriched with attribute values;

- A single-field table-attribute (an unordered list, indicated by the "unordered" suffix) will become a modeling attribute (if it contains literal values) or a modeling relation without attributes (if it contains hyperlinks to other objects), having all the values from the list as RDF objects;
- An ordered single-field table-attribute (a table with an index column and a content column) will preserve the order by annotating each item with the index property (similar to RDF containers). Ordering may be similarly applied to the previous cases.

5. Related works

The related works mentioned in this section converge from different fields: (i) the *relation between metamodels and ontologies* inspired the work on conceptual level; (ii) the *existing diagram export formats* inspired the technical aspects regarding model serialization; (iii) the paradigm of *process-aware information systems* motivates some of the use cases. The contribution relative to this body of work will be emphasized in the conclusive Section 6.1.

Conceptual models are constrained by metamodels, while Linked Data is structured by ontologies (typically from the weaker end of the semantic spectrum). Metamodeling and ontology engineering have different origins: the first was motivated by model-driven software engineering (stimulated by communities such as OMG [46]), whereas the second emerged from artificial intelligence. As commonalities became obvious, authors such as [47] have provided formal mappings between the two notions. In [48], a model is defined as "an abstraction of reality according to a certain conceptualization" (which is the metamodel). In [49], ontologies are defined as formal explicit specifications of shared conceptualizations. The commonality is obvious: both are conceptualizations. With ontologies, pragmatic emphasis is placed on being "shared". With models, emphasis falls on "according to", suggesting an enforced compliance to the metamodel. Ontology sharing can be understood both as a common conceptual view ("shared understanding"), but also on a technical level ("shared information", as they can be distributed in RDF format). Metamodels also rely on common conceptual patterns identified in the domain, but they are rather aimed to educate users in structuring and communicating domain knowledge with a limited set of visual constructs, having specific semantics. In [50], the notion of *model conformance* (validating a model against a metamodel) is differentiated from *instantiation* (deriving a model from a metamodel); while in Semantic Web a third mechanism comes into focus: instances are assigned to a class via *classification* reasoning (assigning a model to a metamodel, based on its properties and usage). Authors of [51] also observe the different origins in ontology-related

and model-related studies, adding that models are focused on realization (they are *prescriptive*), and ontologies on queries and reasoning (they are *descriptive*). Both ontologies and metamodels have the common goal of *answering competency questions*. With ontologies, this relies on discovery of knowledge implied through some rule base. With diagrammatic models, the aim is to generate analysis reports that are comprehensive for the problem under scrutiny, in order to support model-based decision making. However, these goals are interchangeable. From an end-user perspective, ontologies and Linked Data graphs are expressed in some serialized machine-readable format and their visualization is of secondary interest; however, recent research challenges strive to provide visual ways for designing them [52]. On the other hand, diagrammatic modeling brings forth concerns for usability and the quality of diagrammatic notation [53]; still, semantic integration and inferences across multiple layers of abstraction is equally relevant [54]. Important work has been done by OMG in order to bridge ontology engineering and the MOF metamodeling approach (the Ontology Definition Metamodel [55]), with a focus on ontology derivation from UML models. Clearly, there is growing interest in bridging the paradigms of metamodeling and Semantic Web, on the common research ground of *knowledge representation*.

Related works regarding *model interoperability* are available as various model serialization formats provided by different modeling tools. Most of them are rather tool-specific (Visio formats [43], ADONIS ADL [24]), or domain-specific (XPDL [56] or BPEL [37] for business process models). A general purpose model and metadata exchange format is the OMG standard XMI [57]-an XML vocabulary most often used in model-driven engineering, as a medium between UML modeling tools and code generators.

The motivation and benefits of the work have emerged from the paradigm of *process-aware information systems*, as defined in [58]: "a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models". Such systems are presented as an advancement on the traditional (mostly relational) *schema-aware systems*, generating advantages such as: human communication support through models, better flexibility when dealing with changes, automated enactment of business processes, process monitoring and mining opportunities [11]. The Linked Open Models vision promotes domain-specific model-awareness in information systems, by employing the knowledge captured in diagrammatic form to semantically enrich execution time data (e.g., process traces). Consequently, the work is also related to the emerging *Models@runtime* paradigm [59], for which it provides certain enablers pertaining to semantic linking, enrichment of models and run-time/design-time bridging. In the literature on semantic enrichment we can identify a strong bias towards enriching model information with (meta) data [60], while the use cases discussed here aim towards enriching data with model semantics.

6. Concluding discussion and evaluation

6.1. Contribution summary

Compared to the baseline given by the related works, the focus of the work at hand is placed on a Linked Data-compliant serialization of diagrammatic conceptual models. The benefits investigated in the context of the ComVantage research project have been generalized here to formulate a coherent Linked Open Models vision that may stimulate a convergence of concerns from different areas pertaining to the engineering of semantic information systems. As the proposed model serialization covers (in a uniform query-able representation) multiple abstraction layers-meta-metamodel, metamodel, models and their metadata, execution-time instance data -, it aims towards a multi-abstraction integration similar to [54], however in a manner that is interoperable with a Linked Open Data environment, thus open to a wide array of semantic processing possibilities. Exported diagrammatic models can be employed for various purposes, opening new insights on the relation between metamodels and ontologies, especially as they do not necessarily represent the same MOF layer of abstraction (e.g., instance models can become vocabularies for execution data or other models).

The proposed vision of Linked Open Models recommends an "integrated separation of concerns": models can be filtered by restrictive checks (compliance against the metamodel, enforced by a modeling tool) before being released and processed in the Web of Data. Compared to the ODM [54] proposal, the work presented here targets the currently developing Web of Data rather than a fully-fledged Semantic Web, and relies on domain-specific modeling languages to capture the knowledge of business stakeholders in an intuitive and easy-to-educate manner. It also generalizes the notion of "models as ontologies" beyond the scope of UML, as the use case from Section 4.1.4 suggests.

Compared to popular model export formats, the work at hand does not target code generation, but rather the lifting of diagram semantics in a Linked Data environment, potentially fueling a new generation of information systems: *model-aware information systems* can be considered a superclass for both systems with process-awareness and traditional schema-awareness (business process and ER diagrams are, ultimately, conceptual models). The indicated advantages of process-awareness can thus be generalized for domain specific model-awareness. In the context of the ComVantage research project, model-awareness was implemented in an app orchestration engine [21] which helps business stakeholders to redeploy app chains according to the changes made during the redesign of business process models (using orchestration models as intermediary knowledge structures). However, model-awareness can manifest in many other different ways, depending on the types of relations expressed in models. Additionally, unlike various XML export formats, the RDF serialization of models is not intended to be transferred between two parsers, but to become an integral part of the Web of Data, to enrich Linked Data with

user-created semantics extracted from user-friendly, domain-specific modeling tools.

To conclude, the work at hand tackled the key requirements for Linked Open Models by analogy with Linked Open Data (Table 1). The requirements were extended with use cases such as (a) linking inter-organizational models; (b) enabling model-awareness; (c) enriching legacy data queries with model-based constraints; (d) enabling model processing in Linked Data clouds.

6.2. SWOT evaluation and outlook

A proof-of-concept modeling tool has been iteratively implemented for the ComVantage modeling method. It is hosted in the Open Model Initiative Laboratory portal, open on a registration basis to the members of the ComVantage OMILab space [17] and other interested users. The prototype (Fig. 11) was used to support project progress, particularly for experimentation on key features such as model-awareness in run-time components. Extensive usage guidelines, as well as concrete examples of queries over domain-specific Linked Models are available in the modeling guidelines documentation also available at [17].

The feasibility and performance of remote model querying has been tested using a Sesame triplestore [61] (on a i5@2.6 Ghz with 8 GB RAM) and a query client running the Sesame query-over-HTTP protocol [62]. Queries have been run over HTTP on models that reflect project use cases, as well as on improvised models aimed to challenge scalability. The diagram export component (top left side of Fig. 11) enables the user to filter the model information to be exposed as Linked Data (that is, the properties of model elements). A generic query client (bottom-right side of Fig. 11) was used to test queries over POST requests to a RESTful endpoint of choice.

A common query pattern used for testing is highlighted in Fig. 12: starting from a key activity from a business process model, all resources used downstream from that activity are retrieved (e.g., required human roles from organizational models), together with other objects from the resources' context (e.g., departments and companies providing those roles), and finally a particular property of these contextual objects is retrieved (e.g., location). The full property sheets of model elements were exported as Linked Data, although for practical needs this is usually filtered down to some key attributes, significantly lowering the number of generated quads in the output hypergraph.

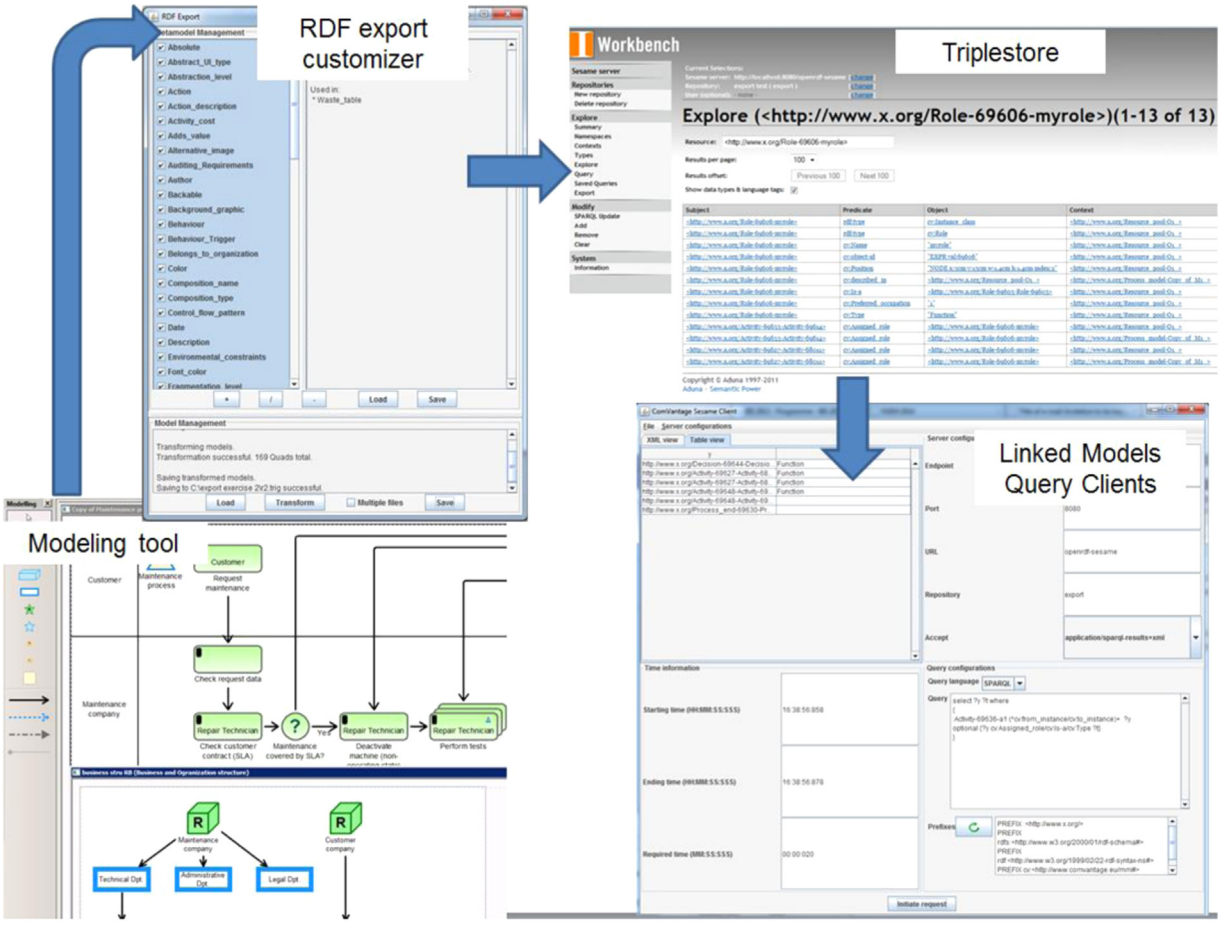

Fig. 11. Components involved in the proof-of-concept implementation.

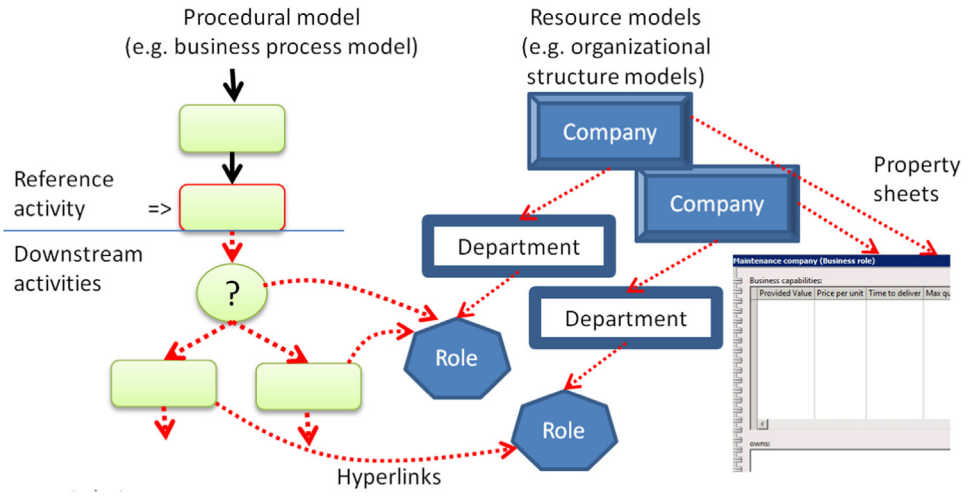

Fig. 12. Modeling relations traversed by SPARQL tests (highlighted).

Table 3
Query test results relative to model size.

Number of modeling objects	Number of modeling relations	Number of Quads exported	Average query performance (ms)
14	15	208	21
30	40	575	22
76	112	1510	27
166	199	2652	30
226	288	3966	34

The tests gave stable results shown in Table 3, with some reliability issues for the last row which used a business process model with 162 process elements and 64 elements spread in 3 organizational charts. However, most practical cases encountered in the project (a) deal with less than 100 modeling elements (across multiple models), (b) reduce the exported attribute set significantly (less than 50%, producing less than 1000 quads per Linked Model Cluster) and (c) use targeted, well constrained queries rather than "searching queries" that must deal with a large search space (such as "give me some data related to whatever there is downstream from an activity").

The conclusions of a SWOT evaluation based on this implementation and the encompassing conceptual work follows below (weaknesses and opportunities are more elaborated, in order to suggest future directions):

6.2.1. Strengths

The mechanism described in this paper exposes diagrammatic models in a Linked Data cloud and enables their linking to other available resources. If the (discussed) pre-conditions are met, it enables model-to-model linking, model-to-data linking, models-as-shared-vocabularies and model transformation that can fuel a new generation of *model-aware information systems* and can drive model-based collaboration with the kind of network effect that is expected for the development of the Web of Data. Therefore, the vision introduces "Linked Open Models" as a novel conceptual layer over the Web of Data, a possible intermediate stage between the pragmatic approach of Linked Open Data and the high level vision of an ontology-driven Semantic Web.

6.2.2. Weaknesses

As with Linked Open Data, several challenges are raised regarding management and publishing practices for Linked Open Models. The current solutions for version management are quite rudimentary: model versioning metadata is applied on the named model graphs and the PUT Graph Store Protocol Method [62] is used to fully replace a model graph after it is edited. However, a better granularity for model editing operations mapped on SPARQL-over-HTTP operations is a key future direction, to the aim of synchronizing model changes with the Linked Models repository. This is only one aspect of a broader work that needs to be developed for setting up a dedicated Linked Models repository that can guarantee quick access to metamodel and model descriptions, to support the sharing and documenting of relevant URIs for the discussed linking scenarios. Additional benefits may come from making URIs dereference-able in ways that are richer compared to the generic Linked Data practices (e.g., internal DESCRIBE queries and HTTP content negotiation [63]).

6.2.3. Opportunities

Ontologists can benefit from the knowledge externalized through modeling tools, as a starting point for engineering fully-fledged domain or task ontologies. They can build on the models themselves (an approach suggested in scenario D. from Section 4.1) or on the Linked Metamodel (e.g., it can be derived from the discussed example that a Subsequent relation has, as domain and range, at least the union of Activity and Decision classes). Competency questions elicited as requirements usually suggest relations and attributes that must be captured by models; from these, domains and ranges can be derived, then organized in a class hierarchy covered by an upper ontology. Following Guarino's [64] classification, certain model

types could be the basis for the *task ontology* (e.g., the process- and goal-related models), others for the *domain ontology* (models describing resource types involved in or triggering processes). Therefore, a wide terrain of investigation for semantics-aware systems opens.

For example, a process-aware systems may use a Linked Model Base as a "process repository", behaving on the same principles as ontology-based knowledge repositories: different actors would host, share and change model serializations with the desired level of detailed (attributes made available to others), while consumer systems would maintain links to these models and their schemata. Business process models are, after all, as [65] explains, nothing more than complex data structures. The graph nature of Linked Data fits more naturally to their complexity compared to the relational and hierarchical ones (typically employed for process repositories). Conceptual diagrams have an underlying graph nature (typically emerging from the meta^{2}models of metamodeling platforms). Fitting diagrams in some relational schema usually has the effect of either inducing an excess of NULL values, or an excess of tables in the repository. Comparisons between relational and RDF data models have been analyzed in the literature based on the initial considerations of [66].

Finally, support for importing Linked Data in modeling tools may open new possibilities: values for modeling attributes may be available in the Linked Data cloud, originating in various sources such as semantically lifted legacy repositories (e.g., organizational structure and employees extracted from an ERP system). Another opportunity is to employ, for model transformations, reasoners and fully-fledged ontologies instead of the query-time rules exemplified in the work at hand.

6.2.4. Threats

There is a risk that the scope of the work seems limited to the ComVantage research project specificity. To avoid this, a discussion on the level of generality was included (Section 4.2) to emphasize the reusability and global value of the proposal. The proposed RDF vocabulary relies on a minimal meta-metamodel derived from the ADOxx metamodeling platform and, just as any RDF Schema, can be extended to include additional specialization. The metamodel elements are derived from the modeling language, hence they will are synchronous with the metamodel itself, acting like a controlled vocabulary for everyone who uses the same modeling tool. Furthermore, the Linked Data serialization of models can be used for notational interoperability, to transfer or transform similarly structured diagrams between different notations.

The success of Linked Data is heavily dependent on the "killer apps" that will convince end-users of the benefits they can get from semantically annotating and describing content. A Linked Data representation of conceptual models may facilitate the impact but also takes some risks, as end-users must be educated to embrace domain-specific modeling languages in order to produce a new type of Web content: Linked Open Models.

6.3. Takeaway message

The vision of Linked Open Models promotes the transfer of diagrammatic conceptual models between a human-oriented knowledge tier (with conceptual models used for knowledge acquisition) and the machine-oriented tier of the Future Internet, in order to enrich the semantics of Linked Data. The Linked Data paradigm requires means of producing data graphs from various types of legacy systems – (a) relational databases, via adapters such as D2RQ [36]; (b) gleaning mechanisms relying on tools like Any23 [67]. *An untapped source for lifting Linked Data are the modeling tools which traditionally rely on underlying graph structures. Richer structures can be injected in the Web of Data via modeling tools, enabling new query federation scenarios. This may be less ambitious than the ontology-centered vision of the Semantic Web, but it follows the pragmatic goals of Linked Data as an enabling technology.*

Acknowledgments

The research leading to these results was funded by the European Community's Seventh Framework Programme under Grant agreement no. FP7-284928 ComVantage.

References

[1] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Sci. Am. 284 (5) (2001) 34–43.
[2] The Linked Data Design Issues. URL ⟨http://www.w3.org/DesignIssues/LinkedData.html⟩ (accessed 10.07.15).
[3] The Open Model Initiative events. Workshops – OMI 2012. URL ⟨http://www.openmodels.at/events⟩ (accessed 10.07.15).
[4] ComVantage Research Project Consortium, Project public deliverables. URL ⟨http://www.comvantage.eu/results-publications/public-deriverables/⟩ (accessed 10.07.15).
[5] W3C, The RDF standard resources and specification. URL ⟨http://www.w3.org/RDF/⟩ (accessed 10.07.15).
[6] O. Pastor, J. Gomez, E. Insfran, V. Pelechano, The OO-method approach for information systems modeling: from object-oriented conceptual modeling to automated programming, Inf. Syst. 26 (7) (2001) 507–534.
[7] The Open Model Initiative Laboratory (OMILab). URL ⟨http://omilab.org⟩ (accessed 10.07.15).
[8] J. Mylopoulos, Conceptual modeling and Telos1, In: P. Loucopoulos, R. Zicari (Eds.), Conceptual Modeling, Databases, and Case – an Integrated View of Information Systems Development, Wiley, 1992, pp. 49–68.
[9] B. Pernici, M. Weske, Business process management, Data Knowl. Eng. 56 (1) (2006) 1–3.
[10] R. Maier, Knowledge Management Systems, Springer, Berlin Heidelberg, 2004.
[11] W. M. P. van der Aalst, Process-aware information systems: lessons to be learned from process mining, In: L. Jensen, W. M. P. van der Aalst (Eds.), LNCS, vol. 5460, Springer, Berlin, Heidelberg, 2009, pp. 1–26.
[12] Protégé, official website. URL ⟨http://protege.stanford.edu/⟩ (accessed 10.07.15).
[13] Future Internet Enterprise Systems cluster, The FInES Research Roadmap 2025. URL ⟨http://cordis.europa.eu/fp7/ict/enet/documents/fines-research-roadmap-v30_en.pdf⟩ (accessed 10.07.15).
[14] D. Karagiannis, H. Kühn, Metamodelling platforms, In: Proceedings of the Third International Conference EC-Web, DEXA, LNCS, vol. 2455, Springer, Berlin, Heidelberg, 2002, p. 182.
[15] BOC-Group, ADOxx tool. URL ⟨http://www.adoxx.org/live/⟩ (accessed 10.07.15).
[16] T. Muench, R. Buchmann, J. Pfeffer, P. Ortiz, C. Christl, J. Hladik, J. Ziegler, O. Lazaro, D. Karagiannis, L. Urbas, An Innovative Virtual

Enterprise Approach to Agile Micro and SME-based Collaboration Networks, In: L. M. Camarinha-Enterprise Approachto Agile Micro- and SME-based Collaboration Networks, In: L. M. Camarinha-Matos, R. J. Scherer (Eds.), Proceedings of the 14th IFIP Conferenceon Virtual Enterprises, Springer, 2013, pp. 121–128.
[17] The ComVantage modeling prototype page at OMILab. URL ⟨http://www.omilab.org/web/comvantage/home⟩ (accessed 10.07.15).
[18] K. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson, Feature-oriented domain analysis (FODA) feasibility study, Software Engineering Institute, Technical Report CMU/SEI-90-TR-021, 1990.
[19] R. Buchmann, Conceptual modeling for mobilem aintenance: the ComVantage case, In: R. H. Sprague Jr. (Ed.), Proceedings of the 47th Hawaii International Conference on Systems Sciences, IEEECS, 2014, pp. 3390–3399.
[20] J. Gordijn, H. Akkermans, E3-value: design and evaluation of e-business models, IEEE Intell. Syst. 16 (4) (2001) 11–17.
[21] J. Ziegler, M. Graube, J. Pfeffer, L. Urbas, 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, 2013, pp. 1–8.
[22] W3C, The SPARQL Query Language specification. URL ⟨http://www.w3.org/TR/sparql11-query/⟩ (accessed 10.07.15).
[23] W3C, The SPARQL Graph Store HTTP Protocol specification. URL ⟨http://www.w3.org/TR/sparql11-http-rdf-update/⟩ (accessed 10.07.15).
[24] BOC-Group, ADONIS Community Edition Tool. URL ⟨http://www.adonis-community.com/⟩ (accessed 10.07.15).
[25] W3C, N-quads, The Official recommendation Page. URL ⟨http://www.w3.org/TR/n-quads/⟩ (accessed 10.07.15).
[26] C. Bizer, R. Cyganiak, The TriG Syntax. URL ⟨http://wifo5-03.informatik.uni-mannheim.de/bizer/trig/⟩ (accessed 10.07.15).
[27] J.J. Carroll, P. Stickler, The TriX syntax. URL ⟨http://www.hpl.hp.com/techreports/2004/HPL-2004-56.html⟩ (accessed 10.07.15).
[28] J. Hayes, C. Gutierrez, Bipartite graphs as intermediate model for RDF, In: S.A. McIlraith (Ed.), ISWC 2004, LNCS, vol. 3298, Springer, 2004, pp. 47–61.
[29] Object Management Group, MetaObject Facility – The Official Page. URL ⟨http://www.omg.org/mof/⟩ (accessed 10.07.15).
[30] W3C, RDB2RDF – The Official Page. URL ⟨http://www.w3.org/2001/sw/rdb2rdf/⟩ (accessed 10.07.15).
[31] Supply Chain Council, The Supply Chain Operations Reference specification. URL ⟨http://supply-chain.org/scor⟩ (accessed 10.07.15).
[32] Dynamic Business Process Formation for Instant Virtual Enterprises, In: N. Mehandjiev, P. Grefen, P (Eds.), Springer, Berlin, 2010.
[33] D. Allemang, J. Hendler, Semantic Web for the Working Ontologist, Morgan Kaufmann, 2011.
[34] Sindice, The Semantic Web Index. URL ⟨http://sindice.com/⟩ (accessed 10.07.15).
[35] Link Discovery with SILK. URL ⟨http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/⟩ (accessed 10.07.15).
[36] D2RQ - The Official Website. URL ⟨http://d2rq.org/⟩ (accessed 10.07.15).
[37] OASIS, BPEL, The Official Page. URL ⟨https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel⟩ (accessed 10.07.15).
[38] J. Morgan, Guidelines for Chaining iOS Apps. URL ⟨http://usabilityetc.com/2012/05/guidelines-for-chaining-ios-apps/⟩ (accessed 10.07.15).
[39] M. Lindemann, S. Junginger, T. Rausch, H. Kühn, ADOLog: An implementation of the Supply Chain Operations Reference Model, In: Proceedings of the 9th European Concurrent Engineering Conference 2002 (ECEC'2002), Society for Computer Simulation (SCS), 2002.
[40] H. Kern, A. Hummel, S. Kuhne, Towards a comparative analysis of meta-metamodels, In: Proceedings of the 11th Workshop on Domain-Specific Modeling, Portland, USA. URL ⟨http://www.dsmforum.org/events/DSM11/Papers/kern.pdf⟩ (accessed 10.07.15).
[41] MetaEdit+ DSM Environment. URL ⟨http://www.metacase.com/products.html⟩ (accessed 10.07.15).
[42] S. Kelly, K. Lyytinen, M. Rossi, MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE Environment, In: P. Constantopoulos, J. Mylopoulos, Y. Vassiliou (Eds.), Proceedings of CAISE 1996, LNCS, vol. 1080, Springer, Berlin, Heidelberg, 1996, pp. 1–21.
[43] MS Visio, The Official Website. URL ⟨http://visio.microsoft.com/en-us/pages/default.aspx⟩ (accessed 10.07.15).
[44] W3C, SKOS Homepage. URL ⟨http://www.w3.org/2004/02/skos/⟩ (accessed 10.07.15).
[45] W3C, RDF Schema. URL ⟨http://www.w3.org/TR/rdf-schema/⟩ (accessed 10.07.15).
[46] The Object Management Group. URL ⟨http://www.omg.org/⟩ (accessed 10.07.15).
[47] B. Henderson-Sellers, Bridging metamodels and ontologies in software engineering, J. Syst. Softw. 84 (2) (2011) 301–313.
[48] G. Guizzardi, Ontological foundations for structural conceptual models, Enschede, 2005.
[49] R. Studer, R. Benjamins, D. Fensel, Knowledge engineering: principles and methods, Data & Knowledge Engineering 25 (1–2) (1998) 161–198.
[50] D. Gašević, N. Kaviani, M. Hatala, On metamodeling in megamodels, in: G. Engels, B. Opdyke, D. Schmidt, F. Weil (Eds.), Proceedings of MoDELS 2007, LNCS, vol. 4735, Springer-Verlag, Berlin, 2007, pp. 91–105.
[51] C. Atkinson M. Gutheil K. Kiko On the relationship of ontologies and models, In: Proceedings of WoMM Meta-Modelling and Ontologies, LNI, Volume P-96, 2006, pp. 47–60.
[52] M. Brade, F. Schneider, A. Salmen, R. Groh, Ontosketch: towards digital sketching as a tool for creating and extending ontologies for non-experts, Proc. IKNOW, ACM, New York, 2013 article no. 9.
[53] D. Moody, The physics of notations: towards a scientific basis for constructing visual notations in software engineering, IEEE Trans. Softw. Eng. 35 (5) (2009) 756–777.
[54] M. Jeusfeld, Metamodeling and method engineering with ConceptBase, in: M. Jeusfeld, M. Jarke, J. Mylopoulos (Eds.), Metamodeling for Method Engineering, The MIT Press, Cambridge, MA, USA, 2009, pp. 89–168.
[55] Object Management Group, The Ontology Definition Metamodel specification. URL ⟨http://www.omg.org/spec/ODM/⟩ (accessed 10.07.15).
[56] Workflow Management Coalition, XPDL The Official Website. URL ⟨http://www.xpdl.org/⟩ (accessed 10.07.15).
[57] Object Management Group, XMI The Official Website. URL ⟨http://www.omg.org/spec/XMI/⟩ (accessed 10.07.15).
[58] M. Dumas, J. Recker, J.M. Weske, Management and engineering of process-aware information systems: Introduction to the special issue, Inf. Syst. 37 (2) (2012) 77–79.
[59] N. Bencomo, R. France, B.H.C. Cheng, U. Aßmann, LNCS, vol. 8378, Springer, Berlin, Heidelberg, 2014.
[60] Y. Lin, A. Soelvberg, Goal annotation of process models for semantic enrichment of process knowledge, in: J. Krogstie, A. Opdahl, G. Sindre (Eds.), Proceedings of CAISE 2007, LNCS, vol. 4495, Springer, 2007, pp. 355–369.
[61] Sesame, The Official Page. URL ⟨http://rdf4j.org/⟩ (accessed 10.07.15).
[62] System Documentation for Sesame 2. Formats and Protocols. URL ⟨http://rdf4j.org/sesame/2.7/docs/system.docbook?viewFormats_and_Protocols⟩ (accessed 10.07.15).
[63] T. Heath, C. Bizer, Linked Data: Evolving the Web into a Global Data Space, Morgan and Claypool. URL ⟨http://linkeddatabook.com/editions/1.0/htoc11⟩, 2011 (accessed 10.07.15).
[64] N. Guarino, Formal ontology and information systems: Proceedings of FOIS 1998, IOS Press, 1998.
[65] S. Jablonski, Do we really know how to support processes? Considerations and reconstruction, In: G. Engels, C. Lewerentz, W. Schafer, A. Schurr, B. Westfechtel (Eds.), Graph Transformations and Model-Driven Engineering, LNCS, vol. 5765, Springer, 2010, pp. 393–410.
[66] T.B. Lee, Relational Databases on the Semantic Web. URL ⟨http://www.w3.org/DesignIssues/RDB-RDF.html⟩ (accessed 10.07.15).
[67] Apache Any23 RDFizer. URL ⟨http://any23.apache.org/⟩ (accessed 10.07.15).