Bus Inf Syst Eng 58(5):341–353 (2016)
DOI 10.1007/s12599-016-0445-1

RESEARCH PAPER

Enriching Linked Data with Semantics from Domain-Specific Diagrammatic Models

Robert A. Buchmann · Dimitris Karagiannis

Received: 1 November 2015 / Accepted: 21 June 2016 / Published online: 8 August 2016

Abstract One key driver of the Linked Data paradigm is the ability to lift data graphs from legacy systems by employing various adapters and RDFizers (e.g., D2RQ for relational databases, XLWrap for spreadsheets). Such approaches aim towards removing boundaries of enterprise data silos by opening them to cross-organizational linking within a "Web of Data". An insufficiently tapped source of machine-readable semantics is the underlying graph nature of diagrammatic conceptual models – a kind of information that is richer compared to what is typically lifted from table schemata, especially when a domain-specific modeling language is employed. The paper advocates an approach to Linked Data enrichment based on a diagrammatic model RDFizer originally developed in the context of the ComVantage FP7 research project. A minimal but illustrative example is provided from which arguments will be generalized, leading to a proposed vision of "conceptual model"-aware information systems.

Keywords Model-aware information systems · Domain-specific modeling · Linked data · Linked models · Metamodeling

Accepted after two revisions by the editors of the special issue.

Conf. Univ. Dr. R. A. Buchmann (✉)
Business Information Systems Department, Faculty of Economic Sciences and Business Administration, University Babes-Bolyai of Cluj Napoca, Str. Teodor Mihali 58-60, 400591 Cluj-Napoca, Romania
e-mail: robert.buchmann@econ.ubbcluj.ro

Prof. Dr. D. Karagiannis
Knowledge Engineering Research Group, Faculty of Computer Science, University of Vienna, Waehringerstr. 29, 1090 Vienna, Austria
e-mail: dk@dke.univie.ac.at

1 Introduction

The growth of the Web of Data and the global acceptance of Linked Data as a pragmatic paradigm (Heath and Bizer 2011; Äuer et al. 2014) are crucially dependent on the ability to derive data graphs from legacy/existing systems – e.g., relational databases, HTML pages, Open Data portals. In order to achieve this, an extensive effort has been made to provide adapters and RDFizers: D2RQ for relational databases (D2RQ 2015), XLWrap for Excel Spreadsheets (Langegger and Wöß 2009), the DBPedia extractor (Lehmann et al. 2009; DBPedia 2015), gleaners/distillers of Web documents (Any23 2015), CSV Open Data lifters (van der Waal et al. 2014) etc. – an extensive list is available at W3C (2015c). Typically the semantics of such data are based on vocabularies that have been derived from the same legacy source (e.g., a table structure, a micro-format dialect).

The work at hand considers, as input for enriching Linked Data, a new source of machine-readable semantics that can be leveraged: domain-specific diagrammatic models, together with the information available on the abstraction levels of their underlying modeling language (i.e., the metamodel and meta^{2}model). The output may act as a semantic bridge between Linked Data of various provenances, as well as a back-end mash-up for run-time systems that must leverage information available in a diagrammatic form (for the purposes of this paper, such systems will be subsumed under the label "conceptual model"-aware information systems).

The value does not come necessarily from standardized languages [e.g., RDF representations for some types of UML diagrams are available (TopQuadrant 2015)], but rather from agilely developed modeling methods/languages that capture domain-specific semantics based on

requirements which are relevant within a limited context (a single enterprise or a collaboration network). This will be illustrated by a minimal, yet representative running example based on (i) a domain-specific modeling demonstrator built on the ADOxx metamodeling platform (BOC 2015) and (ii) a proof-of-concept RDFizer, originally applied to the ComVantage enterprise modeling tool (OMILab 2015a). The minimal example described in this paper was designed to strip down the presentation of project-specific details and to showcase the reusability of the proposal and its key features independently of its application context.

To achieve its goal, this work builds on the discipline of metamodeling as a means of anchoring diagramming shapes/symbols in machine-readable abstractions (metamodels and meta-metamodels). Typically, metamodeling works with a rigid separation of these layers, based on the MetaObject Facility architecture (OMG 2015b). The Resource Description Framework (W3C 2015d) is used here (a) to capture the different abstraction layers in a uniform representation – in this respect, providing an open alternative to works such as Jeusfeld (2009) as well as (b) to enable the linking of models to external resources (e.g., other models, Linked Data or schemata of various provenance), thus leading to a potential "Web of Data and Models".

The paper is built around a running example described in Sect. 5. The example is preceded by an overview of related works (Sect. 2), methodological aspects (Sect. 3), and a motivational scenario (Sect. 4), and is succeeded by a discussion on the relation between diagrams and run-time resources (Sect. 6). The paper ends with an evaluation and discussion of the proposal's business relevance, followed by conclusions derived from a proof-of-concept implementation.

2 Background and Related Works

Due to the rather recent uptake of Linked Data, prototypical RDFizers are still emerging for various types of legacy data sources (some of the prominent ones have been referenced in the Introduction). Currently, the list maintained by W3C (2015c) does not include a tool similar to this proposal. Due to a mostly parallel development of the Conceptual Modeling and Linked Data paradigms, the literature is still poor with respect to their interplay. Model queries have been approached strictly within a modeling tool, lacking openness to existing Web resources (Delfman et al. 2015). In Jarrar and Dikaiakos (2008), the authors convert diagrams to SPARQL queries in order to achieve data mash-ups in a Web 2.0 context. Semantic enrichment approaches are biased towards annotating diagrammatic models with (meta)data (Lin and Soelvberg 2007) rather than enriching the semantics of legacy data with diagrammatic knowledge. Semantic lifting techniques often involve tagging vocabularies/ontologies (Costa and Lima 2013). Diagrams have been typically employed in relation to Linked Data as a means of visualization (Neto et al. 2016), while the availability of knowledge in diagrammatic form was largely overlooked, possibly due to a tendency to subordinate conceptual modeling to the goal of code generation ("modeling is programming"). Therefore diagrammatic models are typically discussed in the context of standardized languages for software engineering and their serialization relies on XML syntactic interoperability – XMI (OMG 2015c), BPEL (OASIS 2015), XPDL (WfMC 2015). Departing from this tradition, we treat modeling as knowledge representation, since the mash-up of diagrammatic models and Linked Data can lead to a conceptual graph base derived from domain-specific languages.

Therefore, this paper highlights new benefits for diagrammatic models at run-time – see also the roadmaps discussed in the Models@runtime seminars. Regarding their research challenges raised with respect to business process management (Redlich et al. 2014), the work at hand provides pragmatic solutions for the aspects of "causal connections" (by enabling semantic linking between run-time data and models) and "reasoning" (by employing query-time graph transformations as production rules applied on models).

The foundations leading to this proposal are provided by the paradigms of (a) metamodeling and (b) Linked Data, while motivational background derives from (c) enterprise modeling, and (d) process-aware information systems engineering:

(a) We transfer the desideratum of data linking towards model linking, so that models become navigable and "open" knowledge structures, benefiting from their underlying graph nature and multi-layered abstractions, as exposed by metamodeling platforms. Thus, from *metamodeling* we build on the notion of a *modeling method* as defined in Karagiannis and Kühn (2002) in terms of its building blocks: (i) the modeling language (the set of modeling constructs described in terms of their metamodel, syntax and semantics – here, considering the linking requirements); (ii) the modeling procedure (steps to be taken by modelers towards their goals – here, including guidelines for model linking); (iii) mechanisms/algorithms (functionality built on model contents – here, the pattern-based RDFization and related usability-oriented functionality).

(b) In order to achieve model navigability and awareness, we employ the *Linked Data* design principles

and the technological space available for linking enterprise data (Äuer et al. 2014): the data model (RDF), the query language (SPARQL), the storage technology and retrieval protocols (Aduna 2015). As an addition to the adapters referenced in Sect. 1, the work at hand contributes a new type of RDFizer that leverages diagrammatic semantics in relation to Linked Open Data.

(c) The practice of *enterprise modeling* uses models to visually describe various aspects of enterprise architectures. Unlike software engineering, enterprise models do not share the same level of abstraction as code. Multiple frameworks and modeling methods have emerged – e.g. (OpenGroup 2015). Languages for business process modeling (OMG 2015a) or requirements modeling (Yu et al. 2011) can also be mentioned here, in the sense that they address facets of enterprise modeling, although they are not concerned with the cross-facet model linking. Multi-level modeling (Frank 2014) has been a key concern where domain-specificity was required in enterprise modeling; however, it commonly addresses design-time requirements, whereas our work focuses on modeling requirements that propagate from run-time systems.

(d) The recent progress in information systems engineering has brought forward an important distinction between data (schema)-awareness and *process-awareness* (van der Aalst 2009). The second capability enables process-aware information systems ("PAIS") to (en)act in accordance to the knowledge captured in business process models: "[a PAIS] manages and executes operational processes involving people, applications, and/or information sources on the basis of process models" (Dumas et al. 2005). A feature that distinguishes between these two generations of information systems is the type of conceptual model that governs their execution at run-time: (i) schema-awareness relies on some data schemata, i.e., some implementation of an ER (entity-relationship) model; (ii) process-awareness relies on workflow/business process models serialized in some machine-readable, XML structure, e.g., BPEL (OASIS 2015), XPDL (WfMC 2015). To subsume both data schema-awareness and process-awareness, we employ here the loose sense of the term "conceptual models", covering both models with ontological scope (e.g., ER diagram, domain models, taxonomies) and models with applicative scope (e.g., business process models). Depending on whether or not business process models are part of a hybrid model repository, we can either (i) pursue the PAIS aim of driving and assisting process execution or (ii) facilitate different kinds of awareness, with models acting as semantically rich configurations for parameterized run-time systems. To this end, a Linked Data-based serialization is preferred to the XML-based process serializations traditionally employed in PAIS.

3 Methodological Aspects

We subsume both *process-awareness* and *data schema-awareness* to *(conceptual) model-awareness* by resorting to the methodological framework of "Agile Modeling Method Engineering" as applied in the ComVantage research project (Buchmann and Karagiannis 2015b). This is based on an iterative, incremental cycle for agilely developing a domain-specific modeling tool driven by two classes of modeling requirements: (i) those derived directly from design-time needs (e.g., decision support) and (ii) those that propagate from run-time systems' requirements and can benefit from semantics captured in non-standard diagrammatic models. The cycle follows specific phases from the identification of relevant modeling concepts in requirements to the deployment of a modeling tool. The technological space employed for leveraging the knowledge expressed in diagrammatic form comes from the Linked Data paradigm – more specifically, the standards for information linking (W3C 2015d) and retrieval (W3C 2015a, b), while the modeling tool itself was developed in one of the available metamodeling environments that enables agility in implementations (BOC 2015).

The underlying graph nature of models and their enabling abstractions (i.e., metamodels) is exposed as a semantic complement to the back-end data consumed by run-time systems. Consequently, run-time awareness extends towards arbitrary types of models produced with modeling methods aiming for domain-specificity, thereby favoring specialization/familiarity at the expense of reusability across domains (agility supports this through a gradual assimilation of semantics in the language).

4 Motivating Scenario

Although the proposal was originally developed for a project-specific modeling method, what we describe here is a running example stripped down of project details and built around a fictive scenario that highlights key principles and design decisions, starting from the following requirements:

- A parking company publishes Linked Open Data that reports in real time on the availability of parking spaces

in various areas, using a fixed vocabulary (more than one company may share the vocabulary to provide such data for different geographical areas).
- A courier company needs a domain specific modeling tool to design tasks for its couriers and to map those tasks on the geographical areas where they must be accomplished.
- Couriers (employees of the courier company) need an app that allows them to check designated tasks, to consult parking space availability and to reserve parking spaces corresponding to their designated tasks.

It can be inferred that the couriers' app must make use of both run-time data (parking availability provided by third parties) and model information (designated tasks) from the courier company. Multiple cases can be extrapolated from the courier's requirements:

1. Retrieval of open data regardless of models (e.g., *Show me all parking areas and their availability*);
2. Retrieval of model information regardless of third party data (e.g., *Show me a list of courier tasks*);
3. Retrieval of model information constrained by open data (e.g., *Show me all tasks for which parking spaces are available*);
4. Retrieval of open data constrained by model information (e.g., *Show me a list of parking areas from the cities where my task will take me*);
5. Retrieval of both open data and model information, including mutual constraints (e.g., *Show me the steps of my task and corresponding parking space availability*). In the remainder of the paper we will only tackle this case, since all the others are simplifications of it.

5 Running Example

5.1 The Third-Party Run-Time Data

It is assumed that data on parking space availability is published by third parties (parking companies) as Linked Open Data. A minimal sample serving our example is listed in Fig. 1, both in a legacy (table) format and its TriG serialization (Bizer and Cyganiak 2007). Typically such data is lifted from legacy systems, for which adapters are openly available (D2RQ 2015; Langegger and Wöß 2009).

5.2 The Metamodel

A modeling tool must answer the courier company's modeling requirements (including the "propagating" run-time app requirements). Its conceptual foundation is a

Tabular view

Parking Areas	
Parking ID	**Availability**
ParkingA	2
ParkingB	0
ParkingC	4
ParkingD	0
ParkingE	3

TriG serialization

```
@prefix : <http://parkingcompany.com#>.
:RuntimeData
{
:ParkingA :availability 2 ; a :ParkingArea.
:ParkingB :availability 0 ; a :ParkingArea.
:ParkingC :availability 4 ; a :ParkingArea.
:ParkingD :availability 0 ; a :ParkingArea.
:ParkingE :availability 3 ; a :ParkingArea.
}
```

Fig. 1 Run-time data sample

metamodel depicting the language terminology. In Fig. 2 we repurpose a UML class diagram to describe a modeling language comprising two types of models:

(a) The *CourierTask* model type is a rudimentary control flow allowing the design of sequences of *Actions* and *Decisions* that describe courier tasks.
(b) The *ParkingMap* model type shows the allocation of *ParkingAreas* (of different types) to geographical areas (*Cities*).

The *Actions* from task models can be mapped to the *Cities* of parking maps (via the cross-model relation *requiresParkingInCity*), therefore a cross-model semantic link is established at the level of the modeling language. In addition, some of the concepts have properties that will be editable as annotations in the modeling tool (for both modeling elements and connectors). This metamodel becomes the basis for consistency checking (on connectors, inter-model links, attributes), imposed primarily through domain, range and cardinality constraints defined in the underlying metamodeling environment. This will also be applicable to imported models, making the modeling tool a validation point for RDFized models with respect to the metamodel.

5.3 The Domain-Specific Modeling Tool

An example showing models designed with this language is depicted in Fig. 3, showing three "linked models":

(a) Two task models: (i) *OnDemandTask*, where the courier must perform all the necessary transportation between the production steps of a shirt; (ii) *ScheduledTask*, which is a minimal task where a product must be taken from one place to another;
(b) One parking map where cities are assigned to task actions and parking areas are allotted to cities.

The arrows between models (e.g., from *Pick material* to *Vienna*) represent the *requiresParkingInCity* relation from the metamodel – that is, semantic links from tasks to cities (also acting as cross-model hyperlinks in our

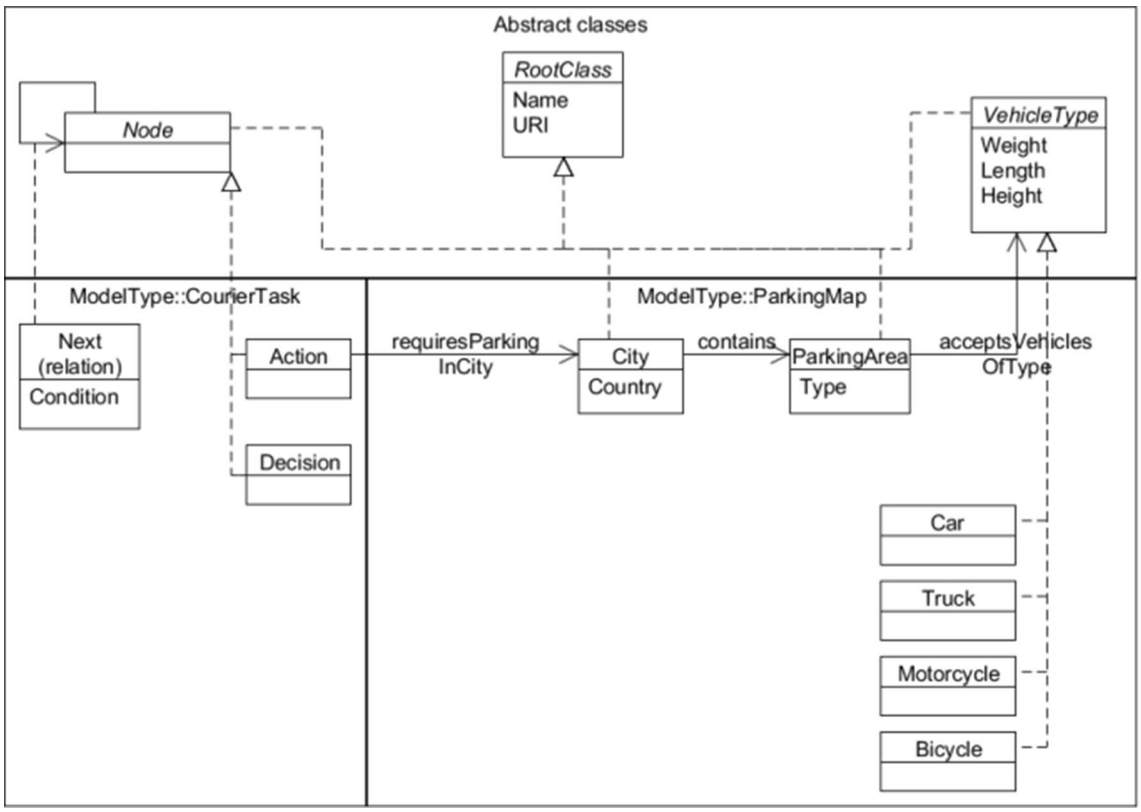

Fig. 2 Metamodel for the exemplary modeling language

implementation). The attributes defined by the metamodel manifest in the tool as pop-up sheets showing editable properties (see also the discussion on Fig. 4).

5.4 Model Linking

The linking of models that are open at the same time in the same modeling tool is controlled at metamodel level, through the domain, range and cardinality constraints on the relations that cross between partitions of the modeling language (model types). The end-user perceives such relations as hyperlinks for model browsing, but also as semantic links that may have their own editable properties. Models created by others may also be imported in the modeling tool by using the reverted RDFizer functionality, in order to benefit from the constraint checking mechanisms imposed by the metamodel implementation.

Additional aspects must however be considered when *linking models to external resources* – that is, when fetching existing URIs that identify the same things as the model elements, or class URIs from some schemata already available in the Web. The Linked Data paradigm envisions multiple means for publishing and acquiring existing URIs – e.g., search engines (SindiceTech 2015), link discovery based on similarity rules (Isele et al. 2015), public SPARQL endpoints (e.g., an endpoint where the parking company publishes the URI for all their parking lots). Several approaches have been tested in this respect.

5.4.1 Linking by Equivalence

In order to achieve model-to-data linking, the third-party parking area URIs (in Fig. 1) should be re-used as identifiers of the parking elements present in the models, thus establishing OWL equivalence between a modeled parking area and an existing parking area. For this purpose, a "universal identifier" attribute is prescribed on the metamodel level (see Fig. 2) in the *RootClass,* from which all modeling concepts inherit it (although for this particular example only the *ParkingArea* concept makes use of it). Filling this dedicated URI property slot (visible in Fig. 4) for the modeled parking areas can be performed through different means – manually or automated, depending on the desired level of usability and streamlining provided by the modeling tool:

(a) The generation of model elements (with pre-filled URIs) from a text-based list of obtained URIs, as suggested in Fig. 4. The source may be an existing URI list or the results of a SPARQL HTTP Protocol query to an existing endpoint. The underlying metamodeling scripting language (AdoScript in our

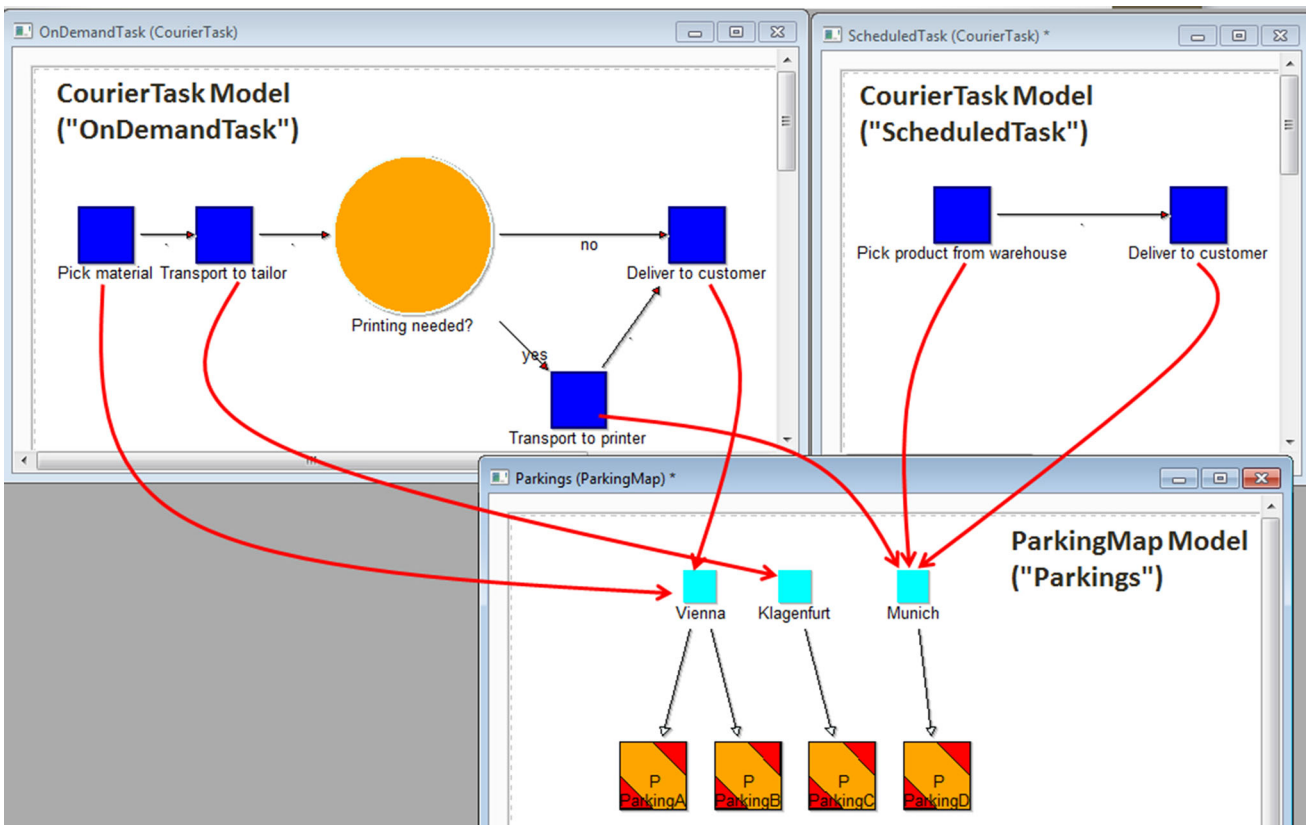

Fig. 3 Model samples for the running example

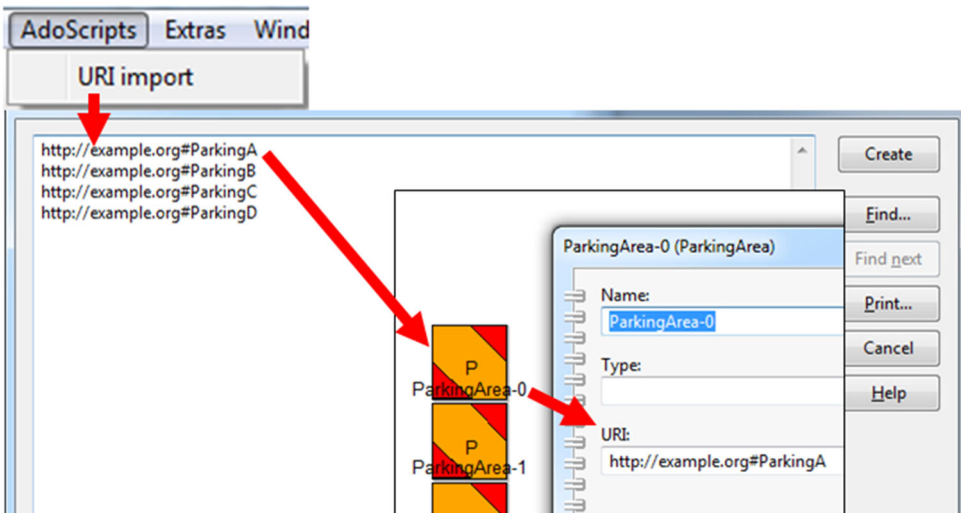

Fig. 4 Model-to-data linking via URI import. Note: Stating equivalence between model elements and resources which already have persistent URIs in the Web of Data might be an oversimplification with respect to the trap of "identity crisis" – a thing should not be considered the same as its representation, therefore a "representation" relation, rather than sameness, should be stated between model elements and modeled resources. An analysis of ways of misusing sameness is available in Halpin et al. (2010). The discussion will be furthered in Sect. 6

demonstrator) allows for the implementation of mechanisms that generate on-the-fly model elements with pre-filled property sheets and some convenient positioning.

(b) Manually typing known URIs in the dedicated slot in order to override the identifiers produced by the RDFizer (based on local names and a namespace provided by the modeler to indicate provenance).

(c) Importing existing RDF serializations of models which are compliant with the RDFizer schema (and have been SPARQL-constructed outside the modeling tool).
(d) Other means may be applied directly to the output of the RDFizer, based on the practices established in the Linked Data community – e.g., link discovery (Isele et al. 2015).

5.4.2 Linking by Modeling Properties

Equivalence is applicable if there is some redundancy in the model-data mash-up – i.e., if a modeling element can be interpreted as being "the same" as some resource described in the Web of Data (with the reservation indicated above). Our example uses equivalence to establish links between model contents and external Linked Data with the parking areas as a bridge.

It is however realistic to assume that city-to-parking mappings would also be available in the run-time open data. This could be handled by stating equivalence for cities; however, a different approach is also possible: if the design-time requirements allow it, the *Parking Map* model type could be dropped out of the modeling language and *requiresParkingInCity* would become a domain-specific editable slot for the (*Action*) model elements, to be filled by modelers with the known URIs of related cities. Such slots become RDF predicates (the metamodel indicates other such properties – e.g., *Country*).

5.4.3 Linking by Arbitrary Properties or Type

Similar to the use of the dedicated *URI* attribute, the RDFizer also provides special treatment for other editable properties that may be inherited by all model elements:

(a) A *property_collector* table allows the modeler to annotate any model element with arbitrary RDF statements for which the selected model element is considered as either a subject or an object; obviously, these will not be constrained by the modeling language, relying instead on external reasoning (via CONSTRUCT queries or persistent custom inference rules, as supported by the recent versions of the Sesame storage technology);
(b) A *type* property allows the modeler to link any model element to a known class from an existing external schema or OWL ontology, thus enriching the typing derived from the language metamodel and opening the model contents towards schema-based reasoning outside the modeling tool.

Examples of these types of links are not presented in the running example, but are discussed in the context of an inventory of diagram serialization patterns in Buchmann and Karagiannis (2015a).

5.5 RDFizing Diagrammatic Models

The RDFizer was developed in the context of the ComVantage FP7 project (ComVantage 2015) within the Open Model Initiative Laboratory (OMILab 2015a) on the ADOxx metamodeling platform (BOC 2015). A vocabulary (Karagiannis and Buchmann 2016) and a platform-independent inventory of diagram serialization patterns (Buchmann and Karagiannis 2015a) can guide implementations for other environments. A summary of key design decisions, formally described through hypergraphs in Karagiannis and Buchmann (2016), is provided here to make the paper self-contained with respect to the running example.

In order to achieve reusability outside the project context, the schema on which the serialization is based was modularized across abstraction layers with different degrees of flexibility. Links can be established between model elements and run-time open data (Fig. 5) at different abstraction levels:

(a) On the *meta*^{2} *layer*, the foundational constructs of common meta^{2}models – i.e., modeling class, visual connector, editable property etc. (see an overview in Kern et al. 2011) – are mapped on specializations of the primitive RDF Schema concepts under the fixed *cv:* namespace (e.g., the classes of all models, all model elements, all connectors, all inter-model hyperlinks, all editable properties);
(b) On the *meta layer*, domain-specific specializations are dynamically generated from the modeling language metamodel, therefore each language will produce its own classes (in our case, the classes of all *Cities*, all *ParkingAreas*) under a namespace decided by the modeler (in our case, *courier:*). This ensures that the RDFizer is reusable for other modeling languages;
(c) On the *model layer*, model contents are RDFized based on recurring diagram patterns. An inventory of pattern-based transformations is provided in Buchmann and Karagiannis (2015a). We will only summarize the key principles here:

- Each model becomes a separate named graph, instance of its model type (and of *cv:Model*), possibly annotated with model-level attributes (e.g., author, designated courier);
- Each editable property (annotation) of a model element is serialized as an RDF predicate (instance of

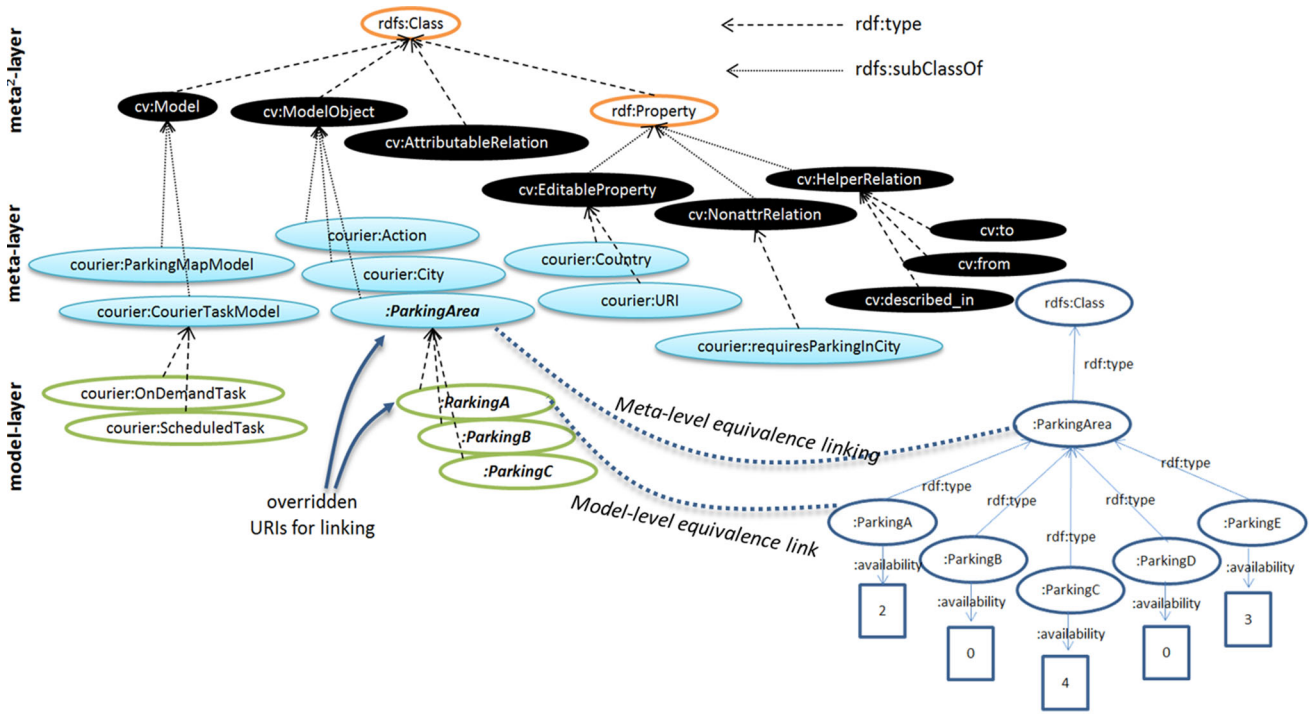

Fig. 5 The model serialization vocabulary and model linking levels

cv:EditableProperty), if it has a single value. More complex properties (lists, tables) can be serialized as rdf:List, but also as patterns communicated by the language designer through concept-level annotations. This includes the *property_collector* mentioned in Sect. 5.4.3;

- Each inter-model hyperlink is serialized as an RDF predicate (instance of *cv:NonattrRelation*), plus a helper property (*cv:described_in*) to indicate the model (named graph) where the target of the hyperlink belongs;
- Each visual connector with no editable attributes is serialized as an RDF predicate (instance of *cv:NonattrRelation*);
- Each visual connector with editable attributes (e.g., the *Next* arrow between task actions has an editable transition *Condition*) is serialized as an n-ary RDF relation, using the helper properties *cv:from* and *cv:to* to distinguish the connector source and target, respectively;
- URIs for all model elements are generated from a namespace provided by the modeler;
- As discussed in Sect. 5.4, links to external resources can be created in the modeling tool, by overriding URIs generated for model elements, but also by freely annotating them with RDF triples (e.g., new types can be declared for model elements, other than those prescribed by the metamodel). Linking at meta-level (e.g., the equivalence of the *ParkingArea* concepts) may also be applied after the serialization, by overriding the meta-level (concept) URIs with a preferred one;
- Usability and streamlining features are dependent on the underlying metamodeling platform (e.g., the filtering of editable properties that should be exported, the RDFized model upload to a repository of choice directly from the RDFizer UI, the generation of model elements with pre-filled URIs). The filtering of editable properties is particularly important since it impacts the number of quads to be produced, depending on a richness/performance trade-off and on what domain-specific details should be obscured.

5.6 Enriching Queries with Model Semantics

A demonstrator model-aware app was implemented to showcase queries that take advantage of the model-to-data links and inter-model links in order to mash-up third party Linked Data with diagram information. It runs SPARQL queries over HTTP using Sesame's REST protocol (Aduna 2015), so it relies on Sesame for storing both kinds of information. A query example is shown here [addressing requirement (5) from Sect. 4]. The example retrieves all cities and available parking spaces, but only for those cities that are assigned to the task called *OnDemandTask*. Assuming that the task name is parameterized with a user

selection, the query retrieves the data necessary to the second app screen (visible in Fig. 6).

The query assumes that the model information and the parking availability data are provided at different endpoints by different companies using different namespaces – prefixed as *courier:* and *park:*, respectively. The third namespace (*cv:*) is fixed by the meta^2 layer vocabulary.

Thus, the example shows a federated query which also hints at how the model information is structured as Linked Data: each diagrammatic model is a distinct named graph annotated with metadata (e.g., name, author, assigned courier etc.) and linked to elements in other diagrams through a semantic version of the *requiresParkinginCity* hyperlink.

```
PREFIX courier:<http://couriercompany.com#>
PREFIX park:<http://parkingcompany.com#>
PREFIX cv:<http://www.comvantage.eu/mm#>
SELECT DISTINCT ?cityname ?parkingname ?spaces
WHERE {

SERVICE <http://couriercompany.com/repository>
{GRAPH courier:graphmetadata
{?taskmodel courier:Name "OnDemandTask"}
GRAPH ?taskmodel
{?action courier:requiresParkingInCity ?city.
?city cv:described_in ?parkmapmodel}
GRAPH ?parkmapmodel
{?city courier:contains ?parking. ?city courier:Name ?cityname.
?parking courier:Name ?parkingname} }

SERVICE <http://parkingcompany.com/repository>
{GRAPH park:RuntimeData {?parking park:availability ?spaces} } }
```

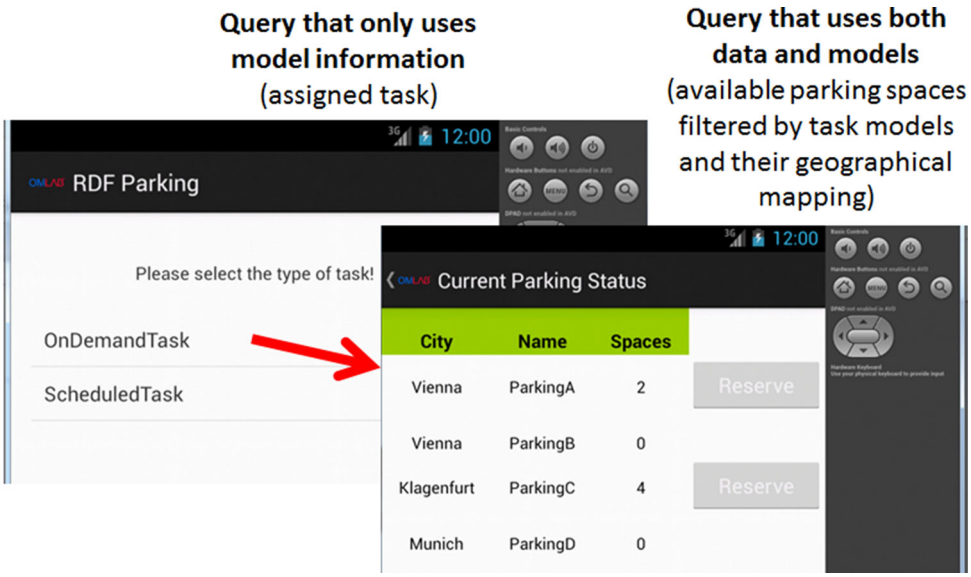

Fig. 6 Query and front-end for model-aware app

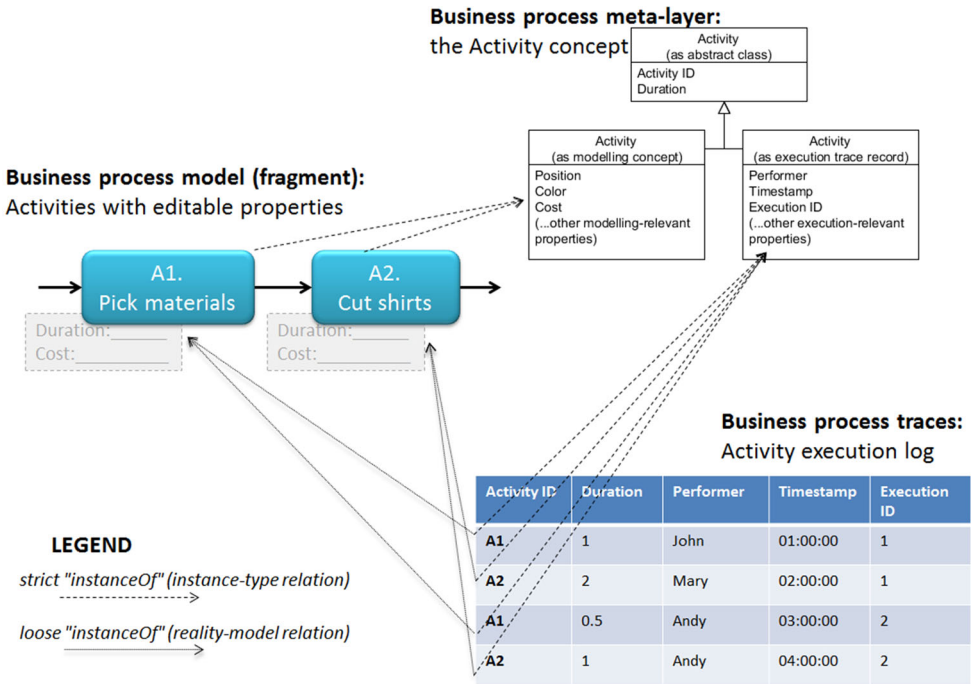

Fig. 7 Possible links between different representations of the activity concept

6 Discussion on Linking Models to External Resources

It is common in conceptual modeling to consider real world phenomena manifestations as instances of model elements. There is a common perception that model elements and the things they represent belong to different layers of abstraction according to some fixed type system – i.e., real things are instances of their diagrammatic representations, see M0 and M1 in the MOF architecture (OMG 2015b). However, for our purposes this view may be relaxed. In Sect. 5.4.1, the possibility of overriding URIs of model elements with those of existing resources is similar to stating an owl:sameAs equivalence between these. One may question the assumption that a model is equivalent to its corresponding real world resource. In order to keep the example minimal, we rely on the interpretation that this is not a standard "identity crisis" case (of having an addressable resource representing a non-addressable resource). Instead, the model element and the external RDF data are complementary (possibly overlapping) descriptions of the same thing and their different provenances may be distinguished by having the model contents grouped in a named graph. The goal is to allow queries to easily aggregate both design-time (diagrammatic) descriptions and run-time description of the same thing. We also consider designating a dedicated subproperty (of rdfs:seeAlso) to state that alternative, diagram-based descriptions are available about the same thing. The dilemma also emerges in other metamodeling contexts:

(a) Multi-level metamodeling (Frank 2014) deals with the creation of modeling languages that include, sometimes in the same model, elements that represent types together with instances. This is quite typical in domain-specific modeling, where certain model elements represent fixed, available entities (e.g., in our case real cities, existing parking lots). The properties of the model element are not necessarily inherited by its run-time corresponding resource, or vice versa. Instead, model elements are typically designed to hold data that serves some design-time use cases (e.g., simulation), while their corresponding run-time resource is described by data relevant to run-time systems.

(b) In business process management, process executions (recorded traces) are considered instances of some process model, although the process model does not necessarily provide a data schema from which all process executions can be instantiated with run-time values. Let's consider the concept of *Activity/Task*, which is present in most business process modeling languages. In a modeling tool, this would have properties that are relevant for modeling purposes (e.g., simulation/evaluation for process reengineering), while in a process trace log it would have properties that are relevant for post-execution auditing/monitoring (e.g., timestamps, performers). Figure 7 suggests the semantic overlapping between (i) the concept of *Activity* as instantiated in model elements and (ii) the *Activity* entity as captured in a

relational database where execution traces are accumulated. We differentiate here between "strict instantiation" (across layers of abstraction) and "loose instantiation" (enactment, between design-time and run-time). Relative to the meta-layer where the underlying *Activity* concept is designed, both the model element and its corresponding execution traces belong to the same layer of abstraction, each of them being an instance of some specialization of the *Activity* abstraction, but aiming for different kinds of requirements/systems (the same activity may be executed multiple times, but also the same activity may be modeled multiple times, even with multiple modeling languages). Re-using the activity identifiers in the two different contexts establishes valuable links to allow navigation between the descriptions that are relevant for design-time purposes and those relevant for run-time systems.

For example, direct enactment relations may be inferred by running SPARQL CONSTRUCTs on the RDFized mash-up of the model, execution data and meta-layer. The following example (although simplified for clarity) illustrates the possible derivation of enactment relations (the "loose instantiation") from the other links, by relying on the reuse of activity identifiers:

```
CONSTRUCT ?x :enactmentOf ?y
WHERE { GRAPH :executionData
        {?x :activityID ?id; a/rdfs:subClassOf :Activity}
        GRAPH :processModel
        {?y :activityID ?id; a/rdfs:subClassOf :Activity}}
```

In the paradigm of process-aware information systems, this predicate will typically be generated at run-time from "work items" (forms that require user interaction to acknowledge the logging of the current step and the advancement through the process). In a Linked Data environment this opens the possibility of having a process model repository that links to execution traces across multiple organizations.

7 Evaluations and Conclusions

7.1 Reusability and Validation in Business Context

The proposed RDFizing approach is reusable beyond the presented example, in two different interpretations of "reusability":

(a) Any modeling language/tool designed on the underlying ADOxx metamodeling platform (BOC 2015) will dynamically generate, through this RDFizer, the classes and properties corresponding to its language design, regardless of the types of diagrams, relations etc. A proof-of-concept was applied in the domain-specific ComVantage modeling method (its language extensively documented in Buchmann and Karagiannis 2015c) and is the basis for the evaluation presented in this section.

(b) The approach uses multi-layered abstractions inspired by the metamodeling discipline, so it can be easily implemented in similar plug-ins for other metamodeling platforms, based on the guidelines in Sect. 5.5 and the detailed inventory of transformation rules published in Buchmann and Karagiannis (2015b).

The business relevance of this work is related to the adoption of domain-specific enterprise modeling (Frank 2014) on a different layer of abstraction than code generation. Although such models have been traditionally employed for enterprise analysis, sense-making and communication, the agile metamodeling discipline imposed sufficient structure to consider such models as being machine-readable knowledge representations that can enrich Linked Enterprise Data graphs even in the absence of fully-fledged ontologies. It is not uncommon to find semantic redundancies between a concept present in a decision-support modeling tool and an entity present in a database used at run-time. This paper advocates a potentially useful bridge in a Linked Data environment, while benefiting from content created by business stakeholders who are not familiar with ontology engineering but are trained to communicate through domain-specific diagrams.

The proposal of this paper was applied in the industrial context of the FP7 research project (ComVantage 2015). An iteration of the proof-of-concept RDFizer is available at OMILab (2015b), with versions of higher usability being transferred for productization to the project partner dealing in the commercial exploitation of modeling tools. Extensive query examples for the domain-specific ComVantage modeling method are described in OMILab (2015c). Model queries were run over HTTP on a repository of models reflecting project use cases. A recurring query pattern is the one that retrieves a property of all enterprise resources linked to the activities which are positioned downstream relative to some fixed step in a business process (translated to our running example: "the capacities of all parking lots where a courier may go after a specific decision"). Table 1 shows the number of N-quads typically exported from models (both for this paper's running example and for ComVantage models), without filtering the editable properties of model elements (which significantly lowers the number of quads, if the modeler decides that certain

Table 1 Query test results relative to model size

Number of modeling objects	Number of modeling relations	Number of exported quads	Average query performance (ms)
The parking running example			
14	16	258	21
ComVantage models			
30	40	575	22
76	112	1510	27
166	199	2652	30
226	288	3966	34

attributes should not be shared, or are irrelevant to the model-awareness requirements).

7.2 Concluding SWOT Evaluation

The paper advocates a bridging between the paradigms of conceptual modeling and Linked Data, for the benefit of enabling *conceptual model-aware information systems* whose key characteristics are (a) the fact that diagrammatic models are brought on the same abstraction layer as execution data (relative to some metamodel/schema of superior abstraction) and (b) that they enrich data with semantics available in diagrammatic form. The following SWOT conclusions highlight both limitations and an outlook to further development opportunities:

- Strengths: Diagrammatic models developed with custom-made modeling methods provide machine-readable knowledge that can complement, in a Linked Data environment, the execution-time data, both for "a priori data" (data published independent of the existence of models) and "a posteriori data" (traces generated in relation with models).
- Weaknesses: The paper does not discuss issues pertaining to the management, versioning and maintenance of model information exposed as Linked Data. For demonstration purposes, model changes are handled by PUT requests that update an entire model graph, no matter how granular the change is. Mechanisms for finer granularity updates are being investigated, as well as new interpretations on traditional Linked Data notions (e.g., how URI dereferencing should work for a model element).
- Opportunities: The vision of *"conceptual model"-awareness* generalizes the paradigm of process-awareness if we consider arbitrary model types or if we extend process descriptions in the sense of enterprise modeling with domain-specific aspects. The current scope of the presented RDFizer is determined by the scope of its originating project. Post-project research is underway to investigate the potential interplay with OWL ontologies, beyond the current approach of linking to external URIs (specifically, designing ontology skeletons with diagrammatic modeling techniques, or importing SPIN rules in the modeling tool in order to apply model checks that are not supported by the metamodel).
- Threats: The uptake of the Semantic Web paradigm is still slow. However, the separation between short-term pragmatic goals (Linked Data adoption) and long-term scientific ambitions (reasoning agents, fully-fledged ontologies) contributes to a separation of concerns and the advancement of the paradigm with actionable proposals such as the one presented here. Just as Linked Data has established a pragmatic foundation for the less actionable Semantic Web, the proposal of Linked Models and model-data mash-ups is meant to enrich Linked Data even in the absence of fully fledged ontologies, and further work is necessary to harmonize the two approaches for semantic enrichment.

References

Aduna (2015) System documentation for Sesame 2. Formats and protocols. http://rdf4j.org/sesame/2.7/docs/system.docbook?view#Formats_and_Protocols. Accessed 1 October 2015

Any23 (2015) Any23 – official website. http://any23.apache.org. Accessed 1 October 2015

Äuer S, Bryl V, Tramp S (eds) (2014) Linked Open Data – creating knowledge out of interlinked data. Springer, Heidelberg

Bizer C, Cyganiak R (2007) The TriG syntax specification. http://wifo5-03.informatik.uni-mannheim.de/bizer/trig/. Accessed 1 October 2015

Buchmann RA, Karagiannis D (2015a) Pattern-based transformation of diagrammatic conceptual models for semantic enrichment in the web of data. In: Ding L, Pang C, Kew LM, Jain LC, Howlett RJ (eds) Proceedings of KES 2015, Procedia Computer Science 60. Elsevier, Amsterdam, pp 150–159

Buchmann RA, Karagiannis D (2015b) Agile modelling method engineering: lessons learned in the ComVantage project. In: Ralyte J, Espana S, Pastor O (eds) Proceedings of POEM 2015. LNBIP, vol 235. Springer, Heidelberg, pp 356–373

Buchmann RA, Karagiannis D (2015c) Modelling mobile app requirements for semantic traceability. Requir Eng. doi:10.1007/s00766-015-0235-1

ComVantage Consortium (2015) Project public deliverables. http://www.comvantage.eu/results-publications/public-deriverables. Accessed 1 October 2015

Costa R, Lima C (2013) An architecture to support semantic enrichment of knowledge sources in collaborative engineering projects. In: Fred A, Dietz JLG, Liu K, Filipe J (eds) Communications in computer and information science, vol 272. Springer, Heildelberg, pp 276–289

D2RQ (2015) D2RQ – official website. http://d2rq.org. Accessed 1 October 2015

DBPedia (2015) DBPedia – official website. http://dbpedia.org. Accessed 1 October 2015

Delfmann P, Steinhorst M, Dietrich HA, Becker J (2015) The generic model query language GMQL – conceptual specificiation, implementation and runtime evaluation. Inf Syst 47:129–177

Dumas M, van der Aalst WMP, ter Hofstede AHM (2005) Introduction. In: Dumas M, van der Aalst WMP, ter Hofstede AHM (eds) Process-aware information systems: bridging people and software through process technology. Wiley-Interscience, Hoboken, pp 3–20

Frank U (2014) Multilevel modeling: toward a new paradigm of conceptual modeling and information systems design. Bus Inf Syst Eng 6(6):319–337

BOC GmbH (2015) ADOxx – official website. https://www.adoxx.org/live/home. Accessed 1 October 2015

Halpin H, Hayes PJ, McCusker JP, McGuiness DL, Thompson HS (2010) When owl:sameAs isn't the same: an analysis of identity links on the semantic web. In: Patel-Schneider PF, Pan Y, Hitzler P, Mika P, Zhang L, Pan JZ, Horrocks I, Glimm B (eds) Proceedings of ISWC 2010, LNCS, vol 6496. Springer, Heidelberg, pp 305–320

Heath T, Bizer C (2011) Linked Data: evolving the web into a global data space, 1st edn. Morgan & Claypool, San Rafael

Isele R, Jentzsch A, Bizer C, Volz J, Petrovski P (2015) Silk – the official page. http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/. Accessed 1 October 2015

Jarrar M, Dikaiakos MD (2008) MashQL: a query-by-diagram topping SPARQL. Proceedings of ONISW 2008. ACM, New York, pp 89–96

Jeusfeld M (2009) Metamodeling and method engineering with ConceptBase. In: Jeusfeld M, Jarke M, Mylopoulos J (eds) Metamodeling for method engineering. MIT Press, Cambridge, pp 89–168

Karagiannis D, Buchmann R (2016) Linked Open Models: extending Linked Open Data with conceptual model information. Inf Syst 56:174–197

Karagiannis D, Kühn H (2002) Metamodelling platforms. In: Bauknecht K, Tjoa AM, Quirchmayr G (eds) Proceedings of EC-Web 2002 – DEXA 2002, LNCS, vol 2455. Springer, Heidelberg, p 182

Kern H, Hummel A, Kuhne S (2011) Towards a comparative analysis of meta-metamodels. In: The 11th workshop on domain-specific modeling, Portland http://www.dsmforum.org/events/DSM11/Papers/kern.pdf. Accessed 1 October 2015

Langegger A, Wöß W (2009) XLWrap – querying and integrating arbitrary spreadsheets with SPARQL. In: Bernstein A, Karger D R, Heath T, Feigenbaum L, Maynard D, Motta E, Thirunarayan K (eds) Proceedings of ISWC 2009. LNCS, vol 5823. Springer, Heidelberg, pp 359–374

Lehmann J, Bizer Ch, Kobilarov G, Äuer S, Becker Ch, Cyganiak R, Hellmann S (2009) DBpedia – a crystallization point for the web of data. J Web Semant 7(3):154–165

Lin Y, Soelvberg A (2007) Goal annotation of process models for semantic enrichment of process knowledge. In: Krogstie J, Opdahl A, Sindre G (eds) Proceedings of CAISE 2007, LNCS, vol 4495. Springer, Heildelberg, pp 355–369

Neto CB, Müller K, Brümmer M, Kontokostas D, Hellmann S (2016) LODVader: an interface to LOD visualization, analytics and discovery in real-time. Proceedings of WWW 2016. International WWW Steering Commitee, Geneva, pp 163–166

OASIS (2015) BPEL – the official website. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel. Accessed 1 October 2015

OMG (2015a) BPMN specification. http://www.bpmn.org. Accessed 1 October 2015

OMG (2015b) The MetaObject facility – official website. http://www.omg.org/mof/. Accessed 1 October 2015

OMG (2015c) XMI – the official website. http://www.omg.org/spec/XMI. Accessed 1 October 2015

OMILab (2015a) The metamodelling page for the ComVantage project. http://www.omilab.org/web/comvantage/home. Accessed 1 October 2015

OMILab (2015b) Proof-of-concept RDFizer within OMILab. http://repo.omilab.org/omirepo/items.xhtml?groupId=181256&phaseId=4&jftfdi=&jffi=items. Accessed 1 October 2015

OMILab (2015c) ComVantage modelling guidelines. http://www.omilab.org/web/comvantage/modelling-guidelines. Accessed 1 October 2015

Redlich D, Blair G, Rashid A, Molka T, Gilani W (2014) Research challenges for business process models at run-time. In: Bencomo N, France R, Cheng BHC, Aßmann U (eds) Models@run.time. LNCS, vol 8378. Springer, Heidelberg, pp 208–236

SindiceTech (2015) Sindice – the semantic web index. http://sindice.com. Accessed 1 October 2015

The Open Group (2015) The Archimate specification. http://pubs.opengroup.org/architecture/archimate2-doc. Accessed 1 October 2015

TopQuadrant (2015) TopBraid composer – official website. http://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/. Accessed 1 October 2015

Van der Aalst WMP (2009) Process-aware information systems: lessons to be learned from process mining. In: Jensen L, van der Aalst WMP (eds) Transactions on Petri nets and other models of concurrency II. LNCS, vol 5460. Springer, Heidelberg, pp 1–26

Van der Waal S, Wecel K, Ermilov I, Janev V, Milosevic U, Wainwright M (2014) Lifting open data portals to the data Web. In: Äuer S, Bryl V, Tramp S (eds) Linked Open Data – creating knowledge out of interlinked data. Springer, Heidelberg, pp 175–195

W3C (2015a) The SPARQL query language specification. http://www.w3.org/TR/sparql11-query/. Accessed 1 October 2015

W3C (2015b) The SPARQL graph store HTTP protocol specification. http://www.w3.org/TR/sparql11-http-rdf-update. Accessed 1 October 2015

W3C (2015c) ConverterToRDF – official website. http://www.w3.org/wiki/ConverterToRdf. Accessed 1 October 2015

W3C (2015d) RDF – official website. http://www.w3.org/RDF/. Accessed 1 October 2015

WfMC (2015) XPDL – the official website. http://www.xpdl.org. Accessed 1 October 2015

Yu E, Giorgini P, Maiden N, Mylopoulos J (eds) (2011) Social modeling for requirements engineering. MIT Press, Cambridge