Streamlining Structured Data Markup and Agile Modelling Methods

Ana-Maria Ghiran[1], Robert Andrei Buchmann[1], Cristina-Claudia Osman[1], Dimitris Karagiannis[2]

[1]Business Informatics Research Center, Babeş-Bolyai University, Romania
{anamaria.ghiran,robert.buchmann,cristina.osman}@econ.ubbcluj.ro
[2]Knowledge Engineering Research Group, University of Vienna, Austria
dk@dke.univie.ac.at

Abstract. Structured Data Markup allows Web developers to embed semantics in HTML pages, thus enabling clients (search engines, client apps etc.) to distil machine-readable resource descriptions from HTML code. This approach emerged from the Semantic Web paradigm as a powerful alternative to traditional Web scraping. Its enablers are dedicated HTML extensions (e.g., RDFa) and controlled vocabularies (e.g., Schema.org). Originating in a different context, Enterprise Modelling methods rely on diagrammatic means for describing and analysing an enterprise system in terms of key properties and conceptual abstractions. Hence, both the Semantic Web and Enterprise Modelling paradigms share a common interest in machine-processable semantics towards the goal of elevating semantics-awareness in information systems and decision support. Inspired by this overlapping, the paper proposes a mechanism for streamlining semantics between Structured Data Markup and enterprise modelling methods. Towards this goal, it employs the Resource Description Framework and the Agile Modelling Method Engineering Framework.

Keywords: Structured Data Markup, Resource Description Framework, Agile Modelling Method Engineering, Schema.org, ADOxx

1 Introduction

Structured Data Markup is being advocated as a search engine optimisation (SEO) technique enabled by semantic technology grafted on traditional Web development practices [1]. The origins of this approach may be traced back to data gleaning from XML documents [2] and to microformat profiles [3]. More recently, the lessons learned from microformats have led to the centralisation of prominent description vocabularies under the Schema.org "umbrella terminology" [4] founded and maintained by the big search engine providers (e.g., Google, Yahoo, Microsoft). From a conceptual perspective, Schema.org can be considered an ontology – i.e., it provides a consensus on terms (categories and properties) that should be used to describe often searched types of resources: organisations, persons, events, actions etc. The Schema.org terminology is complemented by syntaxes that can extend HTML content with machine-readable descriptions of arbitrary resources – e.g., RDFa [5] introduced by the Resource Description Framework (RDF) [6].

In parallel developments, technologies and practices dealing with semantics have also emerged from the Enterprise Modelling paradigm – originating in data modelling, then evolving in complexity towards system modelling, business process modelling and multi-view enterprise modelling [7]. Although conceptual modelling is commonly perceived as being based on standards, the literature on modelling pragmatics [8] raised awareness on the need for domain-specificity or situational customisation of modelling methods, languages and tools. This is especially relevant in Enterprise Modelling where enterprise context or multi-perspective consistency concerns [9-10] may raise requirements on semantic customisation. Methodologies and fast prototyping enablers have emerged to allow knowledge engineers to tailor and deploy modelling methods and languages for narrow domains or situational cases [11-12]. They rely on metamodels that integrate concepts in *graphical language terminologies* which are comparable, to some extent, to ontological terminologies such as Schema.org.

This intuitive observation inspired the work at hand, as it proposes a streamlining between conceptual descriptions made available through semantic HTML markup and modelling languages that are synchronised to this markup with the help of the Agile Modelling Method Engineering (AMME) methodology [11]. The Resource Description Framework (RDF) [6] is employed as a bridging medium.

Therefore, the problem statement of this paper can be outlined as follows: *assuming that an organisation publishes machine-readable conceptual descriptions in their Web pages* (either for SEO purposes or for arbitrary client agents), *how can these be made available to a diagrammatic Enterprise Modelling environment*? The proposed solution extends the previously published method of Agile Modelling Method Engineering (AMME) with a mechanism for importing model contents in a modelling environment, from the RDF knowledge graph that can be distilled from Structured Data Markup; the necessity of AMME comes from the need to customize the targeted modelling language in order to align its semantics with those provided by the Structured Data Markup - especially if the markup uses the Schema.org terminology and not one that is under the control of the organisation using the modelling environment.

The remainder of the paper is organised as follows: Section 2 will introduce the technological and methodological enablers for the work at hand - Structured Data Markup and the Agile Modelling Method Engineering. Section 3 will present the mechanism through a running example. Section 4 will comment on related works. The paper ends with conclusions and outlook.

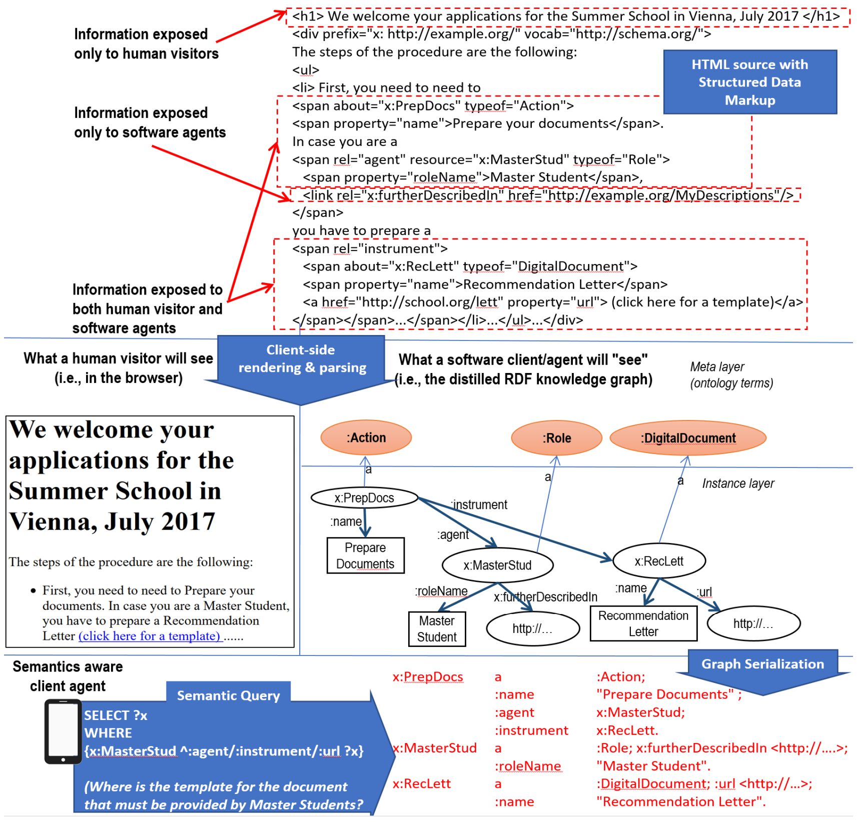
2 Technological and Methodological Enablers

2.1 The Structured Data Markup processing workflow

Structured Data Markup emerged from the convergence between traditional SEO practices and semantic technology. SEO aims to make the contents of HTML documents "understandable" for search engines. Structured Data Markup allows Web developers to embed machine-readable semantics (i.e., resource descriptions governed by some ontology) directly in the HTML source, with the help of dedicated syntactic formats – e.g., RDFa. This brings drastic changes to SEO practices, as the traditional techniques are replaced with precise ways of describing content meaning. The Structured Data Markup may be "distilled" into RDF graphs – a data model that is amenable to knowledge representation and reasoning. Arbitrary client agents (not limited to search engines) can thus shift from traditional string-based scraping towards powerful semantic queries and reasoning over the distilled knowledge graphs.

Complementary to the mentioned syntactic formats, a centralised, extensible terminology was set up at Schema.org [4], incorporating concepts and properties that were previously available in scattered and narrow-scoped microformat profiles or RDF vocabularies (e.g., hCard [13], FOAF [14], GoodRelations [15]).

Fig. 1 shows an example of Structured Data Markup and the typical knowledge flow from a public HTML document to a client agent (not limited to search engine crawlers).



```
<h1> We welcome your applications for the Summer School in Vienna, July 2017 </h1>
<div prefix="x: http://example.org/" vocab="http://schema.org/">
The steps of the procedure are the following:
<ul>
<li> First, you need to need to
<span about="x:PrepDocs" typeof="Action">
<span property="name">Prepare your documents</span>.
In case you are a
<span rel="agent" resource="x:MasterStud" typeof="Role">
  <span property="roleName">Master Student</span>,
  <link rel="x:furtherDescribedIn" href="http://example.org/MyDescriptions"/>
</span>
you have to prepare a
<span rel="instrument">
  <span about="x:RecLett" typeof="DigitalDocument">
  <span property="name">Recommendation Letter</span>
  <a href="http://school.org/lett" property="url"> (click here for a template)</a>
</span></span>...</span></li>...</ul>...</div>
```

We welcome your applications for the Summer School in Vienna, July 2017

The steps of the procedure are the following:

- First, you need to need to Prepare your documents. In case you are a Master Student, you have to prepare a Recommendation Letter (click here for a template) ......

```
SELECT ?x
WHERE
{x:MasterStud ^:agent/:instrument/:url ?x}

(Where is the template for the document
that must be provided by Master Students?
```

```
x:PrepDocs     a            :Action;
               :name        "Prepare Documents" ;
               :agent       x:MasterStud;
               :instrument  x:RecLett.
x:MasterStud   a            :Role; x:furtherDescribedIn <http://....>;
               :roleName    "Master Student".
x:RecLett      a            :DigitalDocument; :url <http://...>;
               :name        "Recommendation Letter".
```

Information exposed only to human visitors
Information exposed only to software agents
Information exposed to both human visitor and software agents
HTML source with Structured Data Markup
What a human visitor will see (i.e., in the browser)
Client-side rendering & parsing
What a software client/agent will "see" (i.e., the distilled RDF knowledge graph)
Meta layer (ontology terms)
:Action
:Role
:DigitalDocument
Instance layer
a
x:PrepDocs
:name
:instrument
:agent
Prepare Documents
x:MasterStud
x:RecLett
:roleName
x:furtherDescribedIn
:url
Master Student
http://...
Recommendation Letter
Semantics aware client agent
Semantic Query
Graph Serialization

Fig. 1. From Structured Data Markup to machine-readable knowledge graphs

On the client side, the document content is formatted by a browser for a human visitor (mid-left side of the figure); the same document can also be distilled into a knowledge graph (bottom side). The example in the figure employs RDFa as a Structured Data Markup syntax [5], Turtle as a graph serialisation syntax [16] and SPARQL as a graph query language [17]. The meta layer of the graph uses concepts (Action, Role, DigitalDocument) and properties (name, agent, instrument) from the Schema.org ontology. The example also indicates three categories of information that can be embedded in the HTML source: (i) statements that are *human-readable but not machine-readable* (i.e., the first statement, written in natural language); (ii) statements that are *machine-readable but not human-readable* in the browser (i.e., that the Master Student role has additional properties at the given URL); (iii) statements that are *both machine-readable and human-readable* (that the application requires a Master Student to provide a Recommendation Letter and a template is available at the given URL).

The conversion to the "pure" knowledge graph is performed by openly available "distillers" (see [18]). The additional step proposed by the work at hand is to further deserialise the graph in an agile modelling environment – this step is supported by a metamodelling plug-in (details in Section 3). The graph semantics may thus be exposed to an agile Enterprise Modelling method for further analysis or extension. The ideal case is to have models of certain enterprise facets (e.g., work procedures, enterprise resource descriptions) generated out of Web pages where they are already described – e.g., in a Linked Enterprise Data environment [19]. The possibility is currently investigated by the project motivating this work, EnterKnow [20].

2.2 Agility at modelling method level

Agile Modelling Method Engineering [11] is a framework and methodology that allows the customisation and alignment of modelling methods (including their associated modelling language and software) with respect to targeted requirements (on language or model-driven functionality). As the name suggests, AMME transfers agile development principles to the practice of *modelling method engineering* – i.e., an incremental development cycle is applied, based on fast prototyping platforms such as ADOxx [21] and a metamodelling approach that agilely customises the building blocks of a "modelling method" defined in [22]. For the purposes of the work at hand, the modelling language must be tailored to accept the contents made available through Structured Data Markup so that the typical modelling procedure may be supported with automated model generation for specific types of models. A multitude of Enterprise Modelling methods have been developed in the Open Models Laboratory collaborative environment (OMiLAB) [23], some of them developed through the AMME methodology (see an inventory of methods in [24]).

3 Running Example

Fig. 2 showcases a custom-made modelling language to be further used as a basis for the running example. The language is tailored to describe "application procedures" –

i.e., bureaucratic processes of applying for certain programs or benefits, extended with descriptions of required documents and dependencies on responsible persons.

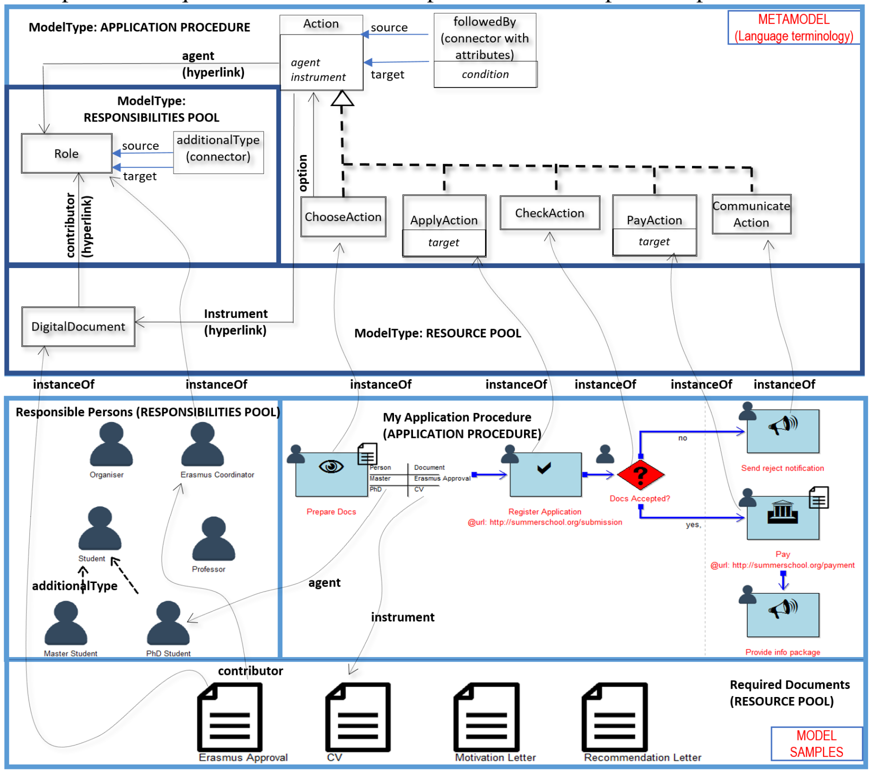


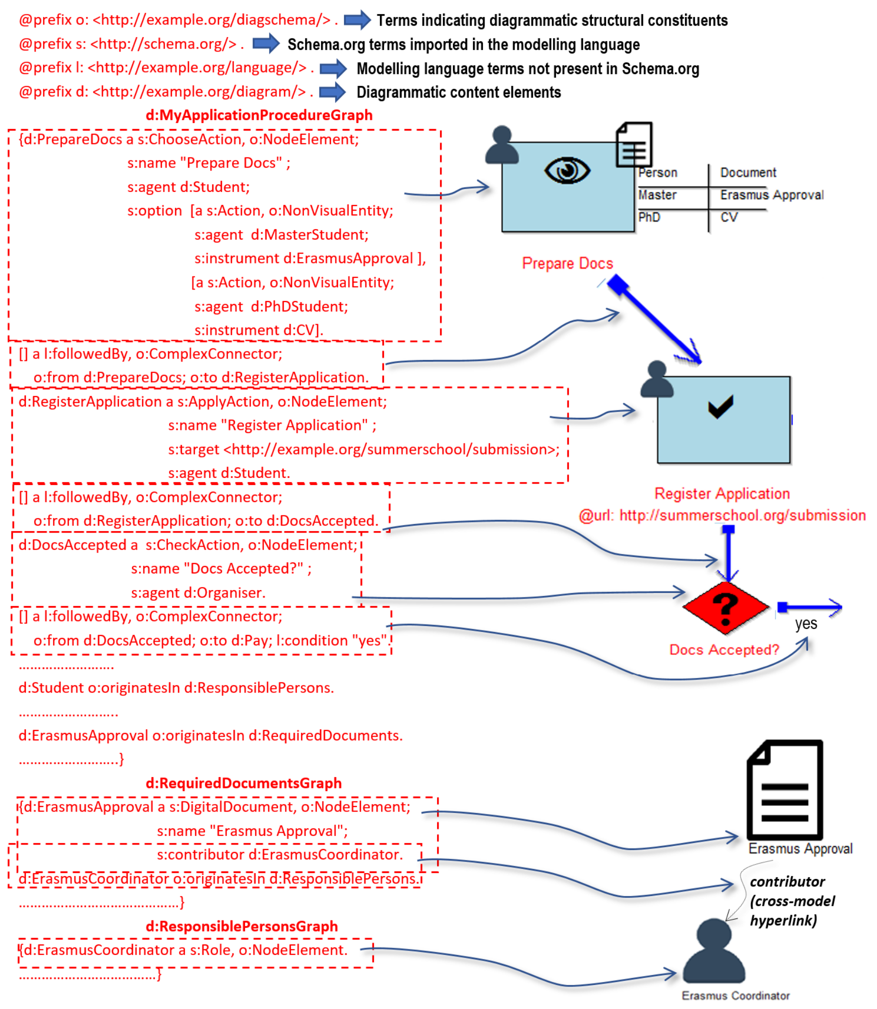
Fig. 2. The Application Procedure Modelling Language

Several customisations may be noticed, compared to the established business process modelling languages: the language concepts (including "action types") are hereby aligned with the Schema.org terminology, with some extensions added whenever properties that are necessary in the modelling environment are not available in Schema.org (e.g., the "condition" attribute, the "followedBy" connector). Certain concepts are enriched with user-editable attributes – e.g., the "ChooseAction" provides a table of alternatives – i.e., what kind of documents must be prepared depending on the applicant type. The language is partitioned into three "views" (types of models): (i) the actual procedure, (ii) the required documents and (iii) the responsibilities. Relations are established across these views (e.g., "agent" to indicate responsibility for an action, "contributor" to indicate required input to a document, "instrument" to indicate documents involved in an action, "target" to indicate the URL where a certain action can be performed on-line). The labels of these concepts and relations are abstracted here to be fully aligned with Schema.org, to keep the example easy to follow. In pragmatic cases, one may encounter cases where they must differ to make models easier to read, hence involving an additional mapping effort. Semantics are reflected in notation by dynamically providing hyperlinks. In terms of syntax, the typical graphical depiction of documentation and responsibilities (e.g., swimlanes) is replaced with hyperlinks, allowing each model type to evolve independently.

The transfer of Structured Data Markup to the modelling environment is governed by a schema comprising two layers: (i) the modelling language terminology aligned with the markup vocabulary (Schema.org in the discussed example); (ii) a fixed RDF schema that maps the markup elements to different types of diagrammatic constituents that are allowed in the modelling environment (e.g., swimlanes, visual connectors, hyperlinks etc.). These are used in the HTML (extended with RDFa) fragment displayed in Fig. 3, which is rendered in the browser as a simple bulleted list of procedure steps (the information visible in the browser is bolded).

```
<div prefix="l: http://example.org/language/ d: http://example.org/diagram/ o: http://example.org/diagschema/"
vocab="http://schema.org/">
<div about="d:MyApplicationProcedureGraph" typeof="l:ApplicationProcedure o:Model">
The steps of the procedure are the following:
<ul><li> First, you need to prepare your documents
  <div about="d:PrepareDocs" typeof="ChooseAction o:NodeElement"><span property="name" content="Prepare Docs"/>
    <div rel="option">In case you are a Master Student
    <div typeof="Action o:NonVisualEntity">
      <span rel="agent"><span about="d:MasterStudent" rel="o:originatesIn" resource="d:ResponsiblePersons"/></span>
      you have to prepare a <span rel="instrument">
      <a about="d:RecommendationLetter" rel="o:originatesIn" href="http://example.org/diagram/RequiredDocuments">
      Recommendation Letter (click for template)</a></span>
      and the    <span rel="instrument">
      <a about="d:ErasmusApproval" rel="o:originatesIn" href="http://example.org/diagram/RequiredDocuments">
      Erasmus Approval (click for template)</a></span>
    </div>
    <div typeof="Action o:NonVisualEntity"> In case you are a PhD Student
      <span rel="agent"><span about="d:PhDStudent" rel="o:originatesIn" resource="d:ResponsiblePersons"/></span>
      you have to prepare a <span rel="instrument">
      <a about="d:CV" rel="o:originatesIn" href="http://example.org/diagram/RequiredDocuments">
      Curriculum Vitae (click for template) </a></span>
      and a <span rel="instrument">
      <a about="d:MotivationLetter" rel="o:originatesIn" href="http://example.org/diagram/RequiredDocuments">
      Motivation Letter (click for template)</a></span>
    </div></div>
  </div>
  <span typeof="l:followedBy o:ComplexConnector">
    <span rel="o:from" resource="d:PrepareDocs"/><span rel="o:to" resource="d:RegisterApplication"/>
  </span></li>
  <li> Next, you need to register your application
  <span about="d:RegisterApplication" typeof="ApplyAction o:NodeElement">
    <span property="name" content="Register Application"/>
    <span rel="agent" resource="d:Student"/> at the following link:
    <a rel="target" href="http://summerschool.org/submission"> Registration </a>
  </span>
  <span typeof="l:followedBy o:ComplexConnector">
    <span rel="o:from" resource="d:RegisterApplication"/><span rel="o:to" resource="d:DocsAccepted"/>
  </span></li>
  <li> Then, we will check your documents
  <span about="d:DocsAccepted" typeof="CheckAction o:NodeElement" rel="agent" resource="d:Organiser">
    <span property="name" content="Docs accepted?"/>
  </span>
  <span typeof="l:followedBy o:ComplexConnector">
    <span rel="o:from" resource="d:DocsAccepted"/><span rel="o:to" resource="d:Pay"/>
    <span property="l:condition" content="yes"/>
  </span></li>....</ul></div></div>
```

Fig. 3. Structured Markup aligned with the Application Procedure Modelling Language



```
@prefix o: <http://example.org/diagschema/> .   Terms indicating diagrammatic structural constituents
@prefix s: <http://schema.org/> .   Schema.org terms imported in the modelling language
@prefix l: <http://example.org/language/> .   Modelling language terms not present in Schema.org
@prefix d: <http://example.org/diagram/> .   Diagrammatic content elements
                d:MyApplicationProcedureGraph
{d:PrepareDocs a s:ChooseAction, o:NodeElement;
                s:name "Prepare Docs" ;
                s:agent d:Student;
                s:option  [a s:Action, o:NonVisualEntity;
                          s:agent  d:MasterStudent;
                          s:instrument d:ErasmusApproval ],
                         [a s:Action, o:NonVisualEntity;
                          s:agent  d:PhDStudent;
                          s:instrument d:CV].
[] a l:followedBy, o:ComplexConnector;
   o:from d:PrepareDocs; o:to d:RegisterApplication.
d:RegisterApplication a s:ApplyAction, o:NodeElement;
                     s:name "Register Application" ;
                     s:target <http://example.org/summerschool/submission>;
                     s:agent d:Student.
[] a l:followedBy, o:ComplexConnector;
   o:from d:RegisterApplication; o:to d:DocsAccepted.
d:DocsAccepted a  s:CheckAction, o:NodeElement;
                s:name "Docs Accepted?" ;
                s:agent d:Organiser.
[] a l:followedBy, o:ComplexConnector;
   o:from d:DocsAccepted; o:to d:Pay; l:condition "yes".
.........................
d:Student o:originatesIn d:ResponsiblePersons.
.........................
d:ErasmusApproval o:originatesIn d:RequiredDocuments.
.........................}
                d:RequiredDocumentsGraph
{d:ErasmusApproval a s:DigitalDocument, o:NodeElement;
                   s:name "Erasmus Approval";
                   s:contributor d:ErasmusCoordinator.
d:ErasmusCoordinator o:originatesIn d:ResponsiblePersons.
........................................}
                d:ResponsiblePersonsGraph
{d:ErasmusCoordinator a s:Role, o:NodeElement.
...................................}
```

Fig. 4. Distilled Application Procedure Model

From the same HTML fragment, RDFa distillers will extract the process description as a machine-readable knowledge graph (Fig. 4). This further becomes the input for the import plug-in prepared for the modelling environment (current implementation is based on ADOxx) to generate model elements. The terms employed in the machine-readable descriptions may be distinguished by their prefices: prefix **s:** (or no prefix in the HTML fragment) corresponds to Schema.org terms; prefix **l:** corresponds to terms that are not found in Schema.org but are necessary in the modelling language (e.g., the connectors between actions); prefix **d:** corresponds to model elements, i.e. those terms that will become diagrammatic constituents (nodes, connectors etc.); prefix **o:** corresponds to the schema of diagrammatic constituents allowed in ADOxx.

The schema of diagrammatic constituents makes an explicit mapping indicating what kind of constructs are to be generated in the modelling tool (rather than relying on labels which may be ambiguous): "Node Element" is the class of all diagrammatic nodes; a "ComplexConnector" corresponds to any connector that has attributes attached to it; Hyperlink properties allow navigation between elements in different models; "NonVisualEntity" is any resource that is not present on the modelling canvas (has no graphical symbol attached), but is editable as table structure annotation attached to model elements. A taxonomy of such constituents is the outcome of analysing the extensive corpus of modelling languages presented in a first volume authored by the OMiLAB global community [24] and is currently being developed towards an ontology of diagrammatic constituents – its origin is in the schema introduced in [25].

4 Discussion on Related Works

Conceptual redundancy often manifests between the *data models driving run-time information systems* and the *conceptualisations governing design-time tools* (e.g., modelling tools). The bridging of these two facets traditionally takes the form of model-driven code generation or of process-aware information systems [26] taking semantic input from some process representation. The proposal of this paper reverses the typical "flow of semantics" by enabling the retrieval of machine-readable semantics from a run-time system (i.e., a running Web page) to a modelling environment tailored through AMME to accommodate the imported knowledge graphs. The reverse flow of RDF graphs, from modelling tools to Linked Data-driven applications was previously discussed in other works [27]. Complementing those works, this paper completes a two-way interoperability channel for modelling environments, a feature traditionally limited to XML-based interoperability.

The proposal may also be positioned as a knowledge conversion step, as it bridges machine-oriented and human-oriented knowledge representations, thus complementing efforts such as knowledge extraction from HTML [28] and potentially supporting knowledge transfer systems [29]. The paper also invites discussion on the semantics of instantiation, previously analysed by [30] - the resources mentioned in HTML documents are dually instantiated in relation to their diagrammatic manifestation and a metamodel. The interplay of Semantic Web and Enterprise Modelling has traditionally focused on the consistency of modelling languages, i.e., their ontological commitment [31], or on the ability to infer relations on certain enterprise model types [32]. Our work pursues a descriptive purpose rather than a prescriptive one, by advocating a more flexible and agile notion of "modelling languages", with models taking input from machine-readable descriptions that are currently spreading across the Web to fulfil the global knowledge graph ambition of the Semantic Web.

5 Conclusions

The paper advocates a streamlining of semantics between the Semantic Web and Enterprise Modelling paradigms. The proposal is supported by a description of technological and methodological enablers - RDF for interoperability and AMME for tailoring a modelling language to targeted semantic requirements. The streamlining proposal presented in the paper is being investigated in the EnterKnow [20] project on more realistic use cases than the showcase modelling language presented here.

Schema.org is frequently enriched and it may take valuable input from the use cases served by the work at hand (e.g., currently the Action concepts in Schema.org are lacking properties required to express action flows). Enterprise Modelling use cases may inspire the addition of concepts for describing enterprise assets.

In terms of limitations, the current implementation is not fully optimised, as it requires manual manipulation to prepare the import – i.e., to add position coordinates for the graphical layout of models (such information cannot be expected to be available in the general case for Structured Data Markup); since named graphs are not supported by the current specification of RDFa, conventions are necessary (e.g., one graph per HTML page). Finally, technical evaluations are still necessary to assess the usability and speed of model generation compared to manual creation of comparable models.

Acknowledgment: This work is supported by the Romanian National Research Authority through UEFISCDI, under grant agreement PN-III-P2-2.1-PED-2016-1140.

References

1. Google Structured Data (2017) https://developers.google.com/search/docs/guides/intro-structured-data
2. W3C (2007) Gleaning Resource Descriptions from Dialects of Languages (GRDDL) https://www.w3.org/TR/grddl/
3. Microformats.org (2017) http://microformats.org/wiki/about
4. Schema.org – official website (2011) http://schema.org/
5. W3C (2015) Rich Structured Data Markup for Web Documents https://www.w3.org/TR/rdfa-primer/
6. W3C (2014) RDF 1.1 – official website, https://www.w3.org/TR/rdf11-concepts/
7. Frank U (2002) Multi-Perspective Enterprise Modeling (MEMO) – Conceptual Framework and Modeling Languages, In: R. H. Sprague Jr. (ed.) Proceedings of the 35th Hawaii International Conference on System Sciences, IEEE, pp 1258-1267
8. Bjeković M, Proper HA, Sottet JS (2014) Embracing Pragmatics. In: Yu E, Dobbie G, Jarke M, Purao S (eds) Conceptual Modeling. ER 2014. LNCS, vol 8824. Springer, pp. 431-444
9. Jeusfeld M A (2016) SemCheck: Checking Constraints for Multi-perspective Modeling Languages. In Domain-Specific Conceptual Modeling, Springer International Publishing, pp. 31-53.
10. Karagiannis D, Buchmann R A, Bork D (2016) Managing Consistency in Multi-View Enterprise Models: an Approach based on Semantic Queries. In: Proceedings of ECIS 2016, Association for Information Systems, p. 53.

11. Karagiannis D (2015). Agile modelling method engineering. In: Karanikolas N, Akoumianakis D, Mara N, Vergados D, Michalis X (eds.), Proceedings of the 19th Panhellenic Conf. on Informatics, ACM, pp 5-10
12. Frank U (2013) Domain-specific modelling languages: requirements analysis and design guidelines. In: Reinhartz-Berger I, Sturm A, Clark T, Cohen Sh, Betin J (eds.) Domain Engineering, Springer, pp 133–157
13. Microformats.org (2017) H-Card http://microformats.org/wiki/h-card
14. Brickley D, Miller L (2014) FOAF Vocabulary Specification 0.99 http://xmlns.com/foaf/spec/
15. Hepp M (2008) Goodrelations: An ontology for describing products and services offers on the web. In International Conference on Knowledge Engineering and Knowledge Management (pp. 329-346). Springer Berlin Heidelberg. http://www.heppnetz.de/projects/goodrelations/
16. W3C (2014) Terse RDF Triple Language https://www.w3.org/TR/turtle/
17. W3C (2013) SPARQL 1.1 Query Language https://www.w3.org/TR/sparql11-query/
18. W3C (2013) RDFa 1.1 Distiller, https://www.w3.org/2012/pyRdfa/Overview.html
19. Wood D (Ed.) (2010) Linking enterprise data. Springer Science & Business Media.
20. EnterKnow project (2017), http://enterknow.granturi.ubbcluj.ro/
21. BOC-Group (2017) ADOxx tool https://www.adoxx.org/live/home
22. Karagiannis D, Kühn H (2002) Metamodelling platforms. In: Bauknecht K, Min Tjoa A, Quirchmayer G (eds.) Proceedings of the Third International Conference EC-Web 2002 – DEXA 2002, LNCS, vol. 2455, Springer, p. 182
23. The Open Models Initiative Laboratory (2017) http://www.omilab.org/psm/home
24. Karagiannis D, Mayr HC, Mylopoulos J (eds.) (2016): Domain-specific Conceptual Modelling, Springer
25. Karagiannis D, Buchmann RA (2016). Linked open models: extending linked open data with conceptual model information. Information Systems, 56:174-197
26. Dumas M, van der Aalst WMP, ter Hofstede AHM (eds.) (2005) Process-Aware Information Systems: Bridging People and Software through Process Technology, New York: Wiley-Interscience.
27. Buchmann R A, Karagiannis D (2017) Domain-specific diagrammatic modelling: a source of machine-readable semantics for the Internet of Things. Cluster Computing, 20(1), 895-908.
28. Wu X, Cao C, Wang Y, Fu J, Wang S (2016) Extracting Knowledge from Web Tables Based on DOM Tree Similarity. In: Lehner F, Fteimi N (eds.), Proceedings of KSEM 2016, Springer, pp 302-313
29. Marumo N, Beppu T, Yamaguchi T (2014) A Knowledge-Transfer System Integrating Workflow, a Rule Base, Domain Ontologies and a Goal Tree. In: Buchmann R, Kifor CV, Yu J (eds.), Proceedings of KSEM 2014, Springer, pp 357-367
30. Laarman A, Kurtev I (2009) Ontological metamodeling with explicit instantiation. In International Conference on Software Language Engineering, Springer, Berlin, Heidelberg,pp. 174-183
31. Guizzardi G (2005) Ontological Foundations for Structural Conceptual Models. CTIT, Centre for Telematics and Information Technology, PhD Thesis Series, No. 05-74
32. Lantow B, Sandkuhl K, Fellmann M (2016) Visual Language and Ontology Based Analysis: Using OWL for Relation Discovery and Query in 4EM. In International Conference on Business Information Systems, Springer, Cham, pp. 23-35