



# The “What” Facet of the Zachman Framework – A Linked Data-Driven Interpretation

Alisa Harkai, Mihai Cinpoeru, and Robert Andrei Buchmann<sup>(✉)</sup>

Faculty of Economics and Business Administration, Business Informatics  
Research Center, Babeş-Bolyai University, Cluj-Napoca, Romania  
{alisa.harkai,mihai.cinpoeru,  
robert.buchmann}@econ.ubbcluj.ro

**Abstract.** The recommended interpretation of the “What” facet in the Zachman Framework is that it serves as a data-centric viewpoint on the enterprise, capturing data requirements across several layers of abstraction – from high-level business concepts down to implemented data entities. In enterprise modelling, these have been traditionally approached through well-established practices and modelling techniques – i.e., Entity-Relationship models, UML class models and other types of popular data model types. In the current context of digital transformation and agile enterprise relying on distributed information systems, certain technological specifics are lost when employing traditional methods acting on a high level of abstraction. For example, the Linked Data paradigm advocates specific data distribution, publishing and retrieval techniques that would be useful if assimilated on a modelling level - in what could be characterised as *technology-specific modelling methods* (mirroring the field of domain-specific languages, but from a technological perspective). This paper proposes an agile modelling language that provides a diagrammatic and, at the same time, machine-readable integration of several of the Zachman Framework facets. In this language, the “What” facet covers concepts met in a Linked Enterprise Data environment – e.g., graph servers, graph databases, RESTful HTTP requests. These have been conceptualised in the proposed language and implemented in a way that allows the generation of a particular kind of code – process-driven orchestration of PHP-based SPARQL client requests.

**Keywords:** Zachman Framework · SPARQL orchestration  
Resource Description Framework · Agile Modelling Method Engineering

## 1 Introduction

With respect to data modelling, we are still relying on the traditional, highly abstract modelling techniques based on, e.g., ER diagrams, UML class diagrams. Technology-specific patterns and properties are not in the scope of such languages. This paper makes initial steps towards filling this gap, considering the context of a Linked Data-driven enterprise, where execution of business processes must be supported by orchestrated Linked Data requests. The key categories of resources in such a

context are graph database servers, graph databases, named graphs, HTTP-based CRUD operations over such graphs - typically performed within a RESTful architecture where combinations of SPARQL queries and HTTP methods, headers and parameters can operate flexibly on the graph storage content [1].

We benefit from methodological enabler such as the Agile Modelling Method Engineering Framework (AMME) [2] and the Resource Description Framework (RDF) [3] – firstly, to customise and extend a business process modelling language with technology-specific concepts and properties included as first-class citizens; secondly, to generate executable PHP-based orchestrations of HTTP requests that could be integrated in scripts supporting various process tasks (e.g., assuming a workflow management system backed by Linked Data resources).

Therefore we hereby propose the notion of “technology-specific modelling languages” (TSML), which reflects the tradition of domain-specific modelling languages (DSML) – however, “domain” is replaced by “technology” since the productivity goal advocated by DSMLs is transferred to a specific technological space (here, Linked Data). Just as in the case of DSMLs, this comes with a trade-off between reusability and productivity, with the second quality being further enhanced by the agility benefits of applying the Agile Modelling Method Engineering Framework (for customising the modelling language and tool). This will further allow the adaptations necessary to extend the proposed modelling tool to generate other kinds of code than the current scope (of PHP scripting). Therefore AMME is a key enabler for the generalisation of the proposal towards other targeted programming environments.

The origins of this work stand in the attempt of establishing an RDF-based approach to representing linked versions of the Zachman Framework enterprise description facets [4] in an agile modelling language [5]. Due to project-based focus [6], only some of those facets have been assimilated in the modelling language – How/When (as processes), Where (as geographical coverage models) and Who (as organisational structures). This paper considers the additional data facet (the “What”), based on the assumption that a Linked Data back-end fuels a business process management system. Data requirements in such a system are not limited to the abstractions allowed by ER or class diagrams – they must also consider the technological context and must be linked to process descriptions in a machine-readable way. Consequently, the mentioned agile modelling language was given the aforementioned TSML quality. This specificity is then further employed by a code generation mechanism that produces PHP-code making use of the EasyRDF library constructs [7] in order to execute orchestrated REST-based SPARQL operations over graph databases.

The Zachman Framework is commonly treated as an ontology, but it is not a formal one in the sense discussed by [8] or [9]. It is also not a modelling method in the sense defined by [10] – we employ it as schema to guide an enterprise metamodel considering the technology-specific definition, design and analysis of architectural information. Our aim is to derive machine-readable knowledge from a technology-specific enterprise architecture design and to query it in order to cross the design-time/run-time bridge with the help of a model-to-RDF transformation plug-in made available for the ADOxx metamodeling platform [11].

The paper is structured as follows: Sect. 2 provides background information about the Zachman Framework, the AMME Framework, the RDF technological space and

about the EasyRDF library employed as a target programming environment for code generation. Section 3 comments on related works. Section 4 discusses the design decisions for the proof-of-concept presented in Sect. 5. The paper ends with conclusions.

## 2 Motivation, Methodology and Enablers

### 2.1 Motivation: The Zachman Framework

The Zachman Framework (ZF) is an enterprise information systems ontological frame, originating in a business system planning project [4], as a method by which information architecture of organisations can be designed and analysed according to an overarching structure serving multiple perspectives. ZF is a matrix of processes, roles, locations, goals, data structures required by the organisation. A common interpretation of the ZF facets is: (i) What – data requirements/services, (ii) How – processes, (iii) Where – locations, (iv) Who – roles and responsibility assignments, (v) When – timing and (vi) Why – goals; and the abstraction layers typically reflect different stakeholder perspectives: (i) execution perspective (contextual level); (ii) business management perspective (conceptual level); (iii) architect perspective (logical level); (iv) engineer perspective (physical level); (v) technician perspective (as built); (vi) enterprise perspective (functioning). Using these levels and views, an enterprise can be described in different ways for different purposes - this has also been recognised in multi-view enterprise modelling [12, 13]. We employ the Agile Modelling Method Engineering framework to produce modelling tools that can capture in a semantically integrated way the facets of ZF – this paper will focus on the What facet, considering the technological specificity of Linked Data (the “domain-specific” quality is translated to a “technology-specific” viewpoint).

### 2.2 Methodology: The Agile Modelling Method Engineering

Agile Modelling Method Engineering (AMME) [2] can be considered a Design Science [14] approach specialised for the creation of modelling methods and modelling tools tailored for various kinds of specificity – including the technological specificity hereby discussed. AMME gives methodologists the ability to create and evolve a modelling tool in an agile manner with respect to semantics, syntax and functionality - an environment such as ADOxx [11] is commonly used for prototyping.

The management practices of today’s enterprises adopt both the notions of Agile Enterprise [15] and Agile Knowledge Management [16], and the Enterprise Architecture Management is based on an agile form of knowledge representation that is synchronised with software engineering processes. Moreover, the Linked Open Models vision [17] shows that models can be exposed to knowledge-driven information systems using Resource Description Framework (RDF) in order to expose model contents

to a code generation framework. In this respect, this paper makes some initial steps targeting a PHP development environment based on the EasyRDF library for Linked Data retrieval.

### 2.3 Technological Enablers

The Resource Description Framework (RDF) is a standard adopted World Wide Web Consortium (W3C) [3] - a family of specifications originally designed as a metadata data model. It evolved as a technological foundation for the Semantic Web and it can be used to describe Web resources in Linked Data environments. Its constructs are graph structures – most nodes in the RDF graphs are URIs (Uniform Resource Identifier) that identifies a Web resource about which machine-readable statements can be stored as subject-predicate-object triples. In Fig. 1 the key aspects are illustrated: (i) a graphical representation of a graph; (ii) a human-friendly RDF serialisation - the Turtle format [18]; (iii) A SPARQL query example.

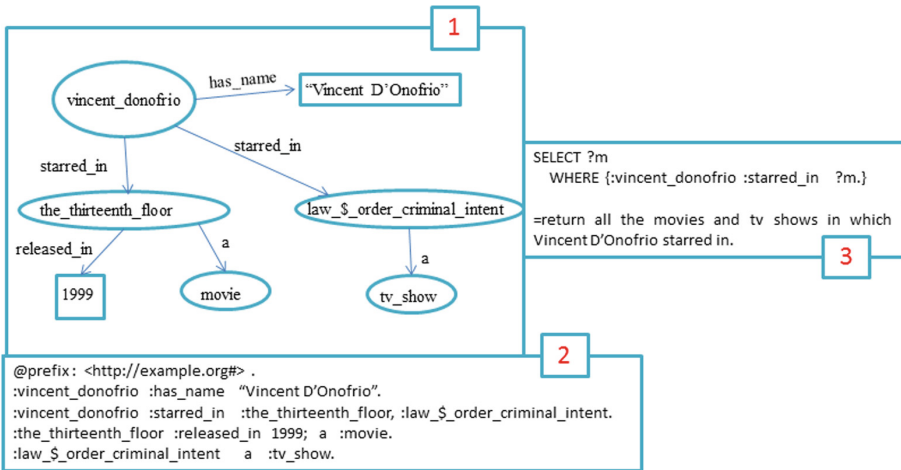


Fig. 1. RDF sample graph, serialisation and SPARQL query example

Another technological enabler is the EasyRDF library [7] which allows PHP scripts to consume and produce RDF based on a series of patterns (that are also captured in our modelling language to support code generation for the use of this particular library). Data is loaded into an EasyRdf Graph object and SPARQL queries can be sent over HTTP using the EasyRdf Sparql Client class providing some object-oriented methods for CRUD operations – some of them executed on full graphs (insert, clear), others sending more granular queries (query, update). The code below exemplifies these CRUD operations.

```

<?php
require 'vendor/autoload.php';
$client1=new EasyRdf_Sparql_Client("http://localhost:7200/repositories/movies");
$client2=new EasyRdf_Sparql_Client("http://localhost:7200/repositories/movies/statements");
$graph=new EasyRdf_Graph();
$graph->load("http://localhost:7200/movies/resource?url=".urlencode("http://www.example.org#mymoviegraph"));
$prefix=new EasyRDF_Namespace();
$prefix->set("", "http://www.example.org#");
$query1="prefix : <http://www.example.org#> describe :vincent_donofrio";
$result1=$client1->query($query);
print $result1->dump();
$query2="delete {the_thirteenth_floor :released_in ?x.} insert {the_thirteenth_floor :released_in 2000.} where
{the_thirteenth_floor :released_in ?x}";
$result2=$client2->update($query2);
$graph->addResource("vincent_donofrio" "lives_in" "Arizona");
$client2->insert($graph, "http://www.example.org#mymoviegraph");
$client2->clear("http://www.example.org#mymoviegraph");
?>

```

The example instantiates two SPARQL clients, one for the read address and one for the write address of a GraphDB query service, which connects to the graph in Fig. 1 with the help of the `EasyRdf_Sparql_Client` constructor. For the first type of operation (Query) we used a string variable with a DESCRIBE query, for the second (Update) we used another string variable with an update (INSERT/DELETE) query which updates the year of the movie, for the third we directly inserted a new statement using the graph URI and for the last we directly deleted the graph using the `Clear()` method.

We do not aim to establish an RDF-centric data modelling technique on the level of abstraction of, e.g. ER, but rather to capture in a modelling language the architectural principles of interacting with Linked Data through the EasyRDF library – however, these are common patterns present in other libraries (e.g., `rdflib` for Python) and further investigation across different programming environments will be necessary to confirm the reusability of these patterns. Moreover, our code generation approach relies on the Linked Open Models vision [17], where several patterns were introduced for converting diagrammatic models into RDF graphs - a plug-in is available for ADOxx in order to achieve this regardless of the type of RDF model. Once models are serialised, they are loaded in GraphDB [19] which is a scalable RDF database to which libraries such as EasyRDF can connect. As we have already discussed, our modelling tool has been extended with a new type of diagram that respects the “What” facet of the ZF considering the technological specificity of Linked Data – i.e., the concepts of graph database, graph, HTTP request.

### 3 Related Works

The notion of “technology-specific modelling” (TSM) is often mentioned in non-diagrammatic modelling approaches [20], but less so in diagrammatic conceptual modelling, although there is rich a tradition of domain-specific diagrammatic modelling (DSM) [21–23], itself deriving from situational method engineering [24]. DSM was probably intended to subsume TSM, however the distinction is worth discussing especially when model-driven software engineering is considered – i.e., specific

technology concepts will benefit from a direct mapping when bridging the design-time and run-time facets. In this work we adopt concepts that are specific to the Linked Data technological space and also have correspondences to the EasyRDF library for PHP, where code must be generated.

Enterprise knowledge can be represented with semantic technology – e.g., RDF graphs coupled with ontologies. Moreover, modelling languages driven by domain-specific requirements are becoming more prominent [23]. The integration of such languages with analytical tools via semantic technology has been discussed before in rather generic terms or decoupled from code generation goals [25, 26]. We think that AMME is a key enabler in adding technology-specific semantics to a modelling language; moreover, we employ it to partition the modelling language with appropriate granularity thus separating modelling concerns pertaining strictly to business views from the technological specificity – at the same time, these models can be linked in meaningful ways so their relations (e.g., from business tasks to HTTP requests to graphs) are traceable for a code generator. This is exploited by the Linked Open Models approach employs the fact that the resulting models/structures are made available to semantic information systems.

The Zachman Framework is used to structure enterprise knowledge according to a two dimensional schema which prescribes six facets and also perspectives for enterprise descriptions but it does not specify neither how to bridge them in machine-oriented ways and make them available to external processing. In time this framework has been extended by other frameworks (e.g., Evernden, The Integrated Architecture Framework) [27]. Moreover, ZF is used to map various processes which are relevant to enterprise architectures (e.g., analysis of the Rational United Process [28], Model-driven architecture [29], TOGAF [30]). Means of bridging the data and process facets have emerged from the process mining community for analytical purposes [31]. A Linked Data-oriented approach focusing on geoprocessing workflows was proposed in [32]. In this context, the contribution of this paper is to describe the data perspective (the What facet of ZF) with means that are amenable to a technology-specific approach to code generation.

## 4 Design Decisions

The proposed modelling language combines ZF, AMME and RDF to create a multi-perspective conceptual frame available to semantic queries, which also exposes specific Linked Data retrieval concepts characterised by technology-specific properties - e.g., HTTP requests characterised by the CRUD operation that is performed and possibly by the explicitly annotated SPARQL query, as required by some business process task. Such technological details are then queried from models and concatenated in EasyRDF code that can actually execute those queries in the sequence dictated by the business process model to which HTTP requests are linked.

We developed the modelling tool starting from a class diagram which represents the meta-model governing the modelling language which can be changed and evolved using AMME framework as new requirements are adopted. Due to the fact that the

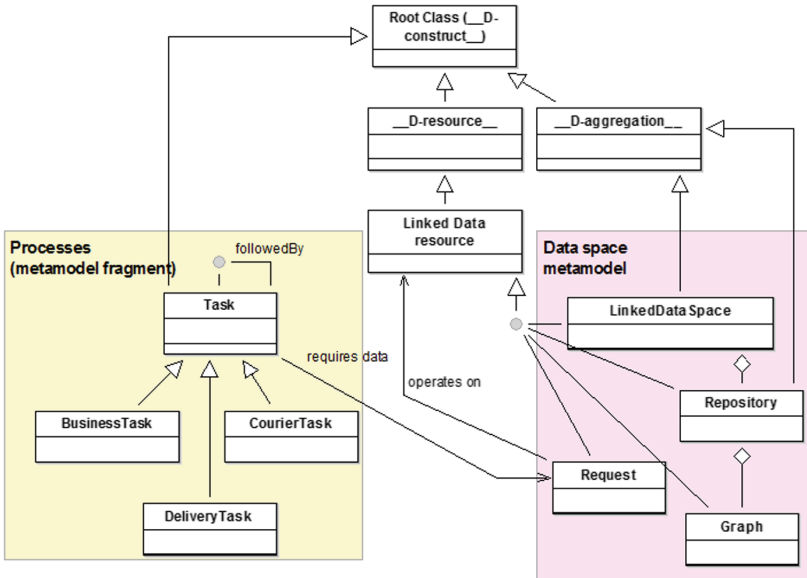


Fig. 2. Metamodel fragment (How and What facets)

work at hand focuses on the What facet of ZF we show in Fig. 2 only a part of the entire meta-model (a fragment which contains the classes for How and What facets).

The metamodelling platform used to develop this agile modelling tool is ADOxx which provides some abstract classes (e.g., \_\_D-construct\_\_, \_\_D-resource\_\_, \_\_D-aggregation\_\_ etc.). \_\_D-construct\_\_ is the root class from which every other class inherits properties and is used in every metamodel. Several types of models have been created to describe transportation processes, locations and participants [5] – however in this paper we only focus on the semantic interface between process tasks and an additional type of model – models of Linked Data resources (the What facet). These have specific symbols that cover ZF facets and in Fig. 3 we emphasised the symbols for the What facet.

Across these types of diagrams hyperlinks are established as depicted in Fig. 4. Different types of links are available: the task Deliver has assigned an employee (Jim) who becomes the responsible person for that task and has as a role Big Car Driver. Moreover the Deliver task has some required data (HTTP) requests, such as Request 2 from the depicted graph which is sent to a REST server that contains a graph database and the graph that we want to query.

All the models can be exported and converted in RDF graphs which are machine-readable ready – i.e., available to queries over the linked models, where certain runtime parameters are stored in the attributes of model elements (e.g., the SPARQL query, the HTTP operation, the address of the graph repository). A graphical representation of the graph which contains the types derived from the meta-model and the links between models is shown in Fig. 4. SPARQL queries can retrieve from these models the overall dependency relations (e.g., which graph is on which database server,





Symbol	ZF facet
task decision  	How + When
graph request  	What
server endpoint (as container) database (as container)	

Fig. 3. Specific symbols covering How and What ZF facets considering Linked Data concepts

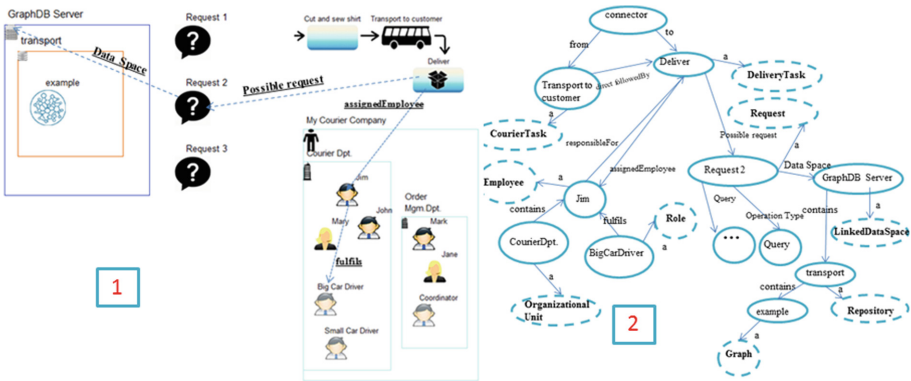


Fig. 4. Linked model fragments (1) and derived RDF graph (2)

which query operates on which graph) – such results are then concatenated in EasyRDF-compatible PHP code to actually execute those queries on those databases, in an order orchestrated according to the process flow linked to those requests.

## 5 Proof-of-Concept

We adopted the case of a transport company that needs to have its courier processes mapped on human resources and addressable locations. Moreover, we want to obtain information which is necessary to perform some activities (e.g., pieces of information about an invoice, information about clients - retrievable from a Linked Data space encompassing multiple graph servers exposing SPARQL endpoints that follow the typical RESTful recommendations for remote data retrieval). The human resources in the example are described both in terms of roles and instance performers, grouped by



departments or organisational units depicted as visual containers, e.g.: Production; Research/Development; Marketing; Finance; Resources.

Figure 5 depicts only two types of diagrams relevant for this work, namely: Model of make-to-order process (M1); Model of linked data space (M2). The first diagram contains a business process made from three types of tasks (i.e., Business Task, Courier Task, Delivery Task) and decisions. These tasks are assigned to user roles (e.g., couriers) and we took into account that at some point in the process the courier needs information about his client in order to deliver the products. The second type of diagram (M2) comprises four major concepts as highlighted in the metamodel: (i) the server concept (**Linked Data Space**), (ii) the graph database concept (**Repository**), (iii) the **Graph** itself concept, (iv) the **Request** concept. Thus, the task Deliver is associated with the Request 2 which has as attributes the EasyRDF operation type (Query) and the query itself annotated as a string in the modelling tool. This request is described as being sent to the GraphDB server which contains the graph database where the relevant graph is hosted (with the information about the clients) – the annotations are illustrated in Fig. 6. Containment relations (inspired by the swimlanes in BPMN) establish the hierarchical subordination of graphs to graph repositories, and further to graph servers. Requests can be sent either directly to graphs (for graph-level operations such as a inserting a whole graph) or to graph databases (for granular SPARQL queries).

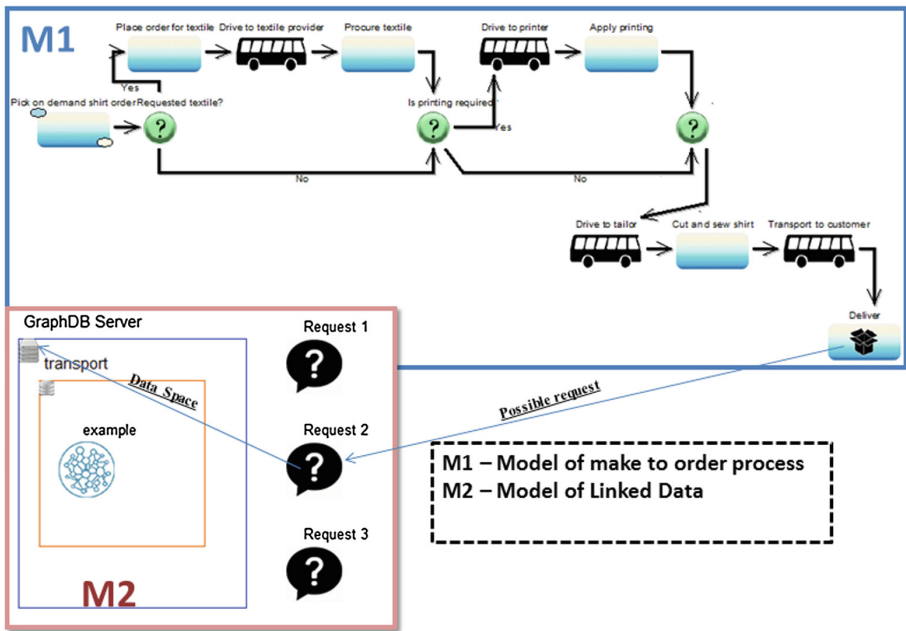


Fig. 5. Samples of business process model (How) and linked data space (What)

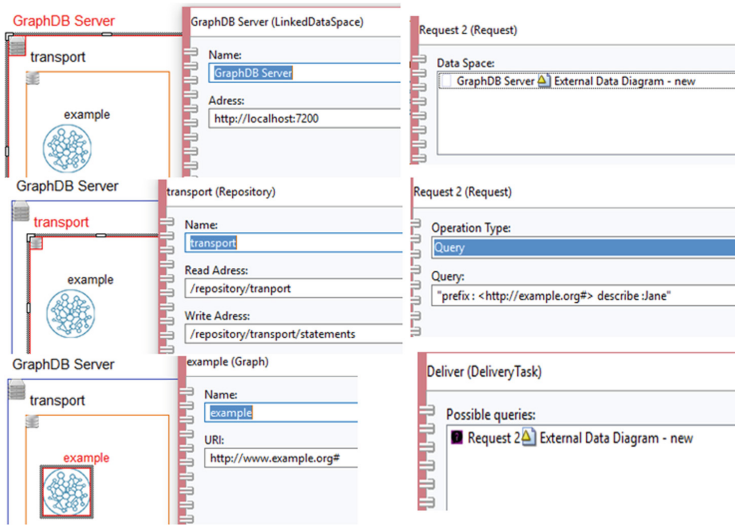


Fig. 6. Example of external data model

```
<?php
require 'vendor/autoload.php';
$graph=new EasyRdf_Graph();
$graph->load("http://localhost:7200/transport/resource?uri=".urlencode("http://www.example.org#Business_Process_Distribute_Shirts"));
$prefices=new EasyRDF_Namespace();
$prefices->set("", "http://www.example.org#");
$part1=getResource("GraphDB Server","Address");
$part2=getResource("transport","Read Address");
$runtimeTarget=$part1 . $part2;
$query=getResource("Request 2","Query");
$client=new EasyRdf_Sparql_Client($runtimeTarget);
$client->query($query);
?>
```

In this piece of EasyRDF code we get all parts of the REST target address in the variable \$runtimeTarget - those address parts are taken directly from the properties of the modelled concepts. Further, we get the query also from the modelled request annotation and we establish a new REST connection with the EasyRdf\_Sparql\_Client class – this will actually run the query to retrieve information necessary in the process task that is served by this script. We used the REST address for read operations (http://localhost:7200/repository/transport) and a DESCRIBE query (annotated in Fig. 6) to obtain the information about Jane (client) which is useful for the driver who has to deliver her some shirts. Other examples:

- insert data {graph :example { :Andreea :hasAddress :NewStreetNo23}} – inserts in the graph a new statement of Andreea (client);
- select distinct ?client (count (?client) as ?count) where {graph <http://www.example.org#example> {?client :hasAddress :Street40} – selects the distinct clients who live at Street40.

## 6 Conclusions

The paper proposes an agile modelling tool as an enabler for technology-specific model-driven engineering. The proposal illustrates the notion of TSML, relying on agile engineering methodologies to adapt a modelling language for code generation scenarios that do not rely on standards. A current limitation is that code generation is only semi-automated, as certain information is annotated manually – i.e., the SPARQL queries. Only the architectural deployment of those queries is described in a diagrammatic manner. A modelling language for SPARQL queries is a key opportunity for future developments. The current practices of code generation rely on standards and stable model compilers confined to the fixed semantic space established by those standards. With this paper we advocate the idea that agile modelling methods combined with the ability to export arbitrary types of models in RDF knowledge graphs could productively feed a code generation framework based on programming libraries whose constructs can be assimilated as first-class citizens in a TSML. AMME is a key ingredient to ensure the fast reprototyping of such tools, thus contributing to a more agile modelling and code generation paradigm compared to traditional model-driven software engineering approaches.

**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. RDF4J Server REST API. <http://docs.rdf4j.org/rest-api>
2. Karagiannis, D.: Agile modeling method engineering. In: Proceedings of the 19th Panhellenic Conference on Informatics, pp. 5–10. ACM (2015)
3. RDF - Semantic Web Standards. <https://www.w3.org/RDF>
4. Zachman, J.A.: Business systems planning and business information control study: a comparison. *IBM Syst. J.* **21**, 31–53 (1982)
5. Harkai, A., Cinpoeru, M., Buchmann, R.A.: Repurposing Zachman Framework principles for “Enterprise Model”-driven engineering. In: Proceedings of ICEIS 2018, pp. 682–689. SCITEPress (2018). <https://doi.org/10.5220/0006710706820689>
6. EnterKnow Project Page. <http://enterknow.granturi.ubbcluj.ro/>
7. EasyRDF Homepage. <http://www.easyrdf.org/>
8. Smith, B.: Beyond concepts: ontology as reality representation. In: Proceedings of the Third International Conference on Formal Ontology in Information Systems, pp. 73–84. IOS Press (2004)
9. Guarino, N.: Formal ontology, conceptual analysis and knowledge representation. *Int. J. Hum Comput Stud.* **5–6**, 625–640 (1995)
10. Karagiannis, D., Kühn, H.: Metamodeling platforms. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) *EC-Web 2002*. LNCS, vol. 2455, p. 182. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45705-4\\_19](https://doi.org/10.1007/3-540-45705-4_19)
11. BOC-Group GmbH, ADOxx platform page – official website. <http://www.adoxx.org/live>

12. Bork, D.: Using conceptual modelling for designing multi-view modelling tools. In: Proceedings of the 21st Americas Conference on Information Systems. Association for Information Systems (2015)
13. Kingston, J., Macintosh, A.: Knowledge management through multi-perspective modelling: representing and distributing organizational memory. *J. Knowl.-Based Syst.* **13**(2–3), 121–131 (2000)
14. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. *J. Manag. Inf. Syst.* **24**(3), 45–77 (2007)
15. Goldman, S., Naegel, R., Preiss, K.: *Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer*. Wiley, New York (1994)
16. Levy, M., Hazzan, O.: Agile knowledge management. In: *Encyclopedia of Information Science and Technology*, pp. 112–117. IGI Global (2008)
17. Karagiannis, D., Buchmann, R.: Linked Open Models: Extending Linked Open Data with conceptual model information. *Inf. Syst.* **56**, 174–197 (2016)
18. RDF 1.1 Turtle. <http://www.w3.org/TR/turtle>
19. GraphDB Homepage. <http://graphdb.ontotext.com>
20. Koehler, A., Peyer, F., Salzmann, C., Saner, D.: Probabilistic and technology-specific modeling of emissions from municipal solid-waste incineration. *Environ. Sci. Technol.* **45**(8), 3487–3495 (2011)
21. Frank, U.: Domain-specific modeling languages: requirements analysis and design guidelines. In: Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. (eds.) *Domain Engineering*, pp. 133–157. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-36654-3\\_6](https://doi.org/10.1007/978-3-642-36654-3_6)
22. Kelly, S., Lyytinen, K., Rossi, M.: MetaEdit+ a fully configurable multi-user and multi-tool CASE and CAME environment. In: Constantopoulos, P., Mylopoulos, J., Vassiliou, Y. (eds.) *CAiSE 1996*. LNCS, vol. 1080, pp. 1–21. Springer, Heidelberg (1996). [https://doi.org/10.1007/3-540-61292-0\\_1](https://doi.org/10.1007/3-540-61292-0_1)
23. Karagiannis, D., Mayr, H.C., Mylopoulos, J.: *Domain-Specific Conceptual Modeling*. Springer, Cham (2016). <https://doi.org/10.1007/978-3-319-39417-6>
24. Kumar, K., Welke, R.: Methodology engineering: a proposal for situation-specific methodology construction. In: Cotterman, W.W., Senn, J.A. (eds.) *Challenges and Strategies for Research in Systems Development*, pp. 257–269. Wiley, New York (1992)
25. Blackburn, M.R., Denno, P.O.: Using Semantic Web technologies for integrating domain-specific modeling and analytical tools. *Procedia Comput. Sci.* **61**, 141–146 (2015)
26. Nassar, N., Austin, M.: Model-based systems engineering design and trade-off analysis with RDF graphs. *Procedia Comput. Sci.* **16**, 216–225 (2013)
27. Schekkerman, J.: *How to Survive in the Jungle of Enterprise Architecture Frameworks*. Trafford, Bloomington (2003)
28. de Villiers, D.J.: Using the Zachman Framework to assess the rational unified process. In: *The Rational Edge*, Rational Software (2001)
29. Frankel, D.S., Harmon, P., Mukerji, J., Odell, J., Owen, M., Rivitt, P., Rosen, M., Soley, R. M.: *The Zachman Framework and the OMG's model driven architecture*. White paper. *Business Process Trends* (2003)
30. The Open Group: *ADM and the Zachman Framework* (2017). <http://pubs.opengroup.org/architecture/togaf8-doc/arch/chap39.html>
31. Alizadeh, M., Lu, X., Fahland, D., Zannone, N., van der Aalst, W.M.P.: Linking data and process perspectives for conformance analysis. *Comput. Secur.* **73**, 172–193 (2018)
32. Yue, P., Guo, X., Zhang, M., Jiang, L., Zhai, X.: Linked Data and SDI: the case on Web geoprocessing workflows. *ISPRS J. Photogramm. Remote Sens.* **114**, 245–257 (2016)