Applied Intelligence (2023) 53:8510–8535
https://doi.org/10.1007/s10489-022-04440-0

The analysis of data metamodels' extensional layer via extended generalized graph

Marcin Jodłowiec[1] · **Marek Krótkiewicz**[1] · **Piotr Zabawa**[1]

Accepted: 28 December 2022 / Published online: 15 March 2023

Abstract

There are several limitations known in data modeling discipline, which are related directly to the traditionally used data modeling languages expressiveness. The strong limitations of the expressiveness of the existing well known data modelling languages combined with the lack of a very general universal data modeling language have negative impact to modelling naturalness. As the result of mentioned limits the reality must be transformed to avoid (workaround) the limits introduced by the modelling language. In turn, the transformation process requires extra effort. The problem is strengthened by the lack of mechanisms, which can be used to measure the expressiveness of a particular data modeling language. Some limitations of the existing data modeling languages result from both their metamodel (abstract syntax) and model (metamodel instance) graph-like structure constraints. This kind of limits also has negative impact to a domain-specific modeling naturalness. The paper addresses all problems mentioned above. The problems can be solved with the help of the EGG data modeling language introduced in the paper. First, a universal and customizable EGG data modeling language together with the customization mechanisms (extensions and generalizations) is introduced. According to the first usage scenario the EGG may be applied for domain-specific data modelling tasks in place of other data modeling languages. Second, the paper proposes and applies (for some data modeling languages: RDF, XML, RDBM, UML and AOM) a novel concept of measuring and comparing data modelling languages via mapping their metamodels to the EGG metamodel. So, according to the second usage scenario the EGG metamodel can be used as a reference metamodel for the data modeling language expressiveness comparative studies. It may also support the decision process when a data modeling language must be chosen for a particular domain-specific data modeling task. Third, the EGG introduced in the paper helps to avoid transforming reality to the needs resulting from the data modeling language as the EGG is general enough for the domain data modeling task. Complete abstract syntax of the Extended Generalized Graph is introduced and is expressed through its implementations in terms of the Association-Oriented Metamodel and the Unified Modeling Language. Semantics of each syntactical category of abstract syntax is described. Two complete concrete syntaxes for the Extended Generalized Graph are also introduced in the paper. The case studies related to both social network and knowledge modeling illustrate the applicability and usefulness of the EGG. Abstract syntax is compared to several other metamodels. The comparative study of the case study models created first in different metamodels and then expressed in the Extended Generalized Graph metamodel is summarized quantitatively in the form of a proposed measure.

Keywords Graph · Hypergraph · Metamodel · Data structure · Data model · Metamodeling · Extended generalized graph · Knowledge model · Social network · Abstract syntax · Concrete syntax · Semantics

This article belongs to the Topical Collection: *Emerging Topics in Artificial Intelligence Selected from IEA/AIE2021*
Guest Editors: Ali Selamat and Jerry Chun-Wei Lin

Marcin Jodłowiec, Marek Krótkiewicz and Piotr Zabawa contributed equally to this work.

✉ Marcin Jodłowiec
marcin.jodlowiec@pwr.edu.pl

Extended author information available on the last page of the article.

1 Introduction

Data structures play important role in several domains. They are represented by graphs as well as by more general data structures like hypergraphs [1, 2] and ultragraphs [3].

One domain the data structures mentioned above are extensively used is software engineering, which explores different data structure representations when both the metamodels for modeling languages and the models, which are compliant to the metamodels are to be constructed.

Other domains constitute very interesting application fields of the data structure representations as well. A good example of such a domain is knowledge mod0eling, which is classified as a part of *Artificial Intelligence* (AI).. Knowledge modeling requires and explores extensively both data representations (not necessarily data structures representation) and data themselves for the purpose of representing knowledge. Knowledge representation is, in turn required by expert systems also well known in AI domain.

This paper is an extension of the conference paper [4] and is dedicated to the presentation of the latest research results led by the authors in the domain of data representation. A new approach to data structures representation was introduced in [5] and then extended in [4]. The subjected data representation named *Extended Generalized Graph* (EGG) is defined precisely and completely in this paper. Another important approach to knowledge representation, the *Resource Description Framework* (RDF) is explored in the paper and is compared to the EGG data representation. The EGG data structures representation is also used in the paper for an illustration of its applicability for knowledge representation. There are some commonly used approaches to data representation, which are traditionally used in knowledge modeling. Thus, the EGG-based approach constitutes an alternative concept of data representation in knowledge modeling domain. A shortened version of the application case study presented in [4] is extended in this paper. Another case study, which is related directly to the knowledge modeling domain is introduced in the paper as well to illustrate the features and possible advantages of the EGG when applied to this AI domain.

1.1 Application of the metamodels for data modeling

Metamodel is a key notion in software engineering domain. The metamodel term refers to a representation (a model) of a modeling language. However, this term may be also related to several data representation languages, which must have their metamodels. A discussion of several metamodels and the data modeling languages was presented in [4]. Such the data modeling languages like *Extensible Markup Language* (XML), *Relational Database Model* (RDBM), *Unified Modeling Language*™ (UML) (when applied to class/object models) and *Associaton-Oriented Metamodel* (AOM) were taken into account. The set of these metamodels is extended in the paper by the RDF which is also used to represent data. Moreover, it is worth noticing that the metamodels constitute the data models as well.

1.2 Motivation for representing metamodels in terms of the EGG

There are several data modeling languages known from the software engineering domain. The UML [6] is one best known. Its characteristic feature is that the metamodel structure has the form of a graph. However, this language allows to create models of more complex structures. Nevertheless, extensional data structures in UML are very simple and described very generally in the specification. Formally, and within UML's abstract syntax it is possible to model higher-level graph structures, but in practice it is hard to find a suitable semantics for such the structures, therefore the modellers utilize that very rarely.

Despite the generality of graphs they are not general enough to model a complex reality correctly. Some simplifications are made in such the case what results in the reality deformations. One example of such the simplifications is the reification of the n-ary relationships [7]. There are two main reasons of applying graphs for a more complex problem in software engineering. One results from the limited competences in the modeling domain. Another one results from the limits of the contemporary modeling languages. The most critical limit is the unavailability of the hypergraphs [8] and/or the ultragraphs (a.k.a. the ubergraphs) [9, 10]. These graph generalizations are known from the graph theory but are rarely applied when modeling. The redundancy of the modeling language constructs being the result of the more complex metamodels makes it possible to optimize models and to fit them to the real needs better.

Some extensions and generalizations of the graph structures, named EGG were introduced in [5]. Some more complex graph-like structures with data associated with these structures can be created with the help of the EGG notion in a uniform way. Modeling languages are applied to order reality but, paradoxically the metamodels, even as complex as the UML, are out of the scope of such ordering efforts. That is why the EGG concept was introduced.

1.3 Research problem

One important feature of knowledge representation used in AI domain is its universality and flexibility. Both features are achieved through the simplicity of data representation and the lack of applying a general enough representation of their structure. Data is usually represented by a flat structure of 3-tuples (subject, predicate and object) to represent knowledge.

The authors argue that the lack of a more complex structural data representation results from the lack of the

general enough and configurable data representation model. The research problem that has been undertaken is as follows. There is no model for structure for data and knowledge representation that is highly configurable and which has varying level of expressiveness. Thus, data metamodels which are designed under different assumptions are hard do compare in terms of data structures that they allow to model. The authors have proposed an extensible and general network-like structure which could be configured to express complex data. The proposed solution offers such a universal and a uniform approach to represent data.

Thus, the EGG has a chance to be widely used in the AI discipline for knowledge representation.

1.4 Contributions

There are several contributions of the paper. One are the implementations of EGG's abstract syntax in terms of the AOM and the UML modeling languages as well as definitions of EGG's concrete syntaxes - a symbolic and a graphical ones. The previously published definition and the metamodel implementations [4, 5] are completed in the paper. Other contribution is a description of EGG categories semantics. An illustrative discussion of the applicability of the EGG for social networks (extended from [4]) and knowledge modeling (introduced in the paper) are also novel. The set of the analysed modeling languages presented in [4, 5] is extended in the paper by the RDF as a knowledge modeling language with its metamodel. The RDF categories are mapped to the EGG categories and the evaluation of data complexity in the RDF data metamodel in relation to the EGG categories is led, which is also a novel element of the paper. Another new concept presented in the paper is the approach to measuring the expressiveness of a particular metamodel through a model created in this metamodel, transformation of this model to the one expressed in terms of the EGG metamodel and then measuring the distance between both models.

Section 3.1 contains more traditional and less traditional implementations of EGG abstract syntax. Definitions of two EGG concrete syntaxes: more universal symbolic concrete syntax (Section 3.2.1) and more readable graphical concrete syntax (Section 3.2.2) are introduced after implementations of EGG abstract syntax. Several languages are mapped in Section 4 to the EGG in order to show how the categories from these languages can be expressed in terms of the very general EGG categories. Then, some case studies are presented – in Section 5.1 a way of using the EGG for a social training group (in relation to the case study from [5]) is illustrated and in Section 5.2 the application of the EGG for representing knowledge is shown. In the rest of the text the complexity of the chosen modeling languages is analysed. The complexity is related for each language to the usage level of all possible categories offered by the EGG. Finally, the conclusions arising from the achieved research results are presented.

2 Related work

There are several domains and threads in the scientific literature, which are related to the subject of the paper.

The fundamental question regarding the approach to representing the data structures, which is used in the paper is formulated in [11]. The question is if the mathematical models or metamodels should be applied to define the data structures in computer science (informatics). The paper shows the fact, which is supported by the examples that the metamodels are more useful in the IT community. Because the metamodels not only better fit well-established solutions in computer science, e.g. the programming languages, but also thanks to their high readability and support with graphic notations, they better meet the needs of a wide group of users. In the cited work, a number of approaches used to define data structures is given.

The authors of this publication share the opinions contained in the work [11] – therefore, after defining the EGG with the help of some set-theoretic concepts, they consistently apply an approach based just on metamodeling. Moreover, in contrast to the work [11], the authors propose a far-reaching generalization of a simple definition of a graph as an alternative approach to the considerations and the comparisons presented in [11].

There are also several publications dedicated to the different graph representations, including some generalizations of graphs, like hypergraphs [1, 2] and ultragraphs [3], which were already mentioned in Section 1. A different and a very general approach to the graph-like representations was proposed by the authors in [4, 5]. Instead of defining the special cases of even very general representations the mentioned approach relies on defining a general data model together with some mechanisms of limiting its generality and extended character to the needs required by a user of the approach.

Another thread of publications is related to the data modeling problems. One important publication is [12], which may constitute a reference for the elaboration of the tools and the standards originated by the EGG-related approach. An interesting problem connected to the transformations of data models expressed in different metamodels is analysed in [13]. This problem can be related to the data model representation introduced by the EGG as one application of the presented concept of the data representation.

Large group of publications is connected to the particular application domains, also mentioned in Section 1, where

data modeling is explored and plays an important role for these domains.

One important domain where the data representation plays the key role is defining the modeling languages and the models in these languages, which is characteristic for software engineering and can be used for the model-driven automated generating of the software systems [14]. There are some commonly known standards in the software engineering domain, which are managed and developed by the *Object Management Group*™ (OMG). Some standards, like the *Meta-Object Facility*™ (MOF), UML and several other ones support the approach to the software development processes according to the *Model Driven Architecture*® (MDA) [15]. All standards mentioned above and especially their metamodels are based on the basic version of the graph notion. This fact limits the possibility of full use of the modeling potential. There are two kinds of the modeling languages known from the software engineering domain, namely the *Domain-Specific Modeling Languages* (DSML) explored in [16, 17] as well as the *General Purpose Modeling Languages* (GPML) mentioned in [18]. Application of the EGG for both the DSML and the GPML gives a chance to break the limits of these languages and introduce a basis for a common approach to constructing them.

A domain of special interest in the paper is knowledge modeling as a branch of artificial intelligence. One important knowledge representation technique is semantic networks. Usually, in semantic networks the binary edges are assumed according to the RDF standard discussed in the paper. However, there are the hypergraph generalizations known in the semantic network models [19]. Semantic networks are well established in computer science and information systems [20, 21] and are still subject of interest [22].

Data models of the higher expressive power, such as the hypergraph models are used also in network analysis [23], scientometrics [24], natural language processing [25].

It should be underlined that the importance of data modeling is confirmed by extensive standardization efforts already mentioned above. In this paper we address the limits of the standards, which result in the difficulties when the domains relying on the standards are explored and developed.

Taking into consideration all the state-of-the-art solutions mentioned above, they (explicitly or implicitly) implement the graph-like structures with different expressive power. In contrast to the state-of-the-art papers, our proposed solution can be perceived as a single formalism which has the superset of graph-like properties and encapsulates them in the form of the generalizations and the extensions. To describe a graph or a hypergraph, or even an ultragraph structure one can use EGG formalism and configure its properties in order to be well-suited to the modelled system. Due to this fact, the EGG is featured by universality in terms of capturing the graph orientation with the different properties. On the other hand, the specific solutions are restricted to their own expressive power and cannot cross beyond them.

3 Definitions of the extended generalized graph (EGG)

The EGG definition was introduced, in its original version in [5] and then it was slightly expanded in [4]. A characteristic feature of these definitions was to focus more on the definition of the graph itself than on the overall concept of the EGG. In this article, the above-mentioned definitions have been extended to include just the EGG notion as a whole. The new concepts introduced to the EGG are expressed in the paper in the form of the AOM and the UML implementations of the EGG metamodel.

The very idea of the EGG concept is based on certain assumptions, which are summarized below. The key such an assumption is that each category of a metamodel should have a function (it can be expected of any metamodel, not only just the EGG one). These functions may be characterized for the particular EGG categories as follows:

- the interrelating categories:
  - composition – a lifetime dependence and exclusive ownership relationship;
  - reference – no lifetime dependence and no exclusiveness relationship;
- the interrelated categories:
  - *Container* is a category physically COMPOSING (in an extensional sense) all other categories, that is it owns all other categories through the composition relationship;
  - *Egg* is a GROUPING category, i.e. it is something like a set, it semantically groups the categories on the basis of a reference relationship;
  - *Edge* is a CONNECTING category, and its semantics focuses on the fact that it represents the n-ary relationship. It has a referential relationship nature. Within the EGG concept, it can also be a connected category;
  - *Vertex* is a CONNECTED category, semantics focuses on the fact that it can participate in the reference relationships, it cannot connect other categories;

– *Label* is a DESCRIBING category, i.e., semantically speaking, it is a place for information that aims to associate any information with *Egg*, *Edge*, *Vertex*. *Label* cannot reference or be referenced (in terms of reference relationship) apart from the one that semantics is about to describe, and therefore provide additional information in any form.

The formal definition of the concept of the EGG characterized above is presented in Section 3.1.

3.1 EGG abstract syntax definitions

EGG abstract syntax in version focused on the graph structure itself has been defined in the previous publications [4, 5] in terms of the set theory. In this paper the enriched metamodel is specified from the perspective of two metamodel implementations. First, the AOM-based implementation and then the UML-based implementation are introduced.

3.1.1 AOM implementation of EGG abstract syntax

The AOM is a novel data metamodel, which has been proposed by Krótkiewicz as a unified platform for data modeling with high semantic capacity. This metamodel covers a symbolic formalism called the *Associaton-Oriented Formal Notation* (AFN), used to model definition. Both syntax and semantics of the AFN can be found in [26].

First, the AOM-based implementation (Fig. 1). The EGG was implemented in the AOM using the AFN as follows:

$$\Diamond Edge \left\langle \begin{array}{l} [1] \xrightarrow{+connection(\Diamond Connection)} [*]\,\Diamond Connection \\ [1..*] \xrightarrow{+Element::descr(\Box Label)} [*]\,\Box Label \end{array} \right\rangle; \quad (1)$$

$$\Diamond Egg \left\langle \begin{array}{l} [*] \xrightarrow{+aggr(\Diamond Edge,\Diamond Egg,\Diamond Vertex)} [*]\,\Diamond Element \\ [1..*] \xrightarrow{+Element::descr(\Box Label)} [*]\,\Box Label \end{array} \right\rangle; \quad (2)$$

$$\Diamond Element\,[1..*] \xrightarrow{+descr(\Box Label)} [*]\,\Box Label \quad (3)$$

$$\left\{ \begin{array}{l} \Diamond Edge \\ \Diamond Egg \\ \Diamond Vertex \end{array} \right\} \xrightarrow{rf^v} \Diamond Element^{\varnothing} \quad (4)$$

$$\Diamond Vertex\,[1..*] \xrightarrow{+Element::descr(\Box Label)} [*]\,\Box Label \quad (5)$$

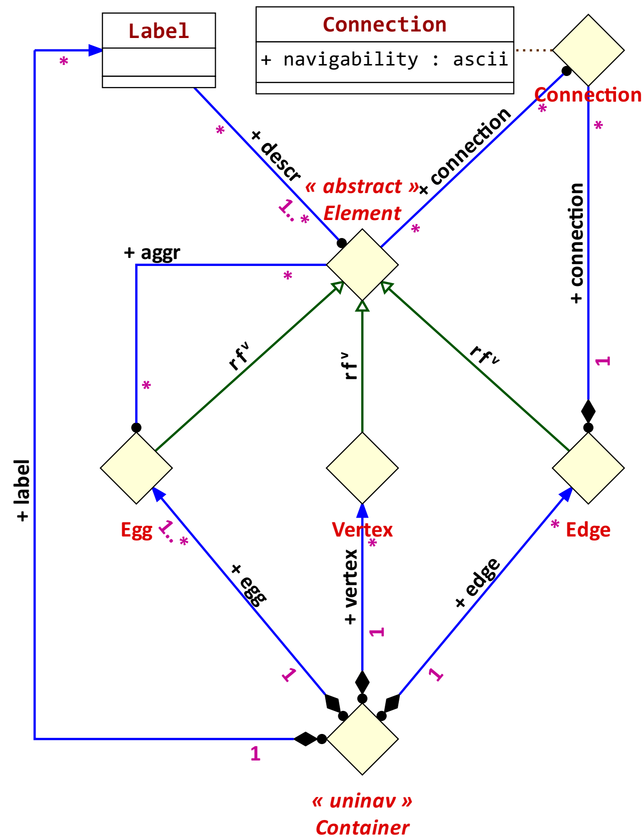


Fig. 1 The AOM implementation of the EGG concept metamodel

$$Connection \cdots \Diamond Connection\,[*] \xrightarrow{+connection(\Diamond Edge,\Diamond Egg,\Diamond Vertex)} [*]\,\Diamond Element \quad (6)$$

$$\Diamond Container \left\langle \begin{array}{l} [1] \xrightarrow{+vertex(\Diamond Vertex)} [*]\,\Diamond Vertex \\ [1] \xrightarrow{+edge(\Diamond Edge)} [*]\,\Diamond Edge \\ [1] \xrightarrow{+egg(\Diamond Egg)} [1..*]\,\Diamond Egg \\ [1] \xrightarrow{+label(\Box Label)} [*]\,\Box Label \end{array} \right\rangle; \quad (7)$$

$$\Box Label\,\langle\rangle; \quad (8)$$

$$\Box Connection \left\langle +navigability : ascii \right\rangle; \quad (9)$$

All roles in the AOM model are referential, i.e. they have the references to the objects. The association-side composition $\Diamond Container$ was used in $+egg$, $+vertex$, $+edge$ and $+label$ roles as the physical instances container to implement the *Container*. Each such the instance belongs to exactly one $\Diamond Container$. The $\Diamond Element$ abstract association implements the polymorphism mechanism in relation to $\Diamond Egg$, $\Diamond Vertex$ and $\Diamond Edge$, which allows $\Diamond Egg$ to

aggregate these instances using the $+aggr$ role. In addition, the $\Diamond Edge$ association is able to connect the listed instance categories via the $\Diamond Connection$ association.

3.1.2 UML implementation of EGG abstract syntax

The UML implementation of the EGG consists of both the UML class diagram expressed in Fig. 2 and the *Object Constraint Language* (OCL) specification of the constraints for the model presented on the diagram.

The diagram from Fig. 2 is adorned by the following constraints which are expressed in the natural language first and then are specified formally via the OCL expressions:

- there is exactly one instance of the $Container$
  context Container inv UniqueContainer: self.oclType().allInstances() -> size() = 1
- there is at least one Egg which is contained in the $Container$
  context Container inv MinCountEgg: self.element.graph -> size() > 0
- there is exactly one Egg (the root Egg), which is not contained in other Egg
  context Egg inv RootEgg: Egg.allInstances() -> select(graph -> size() = 0) -> size() = 1
- Egg cannot be self contained,
  context Egg inv NotSelfEgg: not self.aggr -> includes(self)
- Egg cannot be cyclic in terms of the $aggr$ aggregation,
  context Egg inv AcyclicEgg: self.graph -> closure(graph) -> excludes(self)
- each $Vertex$ must be contained in at least one Egg
  context Vertex inv ContainedVertex: self.graph->size() > 0
- each $Edge$ must be contained in at least one Egg
  context Edge inv ContainedEdge: self.graph->size() > 0

In order to systematize terminology related to EGG abstract syntax, the following terms are introduced, which are used later in the text:

- EGG concept – the whole data structures modeling concept based on EGG abstract syntax,
- EGG metamodel – a definition of the EGG concept, which is composed of the EGG abstract syntax and semantics,
- EGG structure – a term referring to the data structure represented by EGG equivalent to the EGG metamodel instance,
- EGG category – any category of EGG abstract syntax, that is Egg, $Vertex$, $Edge$, $Label$, $Element$,
- EGG category instance – the concretized Egg, $Vertex$, $Edge$, $Label$ categories,
- EGG element – any subcategory of $Element$, that is Egg, $Vertex$, $Edge$,
- *reference* – a no lifetime dependence and no exclusivity relationship,
- *composition* – a lifetime dependence and an exclusive ownership relationship,
- *aggregation* – a kind of the reference relationship,
- *connection* – a kind of the reference relationship,
- *description* – a kind of the reference relationship.

3.2 EGG concrete syntax definitions

This section contains a presentation of two concrete syntaxes (notations) of the EGG. Both concrete syntaxes are compliant to abstract syntax defined in Section 3.1. First, symbolic concrete syntax and then graphical concrete syntax are presented.

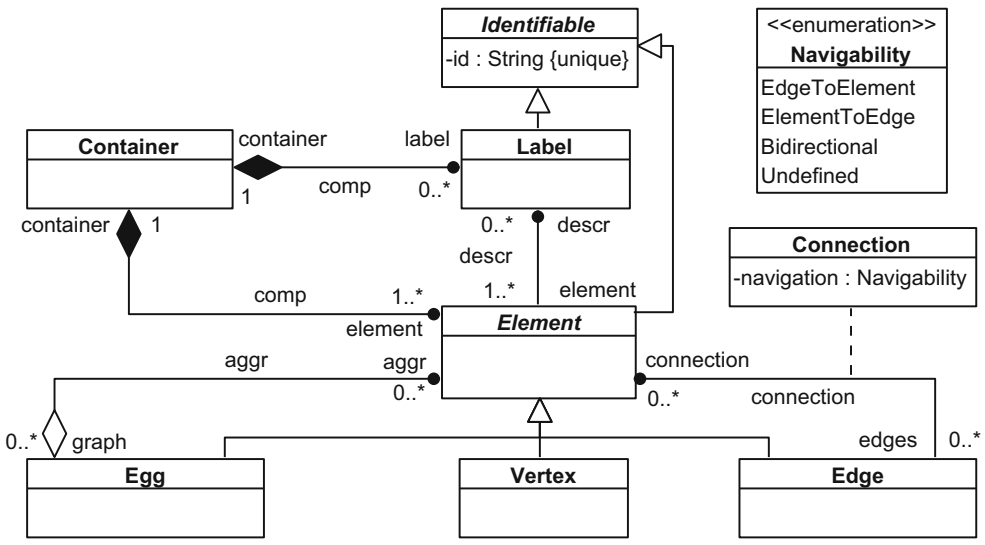


Fig. 2 The UML 2.5.1 and the OCL 2.4 implementation of the EGG concept metamodel

3.2.1 Symbolic EGG concrete syntax

The EGG-based structures are extensional and, as such, they have no constraints beyond those defined in the EGG abstract syntax definition.

The symbols and the rules used:

- The metamodel category instance identifier is unique within the model. The identifier may have any symbolic form.
- The symbols $\langle\cdot\rangle$ represent a comma-separated list of components, e.g. $\langle a, b, c\rangle$.
- The symbols $\{\cdot\}$ are brackets that group items of the same category, such as $\{a, b, c\}$. The curly brackets can also be used to group the items connected with the same type of $connection$.
- The references for each category are denoted as follows: $\blacktriangle g - Egg$, $\bullet v - Vertex$, $\blacklozenge e - Edge$, $\blacksquare l - Label$.
- If several consecutive instances have the same category, then the category designation $\blacktriangle$, $\bullet$, $\blacklozenge$, $\blacksquare$ can be drawn in front of a curly bracket delimiting a set of the instances.

  Some examples:
  $\{\blacktriangle g_1, \blacktriangle g_2\} \equiv \blacktriangle \{g_1, g_2\}$,
  $\{\bullet v_1, \bullet v_2\} \equiv \bullet \{v_1, v_2\}$,
  $\{\blacklozenge e_1, \blacklozenge e_2\} \equiv \blacklozenge \{e_1, e_2\}$,
  $\{\blacksquare l_1, \blacksquare l_2\} \equiv \blacksquare \{l_1, l_2\}$.
- If several consecutive instances are connected by the same kind of a connection, the connection designation can be: $\multimap$, $\rightarrowtail$, $\twoheadleftarrow$, $\twoheadleftarrow\!\!\!\shortmid$, drawn in front of a curly bracket that delimits a set of instances.

  Some examples: $\{\multimap \blacktriangle g_1, \multimap \blacklozenge e_2\} \equiv \multimap \{\blacktriangle g_1, \blacklozenge e_2\}$, $\{\rightarrowtail \bullet v_1, \rightarrowtail \bullet v_2\} \equiv \rightarrowtail \bullet \{v_1, v_2\}$.
- An instance of a given category is marked with an underlined symbol of the category: $\blacktriangle$, $\bullet$ or $\blacklozenge$ together with an identifier and a symbol of the list $\langle\cdot\rangle$, in which there are the components of the instance.

  Some examples:

  $\underline{\blacktriangle\, g}\,\langle\cdot\rangle$ – an Egg category instance with the g identifier,
  $\underline{\bullet\, v}\,\langle\cdot\rangle$ – a $Vertex$ category instance with the v identifier,
  $\underline{\blacklozenge\, e}\,\langle\cdot\rangle$ – an $Edge$ category instance with the e identifier.

  In the case of the $Label$s, the list symbol is not added as it does not aggregate any EGG categories.

  An example: $\underline{\blacksquare\, l}$ – a $Label$ category instance with the l identifier.

  The labels are connected with the $\cdots$ symbol, which is intended only to visually distinguish the annotating relationship of the labels from the other relationships. It is only the so-called syntactic sugar, as the labels cannot be aggregated by the Egg structures and formally there is no need to use a special notation for the $descr$ relationship.

  Some examples

  $\underline{\blacktriangle\, g}\,\langle\cdots\blacksquare\,\{l_1, l_2\}\rangle$ – an Egg category instance with the g identifier is described by the labels with the l_1 and the l_2 identifiers,
  $\underline{\bullet\, v}\,\langle\cdots\blacksquare\,\{l_1, l_2\}\rangle$ – a $Vertex$ category instance with the v identifier is described by the labels with the l_1 and the l_2 identifiers,
  $\underline{\blacklozenge\, e}\,\langle\cdots\blacksquare\,\{l_1, l_2\}\rangle$ – an $Edge$ category instance with the e identifier is described by the labels with the l_1 and the l_2 identifiers.
- The $Label$ values can be optionally specified after the colon symbol.

  An example $\underline{\bullet\, v}\,\langle\cdots\blacksquare l :\ 20\rangle$

A detailed specification of mapping concrete EGG symbolic syntax with EGG abstract syntax is presented in the Tables 1, 2, 3, 4, 5.

The symbols in the context of **concrete** syntax (Tables 6, 7 and 8).

id – an identifier (unique in the scope of the whole EGG structure)

$$
\begin{aligned}
agg &= \blacktriangle\{g_1, g_2, \ldots, g_{gn}\}, \bullet\{v_1, v_2, \ldots, v_{vn}\}, \blacklozenge\{e_1, e_2, \ldots, e_{en}\}\\
dsc &= \cdots\blacksquare\{l_1, l_2, \ldots, l_{ln}\}\\
navi &\in \{\multimap, \twoheadleftarrow, \rightarrowtail, \twoheadleftarrow\!\!\!\shortmid\}\\
cnn &= \begin{array}{l} navi\blacktriangle g_1, navi\blacktriangle g_2, \ldots, navi\blacktriangle g_{gn},\\ navi\bullet v_1, navi\bullet v_2, \ldots, navi\bullet v_{vn},\\ navi\blacklozenge e_1, navi\blacklozenge e_2, \ldots, navi\blacklozenge e_{en}\end{array}
\end{aligned}
$$

3.2.2 Graphical EGG concrete syntax

The root Egg may be omitted in graphical concrete syntax.

4 Mappings between selected data metamodels and EGG

A proposal of a set of possible mapping schemata between some metamodels and the EGG metamodel is contained in this section. Different mappings may be also introduced but the one chosen reflects the characteristic features of each metamodels category well.

One shall notice, that the EGG is a purely extensional data structure. It means, that it does not cover any behavior – and thus gives freedom to define the structural constraints on data. The mapping that the authors have proposed implements gives a possible interpretation of the EGG concepts in terms of the specific data metamodels. The following metamodels are discussed in this section: RDF, XML, RDBM, UML and AOM.

Table 1 Concrete syntax of the $navigability$ category

Syntax	Comment
$\bullet\!\!-$	Undefined navigability. An example: $\underline{\blacklozenge\ e}\,\langle \bullet\!\!- \blacktriangle \{g_1, g_2\}\rangle$ denotes a $connection$ with the undefined $navigability$ of the e instance of the $Edge$ category with the references to the Egg category instances g_1 and g_2.
$\bullet\!\!\leftarrow$	Left-hand navigability. An example: $\underline{\blacklozenge\ e}\,\langle \bullet\!\!\leftarrow \blacktriangle \{g_1, g_2\}\rangle$ denotes a $connection$ with the left-hand $navigability$ of the e instance of the $Edge$ category with the references to the g_1 and the g_2 instances of the Egg category.
$\bullet\!\!\rightarrow$	Right-hand navigability. An example: $\underline{\blacklozenge\ e}\,\langle \bullet\!\!\rightarrow \blacktriangle \{g_1, g_2\}\rangle$ denotes a $connection$ with the right-hand $navigability$ of the e instance of the $Edge$ category with the references to the g_1 and the g_2 instances of the Egg category.
$\bullet\!\!\leftrightarrow$	Bi-directed navigability. An example: $\underline{\blacklozenge\ e}\,\langle \bullet\!\!\leftrightarrow \blacktriangle \{g_1, g_2\}\rangle$ denotes a $connection$ with the bi-directed $navigability$ of the e instance of the $Edge$ category with the references to the g_1 and the g_2 instances of the Egg category.

Table 2 Concrete syntax of the Egg category

Syntax	Comment
$\underline{\blacktriangle\ id}\left\langle \begin{matrix} \textbf{\textit{agg}}, \\ dscr \end{matrix} \right\rangle$	The agg relationship is the aggregate. $aggr$ – a list of the references to the instances of the Egg, $Vertex$ and $Edge$ categories. An example: $\underline{\blacktriangle\ g}\,\langle \blacktriangle \{g_1\}, \bullet \{v_1, v_2\}\rangle$ means that a g instance of the Egg category is composed of the references to the instance g_1 of the Egg category and of two references to the v_1 and v_2 instances of the $Vertex$ category.
$\underline{\blacktriangle\ id}\left\langle \begin{matrix} agg, \\ \textbf{\textit{dscr}} \end{matrix} \right\rangle$	The $dscr$ relationship is a link, that is the $labels$ put on the list are linked to the instance of the category Egg in such the way that they describe it. An example: $\underline{\blacktriangle\ g}\,\langle \bullet \{v_1, v_2\}, \cdots\!\blacksquare \{l_1, l_2\}\rangle$ means that a g instance of the Egg category is composed of two references to the v_1 and v_2 instances of the $Vertex$ category and is linked to two references to the l_1 and l_2 instances of the $Label$ category.

Table 3 Concrete syntax of the $Edge$ category

Syntax	Comment
$\underline{\blacklozenge\ id}\left\langle \begin{matrix} \textbf{\textit{cnn}}, \\ dscr \end{matrix} \right\rangle$	cnn relationship is a connection that is the instances put on the list are interconnected through the instance of the $Edge$ category. An example: $\underline{\blacklozenge\ e}\,\langle \bullet\!\!\leftarrow\!\blacktriangle g_1, \bullet\!\!\rightarrow\!\bullet \{v_1, v_2\}\rangle$ means that an e instance of the $Edge$ category interconnects the following instances via the references: the reference to the g_1 instance of the category Egg through the right-hand connection and the reference to the v_1 instance and to the v_2 instance both of the $Vertex$ category through the left-hand connection.
$\underline{\blacklozenge\ id}\left\langle \begin{matrix} cnn, \\ \textbf{\textit{dscr}} \end{matrix} \right\rangle$	$dscr$ relationship is a link, that is the labels put on the list are linked to the instance of the $Edge$ category in such the way that they describe it. An example: $\underline{\blacklozenge\ e}\,\langle \bullet\!\!\rightarrow\!\blacktriangle v_1, \cdots\!\blacksquare \{l_1, l_2\}\rangle$ means that an e instance of the $Edge$ category interconnects the following instances via the references: a reference to the v_1 instance of the $Vertex$ category through the right-hand connection and is described by the references to the l_1 and the l_2 instances of the $Label$ category.

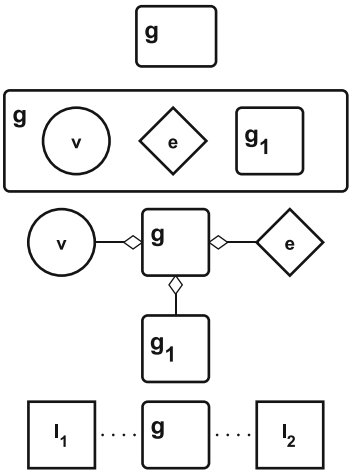
Table 4 Concrete syntax of the $Vertex$ category

Syntax	Comment
$\underline{\bullet\ id}\,\langle dscr\rangle$	$Vertex.descr$ relationship is a descripting link, that is the labels put on the list are related to an instance of the $Vertex$ category in such the way that they describe it. An example: $\underline{\bullet\ v}\,\langle \cdots\!\blacksquare \{l_1, l_2\}\rangle$ means that a v instance of the $Vertex$ category is described by the two references to the l_1 and l_2 instances of the $Label$ category.

Table 5 Concrete syntax of the $Label$ category

Syntax	Comment
$\underline{\blacksquare\ id}$	$Label$ does not have an internal structure and does not exist without a link to at least one instance of the Egg, $Vertex$ or $Edge$ categories. An example: $\underline{\bullet\ v}\,\langle \cdots\!\blacksquare \{l_1, l_2\}\rangle$ means that a v instance of the $Vertex$ category is described by two references to the l_1 and l_2 instances of the $Label$ category.

Table 6 Concrete syntaxes of the *Egg* category

Symbolic	Graphical
$\blacktriangle\, g\,\langle\,\rangle$	
$\blacktriangle\, g\,\langle \bullet v, \blacklozenge e, \blacktriangle g_1 \rangle$	
$\blacktriangle\, g\,\langle \cdots \blacksquare \{l_1, l_2\} \rangle$	

The RDF is a graph-based representation of the objects in the web. If we think of the XML we analyze the pure XML-based hierarchical structures of the documents. The RDBM covers the relational databasemodel. The UML is a modeling languages, which also facilitates creating a data representation on an object diagram containing *InstanceSpecification*s. The AOM is a novel data metamodel, both conceptual and physical, which has two distinguished parts: the extensional the intensional. Here, we focus on the extensional part only. Giving the semantic interpretation of metamodel constructs into the EGG categories makes it possible to designate the EGG configuration for a specific metamodel and further to compare the corresponding data structures. The process extensively uses the reification in terms of representing the higher level relationships if the specific data metamodel is solely data-oriented.

Table 7 Concrete syntaxes of the *Edge* category

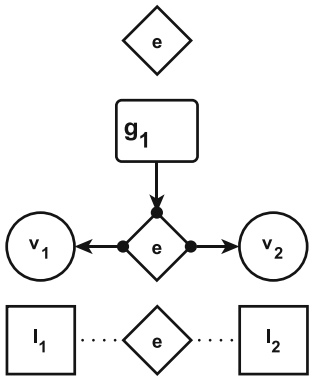
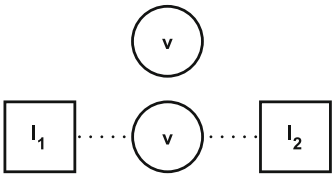
Symbolic	Graphical
$\blacklozenge\, e\,\langle\,\rangle$	
$\blacklozenge\, e \left\langle \begin{array}{l} \bullet\!\!\leftarrow\!\!\blacktriangle g_1, \\ \bullet\!\!\rightarrow\!\!\bullet \{v_1, v_2\} \end{array} \right\rangle$	
$\blacklozenge\, e\,\langle \cdots \blacksquare \{l_1, l_2\} \rangle$	

Table 8 Concrete syntaxes of the *Vertex* category

Symbolic	Graphical
$\blacktriangle\, v\,\langle \cdots \blacksquare \{l_1, l_2\} \rangle$	

Several models were created in the RDF, the XML, the RDBM, the UML and the AOM to present the mappings inference mechanism and the obtained mapping results. Next, the applied constructs were recognized in terms of the EGG. As the result of this approach it was possible to identify which metamodel categories are actually required by the models in terms of their extensional layer. Thus, the metamodel categories which represent instances (model elements) were mapped to the EGG metamodel categories.

The Tables 9, 10, 11, 12 and 13 of the succeeding subsections contain the results of mapping each model category to the EGG metamodel category. The first column of each table contains the categories of a particular metamodel while the second one – the categories of the EGG metamodel.

Mappings presented in this chapter show how the specific categories from a subsequent models can be perceived in terms of the EGG. The mappings proposed might extend the semantics of the mapping result. This means that the resulting EGG category instances might have broader meaning than they have had before mapping.

4.1 Resource description framework (RDF)

The RDF is a simple graph-based model proposed by the W3C, which is used to represent information in the web. The RDF graphs are the sets of the triples, each of which

Table 9 The RDF to the EGG metamodel categories mapping schema

RDF category	EGG category
RDF Graph	*Egg*
RDF Subject	*Vertex*
RDF Predicate	binary, directed *Edge*; `EdgeToElement` connection denotes the RDF Object; `ElementToEdge` connection denotes the RDF Subject
RDF Object	*Vertex*
IRI	*Label*
Literals	*Label*

Table 10 The XML to the EGG metamodel categories mapping schema

XML category	EGG category
node with children	parent is nesting *Egg*
node without children with identifier	*Vertex*
nodes' attributes	*Label*
XML IDREFS (referential attributes)	the attribute is mapped to the *Edge*, the subsequent references are mapped to the directed *Connection* with the `EdgeToElement Direction`; the parent-attribute relationship is mapped to the *Connection* with the `ElementToEdge Direction`.

consists of three elements, namely a subject, a predicate and an object. The RDF specification also covers three types of the RDF data that may occur in the triples. They consist of the IRI, the literal or the blank node. The IRIs are the identifiers which identify some resources. The literals are basic values, which are associated with a specific datatype. The blank nodes are nodes without a global identifier.

The mapping of the RDF to the EGG involves juxtaposition of their graphical structures, as shown in the Table 9.

4.2 Extensible markup language (XML)

The XML is a markup language focused on storing data in both a textual and a hierarchical forms [27]. These features make it intriguing for data modeling and for analysing it from the more general and novel EGG point of view. The XML categories are identified directly from the XML files (models) without any general definitions of the XML structure (e.g. XML Schema). This decision is motivated by the fact that the analysis in this paper refers to the extensional but not the intensional layer. The results of the identified mapping between the XML category instances and the EGG are presented in the Table 10.

It can be seen in the Table 10 that both *Egg* and *Edge* categories are applied with some limits.

Table 11 The RDBM to the EGG metamodel categories mapping schema

RDBM category	EGG category
Relation	*Egg*, set of all RDBM relations is mapped to one *Container*
Tuple	*Vertex*
Value	*Label*
Referential integrity constraint	*Vertices* connected by *Edges*

Table 12 The UML to the EGG metamodel categories mapping schema

UML category	EGG category
InstanceSpecification with classifier of type Class (object)	*Vertex*
Slot with ValueSpecification	*Label*
InstanceSpecification with classifier of type Association (link)	*Edge*
Slot with InstanceValue owned by link	*connection*

4.3 Relational database model (RDBM)

The RDBM constitutes a formal approach to represent the databases. The key notions in the RDBM are *relation* and *tuple*. Originally, it has been defined by Edgar Frank Codd [28] in terms of the set theory. The mapping between data contained in the relational databases (models) and the categories of the EGG category instances is contained in the Table 11.

Relational data are conceptually very simple. The extensional model of data consists of the *relations*, which aggregate the *tuples* (records). The tuples consist of the order *values*. Each tuple has a distinguished set of the values, which stand for the tuple's primary key. To show the relationships between data, *referential integrity constraints* are often used. Using the latter, one can constrain the domain of the value in such a way that it models a reference to the other value, most often fulfilling the role of the primary key in other tuple.

It is clearly seen in the Table 11 that the *Edge* category is applied in the RDBM with some limits.

Table 13 The AOM to the EGG metamodel categories mapping schema

AOM category	EGG category
Object	*Vertex*
Association Object	*Edge*
Role Object	*connection*
Value	*Label*
`Collection` (extensional aspect)	*Egg*, set of all collections are mapped to *Container*
`Association` (extensional aspect)	*Egg*, set of all collections are mapped to *Container*

4.4 Unified modeling language (UML)

The UML [6] is a modeling language, which has been standardized and managed by the OMG for many years. The language is intended for modeling both the structural and the behavioral aspects of the software-intensive systems. This language was taken into account because of its very rich semantic capacity when applied to data modeling. The extensional part, however, seems to be rather simple and straightforward in its inner construction.

The results of the comparative study focused on mapping the EGG category instances to the UML models are presented in the Table 12. The extensional part of the UML structural models is covered by the object diagram. The object diagrams are the instances of the class diagrams and are used to show the snapshot of a system at a specific point in time. The *InstanceSpecifications* are used to show the instantiated classifiers (i.a. classes and associations). These are covered by the objects and the links, respectively. The fact, that an object has some value for a specific attribute is depicted with the slots with *ValueSpecifications*.

It can be noticed in the Table 12 that the *Egg* metamodel category is the only one the UML categories cannot be mapped to. The mapping of the other categories is clear and all their features can be mapped directly.

4.5 Association-oriented metamodel (AOM)

The AOM is a data metamodel the characteristic features of it are: implementability, semantic unambiguity, high semantic capacity and high expressiveness. The fundamental notion in the AOM is `Association`, which is a first-class category [29] according to [30]. `Association` consist of `Roles`, which might be played by `Collections` or by another `Associations`. The AOM enables the very complex means for data abstraction in terms of the data and relationships polymorphism [26]. The AOM's extensional part must follow the structural constraints of the model strictly. `Collections`, `Associations` and `Roles` are instantiated as interlinked `Objects`. The identified mapping between the AOM category and the EGG categories can be found in the Table 13.

The Table 13 shows that the AOM metamodel contains good decomposition of the responsibilities of the metamodel categories in terms of the EGG concept. In the consequence the AOM categories can be easily mapped to all EGG categories. Thus the AOM categories map to all EGG categories in contrast to the UML.

5 Case studies

There are two case studies presented in this section both related to the usage of the EGG data structures and knowledge representation definitions. The first example concerns domain-specific data, namely a structure of an EGG, which represents a social network. The second example is dedicated to illustrating the EGG features like HYPER and ULTRA. Another purpose of the second example is presenting an option of nesting one EGGs in other EGGs. This option is achieved through defining some patterns for simplified structures of the knowledge representation patterns together with a domain-specific example making use of these patterns. Both case studies will be used for further considerations, also in terms of the possibility of their implementation in the selected data metamodels, taking into account the mapping schemes presented in Section 4.

It is clear from the case studies (the extensional level) presented in this section that the EGG concept was introduced to break some limits of the different metamodels already identified in Section 4 on the intensional level.

5.1 Social training group

Social network is a structure consisting of the social actors (e.g. individuals, organizations) and the binary connections between them, which represent their interactions. To improve the semantic capacity [31] of such the network, it might be beneficial to introduce other information patterns regarding the types of the relationships, e.g. the n-ary connections [32], the relationships as participants, grouping, etc. We have employed the EGG to model the structure of the social interactions within a training group.

The structure holds following semantics regarding the training social group: The domain `TrainingDomain` contains a training group `TrainingGroup` of its `members` who interact with each other at the martial arts training. These `members` are represented as *Vertices*. Assume that if specific people train with each other this fact will be represented as `trains with` edge, which connects subsequent vertices. Information about the people's age is associated with the vertices using the data blocks prefixed with `age:`. Some people might be the supervisors; they are introduced inside a shared nested *Egg* `Supervisors` (thus the supervisors are the members and also a part of the training group). Let the fact, that a specific supervisor set is responsible for a specific supervision be represented as the `supervision` edge, which connects this set to a specific `trains with` edge. Some people

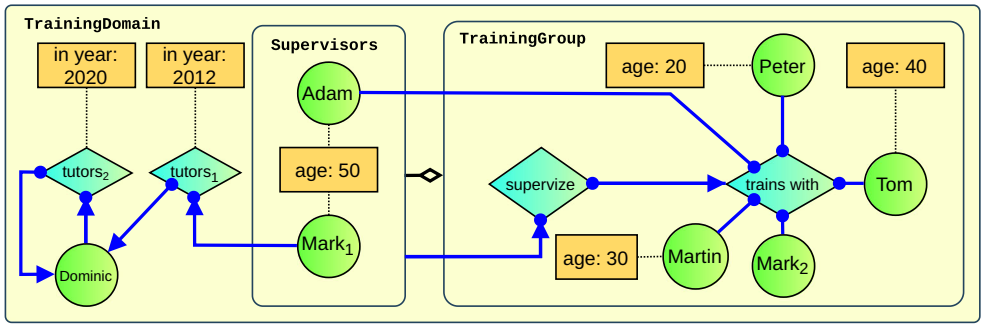


Fig. 3 An example of the EGG^{HUMS}_{FDN} model structure for the domain of a social training group

might be tutors and `tutor` other people (or even themselves). Let us represent this fact as a binary, directed edge, connection for the role of tutor shall be directed towards the edge and the opposite connection towards the node respectively. Moreover, the `tutor` edge should be attributed with data containing temporal information about this tutoring, prefixed with `year:`.

The EGG structure in the Fig. 3 has been created in order to express all possible extensions and generalizations which can be defined within the EGG structures. This means that these structures hold the properties of an EGG^{HUMS}_{FDN} and the proposed EGG is generalized by the following features: HYPER, ULTRA, MULTI, SHARED AGGREGATION and is extended by the following features: FIRST CLASS, DATA, and NAVIGABILITY. Detailed description of the above EGG features has been presented in the paper [5].

Below one can find a specification of the same EGG within a symbolic notation.

List of $Eggs$:

$$\begin{aligned}
&\blacktriangle\ \underline{TrainingDomain}\left\langle \begin{array}{l} \blacktriangle\{TrainingGroup, Supervisors\}, \\ \blacklozenge\{tutors_1, tutors_2\}, \\ \bullet Dominic \end{array} \right\rangle \\
&\blacktriangle\ \underline{TrainingGroup}\left\langle \begin{array}{l} \bullet\left\{ \begin{array}{l} Mark_1, Mark_2, Adam, \\ Martin, Tom, Peter \end{array} \right\}, \\ \blacklozenge\{supervise, trains\ with\} \end{array} \right\rangle \\
&\blacktriangle\ \underline{Supervisors}\left\langle \bullet\{Mark_1, Adam\} \right\rangle
\end{aligned} \tag{10}$$

List of $Vertices$:

$$\begin{aligned}
&\bullet\ \underline{Mark_1}\ \langle \cdots \blacksquare age: 50 \rangle \\
&\bullet\ \underline{Adam}\ \langle \cdots \blacksquare age: 50 \rangle \\
&\bullet\ \underline{Martin}\ \langle \cdots \blacksquare age: 30 \rangle \\
&\bullet\ \underline{Tom}\ \langle \cdots \blacksquare age: 40 \rangle \\
&\bullet\ \underline{Mark_2}\ \langle\ \rangle \\
&\bullet\ \underline{Peter}\ \langle \cdots \blacksquare age: 20 \rangle \\
&\bullet\ \underline{Dominic}\ \langle\ \rangle
\end{aligned} \tag{11}$$

List of $Edges$:

$$\begin{aligned}
&\blacklozenge\ \underline{supervise}\ \langle \multimap \blacktriangle Supervisors, \mapsto \blacklozenge trains\ with \rangle \\
&\blacklozenge\ \underline{tutors_1}\ \langle \bullet\!\!\multimap \bullet Mark_1, \mapsto \bullet Dominic, \cdots \blacksquare in\ year : 2012 \rangle \\
&\blacklozenge\ \underline{trains\ with}\ \langle \multimap \bullet\{Martin, Peter, Mark_2, Tom, Adam\} \rangle \\
&\blacklozenge\ \underline{tutors_2}\ \langle \bullet\!\!\multimap \bullet Dominic, \mapsto \bullet Dominic, \cdots \blacksquare in\ year : 2020 \rangle
\end{aligned} \tag{12}$$

5.2 Semantic network

One well-known group of methods for knowledge representation is semantic networks. Such the networks are built upon the graph-like structure by using vertices and edges in order to represent a declarative knowledge (i.a. definitional or assertional). The vertices in a semantic network may represent terms or concepts respectively. The edges are often directed, thus they might be indicted. Moreover, they can have various semantics and can represent different relationships, e.g. meronymy, hyponymy, hypernymy [33].

The classical models of semantic network consider the binary edges solely. However, there are models which are hypergraphic, due to n-ary hyperedges [19]. The applications of semantic networks comprise thesauri definition [34], knowledge visualization (e.g. financial knowledge [35]), natural language processing [36]. Semantic networks have been in the interest of the researchers in computer science and information systems since a dozen decades [21, 37]. There have been proposed many extensions, i.a. *partitioned semantic networks* [38] and *operator-operand networks* proposed by Krótkiewicz et al. [22]. Partitioned networks are objectifying the networks themselves in such a way that they could participate in edges.

In the Fig. 4 we have presented an exemplary semantic network defined within the EGG concepts. This network represents the definitional relationships between a few concepts in order to show the simplified relationships and the emotions in a family context. The following schematic constraints have been assumed:

- $Eggs$ representing networks and facts (represented as shared, partitioned subnetworks),

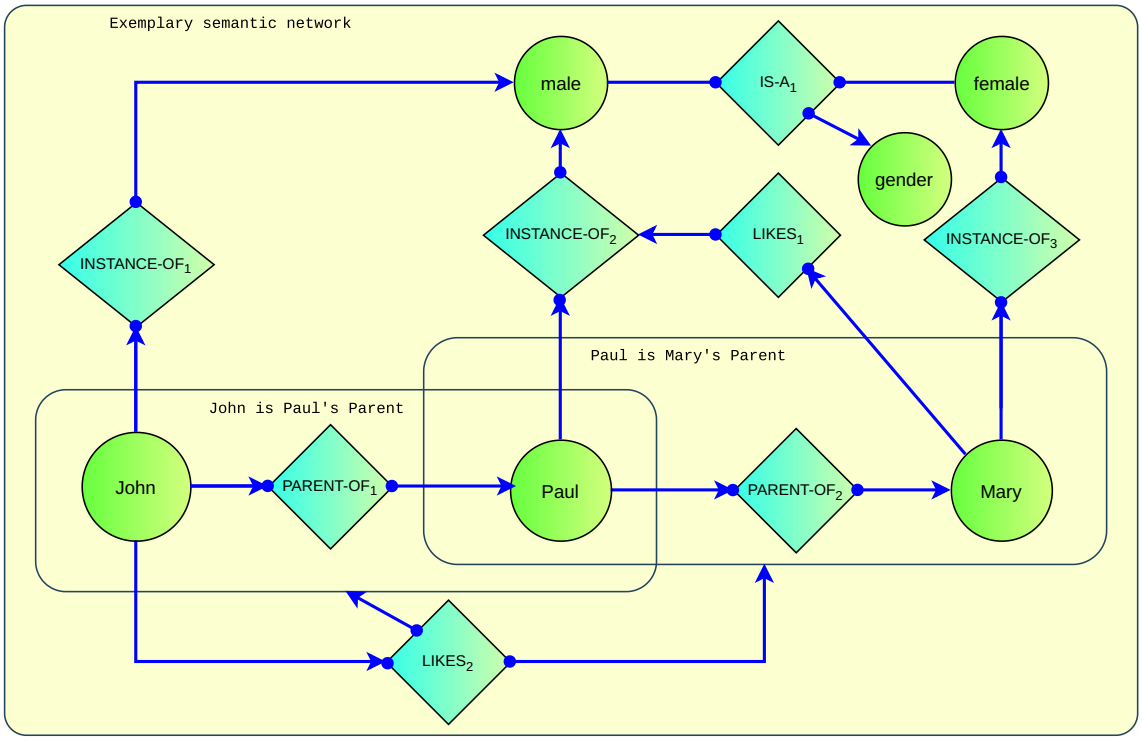


Fig. 4 An example of the EGG_{FN}^{HUS} model structure for an exemplary semantic network

- edges representing predicates, which are asserting linkages of elements in a network,
- vertices representing concepts which participate in a semantic networks,

The following predicate types have been distinguished:

- n-ary predicate $IS-A(subtype, supertype_1, supertype_2, \ldots, supertype_n)$; these predicates connect the specific elements and assert that the *subtype* element is less specific than the other elements, which are more specific within this relationship; the hyperedges are directed, the `ElementToEdge` direction denotes the *subtype*, the `EdgeToElement` direction denotes the *supertypes*;
- n-ary predicate $INSTANCE-OF(instance, type_1, type_2, \ldots, type_n)$; these predicates connect the elements being the instances of another elements being the types; the hyperedges are directed, the `ElementToEdge` direction denotes the *instance*, the `EdgeToElement` direction denotes the *types*;
- n-ary predicate $LIKES(subject, object_1, object_2, \ldots, object_n)$; representing the statement that an actor associated with the $subject$ element likes a series of other elements; it is required for this type of predicate to have directed connections, the `ElementToEdge` direction denotes the *subject*, the `EdgeToElement` direction denotes the *objects*;
- binary predicate $PARENT-OF\,(parent, child)$; representing the family bonds between two actors, the edge has to have directed connections, the `ElementToEdge` direction denotes the *parent*, the `EdgeToElement` direction denotes the *child*;

This semantic network encodes the following data on the information and the knowledge layers.

5.2.1 Defining family members

Specific human entities have been represented as vertices. In our semantic network example, we have distinguished three people called *John*, *Paul* and *Mary* respectively. This fact requires the following vertices to be asserted:

$$\underline{\bullet\ John}\,\langle\rangle \quad \underline{\bullet\ Paul}\,\langle\rangle \quad \underline{\bullet\ Mary}\,\langle\rangle \tag{13}$$

5.2.2 Defining gender

In the semantic network three vertices representing gender (two for concepts representing specific genders and one for the gender itself) have been distinguished:

$$\underline{\bullet\ male}\,\langle\rangle \quad \underline{\bullet\ female}\,\langle\rangle \quad \underline{\bullet\ gender}\,\langle\rangle \tag{14}$$

The fact that $\bullet male$ and $\bullet female$ are specific kinds of $\bullet gender$ is represented by the ternary edge with *IS-A* semantics:

$$\blacklozenge\ \underline{IS\text{-}A_1}\ \langle \bullet\!\!-\ \bullet male, \bullet\!\!-\ \bullet male, \bullet\!\!\rightarrow \bullet gender \rangle \tag{15}$$

5.2.3 Assigning gender (instantiation)

Representation of the fact, that a specific human entity has a gender is modeled as the *INSTANCE-OF* directed edge which connects specific vertices (representing human and gender):

$$\begin{array}{l} \blacklozenge\ \underline{INSTANCE\text{-}OF_1}\ \langle \bullet\!\!\leftarrow \bullet John, \bullet\!\!\rightarrow \bullet male \rangle \\ \blacklozenge\ \underline{INSTANCE\text{-}OF_2}\ \langle \bullet\!\!\leftarrow \bullet Paul, \bullet\!\!\rightarrow \bullet male \rangle \\ \blacklozenge\ \underline{INSTANCE\text{-}OF_3}\ \langle \bullet\!\!\leftarrow \bullet Mary, \bullet\!\!\rightarrow \bullet female \rangle \end{array} \tag{16}$$

5.2.4 Parent-child relationship

Analogously to the instantiation, one can distinguish the binary predicates representing the facts on the *parent-child* relationship:

$$\begin{array}{l} \blacklozenge\ \underline{PARENT\text{-}OF_1}\ \langle \bullet\!\!\leftarrow \bullet John, \bullet\!\!\rightarrow \bullet Paul \rangle \\ \blacklozenge\ \underline{PARENT\text{-}OF_2}\ \langle \bullet\!\!\leftarrow \bullet Paul, \bullet\!\!\rightarrow \bullet Mary \rangle \end{array} \tag{17}$$

5.2.5 Liking

Another n-ary predicate has been used to model liking. In order to represent a fact that *John likes that he is parent of Paul and that Paul is parent of Mary* one need to construct a connection to the Egg, which represents objectified facts, as shown below:

$$\blacklozenge\ \underline{LIKES_2} \left\langle \begin{array}{l} \bullet\!\!\leftarrow \bullet John, \\ \bullet\!\!\rightarrow \blacktriangle \text{John is Paul's Parent}, \\ \bullet\!\!\rightarrow \blacktriangle \text{Paul is Mary's Parent} \end{array} \right\rangle \tag{18}$$

In the case when objectification is not needed, one could refer to the edge itself, as the target of the connection. The structure below represents the fact that *Mary likes Paul being male*.

$$\blacklozenge\ \underline{LIKES_1} \left\langle \begin{array}{l} \bullet\!\!\leftarrow \bullet Mary, \\ \bullet\!\!\rightarrow \blacklozenge\ INSTANCE\text{-}OF_2 \end{array} \right\rangle \tag{19}$$

6 Data complexity of metamodels

The cohesion of the data modeling approaches presented in Section 4 to the data modeling based on the EGG structures is analysed. Due to the fact that that the EGG structures are graph-like at the same time being more general and more extensive than the remaining ones the methodology applied in this section is as follows.

Two case studies are examined: *social training group* (presented in Fig. 3) and *semantic network example* (depicted in Fig. 4). The first one stands for a small and comprehensive EGG which has been modeled in such a way that it forms a fully-featured EGG, i.e. covers all extensions and generalizations. The second one strives with solving the knowledge representation problem in a schemaless structure with abundance of the semantic predicates. The examination covers translation of these structures into the EGG structures which are consistent with the subsequent data metamodels.

The EGG structures have been elaborated in relation to the mappings described in Section 4. These EGG structures hold EGG-specific semantics presented in Section 3, which results from the approach. Another part of semantics has metamodel-specific form, which uses only EGG expressions having reasonable meaning in a particular data metamodel. For example, in the case of the data metamodels with the reference-based relationships, the edges might be binary and navigated in a strictly defined manner.

6.1 Resource description framework (RDF)

The Figs. 5 and 6 show the EGG representations containing the proper case study models expressed in the RDF.

The representations of the RDF expressions are simple because they are based on the triples of the *Subject-Object-Predicate* (SOP). Therefore, all relationships in the original EGG structures of a different arity had to be mapped to the binary edges. Edges of a different nature were reified to nodes. Moreover, the *Subject-Predicate-Object* (SPO) structures enforce a navigability, so a navigability was added in the case the model contained anon directed connection. Thus the RDF in the mapping to EGG has a NAVI property, which is downright forced. The RDF data also has the DATA property - when an element has a label, this is reflected by representing the node as a literal or by attaching an IRI to it. Additionally, it should be noted that it is possible to map the MULTI property, but its occurrence in the *social training groups* model was not mapped due to the reification of the $\blacklozenge tutors_1$ edge that represented such the relationship. The properties FIRST CLASS, SHARED AGGREGATION, HYPER, or ULTRA cannot be represented in the RDF.

6.2 Extensible markup language (XML)

The Figs. 7 and 8 show the EGG representations containing the respective case study models expressed in the XML.

The mapping of the structures *social training group* and *semantic network example* in the XML was done in a way that reflects the specificity of the tree representation of the

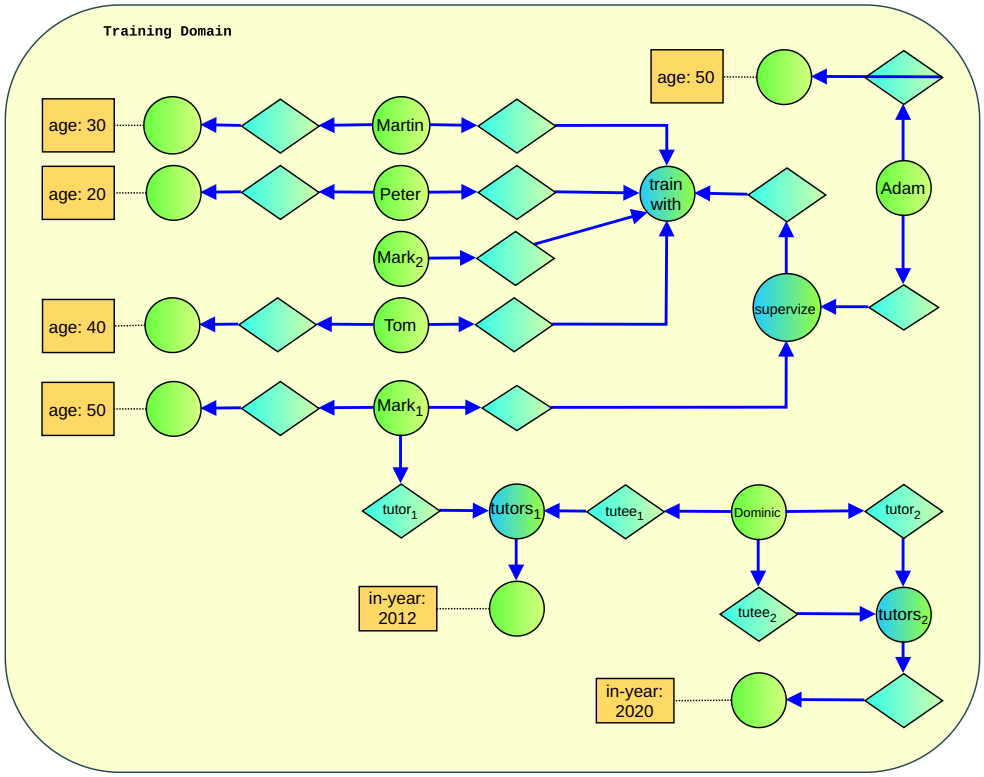


Fig. 5 The EGG structure resulting from the mapping of the *social training group* model to the RDF

documents. The entire document is represented by the EGG, which lists the *Vertices* with *Labels* (representing the attributes). Vertices from the original structure have been represented by *Egg*, if they have the substructure in terms of the tree-based representation. Then, *Edges* implementing binary, unidirectional relationships, characteristic of the tree-based structure of XML have been reified and their semantics has been transferred to the grouping aspect of *Egg*. Vertices forming leaves in this aspect have been mapped directly to vertices. The remaining *Edges* were represented by the reference mechanism defined in the XML schemata. The directionality has been used to represent which attribute refers to the other.

The XML allows for the representation of the HYPER generalization thanks to the *XMLREFS* capability, allowing to refer to the multiple elements within one relationship representation. It is also possible to make the MULTI generalization, as there is no limitation to referring to a given element multiple times within a single relationship representation. In terms of the extensions, all available EGG extensions are realizable with the XML. Due to the fact that some XML nodes are represented by *Egg*, this representation enables the property of FIRST CLASS, i.e. it is possible to refer to it as a reference. Attributes give the opportunity to map *Label*. The NAVI property is strictly defined regarding the direction of individual reference

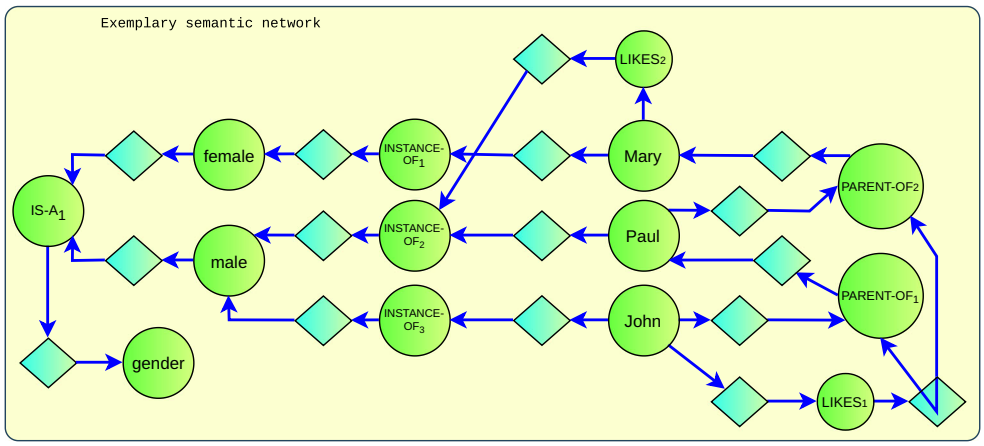


Fig. 6 The EGG structure resulting from the mapping of the *semantic network example* model to the RDF

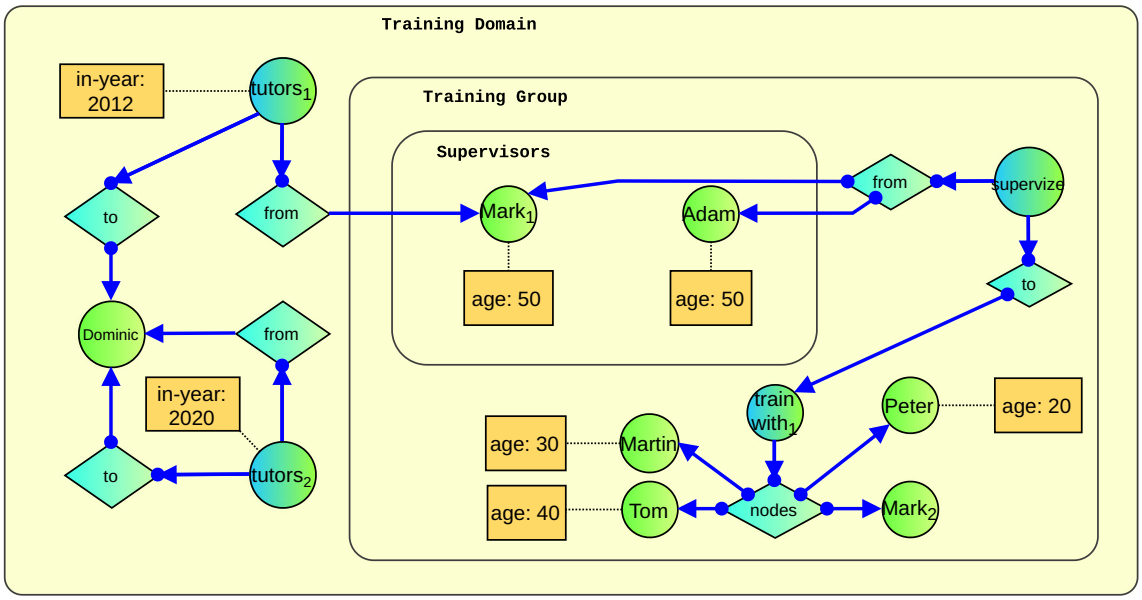


Fig. 7 The EGG structure resulting from the mapping of the *social training group* model to the XML

elements – but it is not possible to model navigable references in a bidirectional manner.

6.3 Relational database model (RDBM)

The diagrams on the Figs. 9 and 10 show the EGG representations containing the case study models expressed in terms of the *Relational Database* (RDB).

The mappings of the EGG structures into the EGG which have semantics and constraints of the RDBM have been based on the representation of both $Vertices$ and $Edges$. $Labels$ are represented as the attributes of these tuples. In order to keep a strong structure and a common set of the attributes for each tuple , data blocks containing $NULL$ values have been generated. Aggregating tuples into the relationships also implies the need to group them. It was mapped into additional $Eggs$ instances, which group structurally the homogeneous elements.

The RDBM allows the representation of the relatively simple EGG structures. The relationships realized by the referential integrity constraints are binary directed, and this direction is always from the referring element to the referred element. This implementation makes the $Edges$ in the RDBM analogous to the RDF. Therefore, the RDBM does not support the HYPER property and has a NAVIGABILITY property with the strongly defined rules. Since the relationships are not the first-class categories and therefore they cannot participate in other relationships, the ULTRA property is not supported. Attributionality of tuples provides the DATA property. The relation category instances

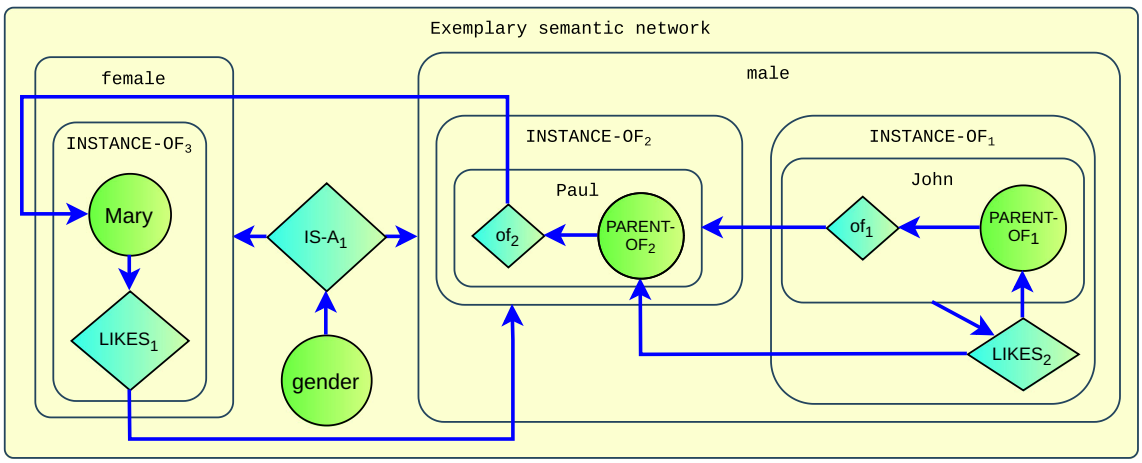


Fig. 8 The EGG structure resulting from the mapping of the *semantic network example* model to the XML

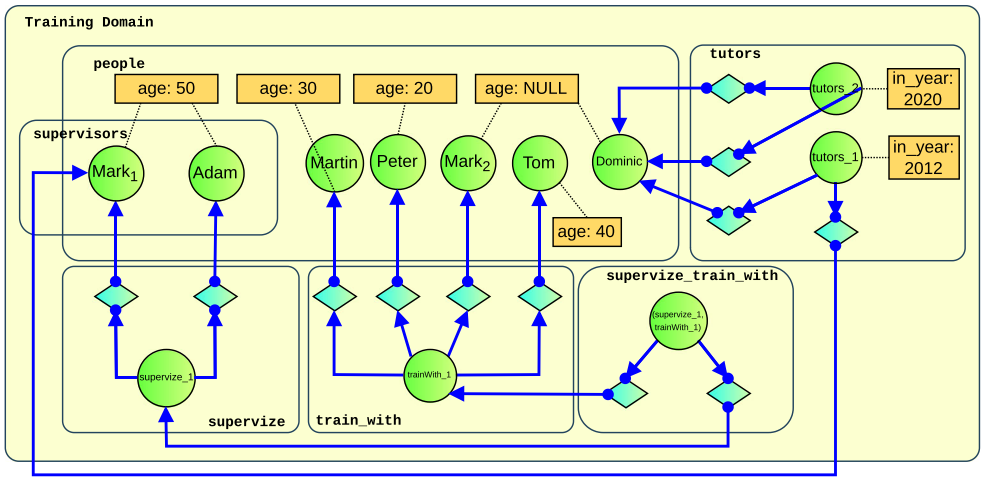


Fig. 9 The EGG structure resulting from the mapping of the *social training group* model to the RDBM

mapped to *Egg* are used to show semantics of grouping, so SHARED AGGREGATION and FIRST CLASS do not exist in the RDBM.

6.4 Unified modeling language (UML)

The diagrams on Figs. 11 and 12 show EGG representations containing the case study models expressed in the UML.

Implementation of structure mappings in the UML was based on an object diagram, within which a number of InstanceSpecification was considered compliant to the model based on a class diagram, classifying nodes as classes and edges as associations. It was assumed that the nodes with the EGG representing people will be matched by the instances of the class with the attribute representing data. The fact of distinguishing a subset of nodes performing the subsumption was taken into account as a subclass. The edges described with data were implemented as the specifications of an association class instance.

The internal structure of the UML instance is quite complex. UML satisfies the DATA extension due to the possibility of specifying for the specific slot in InstanceSpecification. Due to the n-ary links, the instance models are able to have HYPER and MULTI properties. The model specification also does not describe the limitations due to the classification of the instance specifications in terms of combining them – therefore the models have the ULTRA property. It should be noted, however, that class diagrams that classify UML instance models in the form of object diagrams do not in any way support the ultragraph relationships. Due to the lack of a direct equivalent of the EGG, the properties directly related to it, i.e. FIRST CLASS and SHARED AGGREGATION, are not supported.

6.5 Association-oriented Metamodel (AOM)

The diagrams on Figs. 13 and 14 show the EGG representations containing the case study models expressed in the AOM.

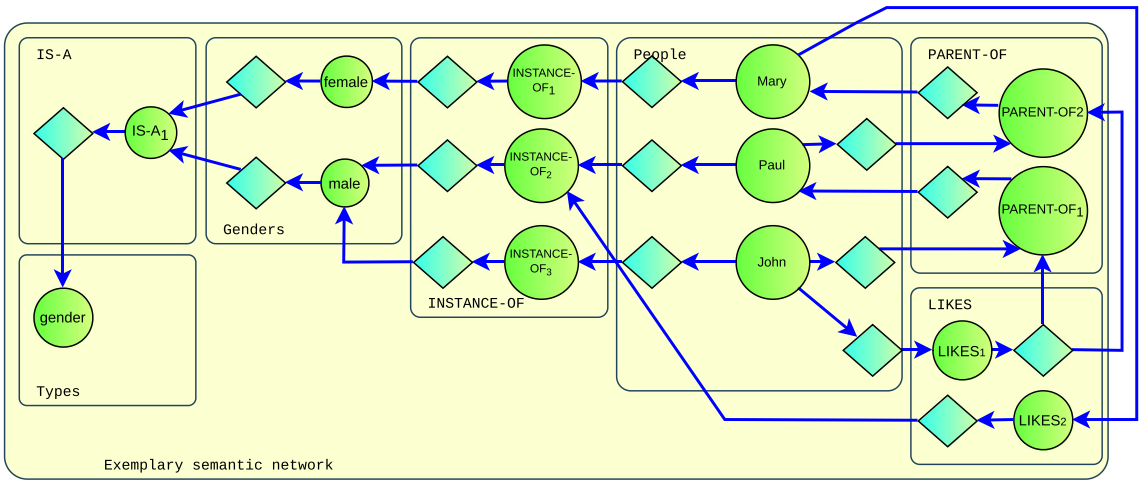


Fig. 10 The EGG structure resulting from the mapping of the *semantic network example* model to the RDBM

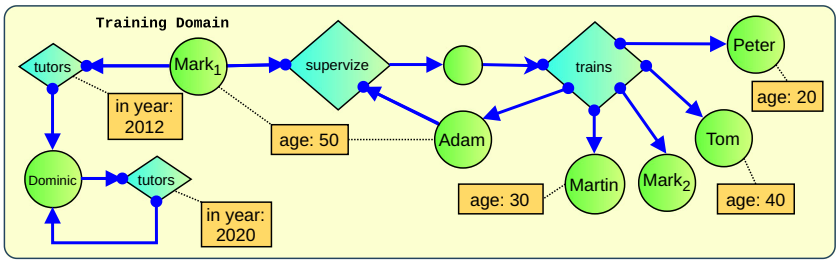


Fig. 11 The EGG structure resulting from the mapping of the *social training group* model to the UML

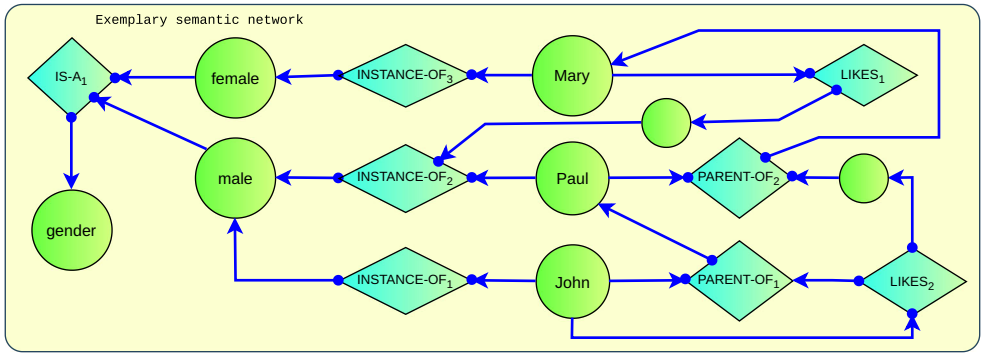


Fig. 12 The EGG structure resulting from the mapping of *semantic network example* model to the UML

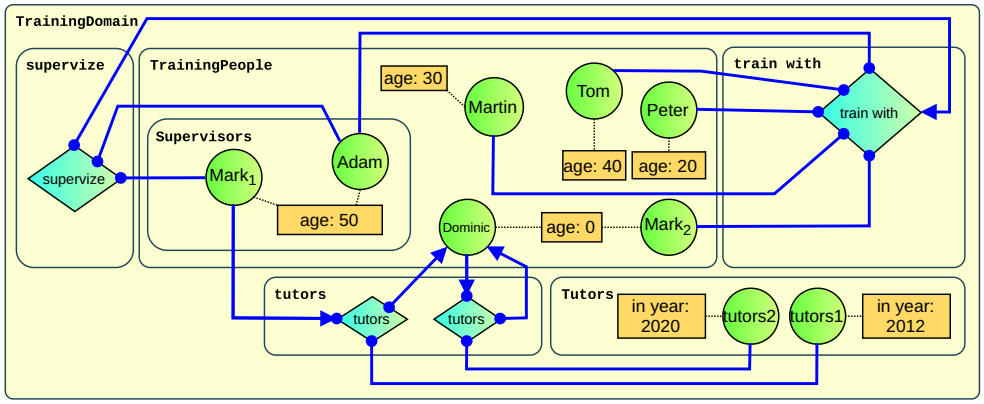


Fig. 13 The EGG structure resulting from the mapping of the *social training group* model to the AOM

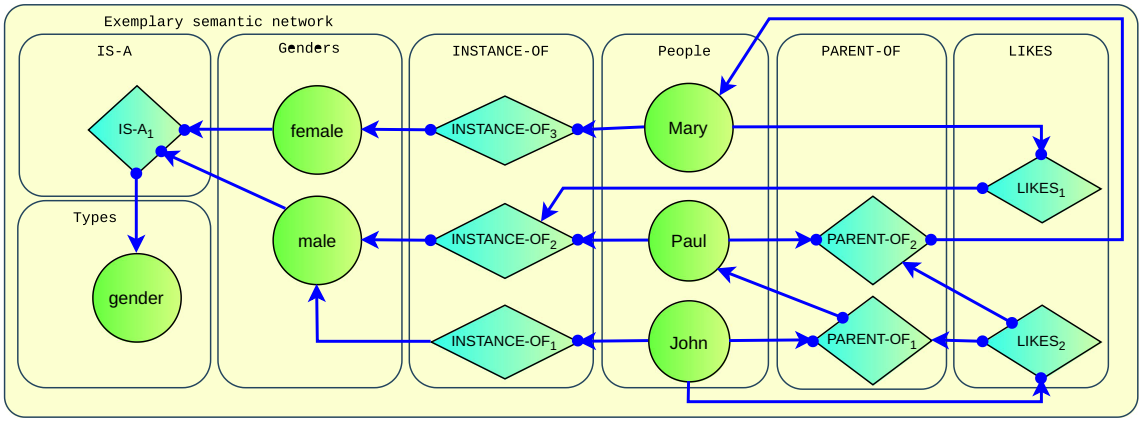


Fig. 14 The EGG structure resulting from the mapping of the *semantic network example* model to the AOM

Listing 1 An XML implementation of the *Social training group*

```
<?xml version="1.0"?>
<trainingDomain>
  <person id="Dominic"/>
  <trainingGroup>
    <supervisors>
      <person id="Mark_1" age="50"/>
      <person id="Adam" age="50"/>
    </supervisors>
    <person id="Marcin" age="30"/>
    <person id="Tom" age="40"/>
    <person id="Peter" age="20"/>
    <person id="Mark_2"/>
    <trainWith id="trainWith_1" nodes="Marcin Tom Peter Mark_2 Tom"/>
    <supervize from="Adam Mark_1" to="trainWith_1"/>
  </trainingGroup>
  <tutors id="tutors_1" from="Mark_1" to="Dominic" in-year="2012"/>
  <tutors id="tutors_2" from="Dominic" to="Dominic" in-year="2020"/>
</trainingDomain>
```

The extensional part of an Association-Oriented Metamodel is a direct consequence of the inherent intensional part of the metamodel, expressing a structure based on the collections and the associations. This part of the metamodel operates on the categories such as `Object`, `Role Object`, and `Association Object`. The nodes represent `Objects`, while the edges represent `Association Objects`. `Role objects`, on the other hand, are mapped as *connections*. The fact that `Objects` have an internal structure, which is represented in the form of values is mapped through labels. Edges requiring data annotation were mapped as instances of the BACT [39] association-oriented structure pattern. Additionally, it should be noted that the Association-Oriented intensional categories such as `Collection` and `Association` have an extensional aspect, i.e. they group the individual object instances, therefore they were also mapped in the resulting *Egg* structure as the grouping *Egg*.

The extensional models of the Association-Oriented metamodel allow all generalizations and *EGG* extensions to be expressed except FIRST CLASS and SHARED AGGREGATION. They assume, in turn, the participation of *Egg* representation in the relationships and in the possibility of sharing the grouped category instances. Such properties of the extensional structures are not directly expressible in the Association-Oriented metamodel.

6.6 Possible implementations

The models and the considerations presented above apply to the case studies expressed in terms of the *EGG*. These structures map to the target metamodels that reflect the data capabilities that are consistent with the selected data metamodels. In order to talk about actual data structures, these *EGG* structures should be implemented in the form of the expressions in the specific metamodels. Below the implementations of the *social training group* (Listing 1) and the *example of semantic network* (Fig. 15) models for XML and the AOM are presented.

The Listing 1 shows one possible implementation of the document that encodes the EGG structure expressed in the Fig. 7. The Figs. 15 and 16 show a possible implementation of the semantic network model in the AOM. The intensional part concerns the data structure, which allows for the

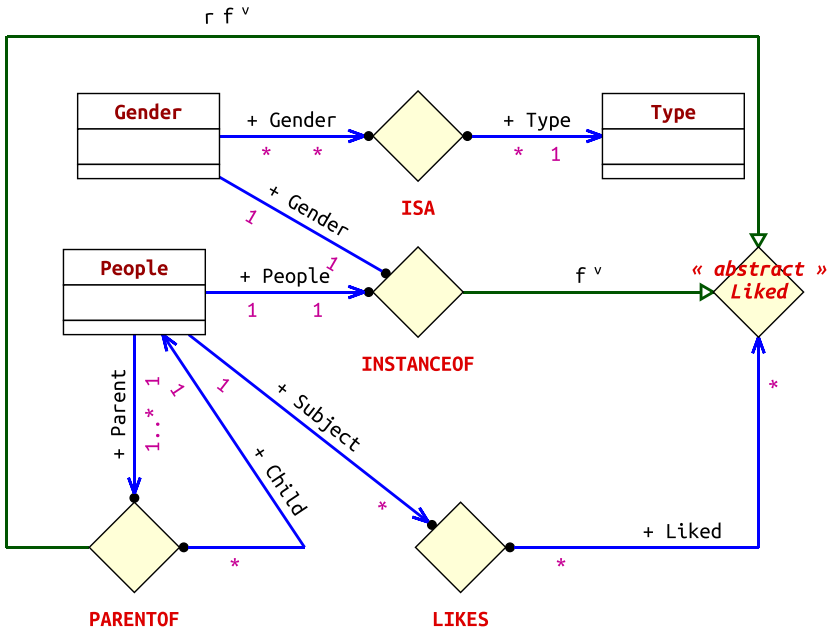


Fig. 15 An Association-Oriented implementation of the *Semantic network example* – the intensional part

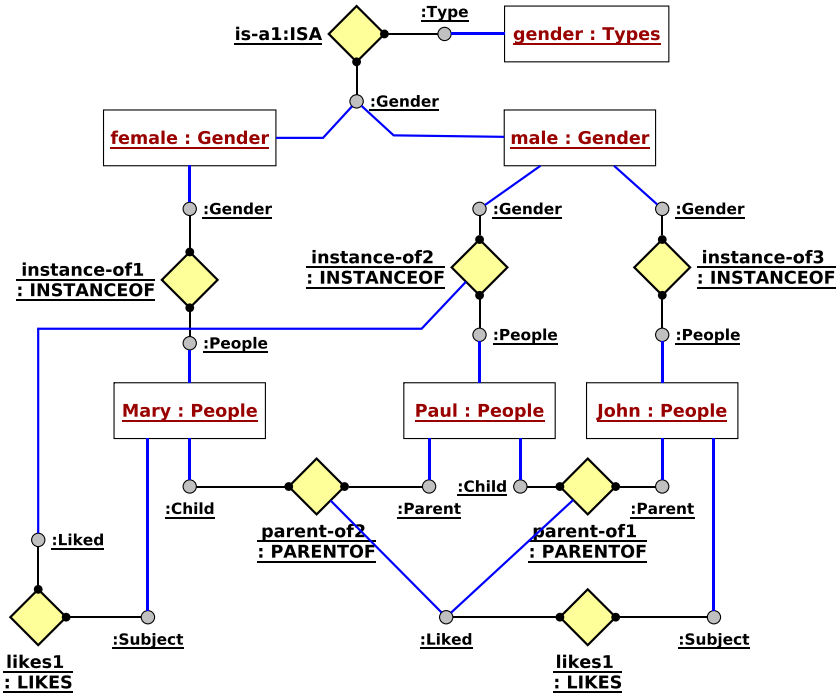


Fig. 16 An Association-Oriented implementation of the *Semantic network example* – the extensional part

expression of data compatible with it. The extensional model contains the direct translation of the EGG structure expressed in the Fig. 14.

6.7 Summary

The EGG features in terms of the analysed data metamodels are summarized in the Table 14. Presence of a specific feature in the extensional layer of a metamodel is marked by the black circle, while its absence - by the dash. The existence of a specific EGG feature in a metamodel but with some syntactical constraints thus enforcing some limitations is denoted by the white circle. The EGG can be used as a reference data metamodel when evaluation of other metamodel expressiveness and semantic capacity [39] is considered. As the result of such the analysis a right metamodel for a specific modeling issue may be selected better. This approach is appropriate if the EGG structure representation complexity of the domain problem is considered.

7 Evaluation of research results

The evaluation of the research results should be related to the measure notion. There are several approaches to defining a measure, which can be applied for the EGG concept:

- a syntax-related
  - a one based on the distances between models created in the different metamodels in relation to the model in the EGG,
  - a one based on the distances between these parts of the metamodels, which were used in a particular model related to the EGG metamodel,

Table 14 The EGG features configuration

EGG features		RDF	XML	RDBM	UML	AOM
Generalizations	HYPER	–	•	–	•	•
	ULTRA	–	–	–	•	•
	MULTI	•	•	•	–	•
	SHARED AGG.	–	–	$\circ^{a}$	–	–
	FIRST CLASS	–	•	–	–	–
Extensions	DATA	•	•	•	•	•
	NAVIGABILITY	$\circ^{b}$	$\circ^{b}$	$\circ^{b}$	$\circ^{c}$	$\circ^{d}$
EGG configuration symbol		EGG^{M}_{DN}	EGG^{HM}_{FDN}	EGG^{MS}_{DN}	EGG^{HU}_{DN}	EGG^{HUM}_{DN}

^{a} – it is only possible formally, ^{b} – without Bidirectional, ^{c} – without `EdgeToElement`, ^{d} – `ElementToEdge`

  - a one based on the distances between the complete metamodels related to the EGG metamodel,
- a semantics-related.

There are also several measures defined for graphs, which are known from minimal-distance structural pattern recognition methods. There are also measures from the *edit distance* group, which are based on expressing the distance in terms of the total minimal cost of the addition or the removal of the graph nodes. The mentioned measures are interrelated.

The semantics-related approach only is taken into account in the presented paper due to the high conceptual and computational complexity of the syntax-related approaches, which are planned to be explored in the succeeding publications.

In order to evaluate the research results obtained we have proposed the measures to compute semantic preservation of the original EGG in the resulting EGG structures. The methodology of the evaluation is as follows. We have examined each EGG category instance of the resulting EGG structures in all data metamodels under consideration. Each category instance has been thoroughly evaluated and classified into one of the three groups:

- *Translated* (T) – the category instances that are in the resulting structure and are direct mapping of the category from the original EGG,
- *Altered* (A) – the category instances that are in the resulting structure however their function has been altered during mapping, i.e. via reification,
- *Forced* (F) – the category instances that are in the resulting data metamodel due to its nature.

Moreover, there is the 4$^{\text{th}}$ category *Lost* (L) which categorizes the category pseudo-instances that are absent in the resulting EGG structure but were present in the original one.

Taking into consideration this grouping, the following metrics have been proposed.

Regarding the Egg category:

$$Eg_T = \frac{\text{number of } Egg \text{ instances classified as } T}{\text{total number of } Egg \text{ instances and pseudo-instances}} \tag{20}$$

$$Eg_A = \frac{\text{number of } Egg \text{ instances classified as } A}{\text{total number of } Egg \text{ instances and pseudo-instances}} \tag{21}$$

$$Eg_F = \frac{\text{number of } Egg \text{ instances classified as } F}{\text{total number of } Egg \text{ instances and pseudo-instances}} \tag{22}$$

$$Eg_L = \frac{\text{number of } Egg \text{ instances classified as } L}{\text{total number of } Egg \text{ instances and pseudo-instances}} \tag{23}$$

Regarding the $Vertex$ category:

$$V_T = \frac{\text{number of } Vertex \text{ instances classified as } T}{\text{total number of } Vertex \text{ instances and pseudo-instances}} \tag{24}$$

$$V_A = \frac{\text{number of } Vertex \text{ instances classified as } A}{\text{total number of } Vertex \text{ instances and pseudo-instances}} \tag{25}$$

$$V_F = \frac{\text{number of } Vertex \text{ instances classified as } F}{\text{total number of } Vertex \text{ instances and pseudo-instances}} \tag{26}$$

$$V_L = \frac{\text{number of } Vertex \text{ instances classified as } L}{\text{total number of } Vertex \text{ instances and pseudo-instances}} \tag{27}$$

Regarding the $Edge$ category:

$$E_T = \frac{\text{number of } Edge \text{ instances classified as } T}{\text{total number of } Edge \text{ instances and pseudo-instances}} \tag{28}$$

$$E_A = \frac{\text{number of } Edge \text{ instances classified as } A}{\text{total number of } Edge \text{ instances and pseudo-instances}} \tag{29}$$

$$E_F = \frac{\text{number of } Edge \text{ instances classified as } F}{\text{total number of } Edge \text{ instances and pseudo-instances}} \tag{30}$$

$$E_L = \frac{\text{number of } Edge \text{ instances classified as } L}{\text{total number of } Edge \text{ instances and pseudo-instances}} \tag{31}$$

Regarding the *Label* category:

$$L_T = \frac{\text{number of } Label \text{ instances classified as } T}{\text{total number of } Label \text{ instances and pseudo-instances}} \tag{32}$$

$$L_A = \frac{\text{number of } Label \text{ instances classified as } A}{\text{total number of } Label \text{ instances and pseudo-instances}} \quad (33)$$

$$L_F = \frac{\text{number of } Label \text{ instances classified as } F}{\text{total number of } Label \text{ instances and pseudo-instances}} \quad (34)$$

$$L_L = \frac{\text{number of } Label \text{ instances classified as } L}{\text{total number of } Label \text{ instances and pseudo-instances}} \quad (35)$$

All above metrics are normalized in the range $\langle 0, 1 \rangle$. The metrics have been applied to the *Social training group* case study. The results obtained have been shown in the Table 15.

The results show that in this case study the XML metamodel has been the most expressive in terms of Egg mapping, because has the highest value for Eg_T. This means that this metamodel served as the best one for realizing the nested structures. The direct application of node nesting and mapping the parent vertices to the Egg helped to achieve the high level of this value. The highest values of Eg_F are obtained by the RDB and the AOM. This stems from the fact, that those metamodels strictly follow the data schemata in order to implement the structural databases, thus require manifestations of the additional Egg instances. V_T is the highest for the XML and the UML, which shows strict correspondence of EGG $Vertex$ category instances to its mappings. Metamodels such as the RDF and the AOM need additional vertices, so they have non-zero V_F values. This happens in RDF due to the triple nature, which requires that describing with $Label$ creates artificial $Vertex$. For the AOM this is the result of the additional collection creations, because the associations cannot be data-labeled. The highest metrics for the translated edges E_T was obtained by the AOM, due to the high semantic capacity of the associations and the roles. The lowest is for the RDF and the RDB, since they do not convey a lot of information in edge representations. For the RDF, most of semantics is determined by the RDFS which is located on the intensional level and thus is out of the scope of this paper. The RDB does not have the direct relationships thus they have been mapped to the referential integrity constraints (the foreign key constraints). Regarding the $Label$ metrics, almost all data metamodels obtained the maximum value. The metamodels with value 0.9 force the additional labels, since strict structuring requires them to introduce neutral *NULL*-like values.

Table 15 The results of the *Social training group* EGG case study evaluation

	Metrics	RDF	XML	RDBM	UML	AOM
Egg	Eg_T	0.3	1.0	0.3	0.3	0.3
	Eg_A	0.0	0.0	0.0	0.0	0.0
	Eg_F	0.0	0.0	0.7	0.0	0.7
	Eg_L	0.7	0.0	0.0	0.7	0.0
Vertex	V_T	0.4	1.0	0.6	1.0	0.6
	V_A	0.2	0.0	0.4	0.0	0.0
	V_F	0.4	0.0	0.0	0.0	0.4
	V_L	0.0	0.0	0.0	0.0	0.0
Edge	E_T	0.0	0.1	0.0	0.8	1.0
	E_A	0.2	0.0	0.0	0.3	0.0
	E_F	0.8	0.9	1.0	0.0	0.0
	E_L	0.0	0.0	0.0	0.0	0.0
Label	L_T	1.0	1.0	0.9	1.0	0.9
	L_A	0.0	0.0	0.0	0.0	0.0
	L_F	0.0	0.0	0.1	0.0	0.1
	L_L	0.0	0.0	0.0	0.0	0.0

Table 16 The aggregated results of the *Social training group* EGG case study evaluation

	Eg	*V*	*E*	*L*	*Agg*
RDF	0.33	0.39	0.00	1.00	0.49
XML	1.00	1.00	0.14	1.00	0.83
RDBM	0.29	0.58	0.00	0.88	0.51
UML	0.33	1.00	0.75	1.00	0.85
AOM	0.29	0.58	1.00	0.88	0.71

We have aggregated the obtained results using the following aggregation function:

$$Agg = \frac{\sum\limits_{\mu\in\{Eg,V,E,L\}} w_\mu \cdot \frac{\mu_T}{\mu_T+\mu_A+\mu_F+\mu_L}}{\sum\limits_{\mu\in\{Eg,V,E,L\}} w_\mu} \tag{36}$$

where w denotes the weight. The weight vector contains the number of the specific category instances in the original Egg. The aggregation results along with the partial aggregations have been presented in the Table 16. Results show that the UML and the XML realize the original Egg in the most data semantics preserving way. The RDB and the RDF are the opposite.

8 Conclusions

The EGG was defined to create the data models and due to its generality it forms a concept, which may be used for a comparative studies of the features of the other data modeling oriented metamodels. As the result, the EGG constitutes a reference metamodel and the modeling language when the data modeling problems are investigated. Its referential nature was demonstrated in Section 4 for mapping between the EGG and selected metamodels as well as in Section 6 for identifying the EGG features availability in the selected metamodels. The set of the metamodels taken into account can be extended to the other existing metamodels. Alternatively, the proposed mechanism can be applied for identifying the features of the newly discovered metamodels in future.

The EGG structures may also be used in various applications in computer science and machine learning tasks e.g. for computational problems which rely on network-like structures (Bayesian networks [40] or neural networks [41]) and for complex methods for graph signal processing [42].

The paper introduces a novel concept of measuring the expressiveness of metamodels in comparison to the EGG metamodel expressiveness. The measure is semantics-related one, so it compares the metamodels expressiveness via comparison of particular models created in the original model with the model transformed to EGG abstract syntax. This measure may help decide which metamodel is sufficient for modeling a particular reality. Other measures, the syntax-related ones may be also applied for leading comparative studies, what is planned to show in the succeeding publications.

In order to achieve the challenge of measuring the data metamodels expressiveness complete EGG abstract syntax was introduced in the paper. It was done through its implementations in the AOM and the UML modeling languages. Then two concrete syntaxes were defined and EGG categories semantics was specified. All these aspects together form the strong and the well defined mechanisms for working with the EGG. They make it possible to apply the EGG concept to very wide application domains. Its applicability is shown on the example of two case studies. One case study is an extension of the case study shown in the previous publication. The second one is dedicated to knowledge modeling, so it shows that the EGG is applicable for AI domain as well.

Abbreviations AFN, *Associaton-Oriented Formal Notation.*; AI, *Artificial Intelligence.*; AOM, *Associaton-Oriented Metamodel.*; BACT, *Bicompositive Tandem Association-Collection.*; DSML, *Domain-Specific Modeling Languages.*; EGG, *Extended Generalized Graph.*; GPML, *General Purpose Modeling Languages.*; IRI, *Internationalized Resource Identifier.*; MDA, *Model Driven Architecture®.*; MOF, *Meta-Object Facility™*; OCL, *Object Constraint Language.*; OMG, *Object Management Group™*; RDB, *Relational Database.*; RDBM, *Relational Database Model.*; RDF, *Resource Description Framework.*; RDFS, *RDF Schema.*; SOP, *Subject-Object-Predicate.*; SPO, *Subject-Predicate-Object.*; UML, *Unified Modeling Language™.*; W3C, *World Wide Web Consortium.*; XML, *Extensible Markup Language*.

References

1. Li X, Lyu M, Wang Z, Chen C-H, Zheng P (2021) Exploiting knowledge graphs in industrial products and services: a survey of key aspects, challenges, and future perspectives. Comput Ind 129:103449
2. Fischer MT, Frings A, Keim DA, Seebacher D (2021) Towards a survey on static and dynamic hypergraph visualizations. In: 2021 IEEE visualization conference (VIS). IEEE

3. Lapshin VS, Rogozov YI, Kucherov SA (2021) Method for building an information model specification based on a sensemaking approach to user involvement in the development process. J King Saud Univ - Comput Inf Sci
4. Jodłowiec M, Krótkiewicz M, Zabawa P (2021) The extended graph generalization as a representation of the metamodels' extensional layer. In: Fujita H, Selamat A, Lin JC-W, Ali M (eds) Advances and trends in artificial intelligence. Artificial intelligence practices. Springer, pp 369–382
5. Jodłowiec M, Krótkiewicz M, Zabawa P (2020) Fundamentals of generalized and extended graph-based structural modeling. In: Nguyen NT, Hoang BH, Huynh CP, Hwang D, Trawinski B, Vossen G (eds) Computational collective intelligence - 12th international conference, ICCCI 2020, Da Nang, Vietnam, 30 Nov - 3 Dec 2020, proceedings. Lecture notes in computer science. Springer, vol 12496, pp 27–41
6. Cook S, Bock C, Rivett P, Rutt T, Seidewitz E, Selic B, Tolbert D (2017) Unified modeling language (UML) version 2.5.1. standard, object management group (OMG). https://www.omg.org/spec/UML/2.5.1
7. Giunti M, Sergioli G, Vivanet G, Pinna S (2019) Representing n-ary relations in the semantic web. Logic J IGPL 29(4):697–717
8. Smarandache F (2020) Extension of hypergraph to n-superhypergraph and to plithogenic n-superhypergraph, and extension of hyperalgebra to n-ary (classical-/neutro-/anti-)hyperalgebra. Neutrosophic Sets and Systems 33:18
9. Joslyn CA, Aksoy S, Arendt D, Firoz J, Jenkins L, Praggastis B, Purvine E, Zalewski M (2020) Hypergraph analytics of domain name system relationships. In: International workshop on algorithms and models for the web-graph. Springer, pp 1–15
10. Yadati N (2020) Neural message passing for multi-relational ordered and recursive hypergraphs. Adv Neural Inf Process Syst, vol 33
11. McDonald-Maier KD, Akehurst DH, Bordbar B, Howells WGJ (2008) Maths vs (meta)modelling - are we reinventing the wheel? In: Cordeiro J, Shishkov B, Ranchordas A, Helfert M (eds) ICSOFT 2008 - proceedings of the third international conference on software and data technologies. INSTICC Press, pp 313–322
12. Komar KS, Santra A, Bhowmick S, Chakravarthy S (2020) Eer → MLN: EER approach for modeling, mapping, and analyzing complex data using multilayer networks (MLNs). In: Conceptual modeling. Springer, pp 555–572
13. Boyd M, McBrien P (2005) Comparing and transforming between data models via an intermediate hypergraph data model. J Data Semant:69–109
14. Iung A, Carbonell J, Marchezan L, Rodrigues E, Bernardino M, Basso FP, Medeiros B (2020) Systematic mapping study on domain-specific language development tools. Empir Softw Eng 25(5):4205–4249
15. Mellor SJ, Scott K, Uhl A, Weise D (2004) MDA distilled: principles of model-driven architecture. Addison-Wesley Professional, Boston
16. Zabawa P, Hnatkowska B (2017) CDMM-F - domain languages framework. In: Borzemski L, Swiatek J, Wilimowska Z (eds) Information systems architecture and technology: proceedings of 38th international conference on information systems architecture and technology - ISAT 2017 - Part II, Szklarska Poręba, Poland, 17-19 Sept 2017. Advances in intelligent systems and computing. Springer, vol 656, pp 263–273
17. Krótkiewicz M, Zabawa P (2018) AODB And CDMM modeling - comparative case-study. In: Nguyen NT, Hoang DH, Hong T, Pham H, Trawinski B (eds) Intelligent information and database systems - 10th asian conference, ACIIDS 2018, Dong Hoi City, Vietnam, 19-21 March 2018, proceedings, Part II. Lecture notes in computer science. Springer, vol 10752, pp 57–68
18. Zabawa P (2018) Meta-modeling - decomposition of responsibilities. In: Nguyen NT, Hoang DH, Hong T, Pham H, Trawinski B (eds) Intelligent information and database systems - 10th asian conference, ACIIDS 2018, Dong Hoi City, Vietnam, 19-21 March 2018, proceedings, Part II. Lecture notes in computer science. Springer, vol 10752, pp 91–101
19. Jodłowiec M, Krótkiewicz M, Wojtkiewicz K (2019) Defining semantic networks using association-oriented metamodel. J Intell Fuzzy Syst 37(6):7453–7464
20. Han J, Sarica S, Shi F, Luo J (2021) Semantic Networks For Engineering Design: A Survey. Proc Design Society 1:2621–2630
21. Shapiro SC, Rapaport WJ (1987) SNEPS considered as a fully intensional propositional semantic network. In: The knowledge frontier. Springer, pp 262–315
22. Krótkiewicz M, Jodłowiec M, Wojtkiewicz K (2017) Semantic networks modeling with operand-operator structures in association-oriented metamodel. In: International conference on computational collective intelligence. Springer, pp 24–33
23. Chodrow P, Mellor A (2020) Annotated hypergraphs: models and applications. Appl Netw Sci, vol 5(1). https://doi.org/10.1007/s41109-020-0252-y
24. Xie Z (2021) A distributed hypergraph model for simulating the evolution of large coauthorship networks. Scientometrics 126(6):4609–4638. https://doi.org/10.1007/s11192-021-03991-2
25. Lin JC-W, Shao Y, Zhou Y, Pirouz M, Chen H-C (2019) A bi-LSTM mention hypergraph model with encoding schema for mention extraction. Eng Appl Artif Intell 85:175–181
26. Krótkiewicz M (2018) A novel inheritance mechanism for modeling knowledge representation systems. Comput Sci Inf Syst 15(1):51–78
27. Singh P, Sachdeva S (2020) A landscape of XML data from analytics perspective. Procedia Comput Sci 173:392–402
28. Date CJ (2019) E.F. Codd and relational theory lulu publishing services
29. Krótkiewicz M (2019) Cyclic value ranges model for specifying flowing resources in unified process metamodel. Enterprise Inf Syst 13(7-8):1046–1068
30. Bildhauer D (2011) Associations as first-class elements. In: Databases and information systems VI. IOS Press, pp 108–121
31. Krótkiewicz M, Jodłowiec M (2018) Modeling autoreferential relationships in association-oriented database metamodel. In: Świątek J, Borzemski L, Wilimowska Z (eds) Information systems architecture and technology: proceedings of 38th international conference on information systems architecture and technology – ISAT 2017. Springer, pp 49–62
32. Krótkiewicz M (2017) Association-oriented database model - n-ary associations. Int J Softw Eng Knowl Eng 27(2):281–320
33. Cai Y, Pan S, Wang X, Chen H, Cai X, Zuo M (2020) Measuring distance-based semantic similarity using meronymy and hyponymy relations. Neural Comput Appl 32(8):3521–3534
34. Tsatsaronis G, Varlamis I, Vazirgiannis M (2010) Text relatedness based on a word thesaurus. J Artif Intell Res 37:1–39
35. Dudycz H (2017) Application of semantic network visualization as a managerial support instrument in financial analyses. Online J Appl Knowl Manag 5(1):112–128
36. Žáček M, Homola D (2017) Analysis of the english morphology by semantic networks. In: AIP conference proceedings. AIP Publishing LLC, vol 1906, p 080006
37. Yoo S, Jeong O (2020) Automating the expansion of a knowledge graph. Expert Syst Appl 141:112965. https://doi.org/10.1016/j.eswa.2019.112965
38. Tanwar P, Prasad T, Dutta K (2022) A tour of various knowledge representation techniques in artificial intelligence for making

machines intelligent. In: Empowering artificial intelligence through machine learning. Apple academic press, pp 1–29
39. Jodlowiec M, Pietranik M (2019) Towards the pattern-based transformation of SBVR models to association-oriented models. In: Nguyen NT, Chbeir R, Exposito E, Aniorté P, Trawinski B (eds) Computational collective intelligence - 11th international conference, ICCCI 2019, Hendaye, France, 4-6 Sept 2019, proceedings, part I. Lecture notes in computer science. Springer, vol 11683, pp 79–90
40. Wu Q, Liu X, Qin J, Wang W, Zhou L (2021) A linguistic distribution behavioral multi-criteria group decision making model integrating extended generalized TODIM and quantum decision theory. Appl Soft Comput 98:106757. https://doi.org/10.1016/j.asoc.2020.106757
41. Huk M (2019) Training contextual neural networks with rectifier activation functions: role and adoption of sorting methods. J Intell Fuzzy Syst 37(6):7493–7502. https://doi.org/10.3233/jifs-179356
42. Saito N, Shao Y (2022) eGHWT: the extended generalized haar-walsh transform. J Math Imaging Vis 64(3):261–283. https://doi.org/10.1007/s10851-021-01064-w

Marcin Jodłowiec holds a doctoral degree in the field of information and communication technology awarded by Wroclaw University of Science and Technology in 2023. He currently works as a research and teaching assistant in the Process Analysis and Semantic Systems Group at the Department of Applied Informatics at Wroclaw University of Science and Technology. He is an author or co-author of over 25 scientific publications on topics including data modeling and metamodeling, database processes, methods and systems for knowledge representation, and artificial intelligence. He is a member of the organizing committees of two scientific conferences: The Asian Conference on Intelligent Information and Database Systems (ACIIDS) and the International Conference on Collective Intelligence (ICCCI), where he serves as the Publicity Chair. He has participated in a number of scientific research projects related to the design and production of IT systems. The projects he has worked on include signal analysis, image analysis, application of IT technologies in quality management of production units, as well as support for rescue and search operations.

Marek Krótkiewicz holds the position of Associate Professor in the Faculty of Information and Communication Technology (FICT) of Wrocław University of Science and Technology (WUST). He earned his computer science doctorate in 2001 and a DSc degree (habilitation) in information and communication technology at Wrocław University of Technology in 2020. His main interest is in data metamodeling, database models, software engineering, knowledge representation, signal processing, and classification. He is an author of over 50 international scientific publications published in journals and at conferences. One of his achievements is integrated, script-based graphic processing software that integrates commonly known algorithms and a wide range of his solutions, previously published in national and international journals. He is also an author of the Association-Oriented Database model that Knowledge and Information Engineering Group has used in Semantic Knowledge Base development. He participated in many scientific and R&D projects. The developed advanced scientific and technical solutions are commercialized within the Science in Software LTD company established for this purpose, of which he is the president.

Piotr Zabawa holds a position of an assistant professor in the Department of Applied Informatics in the Faculty of Information and Communication Technology at Wrocław University of Science and Technology. He studied electronics, automatic control and computer science and got his M.Sc. in 1991. Then he got his awarded Ph.D. degree in automatic control at AGH University of Science and Technology in 1993. His main interest is in software engineering, software development processes and their automation, software architecture and especially in modeling languages and meta-modeling languages. One of his achievements is the Context-Driven Meta-Modeling approach to defining modeling languages, which are applicable to modeling software-intensive systems. This approach consists of a very general meta-modeling language which makes it possible to define modeling languages on a user demand at run-time. The software project artifacts can be generated in models created in the mentioned above user-defined modeling languages automatically.

Affiliations

Marcin Jodłowiec[1] · **Marek Krótkiewicz**[1] · **Piotr Zabawa**[1]

Marek Krótkiewicz
marek.krotkiewicz@pwr.edu.pl

Piotr Zabawa
piotr.zabawa@pwr.edu.pl

[1] Department of Applied Informatics, Wrocław University of Science and Technology, Wybrzeże Stanisława Wyspiańskiego 27, Wrocław, 50-370, Lower Silesia, Poland