Towards Conceptual Modelling Interoperability in a Web Tool for Ontology Engineering

Germán Braun[1,2,3], Christian Gimenez[1], Pablo Fillottrani[3,4], and Laura Cecchi[1]

[1] *Grupo de Investigación en Lenguajes e Inteligencia Artificial*
Departamento de Teoría de la Computación - Facultad de Informática
Universidad Nacional del Comahue
[2] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)*
[3] *Laboratorio de I&D en Ingeniería de Software y Sistemas de Información*
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
[4] *Comisión de Investigaciones Científicas de la Prov. Bs.As (CIC)*

Abstract The definition of suitable visual paradigms for ontology modelling is still an open issue. Despite obvious differences between the expressiveness of conceptual modelling (CM) languages and ontologies, many proposed tools have been based on UML, EER and ORM. Additionally, all of these tools support only one CM as visual language, reducing even more their modelling capabilities. In previous works, we have presented *crowd* as a Web architecture for graphical ontology designing in UML and logical reasoning to verify the relevant properties of these models. The aim of this tool is to extend the reasoning capabilities on top of visual representations as much as possible. In this paper, we present an extended *crowd* architecture and a new prototype focusing on an ontology-driven metamodel to enable different CMs visual languages for ontology modelling. Thus facilitating inter-model assertions across models represented in different languages, converting between modelling languages and reasoning on them. Finally, we detail the new architecture and demonstrate the usage of the prototype with simple examples.

1 Introduction and Motivation

The definition of suitable visual paradigms for ontology modelling is still an open issue [1]. Despite obvious differences between the expressiveness of conceptual modelling (CM) languages and ontologies languages, many proposed modelling tools have been based on Unified Modeling Language (UML) [2], Extended Entity-Relationship (EER) [3] and Object-Role Modeling (ORM) [4]. Additionally, all of these tools support only one CM as visual language, reducing even more their modelling capabilities.

Ontologies are conceptualisations of domains, often shared by a community of users. One of the most commonly used ontology modelling languages is the Web Ontology Language (OWL) [5]. OWL formal underpinning is provided by

description logics (DLs) [6], which is a family of knowledge representation formalisms with well-understood formal properties. Particularly, there exist important applications of DLs such as expressing the ontology of data sources and integrating multiple sources [7]. Both topics have been highlighted again in [8], whose authors have emphasised the needs for multi-models, which arises when semantically related data is organised under different schemes. Linking different ontologies into the Semantic Web or ontology-based systems emphasises the importance of the interoperability amongst conceptual data models. Based on previous results [9, 10, 11], each model can be logically reconstructed so as to automatically reason over them using the standard DL properties. Thus, ontologies can be expressed at a level closer to that of human conceptualisation (e.g., representing conceptual schemas), determining consistency and automatically classifying descriptions. In this direction, we have started the formalisation of an novel approach integrating visualisations to this multi-model proposal [12, 13]. Aiming at developing user-friendly paradigms for presenting knowledge of the data in a way that helps in formulating queries, we expect understanding and defining what the relevant information in a given context is, and representing it in an appropriated way.

Based on this motivation, an ontology-driven metamodel, known as Keet-Fillottrani (KF), has been designed and formalised in OWL 2 [14] to enable different domains views by means of logic-based reconstructions and inter-model assertions. Currently, its specification unifies UML v2.4.1, EER and ORM2. While such languages seem similar, they are known to be distinct and no unifying framework exists respecting all of their language features. Few approaches have been partially proposed, where the strengths and weaknesses of the languages are highlighted and some formal procedures are given to derive "well-formed" input diagrams into "well-formed" output ones [15, 16]. Nevertheless, linking and converting between graphical modelling languages are not considered in depth.

crowd [17, 18] is a graphical modelling tool being supported by both Universidad Nacional del Comahue and Universidad Nacional del Sur of Argentina. The intention behinds the tool is to assist users to design ontologies and conceptual models adopting standard CM languages and employing complete logical reasoning to verify the satisfiability of specifications, infer implicit constraints and suggest new ones. Despite obvious differences between the expressiveness of conceptual modelling (CM) languages and ontologies languages, many tools have partially validated this claim [19, 20, 21]. Nevertheless, the aim of *crowd* is going towards a multi-model support in order to reason on top of visual representations of ontologies as much as possible. Empowered by Web technologies, *crowd* has been designed as a scalable and maintainable architecture for adapting new engines, graphical languages, design methodologies and back-end reasoners. It has been conceived from scratch as a graphical-centric tool for ontology modelling, supporting standard CM languages and considering the possibility to expand its graphical primitives for more expressiveness. Currently, it supports a subset of UML and is connected to the RACER logic-based reasoner [22]. A prototype

is available at `http://crowd.fi.uncoma.edu.ar`, while its source code is to be available for downloading after releasing the first beta version.

Particularly, in this work we present an extended *crowd* architecture and a prototype focusing on the metamodelling approach previously described. Furthermore, we provide more mature support of the UML language and the capability of incorporate ORM2 and EER required for the KF metamodel. From an implementation point of view, the transitions amongst these languages is achieved by creating a corresponding metamodel expressing the input model by means of the formalisation in [14, 23]. After that, we unify concepts in common of each graphical language considering their discrepancies applying mapping rules as defined in [24]. The current prototype supports a subset of static entities composes by Object Types, Subsumption and Attributes and only converts graphically from UML to EER models.

This work is structured as follows. Section 2 gives a summary of the KF metamodel and explains its main objectives. Section 3 includes a review of the first architecture of *crowd* and introduces a new one integrating multi-model support based on the KF metamodel. A metamodelling prototype together with some examples of use are presented in section 4. To conclude the paper, section 5 details some related works, while section 6 elaborates on final considerations and directions for future works.

2 An Overview of the KF Metamodel

The intention behinds the ontology-driven KF metamodel is to provide interoperability, integration and conversion of conceptual data models represented in different languages. For this purpose, the metamodel specifies an approach for transforming a model in one language into another in order to be able to assert semantically proper links between them. It unifies all main static entities and hierarchy of constraints. The first one details entities as Object Types, Relationships (such as Attributes and Subsumptions) and Roles, amongst others. The second one reconciles mandatory and uniqueness constraints, which are the most basic ones common to all CM languages, as well as cardinality, identifiers and relationship constraints. In this way, the metamodel covers all the native features of the CM languages. Fig. 1 depicts the available static entities as a UML Class Diagram (only for facilitating its readability). A white fill of a class icon means that that entity is not present in any of the languages, a light gray fill means that it is present in one language, dark gray that it is present in two, and a black fill that it is present in all three families of languages. An example of the use of the metamodel for integration of conceptual data models has been extracted from [25] and is shown in Fig. 2. The EER diagram depicts a generic termbank while the UML specifies aspects of a specific termbank as the isiZulu[1], which is the most widely spoken home language in South Africa. The example shows how the integration can be done asserting links between entities in the

[1] `http://africanlanguages.com/zulu/`

two models. Some of them are obvious as the subsumption between `isiZuluTerm` and `Term`, although others are not, such as the mapping between the `Morphinfo` EER relationship and the UML association relating `MorphologicalSyntaxInfo` and `isiZuluTerm`. This case seems to be more complicated since they define different cardinalities so that modellers should be asked to accept or not such a relation.

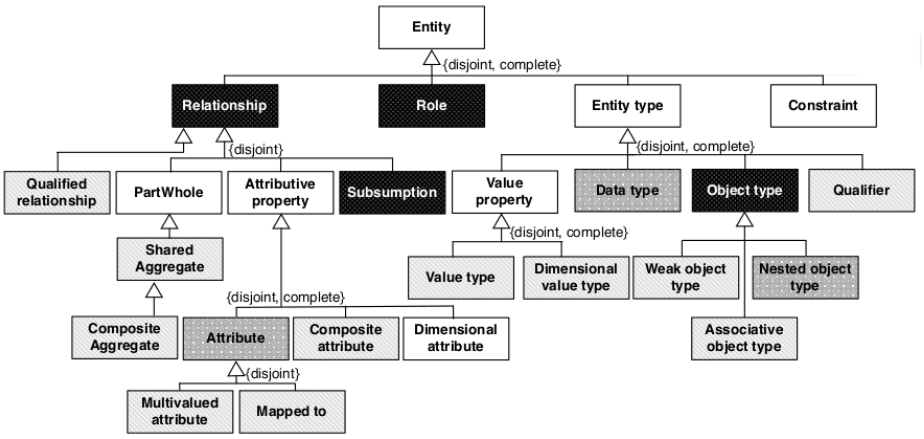

Figure 1. Metamodel static entities as depicted in [25]. Disjointness axioms amongst the subclasses of Relationship are: {PartWhole, Attributive property, Subsumption } and {Qualified relationship, Attributive property, Subsumption}

In the same direction, the KF metamodel also defines four groups of mapping rules divided in: 1:1 mappings, transformations, approximations, and no alternatives. 1:1 mappings are those where the elements are the same from an ontological point of view and as a consequence, the conversions are in simple steps. One of the simplest case is Subsumption. Secondly, transformations involve elements which are essentially the same but not from a syntax point of view. Such transformations take place, for instance, from ORM2 value types to UML and EER attributes. Approximations are special kinds of rules where modellers are required to accept or reject the conversions or links. These rules are based on patterns which could lead to different outcomes. Finally, since these languages do not have the same expressiveness, some of their features can not be neither represented nor approximated in a target language. As an example, the conversion from ORM2 compound cardinality to UML is not possible.

To depict these mapping rules and conversion capabilities, we present in Fig. 3 another example where a new EER model is generated from an initial UML by executing three 1:1 mappings and six transformations. The 1:1 mappings are executed for both UML classes (`Person` and `Student`) generating the corresponding EER entities and the subsumption between these classes generating

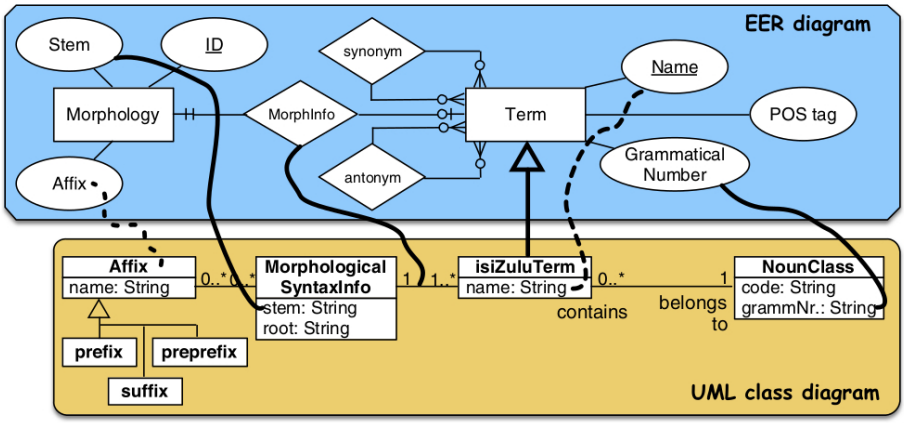

Figure 2. Example extracted from [25]. It shows an integration scenario in which the metamodel could help us to map their entities.

also the corresponding subsumption in the EER model. On the other hand, one transformation is run for each attribute to model the same attributes in EER changing the graphical syntax but keeping the very same meaning.

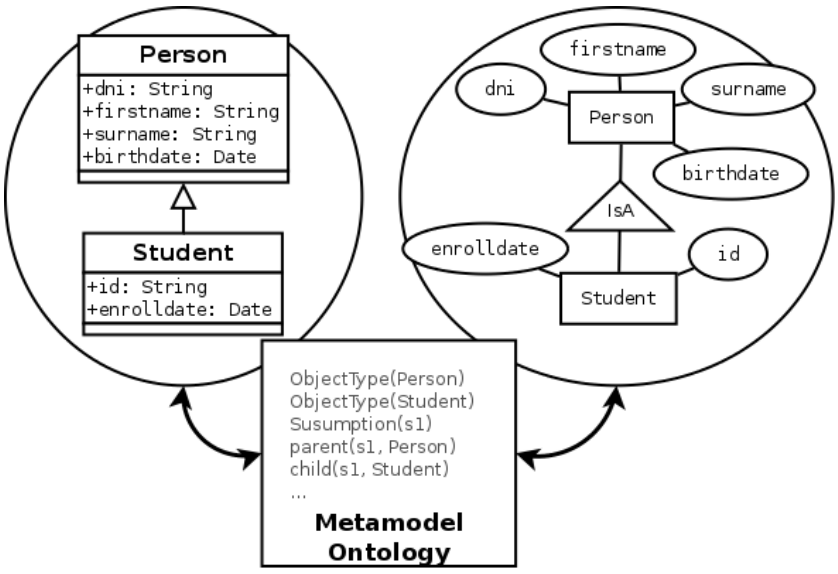

Figure 3. This conversion scenario depicts another possible application of the metamodel. In this case, the first UML model is converted to the EER one through 1:1 mappings for classes and subsumption and transformations for attributes.

3 Conceptual Modelling Interoperability in *crowd*

In this section, we review the first *crowd* architecture and give an overview of its main features. After that, we show the new *crowd* architecture for the metamodel-driven interoperability support.

3.1 *crowd* Overview

The intention behinds *crowd* is to assist users to design ontologies and conceptual models adopting standard CM languages and providing graphical support for developers. Complete logical reasoning is employed by the tool to verify the satisfiability of specifications, infer implicit constraints and suggest new ones. The leverage of automated reasoning is enabled by a precise semantic definition of all the elements of the diagrams. Hence, diagrams constraints are internally translated into a logic-based formalism capturing typical features of models. To this end, the tool is fully integrated with a powerful logic-based reasoning server acting as a background inference engine. Moreover, since *crowd* is based on a deduction-complete notion of reasoning support relative to the diagram graphical syntax, users will see the original model graphically completed with all the deductions and expressed in the graphical language itself. This includes checking class and relationship consistency, discovering implied class and cardinality constraints. *crowd* only focuses on graphical modelling of schemes, while it does not consider individuals.

As presented in [17], *crowd* allows graphically creating and editing simple UML class diagrams, although more expressive OWL 2 [26] constraints can be appended to models by inserting OWL statements. The communication between the front-end with the reasoner is by means of the OWLlink protocol [27]. Moreover, it supports satisfiability checking on simple UML graphical diagrams encoded in $\mathcal{ALCQI}$ Description Logics (DL) [6], as demonstrated in [9]. Finally, the tool continues being developed with updated and scalable graphical libraries and technologies and recently support to users sessions is at phase of alpha-testing. The background visualisation process of *crowd* has been also presented in [13]. Such a process integrates the theoretical foundations of the tool with logic-based reasoning and metamodelling capabilities. We claim that *crowd* is a tool to evaluate the integration of graphical languages with reasoning systems and thus assisting users by means of methodologies for conceptual modelling and ontology design.

3.2 A New Architecture for Metamodelling Support

Aiming to support multi-model capabilities, *crowd* needs to provide EER and ORM2 both in the back and in the front-end. Users should visualise and edit these graphical models and, on demand, send them to the back-end for mapping them into a metamodel instance and convert them into another desirable model (if possible). As a consequence, interface feature will be also set-up according to the corresponding target model.

Client. Models must be rendered accordingly with its own interface setting up also for editing. For that matter, the *crowd* architecture provides a multi-model front-end for UML, ORM2, and EER, which enables to switch amongst them. To this end, the client side has been totally refactored. As the Fig. 4 depicts, *crowd* client has been designed to support a view-model for each interface matching the structure of the underlying JointJS[2] and BackboneJS libraries.

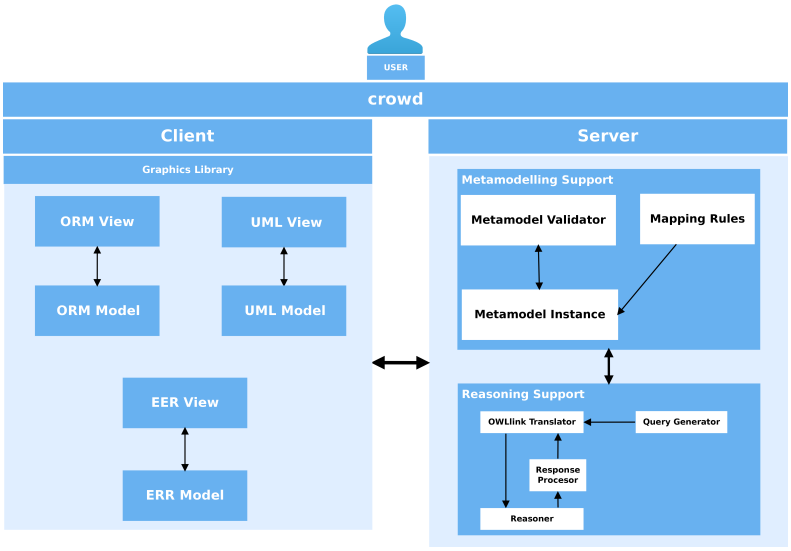

Figure 4. Extended *crowd* client-server architecture integrating metamodelling capabilities. The sub-modules in `Reasoning Support` implement features detailed in [17].

Server. The extended *crowd* back-end now provides a set of modules to enable metamodelling support, which have been fully integrated with the previous ones because of reasoning capabilities. Such modules are `Metamodel Validator`, `Metamodel Instance` and `Mapping Rules`. The first one supplies the metamodel itself for identifying and validating the different primitives of each model and the relationships amongst them. Particularly, this module implements the classes for generating an object-oriented representation of the input model in a metamodel instance. The second one starts the conversion process from any input model to the corresponding metamodel and vice versa. This process is also supported by a set of mapping rules provided by the `Mapping Rules` module and explained in section 2: 1:1 mapping, transformations and approximations. The resulting intermediate representation is a set of object instances of the metamodel static entities for the provided conceptual model. A schematic view of this approach

[2] `http://www.jointjs.com/`

is illustrated in the example of the Fig. 5 showing a metamodel instance from a UML class diagram together with its internal JSON codification in Fig. 6. After this step, the tool is able to express the underlying model in any target modelling language and thus be sent to the `Reasoning Support` sub-modules.

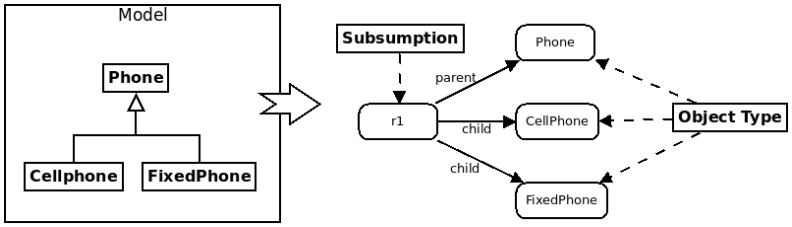

Figure 5. A graphical representation of the metamodel instance generated for the input UML subsumption.

```
{"classes": [
  {"name":"Phone", "attrs":[], "methods":[]},
  {"name":"CellPhone", "attrs":[], "methods":[]},
  {"name":"FixedPhone", "attrs":[], "methods":[]}],
"links": [
  {"classes" : ["CellPhone", "FixedPhone"],
  "multiplicity" : null,
  "name" : "r1",
  "type" : "generalisation",
  "parent" : "Phone",
  "constraint" : [] }] }
```

$\Rightarrow$

```
{"Object type" : [
    {"name" : "Phone"},
    {"name" : "CellPhone"},
    {"name" : "FixedPhone"}],
"Subsumption" : [{"name" : "r1",
    "parent" : "Phone",
    "children" : ["CellPhone",
        "FixedPhone"]}],
"Association" : [],
"Object type cardinality" : [],
"Attribute" : [] }
```

Figure 6. A transformation between an UML JSON representation into a Metamodel one.

The set of modules in `Reasoning Support` have been already detailed in [17]. Briefly, they provide the complete support for automatic reasoning by translating conceptual models to the OWL 2 language together with a set of queries to retrieve their properties. These queries are executed by an off-the-shelf inference engine before returning to the front-end.

Particularly, the core of this support is a graphical-logical mapping [12], which is a complementary formalisation to coordinate them in the context of a graphical-centric tool. There, we coordinate different ways to encode graphical primitives of a language into a decidable logical formalism, as also depicted in Fig. 7. Formally, we have identified a set of graphical elements independent of any language and introduced a mapping function Θ. This function is defined as the union of the logical representations encoding each graphical element. Therefore, Ω is a consistent graphical model if $\mathcal{O}$ is a consistent ontology generated through Θ in a target logic. Likewise, $\Theta(\mathcal{O})$ is a new consistent graphical model if the ontology $\mathcal{O}'$ is also consistent. From Ω and Ω', and their respective un-

derlying ontologies $\mathcal{O}$ and $\mathcal{O}'$ through Θ and Θ^{-1}, we define the integration of the graphical support with reasoning by rendering reasoning results in the same visual notation. The mapping function should then be reversible in order to visualise in the same original graphical formalism the results of the reasoning. At conceptual level, this mapping closes the diagram, however, its validation requires establishing a correspondence between the graphical models.

Finally, the integration between metamodelling, visualisations and reasoning support keeps the previous *crowd* capabilities in order to reason on top of visual representations using any of the conceptual modelling languages.

Figure 7. $\Theta(\Omega)$ is the DL definition for the graphical primitive Ω. These primitives map to many DL expressions and vice versa.

4 *crowd* Metamodelling Prototype and Example of Use

crowd prototype front-end has been developed in JavaScript, while its back-end runs in an Apache server and has been developed in PHP. Currently, this first prototype supports the conversion from UML to EER, which requires both an UML to Metamodel and a Metamodel to EER transformation. Such a conversion is done from simple UML diagrams including only classes, attributes and subsumption (without constraints) to a new EER diagram going through an internal metamodel and applying the mapping rules from Table 1. The EER diagram generated also depicts entities, their attributes and the corresponding subsumption in its own graphical representation. Once the conversion is done, the *crowd* interface remains in an EER mode together with all the front-end features in order to continue editing the new model.

	Metamodel	UML	EER	ORM2
Entity Type	Object Type	Class	Entity	Object Type
	Data Type	Data Type	Not Apply	Data Type
Relationship	Relationship	Association	Relationship	Fact Type
	Subsumption	Subsumption	Subsumption	Subsumption
	Attribute	Attribute (With datatype)	Attribute (Without datatype)	Not Apply

Table 1. This table shows how primitives of the models are converted to others.

To show the use of the prototype, we revisit the conversion example diagrams from Fig. 3 but now rendered in *crowd*. The Fig. 8 depicts the initial

UML Class Diagram modelling `Person` and `Student` classes, their attributes and datatypes `dni:String`, `firstname:String`, `surname:String`, `birthdate:Date`, `id:String`, `enrolldate:Data` and a subsumption relationship. The resulting EER diagram is shown in the Fig. 9.

Before ending this section, it is important to remark that even though datatypes in UML are not converted, they are saved in the metamodel representation. However, restoring the original model is not necessarily aimed by this approach.

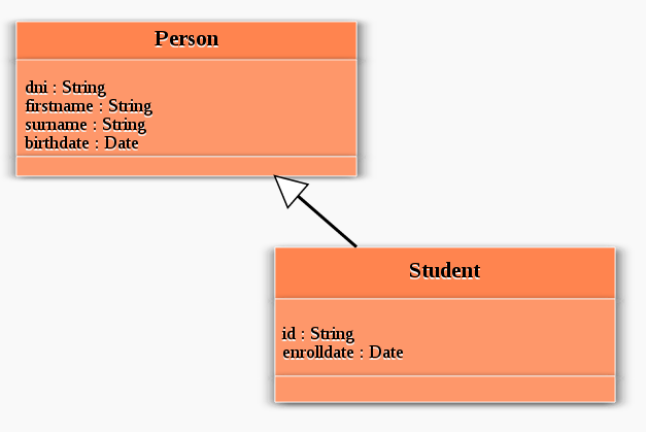

Figure 8. An initial UML model as visualised in the current *crowd* prototype.

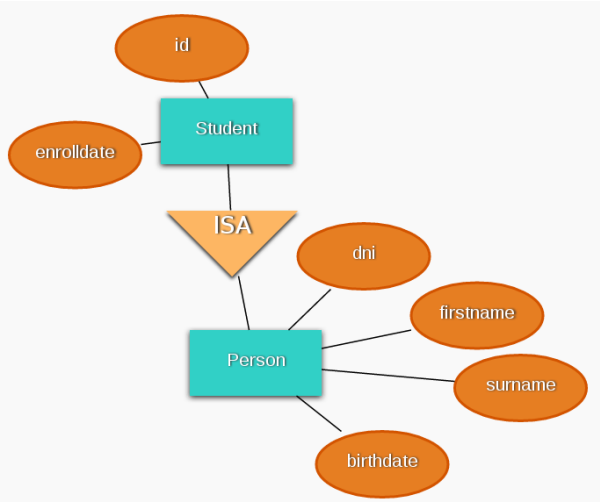

Figure 9. The converted EER model as visualised in the current *crowd* prototype.

5 Comparison with other Tools

From a formal point of view, there is a variety of intents to transforms conceptual models from a graphical language into another without considering validating inter-model assertions. One of the most relevant work is presented in [28] by Boy and McBrien. They support ER, UML and ORM schemes and include transformation rules, but the expressiveness of models is limited since they omit roles, aggregation, weak entity types and several constraints. Halpin [29, 16] analyses mappings from ORM to UML and from ORM to ER. However, these are only theoretical and both mappings are presented as separated approaches as well.

On the other hand, some aspects related to the transformation between CM languages have been undertaken. In this respect, the tool astah[3] allows to convert from UML models to EER and vice versa but mapping mainly UML classes to ER entities and keeping the UML attributes in the same entity. No ORM nor reasoning support is provided. Visual Paradigm[4] is another commercial tool suite for software development for generating Class diagrams from ER ones. The tool maps classes, associations and attributes into entities, associations and attributes maintaining a Baker-like syntax [30] between both models. Similar to the previous suite, no automatic reasoning is provided as well as linking and interoperability capabilities either.

With reference to visualisation process of *crowd*, we have surveyed many related tools with graphical ontological representation, such as ICOM [31], NORMA [20], the well-known Protégé [32], OWLGrED [21], Menthor[5] (OntoUML [33]), amongst others. However, although all of them present different degrees of graphical support, multi-modelling capabilities are missing. Ontologies and conceptual models are depicted using EER, UML or ORM but neither conversion nor linking is provided between these models.

As a conclusion, despite the theoretical or practical achievements, these related works show the needs for an integrated and ontological analysis of UML, EER and ORM together with linking and converting purposes. As far as we know, our new multi-model *crowd* architecture is the only one that proposes this approach to be completely implemented in a graphic-centric Web tool for ontology modelling dealing with back-end reasoning systems.

6 Conclusions and Future Works

Interoperability amongst different CM languages for ontology modelling is key for graphical tools because allows modellers to choose the one of their preference in order to develop complex systems. Furthermore, linking different ontologies or ontology-based models also requires interoperability amongst conceptual data

[3] `http://astah.net/features/convert-diagrams-and-models`
[4] `https://www.visual-paradigm.com/tutorials/generatecdfromerd.jsp`
[5] `http://www.menthor.net`

models. Thus, the definition and implementation of the KF metamodel provides this dimension intended for UML v2.4.1, ORM2 and EER CM languages.

In this work, we have introduced an extended Web architecture for our tool *crowd* by adding metamodelling support. The *crowd* architecture also provides a multi-model front-end for UML, ORM2, and EER, which enables switching amongst them. Additionally, we have implemented a new *crowd* prototype for managing the conversion from UML to EER going through an internal metamodel representation and applying only 1:1 mappings between UML classes and EER entities, subsumptions and attributes transformations. This prototype supports the main primitives of UML and EER entities, attributes and subsumption. In this respect, we are currently developing the modules for full support of EER and ORM2 languages together with the corresponding DL representations for taking advantages of automatic reasoning.

In future, we plan to complete the implementation of *crowd* as has been described in this paper and evaluate the whole approach in depth. Finally, it would be worth considering to expand the metamodel for supporting other visual formalisms different from the already defined ones and thus going towards suitable visualisations in tools.

Acknowledgements

The authors would like to thank the anonymous referees for their comments and suggestions. This work is based upon research partially supported by the Universidad Nacional del Comahue (Project ID: 04/F014), the Universidad Nacional del Sur (Project ID: 24/N038), the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), the Consejo Interuniversitario Nacional (CIN) and the Comisión de Investigaciones Científicas de la prov. de Buenos Aires (CIC).

References

1. Ivanova, V., Lambrix, P., Lohmann, S., Pesquita, C., eds.: Proceedings of the Second International Workshop on Visualization and Interaction for Ontologies and Linked Data co-located with the 15th International Semantic Web Conference, VOILA@ISWC 2016. Volume 1704 of CEUR Workshop Proceedings., CEUR-WS.org (2016)
2. Booch, G., Rumbaugh, J., Jacobson, I.: Unified Modeling Language User Guide. Addison-Wesley Professional (2005)
3. Gogolla, M.: Extended Entity-Relationship Model: Fundamentals and Pragmatics. Springer-Verlag (1994)
4. Halpin, T., Morgan, T.: Information Modeling and Relational Databases. 2 edn. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2008)
5. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C., eds.: OWL 2 Web Ontology Language Profiles. Second edition edn. World Wide Web Consortium (December 2012) `https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/`.
6. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York, NY, USA (2003)

7. Borgida, A., Lenzerini, M., Rosati, R.: The description logic handbook. Cambridge University Press, New York, NY, USA (2003)
8. Abiteboul, S., Arenas, M., Barceló, P., Bienvenu, M., Calvanese, D., David, C., Hull, R., Hüllermeier, E., Kimelfeld, B., Libkin, L., Martens, W., Milo, T., Murlak, F., Neven, F., Ortiz, M., Schwentick, T., Stoyanovich, J., Su, J., Suciu, D., Vianu, V., Yi, K.: Research directions for principles of data management (abridged). SIGMOD Record (2016)
9. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artif. Intell. **168**(1-2) (2005) 70–118
10. Franconi, E., Mosca, A., Solomakhin, D.: ORM2: formalisation and encoding in OWL2. In: On the Move to Meaningful Internet Systems: OTM 2012 Workshops, Confederated International Workshops: OTM Academy, Industry Case Studies Program, EI2N, INBAST, META4eS, OnToContent, ORM, SeDeS, SINCOM, and SOMOCO 2012. (2012)
11. Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: Complexity of reasoning in entity relationship models. In: Proceedings of the 2007 International Workshop on Description Logics (DL). (2007)
12. GILIA: Integrating Graphical Support with Reasoning in a Methodology for Ontology Evolution. Technical report (2015) available at `http://tinyurl.com/nm4nos2` from Jun 7, 2015.
13. Braun, G., Gimenez, C., Cecchi, L., Fillottrani, P.: Towards a visualisation process for ontology-based conceptual modelling. In Baracho, R.M.A., Isotani, S., Almeida, M.B., eds.: ONTOBRAS – Brazilian Ontology Research Seminar (ONTOBRAS). Number 1862 in CEUR Workshop Proceedings, Aachen (2016) 107–118
14. Keet, C.M., Fillottrani, P.R.: An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2. Data & Knowledge Engineering (2015)
15. Bollen, P.: A formal ORM-to-UML mapping algorithm. Technical report, METEOR, Maastricht University School of Business and Economics (2002)
16. Halpin, T.A.: Information analysis in UML and ORM: A comparison. In: Advanced Topics in Database Research, Vol. 1. (2002)
17. Gimenez, C., Braun, G., Cecchi, L., Fillottrani, L.: crowd: A Tool for Conceptual Modelling assisted by Automated Reasoning - Preliminary Report. In: the 2nd Simposio Argentino de Ontologías y sus Aplicaciones SAOA '16 JAIIO '16. (2016)
18. Gimenez, C., Braun, G., Cecchi, L., Fillottrani, P.: Una Arquitectura Cliente-Servidor para Modelado Conceptual Asistido por Razonamiento Automático. In: XVIII Workshop de Investigadores en Ciencias de la Computación. (2016)
19. Fillottrani, P., Franconi, E., Tessaris, S.: The ICOM 3.0 intelligent conceptual modelling tool and methodology. Semantic Web (2012)
20. Curland, M., Halpin, T.A.: The NORMA Software Tool for ORM 2. In: CAiSE Forum. Lecture Notes in Business Information Processing, Springer (2010)
21. Cerans, K., Ovcinnikova, J., Liepins, R., Sprogis, A.: Advanced owl 2.0 ontology visualization in owlgred. In: DB&IS. Frontiers in Artificial Intelligence and Applications, IOS Press (2012)
22. Haarslev, V., Möller, R.: Racer system description. In Goré, R., Leitsch, A., Nipkow, T., eds.: International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, Siena, Italy, Springer-Verlag (2001) 701–705
23. Fillottrani, P.R., Keet, C.M.: KF metamodel formalization. CoRR **abs/1412.6545** (2014)
24. Fillottrani, P.R., Keet, C.M.: Conceptual model interoperability: A metamodel-driven approach. In: Rules on the Web. From Theory to Applications - 8th International Symposium, RuleML 2014, Co-located with the 21st European Conference

on Artificial Intelligence, ECAI 2014, Prague, Czech Republic, August 18-20, 2014. Proceedings. (2014)
25. Keet, C.M., Fillottrani, P.R.: Toward an ontology-driven unifying metamodel for UML class diagrams, EER, and ORM2. In: Conceptual Modeling - 32th International Conference, ER 2013, Hong-Kong, China, November 11-13, 2013. Proceedings. (2013)
26. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The Next Step for OWL. Web Semant. (2008)
27. Liebig, T., Luther, M., Noppens, O., Wessel, M.: Owllink. Semantic Web **2**(1) (2011) 23–32
28. Boyd, M., McBrien, P. In: Comparing and Transforming Between Data Models Via an Intermediate Hypergraph Data Model. Springer Berlin Heidelberg (2005)
29. Halpin, T.A.: Comparing metamodels for er, ORM and UML data models. In: Advanced Topics in Database Research, Vol. 3. (2004)
30. Barker, R.: CASE Method - Entity Relationship Modellierung. Addison-Wesley (1992)
31. Fillottrani, P.R., Franconi, E., Tessaris, S.: The ICOM 3.0 intelligent conceptual modelling tool and methodology. Semantic Web (2012)
32. Knublauch, H., Fergerson, R., Noy, N., Musen, M.: The Protégé OWL plugin: An open development environment for semantic web applications. (2004)
33. Guizzardi, G.: Ontological foundations for structural conceptual models. PhD thesis, University of Twente, Enschede, The Netherlands, Enschede (October 2005)