Journal of Intelligent Information Systems (2023) 60:437–462
https://doi.org/10.1007/s10844-022-00731-7

A framework for interoperability between models with hybrid tools

Germán Braun[1,2] · **Pablo Rubén Fillottrani**[3,4] · **C. Maria Keet**[5]

Received: 24 February 2022 / Revised: 3 July 2022 / Accepted: 18 July 2022 /
Published online: 29 July 2022

Abstract
Complex system development and maintenance face the challenge of dealing with different types of models due to language affordances, preferences, sizes, and so forth that involve interaction between users with different levels of proficiency. Current conceptual data modelling tools do not fully support these modes of working. It requires that the interaction between multiple models in multiple languages is clearly specified to ensure they keep their intended semantics, which is lacking in extant tools. The key objective is to devise a mechanism to support semantic interoperability in hybrid tools for multi-modal modelling in a plurality of paradigms, all within one system. We propose FaCIL, a framework for such hybrid modelling tools. We design and realise the framework FaCIL, which maps UML, ER and ORM2 into a common metamodel with rules that provide the central point for management among the models and that links to the formalisation and logic-based automated reasoning. FaCIL supports the ability to represent models in different formats while preserving their semantics, and several editing workflows are supported within the framework. It has a clear separation of concerns for typical conceptual modelling activities in an interoperable and extensible way. FaCIL structures and facilitates the interaction between visual and textual conceptual models, their formal specifications, and abstractions as well as tracking and propagating updates across all the representations. FaCIL is compared against the requirements, implemented in crowd 2.0, and assessed with a use case. The proof-of-concept implementation in the web-based modelling tool crowd 2.0 demonstrates its viability. The framework also meets the requirements and fully supports the use case.

Keywords Information systems · Conceptual modelling · Semantic interoperability · Ontologies · Logic-based reasoning

Pablo Rubén Fillottrani and C. Maria Keet contributed equally to this work.

✉ Germán Braun
german.braun@fi.uncoma.edu.ar

Extended author information available on the last page of the article

1 Introduction

Complex software development requires the management of models that not only represent but also help to understand, analyse, and predict properties of the different artefacts being engineered. A relevant subset of these models are called *conceptual models* (Guarino et al., 2020), and they are distinguished by including the following characteristics (Mayr and Thalheim, 2021): i) being associated to a conceptual space (the *a priori* semantics, which is provided by a community of practice); and ii) allowing relationships, like generalisation and aggregation, between the modelled elements. Conceptual models are extensively used in software engineering, and several software technologies are grounded on them, for example model-driven development or model-driven engineering (Whittle et al., 2014), method engineering (Jeusfeld et al., 2009), and model-driven architecture (Brown, 2004). Conceptual modelling is the process of building a representation of a conceptual model, requiring one or more modelling languages and involving engineering activities (Thalheim, 2010) which are mostly collaborative in nature. Among the myriad of types of models and languages, we focus on *conceptual data modelling*, which is used in the data analysis stage of database development, application design or big data processing to capture the data storage and manipulation requirements of the universe of discourse. A Conceptual Data Modelling Language (CDML) provides one or more of: diagrammatic techniques and/or natural language features to support communication between stakeholders, and formal meaning to furnish an unambiguous semantics that allows reasoning about the model properties.

There are long-standing different opinions and communities of practice on which CDML is best for a particular task or on the whole. Yet, people need to collaborate, intelligent systems using multiple technologies need to be developed, and disparate systems have to be integrated. The heterogeneity in this case concerns, as a minimum, different visual modelling languages, notably UML Class Diagrams (Object Management Group, 2017), EER (Thalheim, 2009), and ORM (Halpin and Morgan, 2008), which also may be verbalised in a natural language of choice and perhaps also authored in a natural language such as SBVR (Object Management Group, 2022). In addition, if the whole, and possibly integrated, conceptual data model becomes large, it may be necessary to send it to an automated reasoner to check it for any inconsistencies and implicit constraints so as to verify or improve the model's quality (Farré et al., 2013; Sportelli and Franconi, 2016). Further, the models may be used at runtime, such as in ontology-based data access tools (Calvanese et al., 2017), like for data integration and analysis of, e.g., food systems in the Mediterranean (Calvanese et al., 2016). Besides these examples, a practitioner survey found that heterogeneity was also found to be sustained in practice, mainly because of language limitations (ontological completeness and perspective), suitability for stakeholders and familiarity, and size or complexity of the domain (Sabegh and Recker, 2017). Put differently: this situation will not go away.

The heterogeneity also acts out in different ways of representing conceptual models (including certain application ontologies): visualised in a diagram, verbalised in (pseudo-) natural language, serialised for computational use, and possibly also formalised in a logic. Combined, this requires multi-modal modelling in a plurality of paradigms and that is supported by hybrid tools, i.e., systems that not only include but also somehow reconcile representations of two or more styles of representation. The main question then is *how to realise such a reconciliation in a hybrid tool in a way that is systematic, transparent, and extensible?*

This first requires a theoretical foundation on which to base the alignments of the modalities and, underlying that, some form of interoperability across the languages. A recent review identified main challenges, including overlap and inconsistency across models, complexity, maintainability, language variations and tool support, and user comprehension (Ong and Jabbari, 2019). Theoretical approaches to address some of these include a common hypergraph (Boyd and McBrien, 2005), rules with Datalog (Atzeni et al., 2012), and metamodels with mappings and rules (Fillottrani and Keet, 2014; Venable and Grundy, 1995). Regarding tooling, Pounamu (Zhu et al., 2004) and, recently, crowd (Braun et al., 2021) implement a part of the theory to support the heterogeneity. What the theories and tools fall short of are either multiple CDMLs, the verbalisation, or the formalisation for runtime use including inconsistency detection, or they support only fragments of prior CDMLs.

To make this heterogeneity run smoothly nonetheless, a first step is to devise a modelling infrastructure that can handle it. We aim to solve these issues by proposing the Framework for SemantiC Interoperability of conceptual data modelling Languages, FaCIL, as a basis for hybrid modelling tools with relations between components and a workflow that uses them. We instantiate FaCIL for one design specifically: to cater for UML, ER, and ORM2 interoperability. It enables one to seamlessly automatically transform models across CDML barriers, including automatic propagation of updates, and to reason over them if desired, all in one place. The approach was evaluated by implementing the common core fragment that the three main CDMLs agree on (Fillottrani and Keet, 2021) availing of the KF metamodel (Keet and Fillottrani, 2015) and the interoperability rules (Fillottrani and Keet, 2014) together with a formalisation in a Description Logic and serialised in OWL (Motik et al., 2022) as a proof of concept. All these previous works contribute to the theoretical background of FaCIL, mainly related to the KF and their rules. In this work, we integrate the theoretical background to develop a new framework for hybrid modelling tools. The tool crowd 2.0 is extended from Braun et al., (2021) to instantiate the proposed framework, demonstrating its applicability and adaptability.

Distinct advantages of this framework with are as follows:

1. There is one system for related tasks: visual and text-based modelling in multiple modelling languages, automated transformations and update propagation between them, and verification of the model on coherence and consistency;
2. Any visual and text-based conceptual model interaction with the logic is maintained only in one place, instead of separately for each CDML and Controlled Natural Language (CNL);
3. A CNL can be specified on the KF metamodel elements so that it may be applied at once throughout the models and eliminate duplicate work of re-specifications;
4. Additional model management, such as abstraction and modularisation, can be specified either on the KF metamodel or its related logic-based reconstruction, all in one place, and propagate to other models accordingly, rather than reworking the algorithms for each language separately over and over again;
5. The modular design of the framework allows for extensions to each component, be it more variants of visual languages, controlled languages in other natural languages, or more logic-based reconstructions in other logics for more or less scalability.

The remainder of this paper is structured as follows. We first provide a more extensive motivation and background with the plethora of issues in Section 2. The orchestration

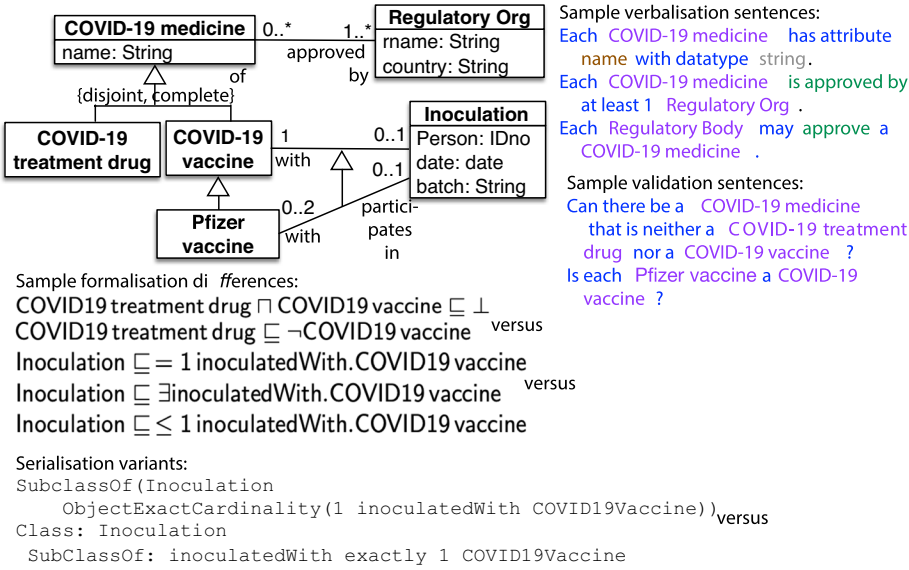

Fig. 1 A selection of a sample conceptual data model about COVID-19 medicines, suboptimal for illustrative purpose (resolved with automated reasoning further below), in UML Class Diagram notation (top-left); two examples of partial logic-based reconstructions of the model (top-right); and two serialisation variants, as statements and as questions to validate

framework is introduced in Section 3 and the procedural side of it in Section 4. Its evaluation by means of an implementation in crowd 2.0, comparison with a reference framework, and a use case are described in Section 5. We compare it to related work and discuss it in Section 6 and conclude in Section 7.

2 Motivational scenario

Let us commence with a scenario about data management of COVID-19 medicines; a section of a UML Class Diagram is shown in Fig. 1. As a model, and for any CDML it may be represented in, it has its own (informal) semantics irrespective of whether there is a logic-based reconstruction. A question that does arise is how to formalise it, which could be some Description Logic (DL) Berardi et al., (2005) or a serialisation in OWL so that it can be sent to an automated reasoner such as Racer (Haarslev et al., 2012). Formalisation questions include heterogeneity in, among others, association/member ends vs. object properties (relationships) and the different algorithms to deal with it Fillottrani and Keet (2015), explicit vs. implied class disjointness, and how much of the UML specification can be reconstructed in the logic. They differ in the details, making it hard to keep track of what is going on and which features are supported in which instance; two examples of partial formalisations are included in Fig. 1.

One can verbalise the model for those users who prefer to read text rather than assess diagrams. Such a verbalisation uses a CNL together with the visual model to produce text by taking the vocabulary from the model and plugging it into a template. A sample

template for a mandatory participation constraint may be `Each {C1} {AssocEnd}` at least one `{C2}` and illustrative CNL-generated verbalisation and validation sentences are shown in Fig. 1. Going back and forth between text and diagram requires management especially when one representation mode covers more or less features than the other and it can be further compounded by multilingualism. Since CNLs are not standardised, there may be a substantial proliferation of them to keep track of.

Then, there is heterogeneity in CDMLs: if the prospective COVID-19 medicine data management system is intended for a RDBMS, then ER may be preferred, and if it originates from business rules, probably ORM2, or they all may be needed as part of a complex system with a database at the back-end and an object-oriented app as front-end and thus needing both simultaneously. ER does not have datatypes, however, and both ER and ORM2 require identifiers that the UML diagram in Fig. 1 does not have; yet, such discrepancies have to be managed. Maintaining the model in all three language families would amount to six ad hoc mappings to and from the specific CDML flavours used in the tools and possibly also for the respective serialisations of the visual models since each tool has its own model serialisation. The ER or ORM2 models may be verbalised as well, using their respective terminology.

Overall, there thus can be very many variants of 'roughly the same' different models. These models and their transformations need to be managed. This would need to include models in 'versions' as diagrams, natural language text, formalisation(s), and serialisations, and that for at least ER, UML Class Diagrams (its static structural components), and ORM2.

3 FaCIL: A framework for hybrid modelling

The orchestration that we propose in Section 3.2 builds upon several key components that have been introduced in prior work. We briefly outline some pertinent theoretical foundations and preliminaries of them first in Section 3.1.

3.1 Context

First, to be able to change from one CDML to another, one needs to know what features are available in the languages. To this end, a unifying metamodel—called the KF metamodel—was designed (Keet and Fillottrani, 2015) and formalised for precision and verification to ensure that there were no inconsistencies or unintended implications. The metamodel unified the ER, ORM2, and UML Class Diagram language (v2.4.1) and harmonised terminology of the features, which facilitated identification of the overlap and uniqueness across the selected CDMLs. While they all have Relationship (association) and Subsumption, Role (association end/relationship component), and Object type (class/entity type), they differ on other constructs (e.g., weak entity type, qualified association), and of the combined 49 different types of constraints, the intersection consists of only cardinality (including Mandatory and Single identification), Disjointness and Completeness over the subsumption relation, and Subset constraints.

These insights formed the basis of a metamodel-driven approach to interoperability of conceptual data models (Fillottrani and Keet, 2014). Instead of requiring a mesh network of mappings, each model maps into the metamodel or out of the metamodel, therewith reducing the number of mappings. These mappings consist of 1:1 alignments for the

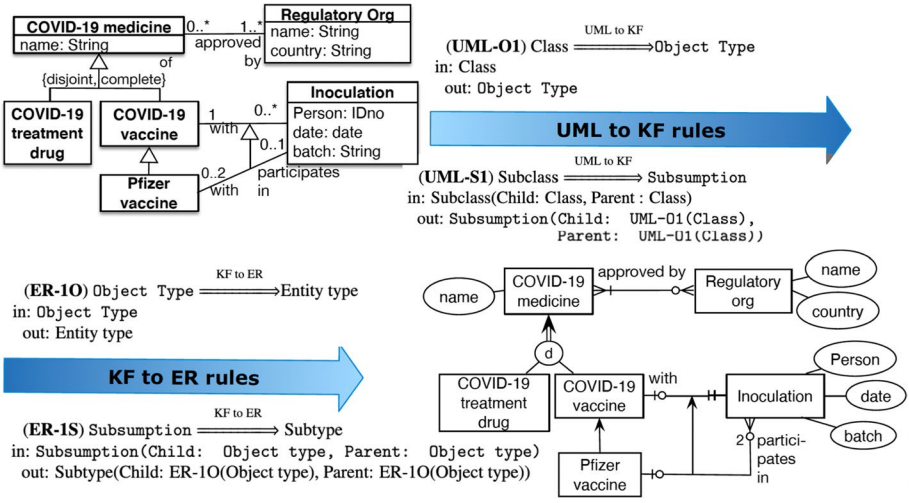

Fig. 2 COVID-19 medicines example from Fig. 1 where interoperability rules are applied to generate the ER version of the model (only a subset of the rules used are shown): first from the model to the metamodel with the relevant "UML to KF rules", where terminology is harmonised with the other supported modelling languages and any transformations are computed, and then the "KF to ER" rules are used to generate the semantics-preserving corresponding ER diagram

feature overlap, and transformations or approximation rules for the rest. More complex transformations are built up from the component rules. For instance, ORM's Value type maps into the metamodel, then within the metamodel there is a rule to transform that into an attribute, and from there it maps out into UML's attribute.

Figure 2 depicts the application of some of the interoperability rules to convert from UML to ER models. For the sake of space, we show only four rules related to both `Object type` and `Subsumption` for the COVID-19 example in Fig. 1. **UML-O1** and **UML-S1** convert the UML Classes and Generalisations to KF Object types and Subsumptions, respectively. **ER-1O** and **ER-1S** take the KF primitives generated in the previous step and generate the corresponding ER primitives: Entity types and Subtypes. Further conversion must be applied to UML Attributes and Associations Constraints (cardinalities/disjointness) (Fillottrani and Keet 2014).

Relevant for realising any implementation is the prioritisation of features based on what is used in publicly available models, for which the assessment presented in Keet and Fillottrani (2015) is used, and, following from that, the specification of corresponding logic profiles, which was first introduced in Fillottrani and Keet (2015).

The static elements summarised in this section do not yet address how these components are supposed to interact to achieve that seamless integration in tools, which is what the framework addresses.

3.2 Framework

The overall orchestration to design a modular hybrid modelling tool based on a systematic approach that is maintainable, is divided into two components: the objects and their relations in the framework—*what* is involved—are depicted in Fig. 3 and will be described in

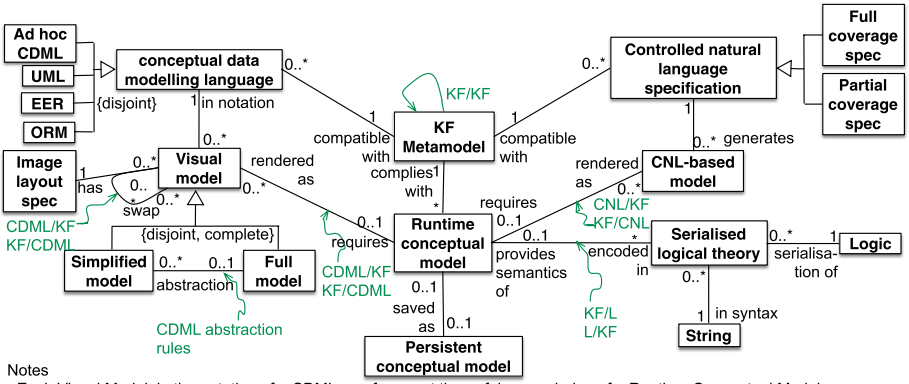

Fig. 3 Framework and rules for multi-modal and hybrid modelling, with the KF metamodel centrally positioned, the various visual languages and orchestration thereof on the left, their text-based counterparts on the top-right, and the link to the logic-based reconstructions in the bottom-right of the diagram. The squiggle arrows point to where which subset of transformation rules are applied

this section; and the dynamic 'workflow' processes—*how* it works—will be described in the next section.

3.2.1 Overview of the main components

The central point for achieving the interoperability is the aforementioned KF Metamodel, which describes all native features and their constraints present in all compatible CDMLs. CDMLs have their own semantics independent of whichever logic is used to formalise them. As such, it means that these elements are self-standing entities and 'first class citizens', not merely syntactic sugar for logical theories in which they are translated.

A conceptual data model (CDM) as visual model (diagram) can be encoded in a Serialised logical theory; it is that and no more and so does not become an ontology even if the logic is that of an ontology language such as OWL 2[1]. The conceptually distinct aspects and artefacts are kept separate in this orchestration, as can be observed from the encoded in. A Controlled natural language specification is a subset of (pseudo-)natural languages that are obtained by restricting the grammar and vocabulary in order to make it easier to be processed and it verbalises a CDM in a chosen natural language.

An instance of the metamodel is a concrete collection of these elements that represents a given conceptual model. Within some hybrid tool, it constitutes a Runtime conceptual model, which is used as pivot to switch between different representations:

- a Visual model graphically represented in a given CDML;
- a CNL-based model, which is its verbalisation following a CNL specification;
- a Serialised logical theory to be interchanged with other systems preserving its semantics based on some formal logic; or

[1] Conversely, an ontology rendered visually using a CDML remains an ontology.

– a Persistent conceptual model that includes tool-specific metadata when saving it into persistent storage.

Both graphical and textual models are intended for human users of the tool. A typical tool interaction involves generating persistent conceptual models that can be sent to a reasoner that later solves queries over the logic-based semantics of this model. Other interactions can be importing or exporting the model to other tools. The runtime conceptual model can be in any format the tool developers prefer, e.g., loaded in some data structures in Java, JavaScript, or even Python, depending on ones particular software design, which is used in-memory. Whichever its actual data structure in a particular implementation, it is constrained by the KF metamodel elements with its constraints.

3.2.2 Achieving interoperability

In FaCIL, interoperability rules, such as those shown in Fig. 2, govern the transformations between the different models a tool can handle, which are textually shown in the middle of each relationship in green/grey in Fig. 3. It is key to specify explicitly which rules it supports in order to understand the impact of performing the transformation on the semantics. It is also important to modularise the rules to provide the opportunity for any hybrid tool to present the user several alternative transformations. In the framework, all of the rules are defined on the KF Metamodel while they become operative on a Runtime conceptual model. Any modification to these rules lead to a new behaviour for such a tool.

Concerning the interoperability with a myriad of logics for reconstructions of CDMLs, there is one key location where this happens: between the runtime conceptual model and the serialised logical theory, where the transformation algorithms (`KF/L` from logic to KF and `KF/L` from KF to logic) are specified once for each computational logic.

Similar rules are necessary to visualise a runtime conceptual model in a CDML, and also to recreate it after editing operations performed by a user, which are handled by the `KF/CDML` and `CDML/KF rules`. Given that the CDMLs differ in expressive power, a roundtrip from two different CDMLs may also involve additional transformations, `KF/KF`, to preserve the semantics of the runtime conceptual model before visualising it into another CDML, such as aforementioned conversion between attribute and value type.

A new CDML can be supported by the framework by relating its visual elements to the KF metamodel, possibly adding new ones, and then specifying these rules for the new CDML. By adhering to the orchestration, a tool can be updated independently while ensuring homogeneous interpretation of the language. This is not the case if one were to repurpose, e.g., the Protégé tool (Musen, 2015) for conceptual modelling, since different plugins can assign different semantics to the same model.

Additionally, a set of rules `KF/CNL` is necessary to generate CNL-based models to be used by human users for validation activities. Again, several sets of these rules can be supported, therewith allowing the user to choose one that is adapted for their needs. Generally, only verbalisation is used, but FaCIL permits rule declarations to enable CNL-based editing of a conceptual model such that it will maintain compliance with the KF metamodel.

Finally, a set of rules is necessary to cater for multiple graphical renderings of the same visual model so as to present a simplified version to the human user, which is useful especially for large models. One can simplify a diagram by removing one or more types of visual elements (a syntactic abstraction), such as hiding attributes or names of relationships,

or summarise a very large diagram to the key salient subject domain content (a semantic abstraction), whose rules are specified with the `CDML abstraction rules`.

3.2.3 Additional functionality

While a fully multi-modal and hybrid tool may be enough for users to complete their tasks, one may wish to add import/export capabilities from/to other tools that are not included in the framework; e.g., a model transformed into a Serialised logical theory in the OWL syntax may be imported into, e.g., Protégé, and vv. This interaction is defined by the respective set of rules `KF/L` and `L/KF`. Interoperability in terms of visual models or CNL specifications from other tools, such as ERwin[2] or draw.io[3], require alignment to particular data structures of those tools for each of them separately, since there is no standard for serialising either visual models or CNL-based models.

4 Possible workflows

In this section, we describe a workflow that avails of FaCIL presented in Section 3. It is split into two processes as illustrated in Figs. 4 and 5, where all the blue and green (darker shades of grey) coloured elements are the key component processes and rules for model interoperability, yellow (light grey) indicate the various models, and white-filled elements indicate the remaining and usual activities in the one-off scenario only. Its purpose is two-fold: i) it highlights that interoperability components are needed, and which ones and where, and ii) it provides a walk-through scenario for usage and to elucidate functionality required for a hybrid tool.

The first instantiation in Fig. 4 is a workflow that operates entirely in the conceptual modelling layer and includes generation of CNL-based models, for collaborative design of a conceptual model, involving modellers from diverse skills that require interoperability between visual an textual models. In the top part of the figure, a tool that implements FaCIL is used by a user to draw a diagram in their preferred CDML from scratch, or they import an externally generated model. This model A is then transformed into a runtime CDM (I') using the `CDML/KF` rules. Then, a CNL textual model is generated for validation purposes using `KF/CNL` rules, labelled N. It generates sentences for each element and constraint, such as the "Each COVID-19 medicine is approved by at least 1 Regulatory Org" from the diagram in Fig. 1 for the 1..* multiplicity. The same visual model A also still may be converted to another CDML using `KF/CDML` rules, obtaining a visual model B (an example will be shown in the use case in Section 5.3 below). The tool enables one user to visually update B into B'. This new visual model may follow two flows: either it may be converted again to a CNL-based model (N') or it is transformed back to the original CDML resulting in model A' due to the updates. These steps require applying the respective `CDML/KF` and `KF/CDML` set of rules. The CNL-based model can be verified by the modeller (or an end user) and updated if needed by using `CNL/KF` rules, producing N'', where such changes then trigger modifications in the corresponding visual models in different `KF/CDML` sets of rules, producing A'' and B''. If B' or B'' is an ORM2 diagram

[2] https://erwin.com/products/erwin-data-modeler/
[3] https://app.diagrams.net/

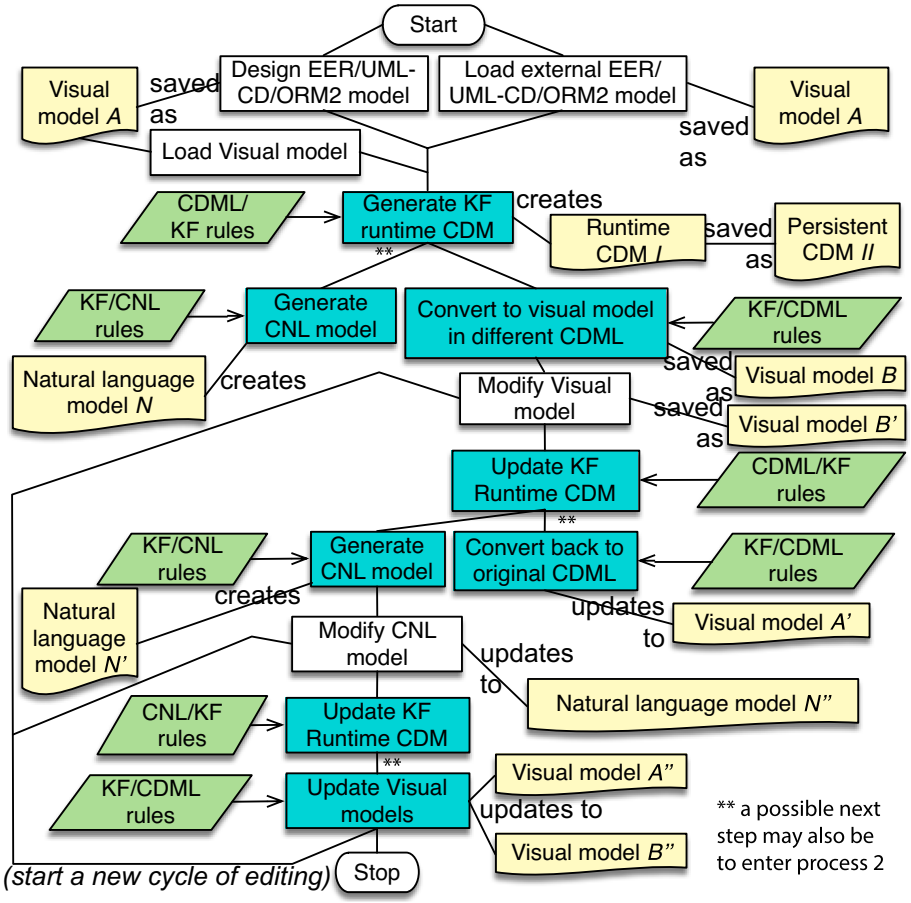

Fig. 4 A possible workflow with interaction between visual conceptual data models and (pseudo-)natural language versions, and their updates that propagate to the others. It commences with a visual model in one's preferred language, from which a KF runtime model is generated; from there, it can go in multiple directions: verbalising it, converting it, or modifying it. After a modification, the KF runtime model is also updated and the changes propagated upon visual model conversion, for as long as the modeller wishes to continue. The coloured (grey) boxes are thanks to our FaCIL and the white ones are from traditional one-off modelling

then for, say, the UML diagram of Fig. 1, the Convert... step includes additional conversions at the level of the KF metamodel, which are done by the `KF/KF` rules. As example, the UML attribute country of the Regulatory Org class is mapped to the KF attribute country(Regulatory Org, `string`), where `string` is a datatype and Regulatory Org is an entity type. Then, a new internal KF primitive, named `MappedTo`, must be asserted into the runtime KF model (using `KF/KF` rules), therewith transforming the attribute into a relationship between `Regulatory Org` and `country`. Finally, such a relationship can be mapped to a ORM2 fact type using a `KF/CDML` rule.

The second part, in Fig. 5, extends the conceptual modelling layer scenario to inclusion of automated reasoning. Once a visual model A is available and a runtime CDM created (I'), as in Fig. 4, one can, as usual, visually update it, save it for later use, or share it

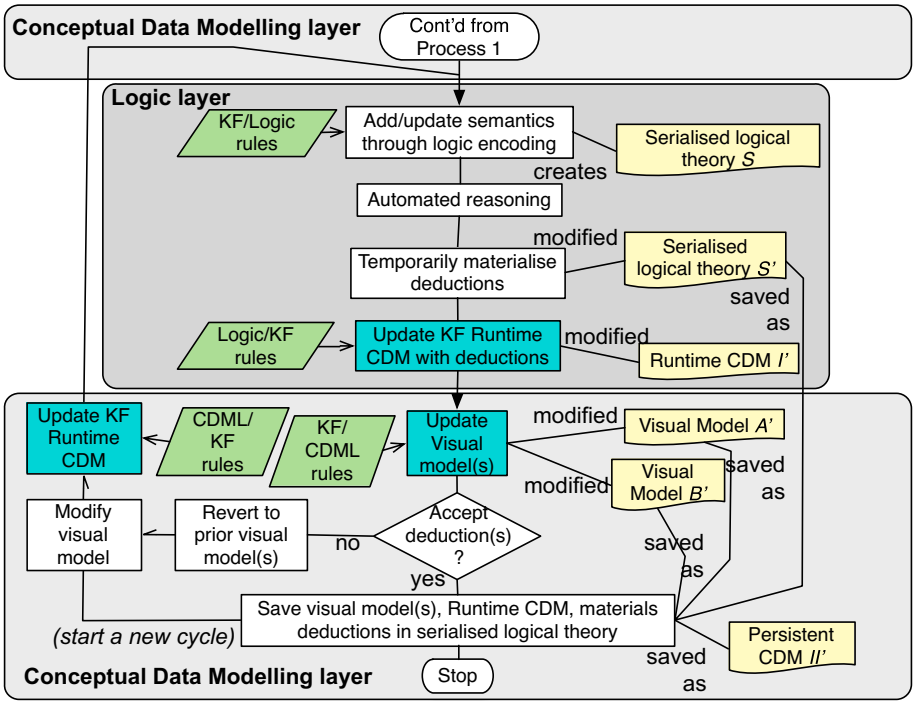

Fig. 5 A possible workflow where conceptual models are also formalised and interact with an automated reasoner. "Process 1" in the top section refers to processes such as in Fig. 4, the "Logic layer" in the middle is responsible for the automated reasoning processes, and the "Conceptual data Modelling Layer" at the bottom deals with the updates to the model and any further modelling tasks following assessing the deductions, such as resolving the problem of an unsatisfiable class

to other modellers with different CDML preferences by using specific `KF/CDML` rules. The emphasis here, however, is reasoning. Since it is well-known that manually 'reasoning' over a visual model, especially when it is large, is difficult due to possible complex constraint interactions and their consequences, we proceed to the logic encoding by applying the `KF/L` set of rules to obtain a serialised logic theory S that is suited for automated reasoning. This is depicted in the middle of Fig. 5. For instance, consider Fig. 1 again: the 1..1 multiplicity on the side of COVID-19 Vaccine may be serialised in OWL with those `KF/L` set of rules as `SubclassOf(Inoculation ObjectExactCardinality( 1 inoculatedWith COVID19Vaccine))`. When serialised into a logic theory, together with a suitable reasoner for it, the prospective hybrid tool then uses queries to the reasoner to make deductions. Depending on the configuration, the deductions may be temporarily materialised (into S') so as to interact with the modeller on whether they really intended to also have those implicit constraints. Assuming they do, this will generate an update to the runtime CDM I' model following the `L/KF` rules. These rules resolve some intricate details and hides them from the user who is typically not a logician. Yet they are implicitly present in any tool with automatic reasoning assistance (although their descriptions are generally brushed over). For instance, a common mismatch between a logical theory and a conceptual model is how they handle class disjointness: it can be declared on the classes in the logic, but conceptual data models require the same constraint

to be asserted over the subsumptions from a common direct parent class. This is a conceptual equivalence, but not logically: a roundtrip results into two different logical theories (a new class has to be added). For three CDMLs, one could end up with four theories then, if it is a direct transformation between the (serialised) logic and each of the visual models. Rather, in the proposed orchestration, we put that in one transformation only, from Serialised logical theory to the Runtime CDM, which then takes care of the rest in a loss-less conversion (w.r.t. disjointness) to visual models in the CDML notation.

After the logic layer, the model-as-logical-theory is passed back to the conceptual modelling layer where a modeller can view the deductions in the visual model, by having been temporarily propagated into I'. The deductions (if any) are analysed and checked on the updated visual model, and accepted or rejected by modellers. This may end the session, where the final model is saved as model I'', or the modeller may continue modelling and enter the workflow again.

5 Implementation and evaluation

We will demonstrate the working of FaCIL by: first, implementing it in a specific hybrid tool, called crowd 2.0, which is a visual web tool for conceptual modelling and ontology engineering tasks; second, evaluating FaCIL against a reference framework for conceptual data modelling; and, third, using the motivational scenario of Section 2 as the basis for a use case. They are presented in sequence in this section.

5.1 A hybrid tool that implements the FaCIL framework

This section summarises the tool we have developed and how it instantiates the framework proposed in the preceding sections.

5.1.1 Overview of crowd 2.0

The intention of crowd, and its successor crowd 2.0, is to involve domain experts and users in modelling tasks by adopting standard CDMLs, providing visual support for them, and logic-based reconstructions (in addition to OWL 2 serialisations) for knowledge engineers. The tool is fully integrated with a powerful logic-based reasoning server acting as a background inference engine. That reasoning is relative to the diagram's graphical syntax so that users will see the original model graphically completed with all the deductions that are expressed in the graphical language itself. crowd focuses on graphical modelling of CDMs (and ontologies) at the type level, and does not consider individuals. It is compliant with W3C standards by allowing the definition of global naming schemes as well as the export of specifications to OWL 2 to interoperate with other tools.

crowd has been released in two versions. The first iteration, as crowd 1.0 Braun et al. (2020), included visual support for only UML diagrams and interoperability via the OWLlink communication protocol (Liebig et al., 2011), with the Racer (Haarslev et al., 2012) or Konclude reasoner (Steigmiller et al., 2014) to validate them. In addition, models were exported as valid OWL 2 serialisations together with their namespaces, which were defined from the user interface. The substantially extended version crowd 2.0 now supports *three* CDMLs supported by the KF metamodel, including visual editors for not only UML, but also ORM2 and ER, and the sets of rules required for conceptual

model interoperability. It does this in the way presented in Fig. 3, which is elaborated on below. The logic-based reconstruction has been refactored (cf. crowd 1.0) in terms of KF instances. crowd 2.0 has been released[4], together with its documentation. The source code is also publicly available[5]. We will demonstrate crowd 2.0 in the use case below.

5.1.2 Framework instantiation in crowd 2.0

To computationally test FaCIL, we implemented the "core fragment" of the three CDML families (Fillottrani and Keet, 2021) and a logic formalisation thereof. In crowd 2.0, a Java KF-API is in charge of receiving models in a given representation and converting them to/from the metamodel by executing the rules `CDML/KF` and `KF/CDML`.

The task of keeping the workflow involving Runtime conceptual model consistent across all the updates of the models, including deductions obtained from the reasoner, has been implemented in a PHP module along with the set of rules `KF/L` and `L/KF`. A KF instance is created as input, which is formalised using the `KF/L` rules, to be serialised in OWL 2, and then sent to one of the two supported reasoners, as the user so prefers. Functionality to import arbitrary OWL files is under development. The complete catalogue of the implemented interoperability rules is available as an online resource[6].

The models persist in a database as JSON objects. Once a model is saved, it may be loaded in any of the visual editors. The client of crowd 2.0 is a web application implemented with AngularJS and JavaScript.

5.1.3 Scalability considerations

There are two key considerations regarding scalability: 1) with respect to the models and 2) in the light of computational costs of the algorithms. We shall address each in turn.

The approach with FaCIL is exceedingly scalable to a much larger set models: they pivot around the KF metamodel in 1:n fashion rather than having to create and maintain $(n-1)!$ mappings in a mesh structure. For instance, with $n = 5$ models in 4 languages, there are 5 mappings to KF (automatically generated) and then only the requested number of, also automatically generated, models in another family of languages in FaCIL (at most 5*4=20), whereas for the model-level mappings it would be $5 * (4-1)! = 5 * 6 = 30$ mappings to maintain already. Concerning larger models than the examples in Fig. 2 and the use case below: this is possible, for as much as any diagramming tool can handle large sizes (Dudáš et al., 2018). The hiding feature in `crowd 2.0` permits condensing the visualisation and, hence, the tool permits more than a plain modelling tool. That is, there are no model size constraints imposed by the back-end, but graphical representations have practical limitations of browsing and a modeller's cognitive load.

The computational costs of automated reasoning over the models are as for any of the chosen logic. For OWL in particular, that is 2NEXPTIME-complete in subsumption reasoning. Informally, the performance is deemed to user satisfaction for 'small' ontologies up to a few thousand entities and axioms, which is well within the typical model size of about 50 entities and constraints (Keet and Fillottrani, 2015).

[4] crowd 2.0 is available at https://crowd-app.fi.uncoma.edu.ar/

[5] At https://github.com/iamcrowd

[6] At https://github.com/iamcrowd/crowd-box

Concerning the analysis of the algorithms that transform between the different modalities of the models, the source of the complexity of them is given by the time spent looping over each primitive of the models and their related ones. As typical examples, for each object type it loops over each attribute associated to it, and for each disjointness or completeness in subsumptions constraint it loops over each object type participating in such a constraint. Then, if we consider n primitives with m related other elements, the total time taken by the algorithms is in $\mathcal{O}(n * m)$. The remaining operations are simple `if-then-else` structures and constant `string` comparisons.

To complement this theoretical evaluation of scalability, we now conduct a runtime performance test aiming at assessing the tool with a set of growing size models. In the absense of real CDMs in crowd 2.0 and to stress the tool with larger models than the average ones (as already said above, about 50 entities and constraints), we create a dataset with the following in-use and real ontologies published in the repository LOV[7]: *time* (Hobbs and Pan, 2004), which is a W3C recommendation; *arco*, *a-loc* (Carriero et al., 2019) and *cis* (Lodi et al., 2017), supported by the Italian Ministry of Cultural Heritage and Activities; *sto* (Bader et al., 2020), for standards in use on the Industry 4.0; *foaf* (Brickley and Miller, 2007), the well-known vocabulary for linking people on the Web; *s4agri* (Poveda-Villalon et al., 2019), for smart agriculture and food chain domain; and *qudt* (QUDT.org, 2011), for quantities, units, dimensions and types. From each ontology, we create a KF instance by taking a subset of logical axioms, composed by all the possible subclasses, complements, unions, and existential and universal quantifications between atomic concepts asserted in the ontology. The average size of the resulting KF instances is of 277 entities: 48 object types, 64 subsumptions (simple and composed with disjointness and completeness constraints), and 14 binary relationships with the respective roles and cardinalities.

We perform the experiments on an Intel Core i5 (3.2 GHz), and 8 GB of memory running Debian GNU/Linux 10 (buster) with 64 bit kernel 4.19.98-1. To browse crowd 2.0, we use Chrome 95.0.4638.69 and also the Chrome DevTools Performance panel to runtimes. The tests were executed in Incognito Mode to ensure that Chrome runs in a clean state. We record the activity into these available categories: *loading*, the time making network requests and parsing HTML; *scripting*, the time parsing, compiling, and running JavaScript code (includes Garbage Collection (GC)); *rendering*, the time doing style and layout calculations; and *painting*, the time painting, compositing, resizing and decoding images. For each test, we record the workflow: (1) import a CDM given as a KF instance in the UML editor of the tool (KF to UML conversion), (2) convert it to ORM 2 (UML to KF and KF to ORM 2 conversions), (3) change the layout, and (4) verbalise the model.

Table 1 summarises the results. Overall, the runtime is affected by the scripting and rendering activities, meaning the execution of JavaScript code on the client side to import, display and arrange the models. We can see that the total runtime increases when the size of each CDM is larger, which is expected if we consider that dealing with larger models is more challenging for the tool. Most of the time is spent by scripting and also by rendering for larger CDMs, while loading (network requests) is relative low, meaning that the conversions between these models on the server side are done with an acceptable performance.

In particular, the tool spends most of the time scripting the two first models, generated from *time* and *sto*, where the majority of the primitives are simple subsumptions (with

[7] https://lov.linkeddata.es/dataset/lov/

Table 1 Runtime performance of crowd 2.0 for increasing CDMs

Ontology	Ontology Metrics	CDM Metrics	Runtime (seconds)			
	#TBox axioms modelled[i]	CDM Size[j]	Loading	Scripting	Rendering	Painting
time[a]	24	46	0.21	15.39	3.1	0.14
sto[b]	40	85	0.37	21.2	6.52	0.14
arco[c]	85	124	0.34	54.23	9.1	0.24
foaf[d]	76	131	0.33	63.93	10.1	0.16
cis[e]	131	287	0.52	54.24	32.05	0.14
xqudt[f]	100	307	0.74	63.66	24.4	0.29
s4agri[g]	148	582	0.66	102.03	57.07	0.26
a-loc[h]	157	655	0.71	114.41	78.25	0.2

[a] http://www.w3.org/2006/time

[b] https://w3id.org/i40/sto#

[c] https://w3id.org/arco/ontology/core

[d] http://xmlns.com/foaf/0.1/

[e] http://dati.beniculturali.it/cultural-ON/cultural-ON.owl

[f] http://qudt.org/schema/qudt

[g] https://saref.etsi.org/saref4agri/

[h] https://w3id.org/arco/ontology/location

[i] TBox axioms modelled = subclasses + complements + unions + existential + universal quantifications

[j] CDM size = Object types + Attributes + Subsumption + Binary Relationships + Roles + Constraints cardinalities/disjointness/completeness)

very few constraints). crowd 2.0 presents similar performances for both the *arco* and *foaf* CDMs, which are close in size (124 vs 131). While the *arco* model has 4 binary relationships (thus, 8 roles and 8 cardinalities), the *foaf* includes a larger set of constraints on composed subsumptions (about 67 subsumption in total and about 16 disjointness), indicating that it is the number of elements in the model, rather than a type of element, that determines the performance.

For the last four models, the tool notably increases the rendering time if we compare *foaf* (10.1 sec) vs. *cis* (32.05 sec). The CDMs for *cis*, *s4agri* and *a-loc* include a greater number of binary relationships (30, 28 and 49, and therefore 60, 56 and 98 roles and cardinalities, respectively), explaining the rendering time w.r.t. the first models. This last observation is consistent if we consider the rendering time of the model *qudt*, which is composed by only subsumptions and is the shortest rendering time (24.4 sec) in these last four models analised, with increasing scripting time.

Concluding this performance analysis of the tool crowd 2.0, it has been validated the feasibility of the tool to manipulate average-size CDMs in a reasonable amount of time.

5.2 Alignment with a reference framework

We have devised workflows for hybrid modelling, which involve a set of activities based on them that we will use later for classifying related approaches. Even so, to theoretically test FaCIL to also demonstrate it is not some 'arbitrary' framework, we consider a "reference

framework" (Delcambre et al., 2018) to align and justify our workflows and activities for conceptual modelling. In line with our contribution, that reference framework of Delcambre et al., 2018 discusses the purposes of a conceptual model—being to provide levels of abstractions (with explicit semantics) to promote communication and understanding among people with different views of domains—and it aims at characterising the field of conceptual modelling by offering a holistic view of definitions, dimensions of analysis and activities involved. Based on that framework, we focus on the activities associated with conceptual models and conceptual modelling languages, and describe briefly how they are covered by FaCIL.

- **Activity 1**: *defining (modelling) a CDM using a CDML*. In FaCIL, a model A can be defined using a particular CDML (UML, ER, ORM2), which is also an instance of a KF model (going through CDML/KF rules). In the same direction, such a model A could be given in terms of structured text and then rendered into a CDML as well.
- **Activity 2**: *implementing a CDM*. This is the activity of transforming a CDM A into an *implemented* artefact or product. As an example, the model A can be defined in FaCIL as a UML Class Diagram and reconstructed into a logic formalism, to be used as a runtime OWL 2 model in an Ontology-Based Data Access (OBDA) system (Calvanese et al., 2016). Related features such as automatically compiling a model into a runtime system could easily be also integrated.
- **Activity 3**: *CDM reverse-engineered from an implemented system*. This is the opposite of the previous activity. A typical example is the extraction of a CDM from a database. For instance, the NORMA tool offers reverse engineering of a database into an ORM diagram, which then could be used in FaCIL. Similarly, an OWL 2 file can be extracted from a database before implementing an OBDA system (Lubyte and Tessaris, 2009). This kind of activity could be supported in FaCIL by importing OWL 2 files through the L/KF rules.
- **Activity 4**: *defining or extending a CDML*. In addition to ground FaCIL, the KF metamodel identifies the commonalities and differences in the modelling principles based on the study of the standard CDMLs (ER, UML, ORM2). In this sense, the KF metamodel provides the baselines to define or extend CDMLs. A new CDML can be supported by FaCIL by relating its visual elements and interoperability rules to the KF metamodel.
- **Activity 5**: *CDM translation*. This is the activity of translating on CDM A in a CDML L_1 to a CDM A in a CDML L_2. As an example, as provided by FaCIL, a CDM given in UML can be translated to a CDM in ER or ORM2, or a model can be verbalised into natural language. Moreover, possible losses during the translations are tracked by the KF metamodel, which keeps the whole semantics of the CDMs.
- **Activity 6**: *mapping from one CDM to other*, where both are expressed in the same CDML. Integration of information is the key motivation of the KF metamodel, which includes reconciling encoding differences. For instance, a typical example from Delcambre et al. (2018) is modelling something as a binary association or as an association class in UML, which easily can be recognised when they are present in two conceptual models in the same CDML. Thanks to the logic-based reconstruction, it could use the patterns and algorithm of Fillottrani and Keet (2017) to translate between the two, or those patterns can be specified in terms of the KF metamodel, so that the transformation happens there, and complete the mapping in three steps (i.e., UML→KF, KF→KF, KF→UML).
- **Activity 7**: *validating a CDM* aims at identifying inconsistencies or contradictions in a model. Such an activity depends on the formal definition of the CDML. In particular,

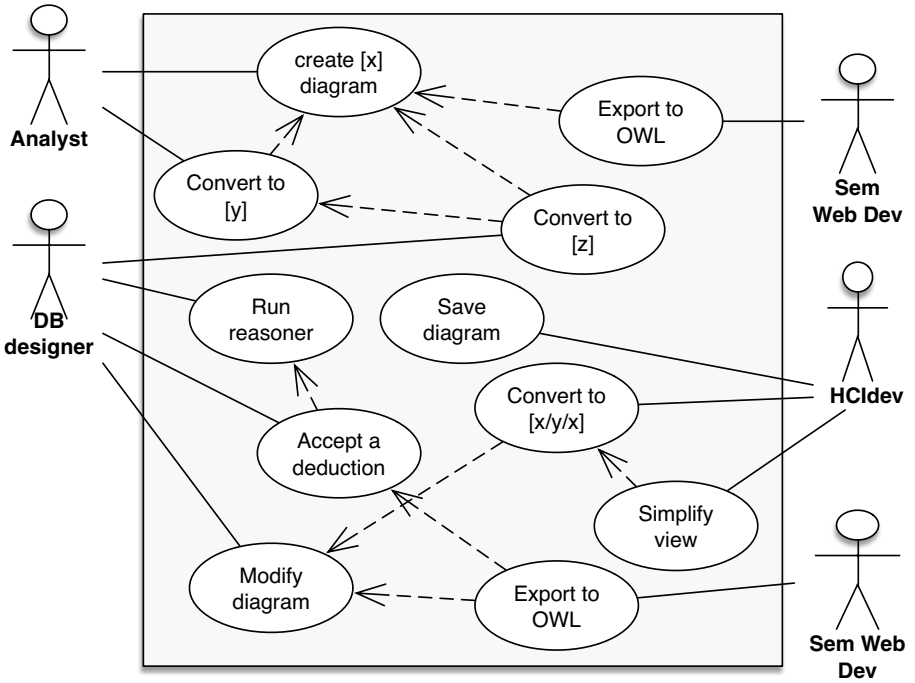

Fig. 6 Sample use case diagram related to the use case described in Section 5.3, involving participants with different roles, such as the database and the interface designer, who may prefer different modelling languages. The [x], [y] and [z] are distinct and either UML, ER, and ORM; the "[ ]" is UML in the use case, but each type of diagram can be simplified and so left blank here

in FaCIL this validation can be executed over a CDM in UML, ER or ORM (being this validation rendered on the same diagrams), or over a CDM in structured text, which is facilitated in particular by the constraints from the KF metamodel and the automated reasoner.

Concluding this assessment of the reference framework, it has been demonstrated that FaCIL meets such requirements.

5.3 Use case: revisiting the motivational scenario

With the theory in place and a hybrid tool that implements its core components, we now turn to the motivational scenario about the modelling of the COVID-19 medicines of Section 2. To do this, we will follow the use case diagram depicted in Fig. 6, which illustrates how users can work collaboratively by exploring a conceptual model from different perspectives that suit their respective competencies and core tasks. For the sake of space, the full description of this scenario is provided as an online resource[8].

[8] https://github.com/iamcrowd/crowd-box

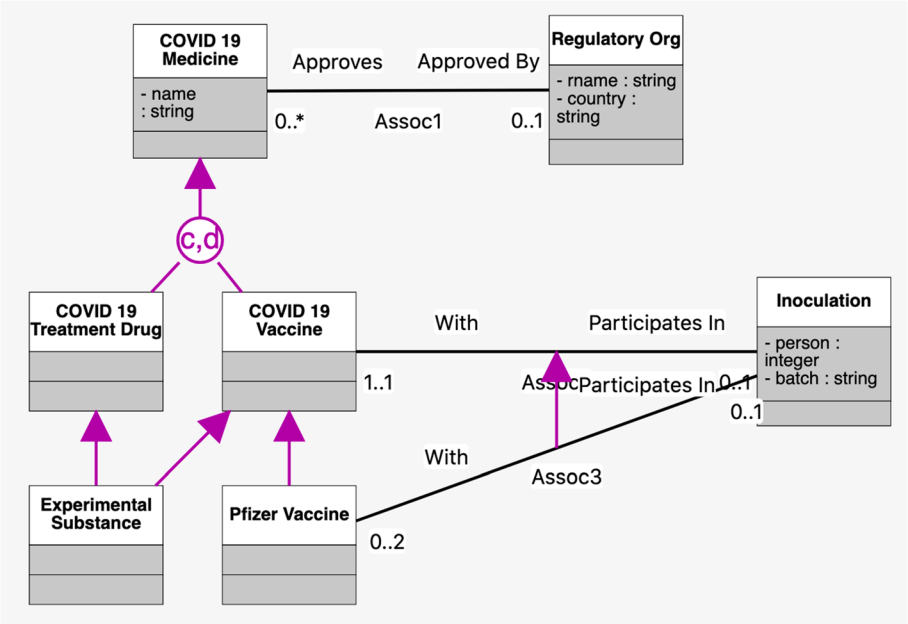

Fig. 7 The same UML diagram as in Fig. 1, but then rendered in crowd 2.0's interface

First, let us put the UML diagram into crowd 2.0 (see Fig. 7). To push the tool's capabilities a little more so as to obtain more interesting results, we add an additional class in the spirit of the CIDO ontology (He et al., 2020), called Experimental substance, which is also shown in Fig. 7. Its back-end runtime conceptual model as KF instantiation and the implementation of the rules were then called to generate automatically their corresponding ER and ORM2 versions, which are depicted in Fig. 8. As can be seen in the figures, there are labels like Assoc3 (abbreviated from the default generated Association-x, where x is an increment count in the interface) and Qf Wzz9 in Fig. 8: ER requires names for relationships, but UML does not, and this discrepancy is fixed with default naming. This is similar for ORM2, where each UML attribute is converted into a ORM2 value type that uses an extra mandatory 1:1 fact type (relationship) for it, which have been added. They all can be modified by the modeller.

Let's assume the modeller is a database analyst, and so we present them the ER diagram. They decide to run the reasoner over the ER diagram to double-check its quality. The deductions are displayed visually as shown in Fig. 9 and, upon clicking it, a brief note appears in the right-hand pane of the tool: Experimental substance is unsatisifiable because its parent classes are disjoint, and the cardinality on the Pfizer vaccine changed from 0..2 to 0..1 because of the cardinality on with in Assoc2 and the subsetting constraint on the relationship. The modeller can accept or reject the deductions.

The scenario continues with more conversions across UML, ER, and ORM2, removing that unsatisfiable class and transforming it back to UML that we started off with. This diagram can be 'decluttered' by toggling the attributes and the role and relationship names, so as to obtain a simplified visual notation whilst maintaining the details in the background. Finally, it demonstrates it is possible to generate an OWL file and JSON from the model. Also, upon the client's request for validating the model, the HCIdev employee generates

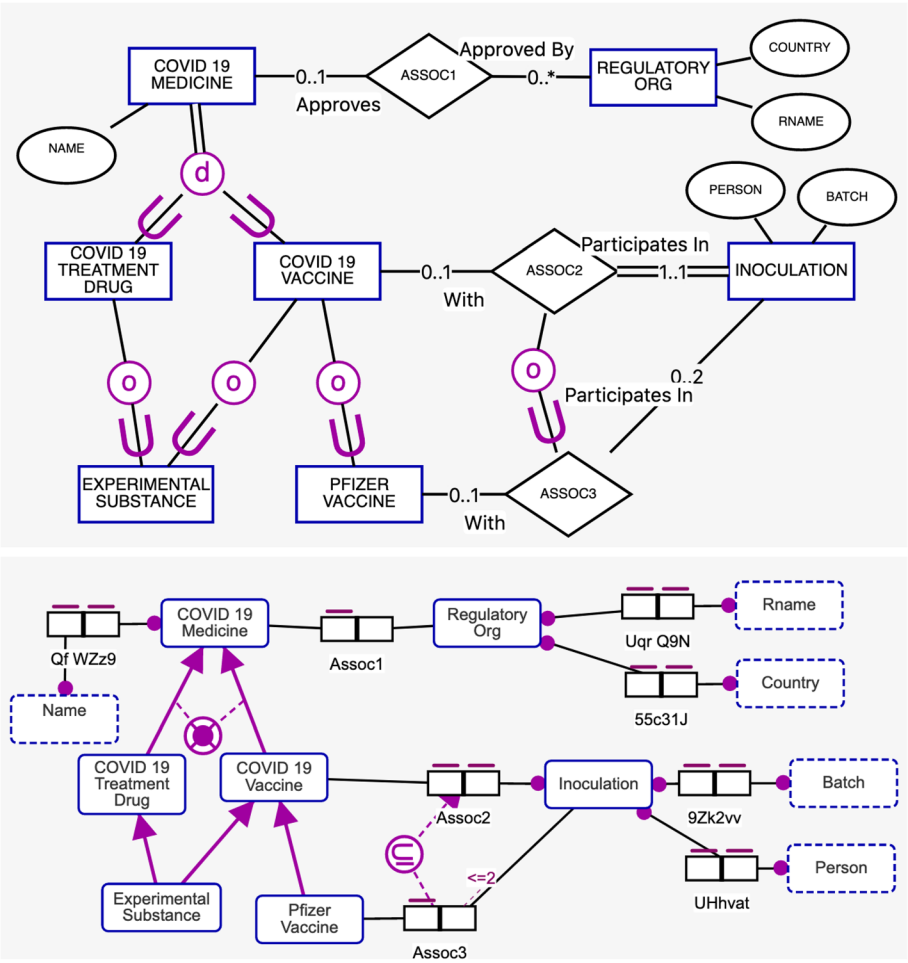

Fig. 8 Automatically generated ER and ORM diagram versions of the original UML class diagram model of Fig. 7

a text-based version with an English CNL. As mentioned, these last features are fully described in the online resource referenced above.

6 Related work and discussion

As noted in Section 1, there are several tools that cater for conceptual modelling, solely or as part of a drawing suite, which are to a greater or lesser extent hybrid tools for multi-model modelling. However, to compare crowd 2.0, we limit ourselves to those tools that aim to support conceptual modelling interoperability, and preferably also have a notion of multi-modality with CNLs and a formal foundation. Thus, we exclude tools from the comparison that offer independent multiple drawing canvases for different CDMLs and tools that offer only one CDML, such as AuRUS (Rull et al., 2015) and OLED-OntoUML

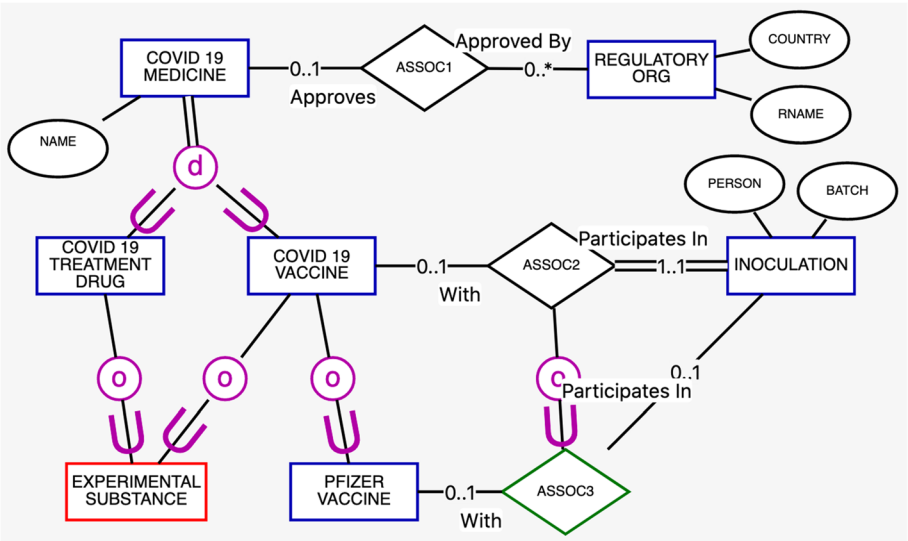

Fig. 9 Deductions over the model, obtained with the automated reasoner: Experimental substance is unsatisfiable and the cardinality on the Pfizer vaccine changed from 0..2 to 0..1

(Guerson et al., 2015). Both of them support only UML, through the off-the-shelf tool ArgoUML[9], and offer reasoning over CDMs by encoding diagrams in DL and Alloy, respectively. Also, FaCIL does not aim to be a multi-level approach (Atkinson et al., 2014) and therefore such tools were excluded from the comparison. Although conceptual models are 'instances' of the KF metamodel in our setting, FaCIL does not allow both types and instances to be mixed in the same model as defined for multi-level approaches. Actually, the KF is only a bridge for supporting CDML translations and multi-modality. ConceptBase (Jeusfeld, 2021), GeRoMe (Kensche et al., 2007) and all the multi-level modelling approaches based on Telos (Koubarakis et al., 2021) are tools that provide powerful generic meta-modelling features to formally characterise different levels of abstractions, including languages, models and instances. These approaches are generic, so visual representations and reasoning is not tailored to specific CDMLs. There are also generic visual tools like MetaEdit+ Kelly et al (2013) and the SyncMeta framework (Nicolaescu et al., 2016). These systems allow one to define collaboratively a visual modeling language and then to generate a model editor for it. They would in principle be capable to express graphically the well-known UML, ORM2 and ER diagrams in their visual view specifications (although some features like cardinalities cannot be represented), but very restricted semantics is attached to these graphical elements, and so the only property that can be verified is model well-formedness. That is, reasoning over models is impossible.

This demarcation of scope reduced the related work to: i) OpenPonk that offers both UML and BORM as well as FSAs and domain-specific languages (Uhnák and Pergl, 2016), ii) Pounamu (Zhu et al., 2004) and the work by Boyd & McBrian, (2005), which both cater for UML class diagrams, ER, and ORM and therewith are most within scope, iii) the MIDST tool Atzeni et al. (200) for translating a model from one language to another,

[9] https://github.com/argouml-tigris-org/argouml

considering ER, UML and physical schema languages, and iv) NORMA with the ORMiE extension Sportelli and Franconi (2016), which is a successor to ICOM Fillottrani et al. (2012) but then tailored to ORM and, like crowd, duals as ontology editor as well. The latter avails of their "Universal Conceptual Modelling Framework" Sportelli and Franconi (2016) that might support transformations in the future but is predicated in the notion of diagrams as 'syntactic sugar' for the logic. The features that they are assessed on are devised based on FaCIL that was presented in Section 3, which are summarised as follows:

- *Multiple CDMLs*: the tool supports more than one CDML to design and maintain conceptual models diagrammatically.
- *Verbalisation (model → text)*: conceptual models in a particular CDML can be verbalised in (pseudo-)natural language sentences by using a CNL.
- *From structured text to model*: conceptual models given as a textual specification with a CNL are written in or imported into the tool and converted into a visual model by using a CDML.
- *Logic-based reconstruction of the model*: the semantics of conceptual models given in a CDML is encoded into a logical specification.
- *Render logical theory diagrammatically*: the tool generates a visual model from a logical specification.
- *Reasoning over a model*: In addition to encoding models into a logical theory, the tool verifies them and then shows the inferences over the same diagram.
- *Reasoning over structured text*: the tool validates models given in structured text and then shows the inferences over that verbalisation.
- *Swap between models in different CDMLs*: the tool supporting multiple CDML provides capabilities to convert between each other.
- *Abstraction or hiding*: The tool offers features to simplify the models, hiding attributes or names, or defining modules and summarisation, among others.

The comparison is included in Table 2. As can be seen, crowd 2.0 compares favourably, although it also does not have all the features implemented yet either. Notably, the interoperability with CNLs is under development (Garrido, 2019) and the systematic semantic abstraction is within reach when integrating NOMSA's abstraction algorithms (Khan and Keet, 2021).

The multi-modality regarding CNL-based conceptual modelling receives little support, both in the tools considered here and in general, with the NORMA tool for ORM2 (and therewith ORMiE) as main exception. It is likely that this is because there is no standard for it other than in SBVR format, which thus leaves plenty of scope for design cf. the fixed syntax of the visual languages, and it requires tailoring to each natural language, whereas the diagrammatic representation is independent of the natural language of the modeller. The state of affairs is similarly patchy for abstractions across the tools (see Khan and Keet, 2021), which likewise looks easy to do superficially, but becomes complicated in the details, for there are many ways how to abstract and it has to be re-implemented for each CDML and modelling tool for as long as there is no common serialisation. Logic-based reconstructions also face a so-called 'embarrassment of the riches', which may be partially due to the plethora of logics, but also because of mismatches in language design principles and purposes (Fillottrani and Keet, 2020).

The framework proposed in this paper contributes to alleviating these issues, and in particular items 1-4 as listed in Section 1, notably: one system for related modelling tasks in different modes, one place to declare any CNL specifications that then may be applied

Table 2 Comparison of crowd 2.0 to similar tools, held against the framework feature requirements for multi-modal modelling

Feature	DSL/CDM	CDM			CDM/Ontologies	
	OpenPonk	Pounamu	B&McB	MIDST	NORMA + ORMiE	crowd
Tool currently available	yes (Linux, Windows)	no	no	no	yes (Windows)	yes (OS-indep.)
Multiple CDMLs	yes	yes	yes	yes	no[a]	yes
Verbalisation (model → text)	no	no	no	no	yes	yes[c]
From structured text to model	no	no	no	no	no	no
Logic-based reconstruction of the model	partially (FSA only)	no	yes (graph)	yes (Datalog)	yes (OWL)	yes (OWL)
Render logical theory diagrammatically	partially (FSA only)	yes	yes	no	no	yes[e]
Reasoning over model	no	no	no	no	yes[b]	yes
Reasoning over structured text	no	no	no	no	yes[d]	no
Swap between models in different CDMLs	no	yes	yes	no	no	yes
Abstraction or hiding	no	no	no	no	modules	attributes roles rels. names

[a] It is possible to generate an ER diagram from an ORM diagram, but not vv.

[b] Reasoning results are not shown over the diagram but they are shown in an independent windows.

[c] crowd 2.0 provides English CNL specifications for the common set of KF primitives.

[d] Derivation rules in CNL are used to express knowledge that is beyond normal ORM capabilities.

[e] It is limited to the expressiveness of diagrams.

throughout, and model management (e.g., abstraction or summarisation) then also can be managed from one central location and propagate rather than administering multiple separate specifications. Further, since it has this realised through a so-called 'separation of concerns' for the different sub-tasks, i.e., with each component of the framework in a separate module, it is extensible by design. For instance, one could add a new CDML to FaCIL by relating its visual primitives to the KF metamodel, and possibly even add new elements or constraints together with their interoperability rules. Similarly, one can add a new CNL for both KF and any particular CDML. FaCIL does not constrain the serialisation technique for a CNL specification, but since its instantiation in crowd 2.0 already has a reasoner, ToCT (Mahlaza et al., 2021) may be of use to define the CNL, so that the CNL specification itself can be sent to the reasoner for verification as well.

7 Conclusions

We have proposed the framework FaCIL for seamless multi-modal modelling in multiple conceptual data modelling languages, supporting all their language constructs, and described a workflow for it. A distinct feature of FaCIL compared to prior work is that it ties together adjacent tasks in an interoperable way, notably multiple modelling languages and automated reasoning over them for quick error detection and updates that propagate throughout the models. Thanks to its theoretical foundations, automated transitions between models in different modelling languages can be carried out coherently and consistently.

The framework was instantiated in a proof-of-concept hybrid tool, called crowd 2.0, which demonstrated the practical feasibility of the theoretical foundations and was shown to compare favourably to related work. FaCIL with crowd 2.0 was also evaluated against a reference framework for conceptual data modelling and shown to be meeting those requirements.

The separation of concerns into distinct modules for separate subtasks plays an essential role in fostering flexible frameworks and facilitates customisation and future extensions, such as adding new CDMLs and then specifying their interoperability rules or specifying new CNL dialects for both KF and particular CDMLs. Indeed, there are several directions for future work, including extending crowd 2.0 with the proof-of-concept Spanish CNL Garrido (2019), incorporating NOMSA's abstraction mechanisms (Khan and Keet, 2021), and adding choices in logic-based reconstructions for both the logic and the encoding decisions. Further, we plan to extend FaCIL as a starting reference framework to help others to evaluate tools, and understand what it has to offer, as well as contribute to the development of tools with user-centred perspectives.

Acknowledgements CMK was financially supported by the National Research Foundation (NRF) of South Africa (Grant Number 120852).

Data Availibility Statement For reproducibility, the models along with the evidence of the tests in the section 5.1.3 are available on the GitHub repository https://github.com/iamcrowd/crowd-box.

Code availability The implementation of crowd 2.0 is available on the GitHub project https://github.com/iamcrowd.

Declaration

Conflicts of interest The authors declare that they have no conflict of interest

References

Atkinson, C., Gerbig, R., & Kühne, T. (2014). Comparing multi-level modeling approaches. In: *17th Int. Conference on Model Driven Engineering Languages & Systems (MoDELS)*. CEUR-WS, vol. 1286, pp. 53–61.

Atzeni, P., Cappellari, P., Torlone, R., Bernstein, P. A., & Gianforme, G. (2008). Model-independent schema translation. *The VLDB Journal, 17*(6), 1347–1370. https://doi.org/10.1007/s00778-008-0105-2.

Atzeni, P., Gianforme, G., & Cappellari, P. (2012). Data model descriptions and translation signatures in a multi-model framework. *Annals of Mathematics and Artificial Intelligence, 63,* 1–29. https://doi.org/10.1007/s10472-012-9277-y.

Bader, S. R., Grangel-González, I., Nanjappa, P., Vidal, M., & Maleshkova, M. (2020). A Knowledge Graph for Industry 4.0. In: 17th International Conference, ESWC, Proceedings. LNCS. Springer, vol. 12123, pp. 465–480. https://doi.org/10.1007/978-3-030-49461-2_27

Berardi, D., Calvanese, D., & De Giacomo, G. (2005). Reasoning on UML class diagrams. *Artificial Intelligence, 168*(1–2), 70–118. https://doi.org/10.1016/j.artint.2005.05.003.

Boyd, M., & McBrien, P. (2005). Comparing and transforming between data models via an intermediate hypergraph data model. J. Data Semant. IV, 69–109. https://doi.org/10.1007/11603412_3

Braun, G. A., Gimenez, C., Cecchi, L. A., & Fillottrani, P. R. (2020). `crowd`: A visual tool for involving stakeholders into ontology engineering tasks. *KI - Künstliche Intelligenz, 34*(3), 365–371. https://doi.org/10.1007/s13218-020-00657-8

Braun, G. A., Marinelli, G., Gavagnin, E. R., Cecchi, L. A., & Fillottrani, P. R. (2021). Web interoperability for ontology development and support with crowd 2.0. In: 30th Int. Joint Conference on Artificial Intelligence, IJCAI, p. 4980–4983. https://doi.org/10.24963/ijcai.2021/707

Brickley, D., & Miller, L. (2007). The friend of a friend (FOAF) vocabulary specification. http://xmlns.com/foaf/spec/

Brown, A. W. (2004). Model driven architecture: Principles and practice. *Software and Systems Modeling, 3*(4), 314–327. https://doi.org/10.1007/s10270-004-0061-2.

Calvanese, D., Cogrel, B., Komla-Ebri, S., et al. (2017). Ontop: Answering SPARQL queries over relational databases. *Semantic Web, 8*(3), 471–487. https://doi.org/10.3233/SW-160217

Calvanese, D., Liuzzo, P., Mosca, A., et al. (2016). Ontology-based data integration in EPNet: Production and distribution of food during the roman empire. *Engineering Applications of Artificial Intelligence, 51,* 212–229. https://doi.org/10.1016/j.engappai.2016.01.005.

Carriero, V. A., Gangemi, A., Mancinelli, M. L., Marinucci, L., Nuzzolese, A. G., Presutti, V., & Veninata, C. (2019). Arco ontology network and LOD on italian cultural heritage. In: Proceedings of the First International Workshop on Open Data and Ontologies for Cultural Heritage Co-located with the 31st International Conference on Advanced Information Systems Engineering, ODOCH@CAiSE. CEUR Workshop Proceedings, vol. 2375, pp. 97–102

Delcambre, L. M. L., Liddle, S. W., Pastor, O., & Storey, V. C. (2018). A reference framework for conceptual modeling. In: 37th Int. Conference on Conceptual Modeling, ER. LNCS. Springer, vol. 11157, pp. 27–42. https://doi.org/10.1007/978-3-030-00847-5_4

Dudáš, M., Lohmann, S., Svátek, V., & Pavlov, D. (2018). Ontology visualization methods and tools: a survey of the state of the art. The Knowledge Engineering Review 33. https://doi.org/10.1017/S0269888918000073

Farré, C., Queralt, A., Rull, G., Teniente, E., & Urpí, T. (2013). Automated reasoning on UML conceptual schemas with derived information and queries. *Information and Software Technology, 55*(9), 1529–1550. https://doi.org/10.1016/j.infsof.2013.02.010.

Fillottrani, P. R., Franconi, E., & Tessaris, S. (2012). The ICOM 3.0 intelligent conceptual modelling tool and methodology. Semantic Web 3(3), 293–306. https://doi.org/10.3233/SW-2011-0038

Fillottrani, P. R., & Keet, C. M. (2020). An analysis of commitments in ontology language design. In: 11th Int. Conference on Formal Ontology in Information Systems (FOIS), vol. 330, pp. 46–60. https://doi.org/10.3233/FAIA200659

Fillottrani, P. R., & Keet, C. M. (2021). Evidence-based lean conceptual data modelling languages. *Journal of Computer Science and Technology, 21*(2), 93–111. https://doi.org/10.24215/16666038.21.e10.

Fillottrani, P. R., & Keet, C. M. (2014). Conceptual model interoperability: a metamodel-driven approach. In: 8th Int. RuleML. LNCS, vol. 8620, pp. 52–66. https://doi.org/10.1007/978-3-319-09870-8_4

Fillottrani, P. R., & Keet, C. M. (2015). Evidence-based languages for conceptual data modelling profiles. In: 19th Conference on advances in databases and information systems (ADBIS). LNCS, vol. 9282, pp. 215–229. https://doi.org/10.1007/978-3-319-23135-8_15

Fillottrani, P. R., & Keet, C. M. (2017). Patterns for heterogeneous tbox mappings to bridge different modelling decisions. In: 14th Extended Semantic Web Conference (ESWC'17). LNCS, vol. 10249, pp. 371–386. https://doi.org/10.1007/978-3-319-58068-5_23

Garrido, M. A. (2019). Verbalización de un Subconjunto de UML en una Herramienta Web. MSc thesis, Univ.Nac. del Comahue, Argentina

Guarino, N., Guizzardi, G., & Mylopoulos, J. (2020). On the philosophical foundations of conceptual models. *Information Modelling and Knowledge Bases, 31*(321), 1. https://doi.org/10.3233/FAIA200002

Guerson, J., Sales, T. P., Guizzardi, G., & Almeida, J. P. A. (2015). OntoUML lightweight editor: a model-based environment to build, evaluate and implement reference ontologies. In: 19th EDOC Workshops. IEEE Computer Society, pp. 144–147. https://doi.org/10.1109/EDOCW.2015.17

Haarslev, V., Hidde, K., Möller, R., & Wessel, M. (2012). The RacerPro knowledge representation and reasoning system. *Semantic Web, 3*(3), 267–277. https://doi.org/10.3233/SW-2011-0032

Halpin, T., & Morgan, T. (2008). Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann

He, Y., Yu, H., Ong, E., et al. (2020). CIDO: the community-based coronavirus infectious disease ontology. In: 11th Int. Conference on Biomedical Ontologies (ICBO). CEUR-WS, vol. 2807

Hobbs, J. R., & Pan, F. (2004). An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing, 3*(1), 66–85. https://doi.org/10.1145/1017068.1017073.

Jeusfeld, M. A. (2021). Multilevel modeling with conceptbase. In: Lukyanenko, R., Samuel, B.M., Sturm, A. (eds.) Proceedings of the ER Demos and Posters 2021 Co-located with 40th Int. Conference on Conceptual Modeling (ER 2021). CEUR-WS, vol. 2958. https://www.ceur-ws.org/Vol-2958/paper1.pdf

Jeusfeld, M., Jarke, M., & Mylopoulos, J. (2009). *Metamodeling for Method Engineering* (1st ed.). Cambridge: The MIT Press.

Keet, C. M., & Fillottrani, P. R. (2015). An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2. *Data & Knowledge Engineering, 98,* 30–53. https://doi.org/10.1016/j.datak.2015.07.004.

Keet, C. M., & Fillottrani, P. R. (2015). An analysis and characterisation of publicly available conceptual models. In: 34th Int. Conference on Conceptual Modeling (ER). LNCS. Springer, vol. 9381, pp. 585–593. https://doi.org/10.1007/978-3-319-25264-3_45

Kelly, S., Lyytinen, K., Rossi, M., & Tolvanen, J. (2013). Metaedit at the age of 20. In: Jr., J.A.B., Krogstie, J., Pastor, O., Pernici, B., Rolland, C., Sølvberg, A. (eds.) Seminal Contributions to Information Systems Engineering, pp. 131–137. https://doi.org/10.1007/978-3-642-36926-1_10

Kensche, D., Quix, C., Chatti, M. A., & Jarke, M. (2007). Gerome: A generic role based metamodel for model management. *Journal on Data Semantics, 8,* 82–117. https://doi.org/10.1007/978-3-540-70664-9_4.

Khan, Z., & Keet, C. M. (2021). Structuring abstraction to achieve ontology modularisation. In: Daramola, O., Moser, T. (eds.) Advanced Concepts, Methods, and Applications in Semantic Computing. IGI Global, pp. 72–92

Koubarakis, M., Borgida, A., Constantopoulos, P., Doerr, M., Jarke, M., Jeusfeld, M. A., et al. (2021). A retrospective on Telos as a metamodeling language for requirements engineering. *Requirements Engineering, 26*(1), 1–23. https://doi.org/10.1007/s00766-020-00329-x.

Liebig, T., Luther, M., Noppens, O., & Wessel, M. (2011). OWLlink. *Semantic Web, 2*(1), 23–32. https://doi.org/10.3233/SW-2011-0027

Lodi, G., Asprino, L., Nuzzolese, A. G., Presutti, V., Gangemi, A., Recupero, D. R., Veninata, C., & Orsini, A. (2017). Semantic Web for Cultural Heritage Valorisation, pp. 3–37. Springer, Cham. https://doi.org/10.1007/978-3-319-54499-1_1

Lubyte, L., & Tessaris, S. (2009). Automated extraction of ontologies wrapping relational data sources. In: Int. Conference on Database and Expert Systems Applications (DEXA). Springer, pp. 128–142. https://doi.org/10.1007/978-3-642-03573-9_10

Mahlaza, Z., & Keet, C. M. (2021). ToCT: A task ontology to manage complex templates. In: FOIS Ontology Showcase, The Joint Ontology Workshops (JOWO). CEUR-WS, vol. 2969. https://www.ceur-ws.org/Vol-2969/paper40-FoisShowCase.pdf

Mayr, H. C., & Thalheim, B. (2021). The triptych of conceptual modeling. *Software and Systems Modeling, 20*(1), 7–24. https://doi.org/10.1007/s10270-020-00836-z.

Motik, B., Patel-Schneider, P. F., & Parsia, B. (2022). OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C recommendation, W3C (accessed 3 February 2022). www.w3.org/TR/owl2-syntax/

Musen, M. A. (2015). The protégé project: a look back and a look forward. *AI Matters, 1*(4), 4–12. https://doi.org/10.1145/2757001.2757003

Nicolaescu, P., Rosenstengel, M., Derntl, M., Klamma, R., & Jarke, M. (2016). View-Based Near Real-Time Collaborative Modeling for Information Systems Engineering. In: Advanced Information Systems Engineering - 28th Int. Conference, CAiSE, vol. 9694, pp. 3–17. https://doi.org/10.1007/978-3-319-39696-5_1

Object Management Group (2017). OMG Unified Modeling Language (OMG UML). Object Management Group. www.omg.org/spec/UML/2.5.1/

Object Management Group (2022) Semantics of business vocabulary and rules (SBVR) – OMG Released Versions of SBVR. www.omg.org/spec/SBVR/1.0

Ong, D., & Jabbari, M. (2019). A review of problems and challenges of using multiple conceptual models. In: vom Brocke, J., Gregor, S., Müller, O. (eds.) 27th European Conference on Information Systems, ECIS 2019. https://www.aisel.aisnet.org/ecis2019_rp/179

Poveda-Villalon, M., Garcia-Castro, R., Daniele, L., de Roode, M. (2019). SAREF4AGRI: an extension of SAREF for the agriculture and food domain. https://saref.etsi.org/saref4agri/v1.1.2/

QUDT.org (2011). FAIRsharing.org: QUDT; Quantities, Units, Dimensions and Types. https://doi.org/10.25504/FAIRsharing.d3pqw7

Rull, G., Farré, C., Queralt, A., Teniente, E., & Urpí, T. (2015). AuRUS: explaining the validation of UML/OCL conceptual schemas. *Software and Systems Modeling, 14*(2), 953–980. https://doi.org/10.1007/s10270-013-0350-8.

Sabegh, M. A. J., & Recker, J. (2017). Combined use of conceptual models in practice: An exploratory study. *Journal of Database Management, 28*(2), 56–88. https://doi.org/10.4018/JDM.2017040103.

Sportelli, F., & Franconi, E. (2016). Formalisation of ORM Derivation Rules and Their Mapping into OWL. In: OTM Conferences in Computer Science, vol. 10033, pp. 827–843. https://doi.org/10.1007/978-3-319-23135-8_15

Steigmiller, A., Liebig, T., & Glimm, B. (2014). Konclude: System description. *Journal of Web Semantics, 27–28,* 78–85. https://doi.org/10.1016/j.websem.2014.06.003.

Thalheim, B. (2009). Extended Entity Relationship Model. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems. Springer, vol. 1, pp. 1083–1091

Thalheim, B. (2010). Towards a theory of conceptual modelling. *Journal of Universal Computer Science, 16*(20), 3102–3137. https://doi.org/10.3217/jucs-016-20-3102.

Uhnák, P., & Pergl, R. (2016). The openponk modeling platform. In: Proceedings of the 11th Edition of the International Workshop on Smalltalk Technologies. ACM. https://doi.org/10.1145/2991041.2991055

Venable, J. R., & Grundy, J. C. (1995). Integrating and supporting entity relationship and object role models. In: 14th Int. Conference of Object-Oriented and Entity-Relationship Modelling. LNCS, vol. 1021, pp. 318–328. https://doi.org/10.1007/BFb0020543

Whittle, J., Hutchinson, J., & Rouncefield, M. (2014). The state of practice in model-driven engineering. *IEEE Software, 31*(3), 79–85. https://doi.org/10.1109/MS.2013.65

Zhu, N., Grundy, J. C., & Hosking, J. G. (2004). Pounamu: a metatool for multi-view visual language environment construction. In: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE Computer Society, pp. 254–256. https://doi.org/10.1109/VLHCC.2004.41

Authors and Affiliations

Germán Braun[1,2] · Pablo Rubén Fillottrani[3,4] · C. Maria Keet[5]

Pablo Rubén Fillottrani
prf@cs.uns.edu.ar

C. Maria Keet
mkeet@cs.uct.ac.za

[1] Universidad Nacional del Comahue, 1400 Buenos Aires, Argentina

[2] Consejo Nacional de Investigaciones Científicas y Técnicas, Godoy Cruz, 2290 (C1425FQB) Buenos Aires, Argentina

[3] Universidad Nacional del Sur, San Andrés 800, Bahía Blanca, Argentina

[4] Comisión de Investigaciones Científicas de la provincia de Buenos Aires, calle 526 e/10 y 11, La Plata, Argentina

[5] Department of Computer Science, University of Cape Town, 18 University Avenue, 7700 Rondebosch, Cape Town, South Africa