A Framework for Interoperability Between Models with Hybrid Tools: Use Case with crowd 2.0*

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

[1]Universidad Nacional del Comahue, Argentina.
[2]Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina.
[3]Universidad Nacional del Sur, Argentina.
[4]Comisión de Investigaciones Científicas de la provincia de Buenos Aires, Argentina.
[5]Department of Computer Science, University of Cape Town, South Africa.

Contributing authors: german.braun@fi.uncoma.edu.ar; prf@cs.uns.edu.ar; mkeet@cs.uct.ac.za;

1 Covid19 Use case with crowd 2.0

This document details motivational scenario about the modelling of the COVID-19 medicines in Fig 1 with the tool crowd 2.0. To do this, we will follow the use case depicted in Fig. 2, which illustrates how users can work collaboratively by exploring a conceptual model from different perspectives that suit their respective competencies and core tasks.

First, let us put the UML diagram into crowd 2.0 (see Fig. 3). 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[1], called Experimental substance, which is also shown in Fig. 3. Its back-end runtime conceptual model as KF instantiation and the implementation of the rules were then called to

*Submitted to the Journal of Intelligent Information Systems
[1]https://www.ebi.ac.uk/ols/ontologies/cido

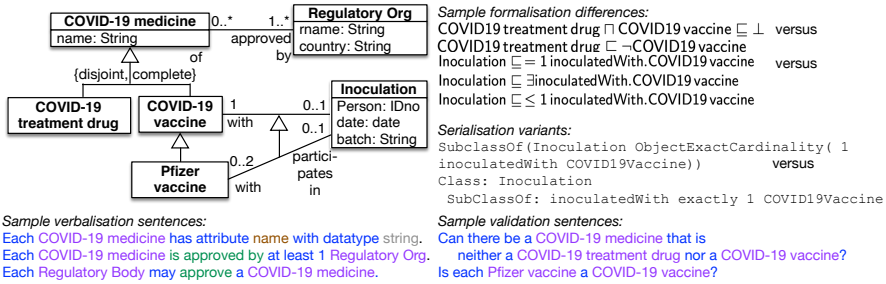


Fig. 1 A selection of a possible conceptual model about COVID-19 medicines (suboptimal for illustrative purpose) and examples of formalisation and serialisation variants.

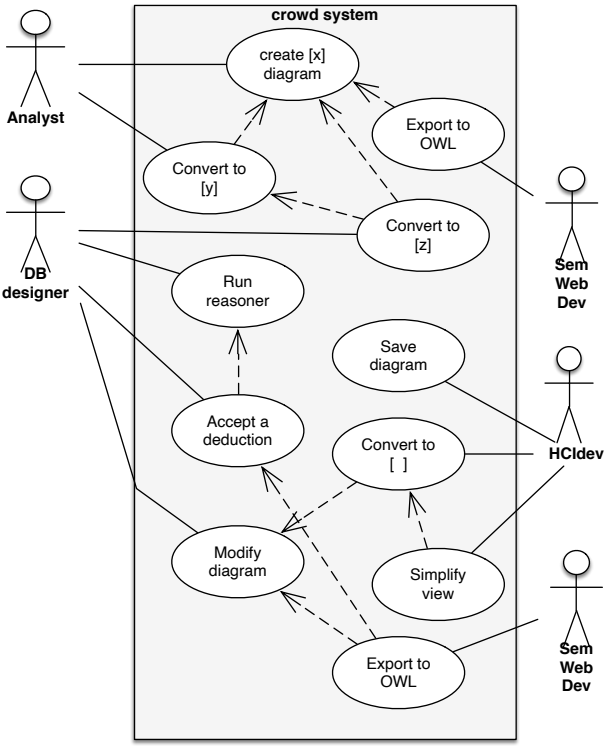


Fig. 2 Sample use case diagram related to the use case described in Section 1 The [x], [y] and [z] are distinct and either UML, EER, and ORM; the "[ ]" is UML in the use case, but each type of diagram can be simplified and so left blank here.

generate automatically their corresponding EER and ORM2 versions, which are depicted in Fig. 4. 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. 4: EER requires names for relationships, but UML does not, and this discrepancy is fixed with default naming. This is similar for ORM2, where a UML attribute is converted into

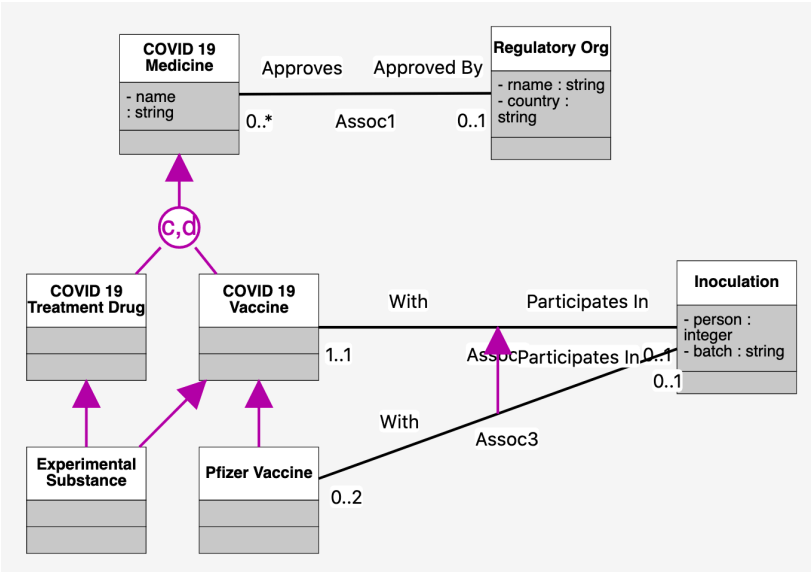


Fig. 3 The same UML diagram as in Fig. 1, but then rendered in crowd 2.0's interface.

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 EER diagram. They decide to run the reasoner over the EER diagram to double-check its quality. The deductions are displayed visually as shown in Fig. 5 and, upon clicking it, a brief note appears in the right-hand pane of the tool, as shown in the two fragments in Fig. 6: 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 relationships. The modeller can accept or reject the deductions.

The database analyst is actually more of an expert in ORM and for now rejects the deductions, automatically transforms the EER diagram into an ORM diagram, and runs the reasoner again. The same deductions are displayed visually. The analyst is now more confident about the deduction on the cardinality constraint on Pfizer vaccine and accepts that deduction. They also remember to add that the regulatory organisation coordinates the Inoculation, and modify the model accordingly. Since their colleague actually does prefer EER, they convert it back to EER, which is shown in Fig. 7 with the relevant section in the GUI of crowd 2.0 (for purpose of readability and layout, these and the following figures are included at the end of the paper in the appendix). There was still that Experimental substance to deal with, and the EER diagram is modified by deleting that unsatisfiable class. A third colleague, in HCIdev group, prefers UML class diagram notation of front-end app development and so it is transformed back into one (Fig. 8), which, cf. the model we started

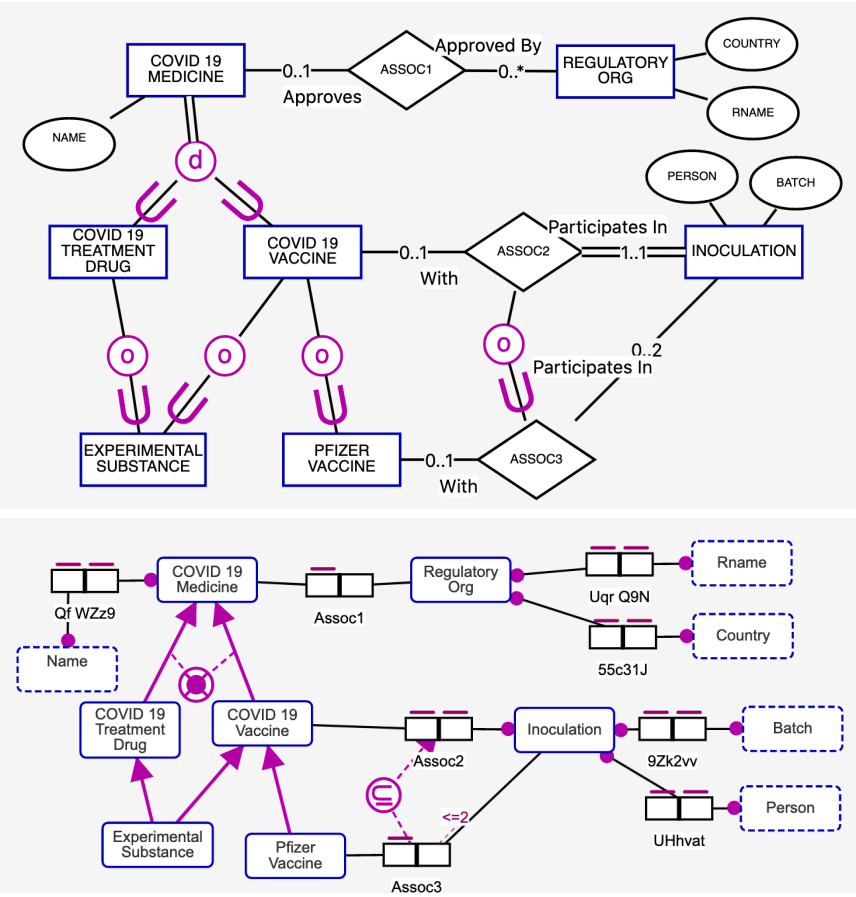


Fig. 4 Automatically generated EER and ORM diagram versions of the original UML class diagram model of Fig. 3.

with, has been updated with the changes that were made to the model when it was an ORM and an EER diagram before.

This UML diagram still can be modified just like it was done with the other two models in the other notations, which then also correspondingly updates the runtime conceptual model and its serialisation in JSON. A section of the JSON serialisation is depicted in Fig. 9. Then, to generate a more compact diagram for an easier overview, the HCIdev modeller can choose to 'declutter' it by toggling the attributes/methods and the role and relationship names, so as to obtain a simplified visual notation whilst maintaining the details in the background (Fig. 10). Also, upon the client's request for validating the model, the HCIdev employee generates a text-based version with an English CNL (see Fig. 11).

Finally, to not lose all the work done, any of the modellers involved can save and, optionally also, export it from crowd 2.0. The tool saves the model

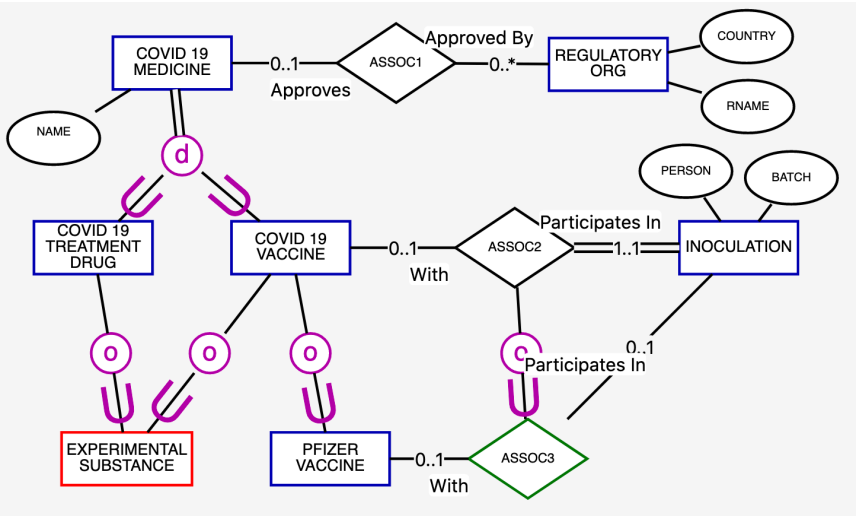


Fig. 5 Deductions over the model, obtained with the automated reasoner: Experimental substance is unsatisfiable and the cardinality on the Pfizer vaccine changed from 0..2 to 0..1.

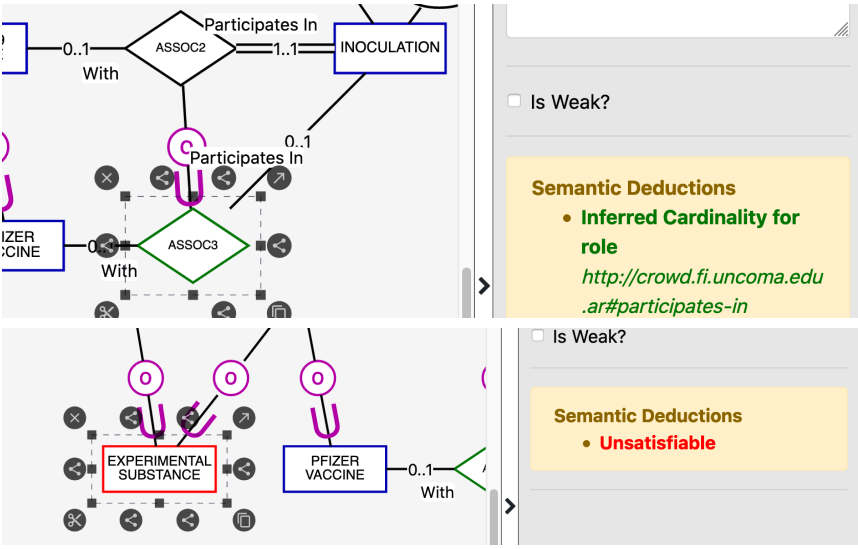


Fig. 6 Clicking on the coloured elements of the model (included in Fig. 5) provides some detail in the right-hand pane for both deduction

in JSON, which is specific to each CDML (see Fig. 12), so as to serialise and persistently store the instantiation of the KF metamodel (see Fig. 13). It can export it to OWL (see Fig. 14) so that it also can be used in an ontology-driven information system that relies on Semantic Web technologies.

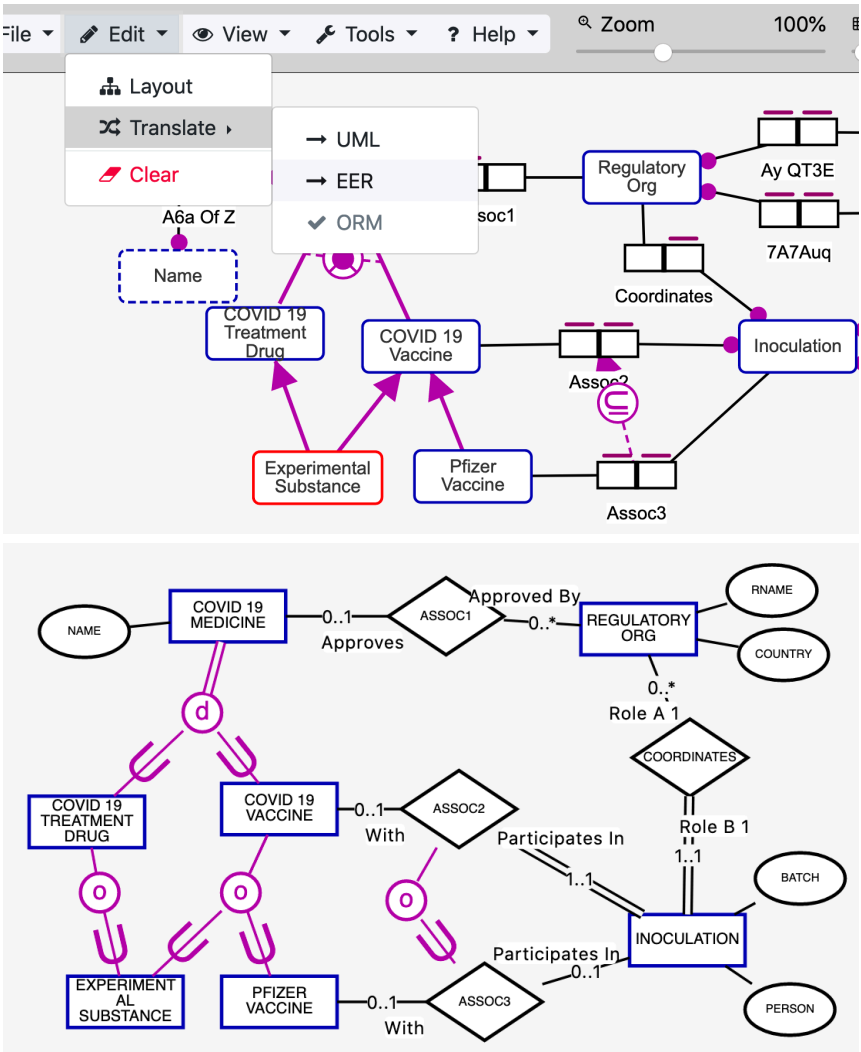


Fig. 7 With the Pfizer vaccine cardinality deduction accepted and the model updated, the intent is to swap back to EER notation (top) and the outcome of that (bottom).

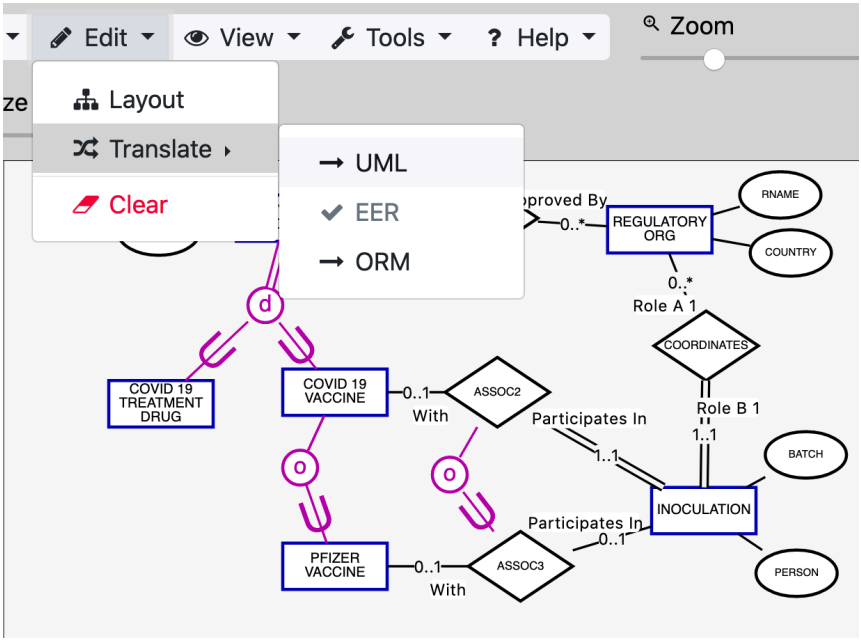
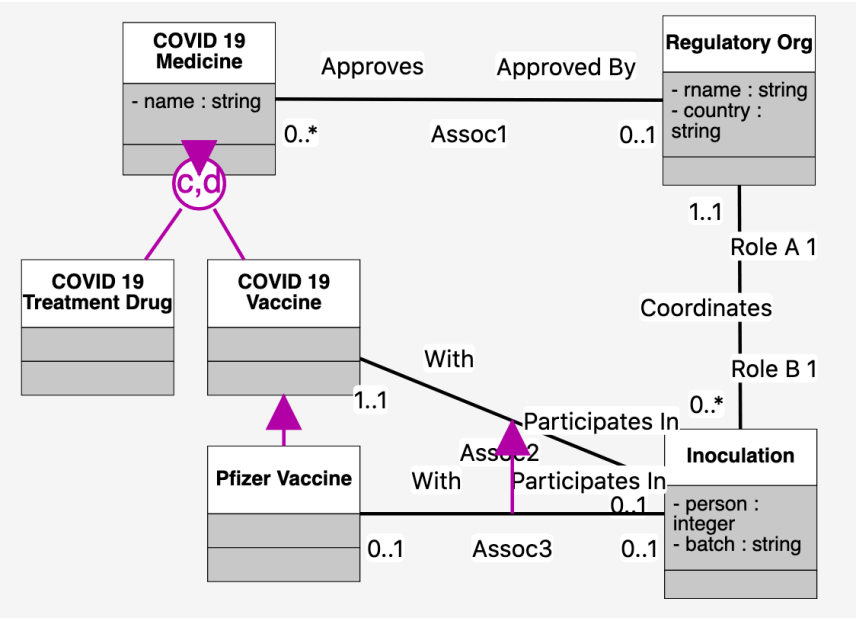


Fig. 8 With the unsatisfiable class removed, the intent is to swap back to UML notation again.



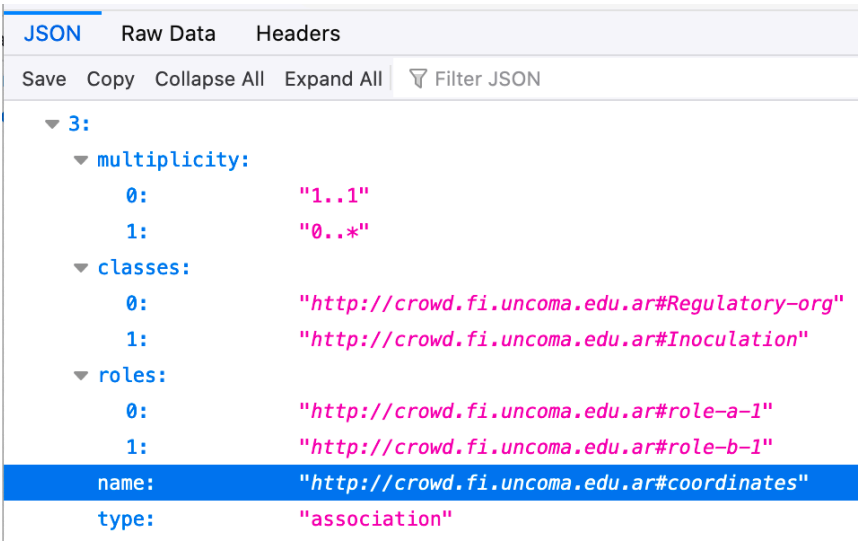


Fig. 9 Back in UML, maintaining that named association (top), which is also reflected in the runtime conceptual model and therefore also its exported serialisation in JSON (bottom).

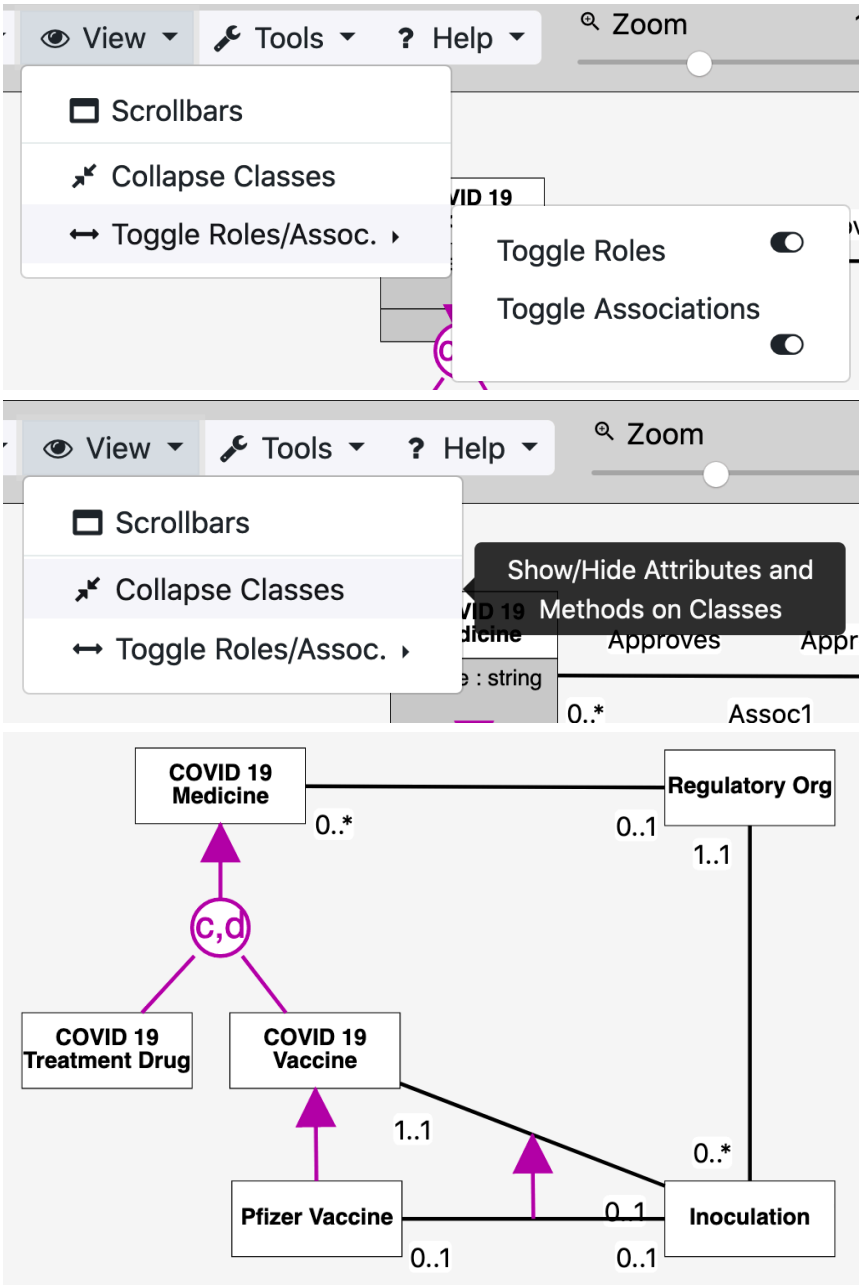


Fig. 10 Toggling to abstract away details and generate a simplified view of the same UML diagram as in Fig. 9.

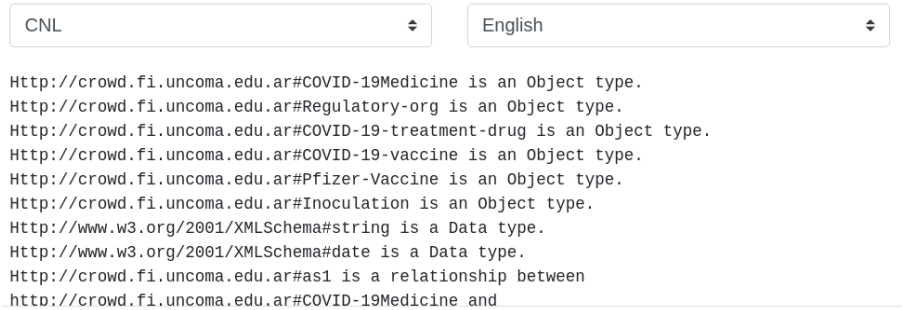


Fig. 11 Screenshot of a section of the CNL output for the final model.

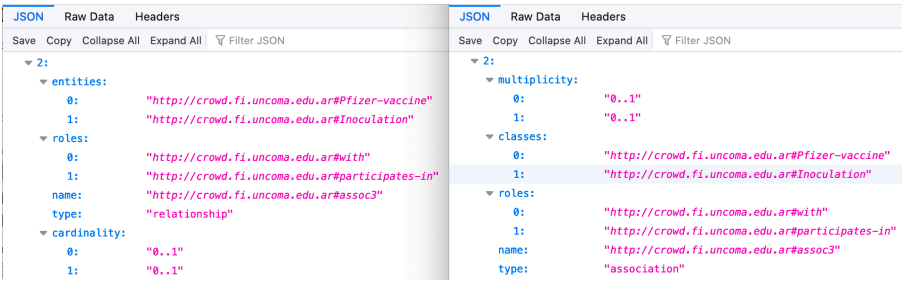


Fig. 12 Screenshots of comparable sections of the respective JSON file that is specific to each CDML. Left: EER; Right: UML

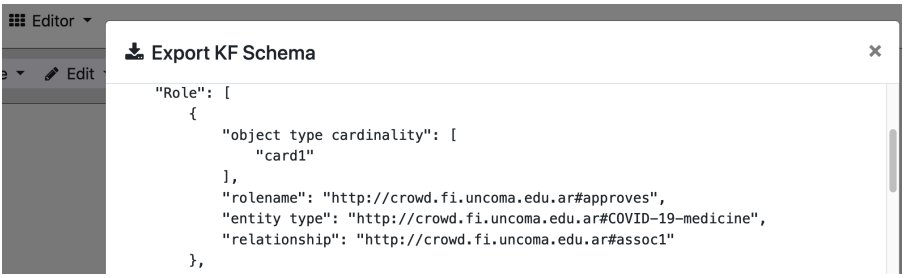


Fig. 13 Screenshot of the export to the KF metamodel persistent storage.

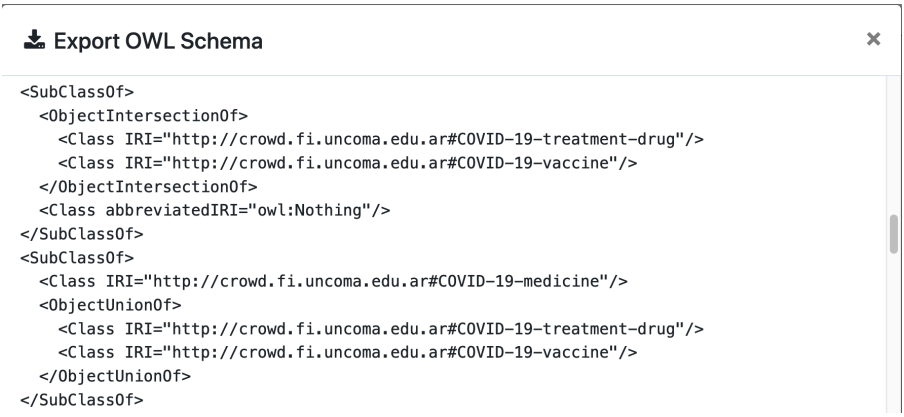


Fig. 14 Screenshot of the KF metamodel encoded into OWL/XML format, at the time of exporting it for use elsewhere.