ExpO: Towards Explaining Ontology-Driven Conceptual Models

Elena Romanenko[1(✉)], Diego Calvanese[1,2], and Giancarlo Guizzardi[3,4]

[1] Free University of Bozen-Bolzano, Bolzano, Italy
{eromanenko,diego.calvanese}@unibz.it
[2] Umeå University, Umeå, Sweden
[3] University of Twente, Enschede, The Netherlands
g.guizzardi@utwente.nl
[4] Stockholm University, Stockholm, Sweden

Abstract. Ontology-driven conceptual models play an explanatory role in complex and critical domains. However, since those models may consist of a large number of elements, including concepts, relations and sub-diagrams, their reuse or adaptation requires significant efforts. While conceptual model engineers tend to be biased against the removal of information from the models, general users struggle to fully understand them. The paper describes ExpO—a prototype that addresses this trade-off by providing three components: *(1)* an API that implements model transformations, *(2)* a software plugin aimed at modelers working with the language OntoUML, and *(3)* a web application for model exploration mostly designed for domain experts. We describe characteristics of every component and specify scenarios of possible usages.

Keywords: Ontology-Driven Conceptual Models · OntoUML · Pragmatic explanation · Software Tools

1 Introduction

A *conceptual model* (CM) is an abstract, high-level representation of the domain of interest, the task that needs to be carried out, or the software itself. In recent years, *ontology-driven conceptual models* (ODCMs) have been proposed as a particular class of conceptual models that gain an advantage by utilizing ontological theories to develop engineering artefacts [15].

Both ODCMs and traditional CMs alike play a fundamental role in organizing communication between people with different backgrounds, such as programmers, ontology engineers, and domain experts. Models are also supposed to be easily reused and extended. Unfortunately, in reality, there is an unspoken disagreement between domain experts and conceptual model engineers—authors of the existing ODCMs. While modelers tend to consider all information specified in the model as necessary, thus, resisting attempts to simplify these artefacts (see experiments in [12]), domain experts, as general users, are struggling to fully understand how to adapt an already existing model to their needs.

J. Araújo et al. (Eds.): RCIS 2024, LNBIP 514, pp. 20–28, 2024.
https://doi.org/10.1007/978-3-031-59468-7_3

In this paper, we present *ExpO* — a prototype system for explaining existing ODCMs. We rely here on the notion of *pragmatic explanation* for ODCMs as discussed in [11] for domain ontologies, and focus on explaining models without changing the form of explanation. ExpO consists of *(1)* an API that implements model transformations that are supposed to help in ODCM understanding, *(2)* an extension of the existing OntoUML Plugin with the API functionality, and *(3)* a web application for model exploration mostly designed for domain experts with a non-technical background. This work is based on several previous papers [5, 7,10,11]. However, our main purpose here is not to formulate a final set of explanation operations needed to reach a better understanding of a given model, but to present the system itself.

The remainder of the paper is organized as follows: Sect. 2 presents background; Sect. 3 proposes ExpO, introduces its architecture, and elaborates on various components of the system; Sect. 4 provides final considerations and outlines future work.

2 Background

Foundational ontologies are a special class of ontologies specifying a general schema for describing objects, their constitution and composition, roles that objects can play, events and their goals, and qualities of objects and events. In a recent special issue of the Applied Ontology journal [1], seven of the most commonly used foundational ontologies were listed, including the *Unified Foundational Ontology* (UFO).

UFO [4] draws contributions from Formal Ontology in Philosophy, Philosophical Logic, Cognitive Psychology, and Linguistics. It is particularly interesting for our research given that it is used as a foundation for OntoUML, one of the most widely used languages in ontology-driven conceptual modeling [15]. *OntoUML* is a language that extends UML class diagrams by defining a set of stereotypes. These stereotypes expand UML's meta-model so that classes and associations decorated with them bring precise (real-world) semantics grounded in UFO. Additionally, OntoUML models satisfy a number of semantically motivated syntactic constraints, ensuring their compliance with the UFO axiomatization [6].

In general, conceptual model engineers are better supported for model development than domain experts for model exploration. For example, *Visual Paradigm* (VP)[1] is a widespread modeling tool that facilitates the development of different types of diagrams, including UML models. For those who are developing ODCMs, there is, e.g., an *OntoUML Plugin* for VP[2] that automatically checks the ontological consistency of an OntoUML diagram in light of UFO. Nevertheless, both proficiency in technical skills and familiarity with UFO, along with the obligation to install specialized software, are essential for utilizing the plugin. At the same time, there is no proper tool for users who would like to

[1] https://www.visual-paradigm.com.
[2] https://github.com/OntoUML/ontouml-vp-plugin.

familiarize themselves with the existing model or simply check if it is suitable for their specific requirements.

To facilitate understanding of existing models, software tools that manage ODCMs should incorporate features that provide explanations for such models. In this context, the term 'explanation' refers to *pragmatic explanation*, where 'pragmatic' signifies an 'instrumentalist' approach to constructing explanations (see [16]), emphasizing the creation of a 'toolbox' designed to assist users in reaching their goals. In [11], we show that complexity management approaches, such as clustering [7] and abstraction [5,10], can be viewed as pragmatic explanation techniques when talking about domain ontologies. Recent experiments revealed that this is also true for ODCMs [12]. However, to the best of our knowledge, there is no uniform tool that offers different transformations of ODCMs, potentially enhancing comprehension of the given model.

Visual Paradigm is a popular modeling tool, but together with the OntoUML Plugin, it is more used by conceptual modelers and ontology engineers for model development rather than by domain experts for model exploration. Currently, it already provides some desired functionality, such as model clustering [7].

Protégé[3] [9] aims at OWL artefacts. ODCMs—exported into Turtle format—may be examined in it with different visualization plugins, e.g., *OntoGraph*[4]. Although these plugins lack important functionality, they provide the possibility to focus or hide the concept of interest. This, however, may lead to a cognitive load of users when dealing with large models [8].

WebVOWL[5] also can load a model in Turtle format, but visualizes individuals as lists, which leads to the same problem of cognitive overload.

Although *Evonne*[6] [8] was developed with a completely different goal—to support interactive debugging of ontologies—it seems to be the most relevant tool to our research since it is aimed at explanations. At the same time, Evonne is aimed at OWL models and, hence, does not support explanation and complexity management for ODCMs in a manner that is aligned with foundational categories.

3 ExpO Architecture

Before presenting the design of the ExpO system, we first need to determine its potential users and their requirements. During a user study we carried out [12], it became obvious that at least two categories of users must be distinguished: *(1) authors of the models* (model engineers, experienced modelers), and *(2) domain experts* without prior knowledge about OntoUML modeling or potential model users who are unfamiliar with the domain.

Taking the defined categories of users into account, we formulated the following design goals for the system.

[3] http://protege.stanford.edu.
[4] https://protegewiki.stanford.edu/wiki/OntoGraf.
[5] http://vowl.visualdataweb.org/webvowl.html.
[6] https://imld.de/en/research/research-projects/evonne.

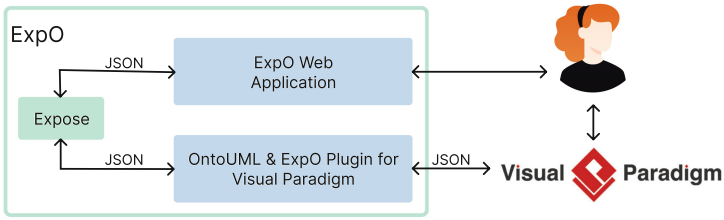

Fig. 1. Broad overview of the approach.

DG1: *Support different levels of expertise.* For users with less modeling experience the system should remain intuitive.
DG2: *Minimize setup difficulties.* Novice users with little technical background should receive a ready-to-use tool.
DG3: *Enable interactive exploration.* Interaction plays a critical role in providing explanations and reaching understanding [8,11].
DG4: *Build on familiar representations.* The system should use standard visual representations of graphs as node-link diagrams.
DG5: *Keep the original layout.* The system should try to keep the original (expert) layout of the ODCM when possible in order not to confuse users, who probably worked with the same model in VP before.

In [14], the author suggests the following strategy for visual information-seeking: "overview first, zoom and filter, then details on demand". Based on this guideline, seven tasks were suggested for systems that provide visual content:

Overview: gain an overview of the entire collection of elements;
Zoom: zoom in on items of interest;
Filter: filter out uninteresting items;
Details-on-demand: select an item or group and get details when needed;
Relate: view relations among items;
History: keep a history of actions to support undo and replay;
Extract: allow extraction of sub-collections.

This approach was further extensively reused, e.g., in [2]. During the development of our system, we took into account these guidelines and the refined associated tasks, and suggested an architecture of the system with three separate components (see Fig. 1):

- *Expose*, a server that provides an API for ODCM transformations;
- *OntoUML & ExpO Plugin for VP*, an updated version of the OntoUML Plugin for VP with extended functionality;
- *ExpO Web Application*, a web interface mostly aimed at domain experts.

The Expose component is written in Python using the FastAPI. It is a server component[7], that provides open routes that implement transformation opera-

[7] Use https://w3id.org/ExpO/expose/health to check if the server is accepting requests.

Table 1. Short description of some of the API routes.

Route	Specification		Description
Focus	*Type* *Parameters*	POST Body: graph, node, hop	Focuses on the given node and keeps only those concepts that are reached by no more than hop relations.
Cluster	*Type* *Parameters*	POST Body: graph, node	If the given node is a *Relator*, applies the relator-centric approach for clustering [7].
Define	*Type* *Parameters*	GET Parameters: concept, number of definitions	Provides several definitions of the given concept that can be found in a dictionary. This route is used by the web application only.
Expand	*Type* *Parameters*	POST Body: graph, node, limit	Finds a similar concept (by name and stereotype) in the OntoUML / UFO Catalog [13] and, if found, extends the hierarchy with the data from the Catalog using no more than limit concepts.
Abstract	*Type* *Parameters*	POST Body: graph, abstraction type	Applies the given type of abstraction (one or more) to the graph. The supported abstraction types are "parthood", "hierarchy", and "aspects" [10].
Fold	*Type* *Parameters*	POST Body: graph, node	Collapses all hierarchical and part-whole relations of the given node.

tions[8]. A short description of some of the open routes as well as additional remarks regarding some of the requests are collected in Table 1.

The *Expand* route keeps an index of all concepts and their stereotypes that were mentioned in all models of the OntoUML / UFO Catalog [13]. In order to get the models with the concept of interest, it sends an API request to the GitHub REST API[9] and tries to extend the current model with the collected information. The *Define* route sends an API request to the *Wiktionary*[10] service to collect likely definitions of the given concept. We assume that this information can be used by novice users not familiar with the domain. All other routes are processed directly by Expose, and, as shown in Table 1, most of them are POST requests. The reason for this is that Expose does not store user data. Thus, the general idea is to take the original model, apply the requested transformation operation to it, and return the result in the right format. All routes respond with JSON, the content of which depends on the type of the output format. For the plugin, the server returns only the model that can be loaded directly into

[8] The full documentation can be found in the corresponding folder of the project on https://w3id.org/ExpO/github and on https://w3id.org/ExpO/expose/docs.
[9] https://docs.github.com/en/rest?apiVersion=2022-11-28.
[10] https://en.wiktionary.org.

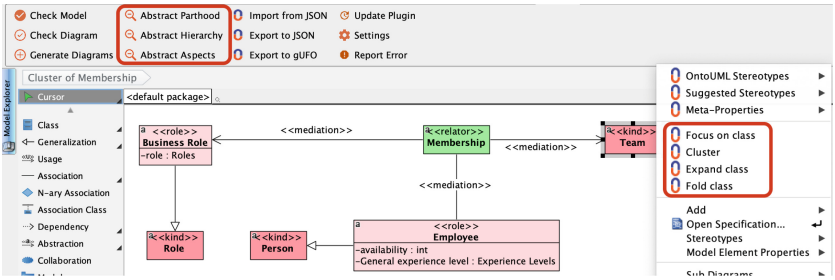

Fig. 2. Changes in the main menu and the context menu of the plugin. In the picture, you can see the result of applying a *Cluster* request to the 'Membership' concept for the model 'jacobs2022sdpontology' from the OntUML / UFO Catalog [13].

VP. For the web application, the result consists of the original graph (for future processing) and its simplification and adaptation for the web interface.

The original OntoUML Plugin for VP was updated in order to support the functionality provided by Expose. Figure 2 shows the updates in the main and context menus that were introduced. The plugin is developed in Java and published on GitHub[11]. However, the user interface is constrained by VP. For instance, while it supports zoom functionality for models, there are limitations when searching for a concept or relation. Users can only see those elements that were found on the active diagram, without the ability to search across the entire project. Additionally, the system has some difficulties in maintaining a comprehensive history of actions; although actions are mostly reversible within a session, each model received from Expose is treated as an entirely new one. These limitations are intrinsic to the existing implementations of VP and the original plugin and cannot be readily overcome. For these reasons, achieving the first three design goals (DG1–DG3) is challenging using only the plugin. For users who do not want to install additional software, miss the required technical skills, or just want to quickly explore the model without the necessity to download it, we suggest using the ExpO Web Application.

The ExpO Web Application[12] is written in JavaScript with the help of the React library[13] and the react-d3-graph component[14]. In order not to confuse the user (who could have used the OntoUML & ExpO Plugin before) and in accordance with the last design goal, the smart colouring scheme (see, e.g., in [7]) and the layout are kept as close to the original model as possible. Note, that full compliance cannot always be achieved, since VP provides an opportunity to divide the project into several models, while the web application combines all of them in one view. Both zoom opportunities and the possibility to perform

[11] https://w3id.org/ExpO/plugin.
[12] https://w3id.org/ExpO.
[13] https://react.dev.
[14] https://danielcaldas.github.io/react-d3-graph.

undo / replay are supported. Search for all elements (concepts, relations, and constraints) is provided across the whole model.

4 Conclusions and Future Work

Ontology-driven conceptual models are supposed to build a bridge for communication between ontology engineers or programmers and domain experts without special technical knowledge. However, before these models can be reused, they need to be understood by their users.

In this paper, we have presented ExpO, a prototype system aimed at producing explanations for ODCMs. The ExpO system consists of three components, each with its own functionality. The *ExpO Server* applies transformation operations to ODCMs. Those operations cover most of the tasks described by the "Visual Information-Seeking Mantra". The rest of the tasks, namely Zoom and History, are covered by the *OntoUML & ExpO Plugin for VP* and by the *ExpO Web Application*. The former is mostly aimed at professionals, who have worked with modeling tools before, while the web application does not require installation or model download, and allows for model investigation on-the-fly.

We expect to continue to extend the functionality of the system. A prospective method can be based on showing the hierarchy below / above the selected concept. For now, most of the explanation transformations are applied in an automatic mode, but the user may be interested in selecting certain model elements that need to be maintained in the final explanation. Also, ideally, each ODCM should be accompanied by a list of *competency questions* [3]. By filtering this list according to concepts that are still left in view, we can help our users determine how far they would like to reduce the model. Finally, a qualitative user study is needed, investigating whether this set of transformation operations is enough to support a full understanding of the model, and the role the web application plays in it.

Resource Availability Statement: The ExpO Server is available at https://w3id.org/ExpO/expose/health. The ExpO Plugin can be downloaded and installed from the GitHub repository https://w3id.org/ExpO/plugin. The ExpO Web Application is available at https://w3id.org/ExpO. The corresponding GitHub repository for the project is https://w3id.org/ExpO/github. The software is distributed under the Apache 2.0 license.

Acknowledgements. This research has been partially supported by the Province of Bolzano and DFG through the project D2G2 (DFG grant n. 500249124), by the HEU project CyclOps (grant agreement n. 101135513), and by the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation.

References

1. Borgo, S., Galton, A., Kutz, O.: Foundational ontologies in action. Appl. Ontol. **17**, 1–16 (2022). https://doi.org/10.3233/AO-220265

2. Golfarelli, M., Pirini, T., Rizzi, S.: Goal-based selection of visual representations for big data analytics. In: de Cesare, S., Frank, U. (eds.) ER 2017. LNCS, vol. 10651, pp. 47–57. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70625-2_5
3. Grüninger, M., Fox, M.S.: Methodology for the design and evaluation of ontologies. In: Proceedings of the IJCAI 1995 Workshop on Basic Ontological Issues in Knowledge Sharing (1995). http://www.eil.utoronto.ca/wp-content/uploads/enterprise-modelling/papers/gruninger-ijcai95.pdf
4. Guizzardi, G., Botti Benevides, A., Fonseca, C.M., Porello, D., Almeida, J.P.A., Sales, T.P.: UFO: unified foundational ontology. Appl. Ontol. **17**(1), 167–210 (2022). https://doi.org/10.3233/AO-210256
5. Guizzardi, G., Figueiredo, G., Hedblom, M.M., Poels, G.: Ontology-based model abstraction. In: Proceedings of the 13th International Conference on Research Challenges in Information Science (RCIS), pp. 1–13. IEEE (2019). https://doi.org/10.1109/RCIS.2019.8876971
6. Guizzardi, G., Fonseca, C.M., Almeida, J.P.A., Sales, T.P., et al.: Types and taxonomic structures in conceptual modeling: a novel ontological theory and engineering support. Data Knowl. Eng. **134**, 101891 (2021). https://doi.org/10.1016/j.datak.2021.101891
7. Guizzardi, G., Sales, T.P., Almeida, J.P.A., Poels, G.: Automated conceptual model clustering: a relator-centric approach. Softw. Syst. Model. **21**, 1363–1387 (2022). https://doi.org/10.1007/s10270-021-00919-5
8. Méndez, J., Alrabbaa, C., Koopmann, P., Langner, R., et al.: Evonne: a visual tool for explaining reasoning with OWL ontologies and supporting interactive debugging. Comput. Graph. Forum (2023). https://doi.org/10.1111/cgf.14730
9. Musen, M.A.: The Protégé project: a look back and a look forward. AI Matters **1**(4), 4–12 (2015). https://doi.org/10.1145/2757001.2757003
10. Romanenko, E., Calvanese, D., Guizzardi, G.: Abstracting ontology-driven conceptual models: Objects, aspects, events, and their parts. In: Guizzardi, R., Ralyté, J., Franch, X. (eds.) RCIS 2022. LNBIP, vol. 446, pp. 372–388. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-05760-1_22
11. Romanenko, E., Calvanese, D., Guizzardi, G.: Towards pragmatic explanations for domain ontologies. In: Corcho, O., Hollink, L., Kutz, O., Troquard, N., Ekaputra, F.J. (eds) EKAW 2022. LNAI, vol. 13514, pp. 201–208. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17105-5_15
12. Romanenko, E., Calvanese, D., Guizzardi, G.: What do users think about abstractions of ontology-driven conceptual models? In: Nurcan, S., Opdahl, A.L., Mouratidis, H., Tsohou, A. (eds) RCIS 2023. LNBIP, vol. 476, pp. 53–68. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33080-3_4
13. Sales, T.P., Barcelos, P.P.F., Fonseca, C.M., Valle Souza, I., et al.: A FAIR catalog of ontology-driven conceptual models. Data Knowl. Eng. **147**, 102210 (2023). https://doi.org/10.1016/j.datak.2023.102210
14. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. In: Proceedings of the 1996 IEEE Symposium on Visual Languages, pp. 336–343. IEEE Computer Society (1996)

15. Verdonck, M., Gailly, F.: Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling. In: Comyn-Wattiau, I., Tanaka, K., Song, I.-Y., Yamamoto, S., Saeki, M. (eds.) ER 2016. LNCS, vol. 9974, pp. 83–97. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46397-1_7
16. Weber, E., Van Bouwel, J., De Vreese, L.: Scientific Explanation. Springer Briefs in Philosophy. Springer, Dordrecht (2013). https://doi.org/10.1007/978-94-007-6446-0