# Modular Graphical Ontology Engineering Evaluated

Cogan Shimizu[1(✉)], Karl Hammar[2], and Pascal Hitzler[1]

[1] Data Semantics Lab, Kansas State University, Manhattan, USA
{coganmshimizu,phitzler}@ksu.edu
[2] Jönköping AI Lab, Jönköping University, Jönköping, Sweden
karl.hammar@ju.se

**Abstract.** Ontology engineering is traditionally a complex and time-consuming process, requiring an intimate knowledge of description logic and predicting non-local effects of different ontological commitments. Pattern-based modular ontology engineering, coupled with a graphical modeling paradigm, can help make ontology engineering accessible to modellers with limited ontology expertise. We have developed CoModIDE, the Comprehensive Modular Ontology IDE, to develop and explore such a modeling approach. In this paper we present an evaluation of the CoModIDE tool, with a set of 21 subjects carrying out some typical modeling tasks. Our findings indicate that using CoModIDE improves task completion rate and reduces task completion time, compared to using standard Protégé. Further, our subjects report higher System Usability Scale (SUS) evaluation scores for CoModIDE, than for Protégé. The subjects also report certain room for improvements in the CoModIDE tool – notably, these comments all concern comparatively shallow UI bugs or issues, rather than limitations inherent in the proposed modeling method itself. We deduce that our modeling approach is viable, and propose some consequences for ontology engineering tool development.

## 1 Introduction

Building a knowledge graph, as with any complex system, is an expensive endeavor, requiring extensive time and expertise. For many, the magnitude of resources required for building and maintaining a knowledge graph is untenable. Yet, knowledge graphs are still poised to be a significant disruptor in both the private and public sectors [17]. As such, lowering the barriers of entry is very important. More specifically, it will be necessary to increase the approachability of knowledge graph development best practices, thus reducing the need for dedicated expertise. Of course, we do not mean imply that *no* expertise is desirable, simply that a dedicated knowledge engineer may be out of reach for small firms or research groups. For this paper, we focus on the best practices according to the eXtreme design (XD) [4] and modular ontology modeling (MOM) [12] paradigms. To this point, we are interested in how tooling infrastructure can improve approachability. In the context of our chosen paradigms and focus on


A. Harth et al. (Eds.): ESWC 2020, LNCS 12123, pp. 20–35, 2020.
https://doi.org/10.1007/978-3-030-49461-2_2

tooling infrastructure, approachability may be proxied by *the amount of effort to produce correct and reasonable output*, where effort is a function of tool-user experience (UX) and time taken. Furthermore, by using tooling infrastructure to encapsulate best practices, it improves the maintainability and evolvability accordingly.

In particular, this paper investigates the use of a graphical modeling tool that encapsulates the pattern-driven philosophies of XD and MOM. To do so, we have developed CoModIDE (the *Comprehensive Modular Ontology IDE* – pronounced "commodity"), a plugin for the popular ontology editing platform, Protégé [16]. In order to show that CoModIDE improves approachability of knowledge graph development, we have formulated for the following hypotheses.

H1. When using CoModIDE, a user takes less time to produce correct and reasonable output, than when using Protege.
H2. A user will find CoModIDE to have a higher SUS score than when using Protege alone.

The remainder of this paper is organized as follows. Section 2 presents CoModIDE. Section 3 discusses related work on graphical modeling and ontology design pattern use and development. We present our experimental design in Sect. 4, our results in Sect. 5, and a discussion of those results and their implications in Sect. 6. Finally, Sect. 7 concludes the paper, and suggests possibilities for future research.

## 2 CoModIDE: A Comprehensive Modular Ontology IDE

### 2.1 Motivator: A Graphical and Modular Ontology Design Process

CoModIDE is intended to simplify ontology engineering for users who are not ontology experts. Our experience indicates that such non-experts rarely need or want to make use of the full set of language constructs that OWL 2 provides; instead, they typically, at least at the outset, want to model rather simple semantics. Such users (and, indeed also more advanced users) often prefer to do initial modeling in pair or group settings, and to do it graphically – whether that be on whiteboards, in vector drawing software, or even on paper. This further limits the modeling constructs to those that can be expressed somewhat intuitively using graphical notations (such that all involved participants, regardless of their ontology engineering skill level, can understand and contribute).

This initial design process typically iterates rapidly and fluidly, with the modeling task being broken down into individual problems of manageable complexity[1]; candidate solutions to these problem pieces being drawn up, analysed and discussed; a suitable solution selected and documented; and the next step

[1] We find that the size of such partial solutions typically fit on a medium-sized whiteboard; but whether this is a naturally manageable size for humans to operate with, or whether it is the result of constraints of or conditioning to the available tooling, i.e., the size of the whiteboards often mounted in conference rooms, we cannot say.

of the problem then tackled. Many times, the formalization of the developed solution into an OWL ontology is carried out after-the-fact, by a designated ontologist with extensive knowledge of both the language and applicable tooling. However, this comes at a cost, both in terms of hours expended, and in terms of the risk of incorrect interpretations of the previously drawn graphical representations (the OWL standard does not define a graphical notation syntax, so such representations are sometimes ambiguous).

The design process discussed above mirrors the principles of *eXtreme Design* (XD) [4]: working in pairs, breaking apart the modeling task into discrete problems, and iterating and refactoring as needed. XD also emphasizes the use of *Ontology Design Patterns* (ODPs) as solutions to frequently recurring modeling problems. Combining ODP usage with the graphical modeling process discussed above (specifically with the need to in an agile manner refactor and modify partial solutions) requires that the partial solutions (or *modules*) derived from ODPs are annotated, such that they can at a later time be isolated for study, modified, or replaced.

In summary it would be useful for our target user group if there were tooling available that supported 1) intuitive and agile graphical modeling, directly outputting OWL ontologies (avoiding the need for the aforementioned post-processing), and 2) reuse of ODPs to create and maintain ODP-based modules. Hence, CoModIDE.

### 2.2 Design and Features

The design criteria for CoModIDE, derived from the requirements discussed above, are as follows:

- CoModIDE should support visual-first ontology engineering, based on a graph representation of classes, properties, and datatypes. This graphical rendering of an ontology built using CoModIDE should be consistent across restarts, machines, and operating system or Protégé versions.
- CoModIDE should support the type of OWL 2 constructs that can be easily and intuitively understood when rendered as a schema diagram. To model more advanced constructs (unions and intersections in property domains or ranges, the property subsumption hierarchy, property chains, etc), the user can drop back into the standard Protégé tabs.
- CoModIDE should embed an ODP repository. Each included ODP should be free-standing and completely documented. There should be no external dependency on anything outside of the user's machine[2]. If the user wishes, they should be able to load a separately downloaded ODP repository, to replace or complement the built-in one.

[2] Our experience indicates that while our target users are generally enthusiastic about the idea of reusing design patterns, they are quickly turned off of the idea when they are faced with patterns that lack documentation or that exhibit link rot.

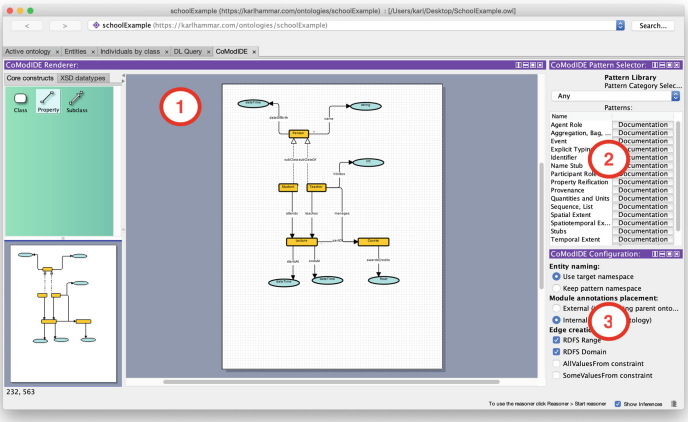


**Fig. 1.** CoModIDE User Interface featuring 1) the schema editor, 2) the pattern library, and 3) the configuration view.

– CoModIDE should support simple composition of ODPs; patterns should snap together like Lego blocks, ideally with potential connection points between the patterns lighting up while dragging compatible patterns. The resulting ontology modules should maintain their coherence and be treated like modules in a consistent manner across restarts, machines, etc. A pattern or ontology interface concept will need be developed to support this.

CoModIDE is developed as a plugin to the versatile and well-established Protégé ontology engineering environment. The plugin provides three Protégé views, and a tab that hosts these views (see Fig. 1). The *schema editor* view provides an a graphical overview of an ontology's structure, including the classes in the ontology, their subclass relations, and the object and datatype properties in the ontology that relate these classes to one another and to datatypes. All of these entities can be manipulated graphically through dragging and dropping. The *pattern library* view provides a set of built-in ontology design patterns, sourced from various projects and from the ODP community wiki[3]. A user can drag and drop design patterns from the pattern library onto the canvas to instantiate those patterns as modules in their ontology. The *configuration* view lets the user configure the behavior of the other CoModIDE views and their components. For a detailed description, we refer the reader to the video walkthrough on the CoModIDE webpage[4]. We also invite the reader to download and install CoModIDE themselves, from that same site.

When a pattern is dragged onto the canvas, the constructs in that pattern are copied into the ontology (optionally having their IRIs updated to correspond with the target ontology namespace), but they are also annotated using

[3] http://ontologydesignpatterns.org/.
[4] https://comodide.com.

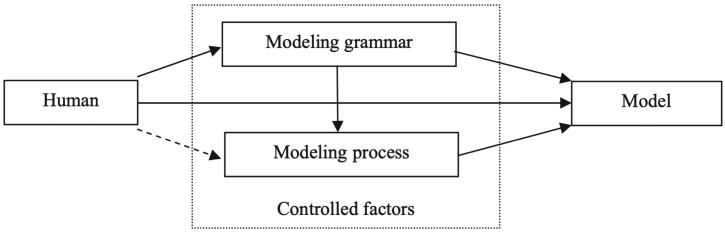


**Fig. 2.** Factors affecting conceptual modeling, from [9].

the OPLa vocabulary, to indicate 1) that they belong to a certain pattern-based module, and 2) what pattern that module implements. In this way module provenance is maintained, and modules can, provided that tool support exists (see Sect. 7) be manipulated (folded, unfolded, removed, annotated) as needed.

## 3 Related Work

**Graphical Conceptual Modeling** [9] proposes three factors (see Fig. 2) that influence the construction of a conceptual model, such as an ontology; namely, the *person* doing the modeling (both their experience and know-how, and their interpretation of the world, of the modeling task, and of model quality in general), the *modeling grammar* (primarily its expressive power/completeness and its clarity), and the *modeling process* (including both initial conceptualisation and subsequent formal model-making). Crucially, only the latter two factors can feasibly be controlled in academic studies. The related work discussed below tends to focus on one or the other of these factors, i.e., studying the characteristics of a modeling language *or* a modeling process. Our work on CoModIDE straddles this divide: employing graphical modeling techniques reduces the grammar available from standard OWL to those fragments of OWL that can be represented intuitively in graphical format; employing design patterns affects the modeling process.

Graphical modeling approaches to conceptual modeling have been extensively explored and evaluated in fields such as database modeling, software engineering, business process modeling, etc. Studying model grammar, [22] compares EER notation with an early UML-like notation from a comprehensibility point-of-view. This work observes that restrictions are easier to understand in a notation where they are displayed coupled to the types they apply to, rather than the relations they range over. [7] proposes a quality model for EER diagrams that can also extend to UML. Some of the quality criteria in this model, that are relevant in graphical modeling of OWL ontologies, include *minimality* (i.e., avoiding duplication of elements), *expressiveness* (i.e., displaying all of the required elements), and simplicity (displaying no more than the required elements).

[1] study the usability of UML, and report that users perceive UML class diagrams (closest in intended use to ontology visualizations) to be less easy-to-use

than other types of UML diagrams; in particular, relationship multiplicities (i.e., cardinalities) are considered frustrating by several of their subjects. UML displays such multiplicities by numeric notation on the end of connecting lines between classes. [13] analyses UML and argues that while it is a useful tool in a design phase, it is overly complex and as a consequence, suffers from redundancies, overlaps, and breaks in uniformity. [13] also cautions against using difficult-to-read and -interpret adornments on graphical models, as UML allows.

Various approaches have been developed for presenting ontologies visually and enabling their development through a graphical modeling interface, the most prominent of which is probably *VOWL*, the *Visual Notation for OWL Ontologies* [15], and its implementation viewer/editor WebVOWL [14,23]. VOWL employs a force-directed graph layout (reducing the number of crossing lines, increasing legibility) and explicitly focuses on usability for users less familiar with ontologies. As a consequence of this, VOWL renders certain structures in a way that, while not formally consistent with the underlying semantics, supports comprehensibility; for instance, datatype nodes and `owl:Thing` nodes are duplicated across the canvas, so that the model does not implode into a tight cluster around such often used nodes. It has been evaluated over several user studies with users ranging from laymen to more experienced ontologists, with results indicating good comprehensibility. CoModIDE has taken influence from VOWL, e.g., in how we render datatype nodes. However, in a collaborative editing environment in which the graphical layout of nodes and edges needs to remain consistent for all users, and relatively stable over time, we find the force-directed graph structure (which changes continuously as entities are added/removed) to be unsuitable.

For such collaborative modeling use cases, the commercial offering *Grafo*[5] offers a very attractive feature set, combining the usability of a VOWL-like notation with stable positioning, and collaborative editing features. Crucially, however, Grafo does not support pattern-based modular modeling, and as a web-hosted service, does not allow for customizations or plugins that would support such a modeling paradigm.

CoModIDE is partially based on the Protégé plugin *OWLAx*, as presented in [19]. This plugin supports one-way translation from graphical schema diagrams drawn by the user, into OWL ontology classes and properties; however, it does not render such constructs back into a graphical form. There is thus no way of continually maintaining and developing an ontology using only OWLAx. There is also no support for design pattern reuse in this tool.

**Ontology Design Patterns.** Ontology Design Patterns (ODPs) were introduced by Gangemi [8] and Blomqvist and Sandkuhl [2] in 2005, as a means of simplifying ontology development. ODPs are intended to guide non-expert users, by packaging best practices into reusable blocks of functionality, to be adapted and specialised by those users in individual ontology development projects. Presutti et al. [18] defines a typology of ODPs, including patterns for reasoning, naming, transformation, etc. The eXtreme Design methodology [4] describes how

[5] https://gra.fo.

ontology engineering projects can be broken down into discrete sub-tasks, to be solved by using ODPs. Prior studies indicate that the use of ODPs can lower the number of modeling errors and inconsistencies in ontologies, and that they are by the users perceived as useful and helpful [3,5].

Applying the XD method and ODPs requires the availability of both high-quality ODPs, and of tools and infrastructure that support ODP use. Recent work in this area, by the authors and others, includes *XDP*, a fork of the WebProtégé ontology editor [10]; the *OPLa* annotations vocabulary that models how ontology concepts can be grouped into modules, and the provenance of and interrelations between such modules, including to ODPs [11]; and the MODL library, a curated and specially documented collection of high-quality patterns for use in many domains [21]. CoModIDE draws influence from all of these works, and includes the MODL library as its default pattern library, using an OPLa-based representation of those patterns.

## 4 Research Method

Our experiment is comprised of four steps: a survey to collect subject background data (familiarity with ontology languages and tools), two modeling tasks, and a follow-up survey to collect information on the usability of both Protégé and CoModIDE. The tasks were designed to emulate a common ontology engineering process, where a conceptual design is developed and agreed upon by whiteboard prototyping, and a developer is then assigned to formalizing the resulting whiteboard schema diagram into an OWL ontology.

During each of the modeling tasks, participants are asked to generate a *reasonable* and *correct* OWL file for the provided schema diagram. In order to prevent a learning effect, the two tasks utilize two different schema diagrams. To prevent bias arising from differences in task complexity, counterbalancing was employed (such that half the users performed the first task with standard Protégé and the second task with CoModIDE, and half did the opposite). The correctness of the developed OWL files, and the time taken to complete each tasks, were recorded (the latter was however, for practical reasons, limited to 20 min per task).

The following sections provide a brief overview of each the steps. The source material for the entire experiment is available online[6].

**Introductory Tutorial.** As previously mentioned, our intent is to improve the approachability of ontology modeling by making it more accessible to those without expertise in knowledge engineering. As such, when recruiting our participants for this evaluation, we did not place any requirements on ontology modeling familiarity. However, to establish a shared baseline knowledge of foundational modeling concepts (such as one would assume participants would have in the situation we try to emulate, see above), we provided a 10 min tutorial

[6] http://urn.kb.se/resolve?urn=urn:nbn:se:hj:diva-47887.

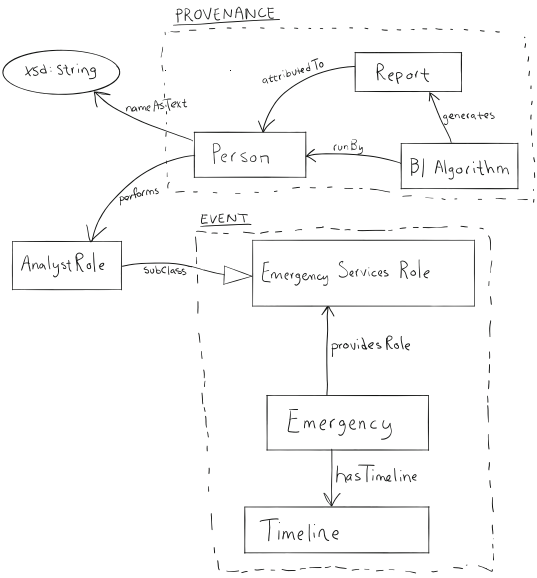


**Fig. 3.** Task A schema diagram

on ontologies, classes, properties, domains, and ranges. The slides used for this tutorial may be found online with the rest of the experiment's source materials.

***a priori* Survey.** The purpose of the *a priori* survey was to collect information relating to the participants base level familiarity with topics related to knowledge modeling, to be used as control variables in later analysis. We used a 5-point Likert scale for rating the accuracy of the following statements.

CV1. I have done ontology modeling before.
CV2. I am familiar with Ontology Design Patterns.
CV3. I am familiar with Manchester Syntax.
CV4. I am familiar with Description Logics.
CV5. I am familiar with Protégé.

Finally, we asked the participants to describe their relationship to the test leader, (e.g. student, colleague, same research lab, not familiar).

**Modeling Task A.** In Task A, participants were to develop an ontology to model how an analyst might generate reports about an ongoing emergency. The scenario identified two design patterns to use:

- **Provenance**: to track who made a report and how;
- **Event**: to capture the notion of an emergency.

Figure 3 shows how these patterns are instantiated and connected together. Overall the schema diagram contains seven concepts, one datatype, one subclass relation, one data property, and six object properties.

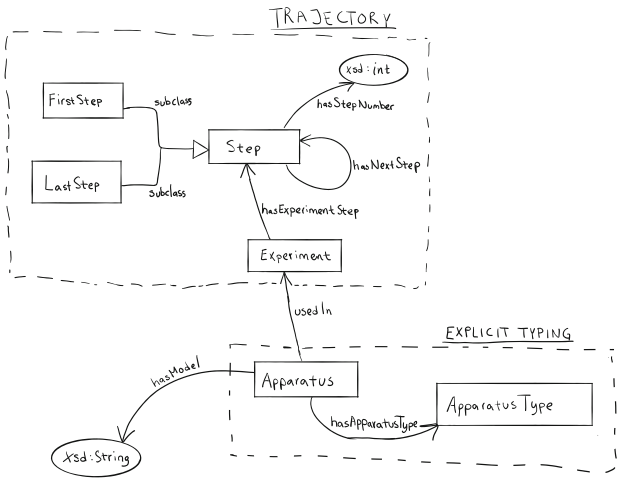


**Fig. 4.** Task B schema diagram

**Modeling Task B.** In Task B, participants were to develop an ontology to capture the steps of an experiment. The scenario identified two design patterns to use:

- **Trajectory**: to track the order of the steps;
- **Explicit Typing**: to easily model different types of apparatus.

Figure 4 shows how these patterns are instantiated and connected together. Overall, the schema diagram contains six concepts, two datatypes, two subclass relations, two data properties, and four object properties (one of which is a self-loop).

***a posteriori* Survey,** The *a posteriori* survey included the SUS evaluations for both Protégé and CoModIDE. The SUS is a very common "quick and dirty," yet reliable tool for measuring the usability of a system. It consists of ten questions, the answers to which are used to compute a total usability score of 0–100. Additional information on the SUS and its included questions can be found online.[7]

Additionally, we inquire about CoModIDE-specific features. These statements are also rated using a Likert scale. However, we do not use this data in our evaluation, except to inform our future work, as described in Sect. 7. Finally, we requested any free-text comments on CoModIDE's features.

## 5 Results

### 5.1 Participant Pool Composition

Of the 21 subjects, 12 reported some degree of familiarity with the authors, while 9 reported no such connection. In terms of self-reported ontology engineering familiarity, the responses are as detailed in Table 1. It should be observed that responses vary widely, with a relative standard deviation ($\sigma$/mean) of 43–67%.

[7] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html.

**Table 1.** Mean, standard deviation, relative standard deviation, and median responses to *a priori* statements

| | mean | $\sigma$ | relative $\sigma$ | median |
|---|---|---|---|---|
| CV1: I have done ontology modeling before | 3.05 | 1.75 | 57% | 3 |
| CV2: I am familiar with Ontology Design Patterns | 3.05 | 1.32 | 43% | 3 |
| CV3: I am familiar with Manchester Syntax | 2.33 | 1.56 | 67% | 1 |
| CV4: I am familiar with Description Logics | 2.81 | 1.33 | 47% | 3 |
| CV5: I am familiar with Protégé | 2.95 | 1.63 | 55% | 3 |

### 5.2 Metric Evaluation

We define our two metrics as follows:

- **Time Taken:** number of minutes, rounded to the nearest whole minute and capped at 20 min due to practical limitations, taken to complete a task;
- **Correctness** is a discrete measure that corresponds to the structural accuracy of the output. That is, 2 points were awarded to those structurally accurate OWL files, when accounting for URIs; 1 point for a borderline case (e.g one or two incorrect linkages, or missing a domain statement but including the range); and 0 points for any other output.

For these metrics, we generate simple statistics that describe the data, per modeling task. Tables 2a and 2b show the mean, standard deviation, and median for the Time Taken and Correctness of Output, respectively.

In addition, we examine the impact of our control variables (CV). This analysis is important, as it provides context for representation or bias in our data set. These are reported in Table 2c. CV1-CV5 correspond exactly to those questions asked during the *a priori* Survey, as described in Sect. 4. For each CV, we calculated the bivariate correlation between the sample data and the self-reported data in the survey. We believe that this is a reasonable measure of impact on effect, as our limited sample size is not amenable to partitioning. That is, the partitions (as based on responses in the *a priori* survey) could have been tested pair-wise for statistical significance. Unfortunately, the partitions would have been too small to conduct proper statistical testing. However, we do caution that correlation effects are strongly impacted by sample size.

We analyze the SUS scores in the same manner. Table 4 presents the mean, standard deviation, and median of the data set. The maximum score while using the scale is a 100. Table 2d presents our observed correlations with our control variables.

Finally, we compare the each metric for one tool against the other. That is, we want to know if our results are statistically significant—that as the statistics suggest in Table 2, CoModIDE does indeed perform better for both metrics and the SUS evaluation. To do so, we calculate the probability $p$ that the samples from each dataset come from different underlying distributions. A common tool, and the tool we employ here, is the Paired (two-tailed) T-Test—noting that it

**Table 2.** Summary of statistics comparing Protege and CoModIDE.

| | mean | $\sigma$ | median |
|---|---|---|---|
| Protégé | 17.44 | 3.67 | 20.0 |
| CoModIDE | 13.94 | 4.22 | 13.5 |

(a) Mean, standard deviation, and median *time taken* to complete each modeling task.

| | mean | $\sigma$ | median |
|---|---|---|---|
| Protégé | 0.50 | 0.71 | 0.0 |
| CoModIDE | 1.33 | 0.77 | 1.5 |

(b) Mean, standard deviation, and median *correctness of output* for each modeling task.

| | CV1 | CV2 | CV3 | CV4 | CV5 |
|---|---|---|---|---|---|
| TT (P) | -0.61 | -0.18 | -0.38 | -0.58 | -0.62 |
| Cor. (P) | 0.50 | 0.20 | 0.35 | 0.51 | 0.35 |
| TT (C) | 0.02 | -0.34 | -0.28 | -0.06 | 0.01 |
| Cor. (C) | -0.30 | 0.00 | -0.12 | -0.33 | -0.30 |

(c) Correlations control variables (CV) on the Time Taken (TT) and Correctness of Output (Cor.) for both tools Protégé (P) and CoModIDE (C).

| | CV1 | CV2 | CV3 | CV4 | CV5 |
|---|---|---|---|---|---|
| SUS (P) | 0.70 | 0.52 | 0.64 | 0.73 | 0.64 |
| SUS (C) | -0.34 | -0.05 | -0.08 | -0.29 | -0.39 |

(d) Correlations with control variables (CV) on the SUS scores for both tools Protégé (P) and CoModIDE (C).

is reasonable to assume that the underlying data are normally distributed, as well as powerful tool for analyzing datasets of limited size. The threshold for indicating confidence that the difference is significant is generally taken to be $p < 0.05$. Table 3 summarizes these results.

### 5.3 Free-Text Responses

18 of the 21 subjects opted to leave free-text comments. We applied fragment-based qualitative coding and analysis on these comments. I.e., we split the comments apart per the line breaks entered by the subjects, we read through the fragments and generated a simple category scheme, and we then re-read the

**Table 3.** Significance of results.

| Time taken | Correctness | SUS evaluation |
|---|---|---|
| $p \approx 0.025 < 0.05$ | $p \approx 0.009 < 0.01$ | $p \approx 0.0003 < 0.001$ |

**Table 4.** Mean, standard deviation, and median SUS score for each tool. The maximum score is 100.

| | mean | $\sigma$ | median |
|---|---|---|---|
| Protégé | 36.67 | 22.11 | 35.00 |
| CoModIDE | 73.33 | 16.80 | 76.25 |

fragments and applied these categories to the fragments (allowing at most one category per fragment) [6,20]. The subjects left between 1–6 fragments each for a total of 49 fragments for analysis, of which 37 were coded, as detailed in Table 5.

Of the 18 participants who left comments, 3 left comments containing no codable fragments; these either commented upon the subjects own performance in the experiment, which is covered in the aforementioned completion metrics, or were simple statements of fact (e.g., *"In order to connect two classes I drew a connecting line"*).

## 6 Discussion

**Participant Pool Composition.** The data indicates no correlation (bivariate correlation $< \pm 0.1$) between the subjects' reported author familiarity, and their reported SUS scores, such as would have been the case if the subjects who knew the authors were biased. The high relative standard deviation for a priori knowledge level responses indicates that our subjects are rather diverse in their skill levels – i.e., they do not consist exclusively of the limited-experience class of users that we hope CoModIDE will ultimately support. As discussed below, this variation is in fact fortunate as it allows us to compare the performance of more or less experienced users.

**Metric Evaluation.** Before we can determine if our results confirm H1 and H2 (replicated in Fig. 5 from Sect. 1), we must first examine the correlations between our results and the control variables gathered in the *a priori* survey.

**Table 5.** Free text comment fragments per category

| Code | Fragment # |
|---|---|
| Graph layout | 4 |
| Dragging & dropping | 6 |
| Feature requests | 5 |
| Bugs | 8 |
| Modeling problems | 5 |
| Value/preference statements | 9 |

H1. When using CoModIDE, a user takes less time to produce correct and reasonable output, than when using Protege.
H2. A user will find CoModIDE to have a higher SUS score than when using Protege alone.

**Fig. 5.** Our examined hypotheses, restated from Sect. 1.

In this context, we find it reasonable to use these thresholds for a correlation $|r|$: 0–0.19 very weak, 0.20–0.39 weak, 0.40–0.59 moderate, 0.60–0.79 strong, 0.80–1.00 very strong.

As shown in Table 2c, the metric *time taken* when using Protégé is negatively correlated with each CV. The *correctness* metric is positively correlated with each CV. This is unsurprising and reasonable; it indicates that familiarity with the ontology modeling, related concepts, and Protégé improves (shortens) time taken to complete a modeling task and improves the correctness of the output. However, for the metrics pertaining to CoModIDE, there are only very weak and three weak correlations with the CVs. We may construe this to mean that performance when using CoModIDE, with respect to our metrics, is largely agnostic to our control variables.

To confirm H1, we look at the metrics separately. *Time taken* is reported better for CoModIDE in both mean and median. When comparing the underlying data, we achieve $p \approx 0.025 < 0.05$. Next, in comparing the *correctness* metric from Table 2b, CoModIDE again outperforms Protégé in both mean and median. When comparing the underlying data, we achieve a statistical significance of $p \approx 0.009 < 0.01$. With these together, we reject the null hypothesis and confirm H1.

This is particularly interesting; given the above analysis of CV correlations where we see no (or very weak) correlations between prior ontology modeling familiarity and CoModIDE modeling results, and the confirmation of H1, that CoModIDE users perform better than Protégé users, we have a strong indicator that we have in fact achieved increased approachability.

When comparing the SUS score evaluations, we see that the usability of Protégé is strongly influenced by familiarity with ontology modeling and familiarity with Protégé itself. The magnitude of the correlation suggests that newcomers to Protege do not find it very usable. CoModIDE, on the other hand is weakly, negatively correlated along the CV. This suggests that switching to a graphical modeling paradigm may take some adjusting.

However, we still see that the SUS scores for CoModIDE have a greater mean, tighter $\sigma$, and greater median, achieving a very strong statistical significance $p \approx 0.0003 < 0.001$. Thus, we may reject the null hypothesis and confirm H2.

As such, by confirming H1 and H2, we may say that CoModIDE, via graphical ontology modeling, does indeed improve the approachability of knowledge graph development, especially for those not familiar with ontology modeling—with respect to our participant pool. However, we suspect that our results are generalizable, due to the strength of the statistical significance (Table 3) and participant pool composition (Sect. 5.1).

**Free-Text Responses.** The fragments summarized in Table 5 paints a quite coherent picture of the subjects' perceived advantages and shortcomings of CoModIDE, as follows:

- *Graph layout:* The layout of the included MODL patterns, when dropped on the canvas, is too cramped and several classes or properties overlap, which reduces tooling usability.
- *Dragging and dropping:* Dragging classes was hit-and-miss; this often caused users to create new properties between classes, not move them.
- *Feature requests:* Pressing the "enter" key should accept and close the entity renaming window. Zooming is requested, and an auto-layout button.
- *Bugs:* Entity renaming is buggy when entities with similar names exist.
- *Modeling problems:* Self-links/loops cannot easily be modeled.
- *Value/preference statements:* Users really appreciate the graphical modeling paradigm offered, e.g., *"Mich easier to use the GUI to develop ontologies"*, *"Moreover, I find this system to be way more intuitive than Protégé"*, *"comodide was intuitive to learn and use, despite never working with it before."*

We note that the there is a near-unanimous consensus among the subjects that graphical modeling is intuitive and helpful. When users are critical of the CoModIDE software, these criticisms are typically aimed at specific and quite shallow bugs or UI features that are lacking. The only consistent criticism of the modeling method itself relates to the difficulty in constructing self-links (i.e., properties that have the same class as domain and range).

## 7 Conclusion

To conclude, we have shown how the CoModIDE tool allows ontology engineers, irrespective of previous knowledge level, to develop ontologies more correctly and more quickly, than by using standard Protégé; that CoModIDE has a higher usability (SUS score) than standard Protégé; and that the CoModIDE issues that concern users primarily derive from shallow bugs as opposed to methodological or modeling issues. Taken together, this implies that the modular graphical ontology engineering paradigm is a viable way to improving the approachability of ontology engineering.

**Future Work.** CoModIDE is under active development and is not yet feature-complete. Specifically, during the spring of 2020 we will implement the following features:

- Wrapping instantiated modules (e.g., in dashed-line boxes) to indicate cohesion and to allow module folding/unfolding.
- An interface feature, allowing design patterns to express how they can be connected to one another; and adding support for this to the canvas, lighting up potential connection points as the user drags a pattern.
- Support for custom pattern libraries; and vocabulary specifications indicating hos pattern libraries should be annotated to be useful with CoModIDE.

In developing CoModIDE we have come across several trade-offs between usability and expressiveness, as discussed in Sect. 2. We intend to follow these threads, using CoModIDE as test bed, to study more precisely how the need for graphical representability affects the use of modeling constructs and/or ontology engineering methods. For instance, we initially assumed that a graphical modeling paradigm would help users verify the correctness of their designs; but the answers to our *a posteriori* survey questions on this matter proved inconclusive.

**Acknowledgement.** Cogan Shimizu and Pascal Hitzler acknowledge partial support from the following financial assistance award 70NANB19H094 from U.S. Department of Commerce, National Institute of Standards and Technology and partial support from the National Science Foundation under Grant No. 1936677.

## References

1. Agarwal, R., Sinha, A.P.: Object-oriented modeling with uml: a study of developers' perceptions. Commun. ACM **46**(9), 248–256 (2003)
2. Blomqvist, E., Sandkuhl, K.: Patterns in ontology engineering: classification of ontology patterns. In: Proceedings of the 7th International Conference on Enterprise Information Systems, pp. 413–416 (2005)
3. Blomqvist, E., Gangemi, A., Presutti, V.: Experiments on pattern-based ontology design. In: Gil, Y., Noy, N. (eds.) K-CAP 2009: Proceedings of the Fifth International Conference on Knowledge Capture, pp. 41–48. ACM (2009)
4. Blomqvist, E., Hammar, K., Presutti, V.: Engineering ontologies with patterns - the eXtreme design methodology. In: Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.) Ontology Engineering with Ontology Design Patterns: Foundations and Applications, Studies on the Semantic Web, chap. 2, vol. 25, pp. 23–50. IOS Press (2016)
5. Blomqvist, E., Presutti, V., Daga, E., Gangemi, A.: Experimenting with eXtreme design. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS (LNAI), vol. 6317, pp. 120–134. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16438-5_9
6. Burnard, P.: A method of analysing interview transcripts in qualitative research. Nurse Educ. Today **11**(6), 461–466 (1991)
7. Cherfi, S.S.-S., Akoka, J., Comyn-Wattiau, I.: Conceptual modeling quality - from EER to UML schemas evaluation. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 414–428. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45816-6_38
8. Gangemi, A.: Ontology design patterns for semantic web content. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 262–276. Springer, Heidelberg (2005). https://doi.org/10.1007/11574620_21
9. Hadar, I., Soffer, P.: Variations in conceptual modeling: classification and ontological analysis. J. Assoc. Inf. Syst. **7**(8), 20 (2006)
10. Hammar, K.: Ontology design patterns in WebProtégé. In: Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Betlehem, USA, 11 October 2015 (2015). No. 1486 in CEUR-WS

11. Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A.A., Presutti, V.: Towards a simple but useful ontology design pattern representation language. In: Blomqvist, E., Corcho, Ó., Horridge, M., Carral, D., Hoekstra, R. (eds.) Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) Co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, 21 October 2017 (2017). No. 2043 in CEUR Workshop Proceedings
12. Hitzler, P., Krisnadhi, A.: A tutorial on modular ontology modeling with ontology design patterns: the cooking recipes ontology. CoRR abs/1808.08433 (2018), http://arxiv.org/abs/1808.08433
13. Krogstie, J.: Evaluating UML using a generic quality framework. In: UML and the Unified Process, pp. 1–22. IGI Global (2003)
14. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: web-based visualization of ontologies. In: Lambrix, P., et al. (eds.) EKAW 2014. LNCS (LNAI), vol. 8982, pp. 154–158. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17966-7_21
15. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with vowl. Semant. Web **7**(4), 399–419 (2016)
16. Musen, M.A.: The Protégé project: a look back and a look forward. AI Matters **1**(4), 4–12 (2015)
17. Noy, N.F., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: lessons and challenges. Commun. ACM **62**(8), 36–43 (2019). https://doi.org/10.1145/3331166
18. Presutti, V., et al.: D2.5.1: a library of ontology design patterns: reusable solutions for collaborative design of networked ontologies. Technical report, NeOn Project (2007)
19. Sarker, M.K., Krisnadhi, A.A., Hitzler, P.: OWLAx: A protégé plugin to support ontology axiomatization through diagramming. In: Kawamura, T., Paulheim, H. (eds.) Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, 19 October 2016. CEUR Workshop Proceedings, vol. 1690 (2016)
20. Seaman, C.B.: Qualitative methods. In: Shull, F., Singer, J., Sjøberg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering, pp. 35–62. Springer, London (2008). https://doi.org/10.1007/978-1-84800-044-5_2
21. Shimizu, C., Hirt, Q., Hitzler, P.: MODL: a modular ontology design library. In: Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019). CEUR Workshop Proceedings, vol. 2459, pp. 47–58 (2019)
22. Shoval, P., Frumermann, I.: Oo and eer conceptual schemas: a comparison of user comprehension. J. Database Manag. (JDM) **5**(4), 28–38 (1994)
23. Wiens, V., Lohmann, S., Auer, S.: WebVOWL editor: device-independent visual ontology modeling. In: Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks. CEUR Workshop Proceedings, vol. 2180 (2018)