

LLM-Based Modeling Assistance for Textual Ontology-Driven Conceptual Modeling

Matheus L. Coutinho^{1,*}, João Paulo A. Almeida¹ and Giancarlo Guizzardi²

¹*Ontology & Conceptual Modeling Research Group (NEMO), Federal University of Espírito Santo (UFES), Brazil*

²*Semantics, Cybersecurity & Services (SCS), University of Twente, The Netherlands*

Abstract

Large Language Models (LLMs) have already shown potentially significant capabilities in assisting users with writing and coding tasks. In this paper, we explore how LLM-based assistance can be leveraged in a modeling environment for textual Ontology-Driven Conceptual Modeling. We integrate the UFO-based textual language ‘Tonto’ with an LLM-powered assistant. We employ detailed UFO-based ‘guidance’ texts which are included by the modeling environment automatically in the context of user prompts along with the current ontology coding artifacts. The tool can take actions such as creating files, changing code, invoking Tonto syntax verification, while still maintaining the modeler in the loop. Our initial exploration shows that a number of modeling tasks can potentially be automated (such as suggesting new elements, summarizing the model, checking consistency of usage of UFO concepts, model fixing, etc.). The tool is proposed as a testbed for empirical user studies.

Keywords

Ontology-Driven Conceptual Modeling, Large Language Models, Textual Modeling

1. Introduction

For more than two decades, the Unified Foundational Ontology (UFO) [1, 2] has been used in a number of tasks in Ontology-Driven Conceptual Modeling. A key result of this research agenda has been the OntoUML profile, which brings to UML class diagrams important ontological distinctions based on UFO. OntoUML employs UFO’s concepts and axioms to enable the creation of high-quality reference ontologies and is supported by a suite of advanced tools [3] for model editing, verification, simulation, transformation, database schema generation, and anti-pattern detection.

Despite the benefits of using UML class diagrams as a basis, which allows OntoUML to leverage UML’s well-known visual notation, there are also drawbacks associated with graphical modeling. These include the manual effort of diagram layout, scalability issues with large diagrams, and the inability to use mature text-based tools for tasks like version control, comparison, merging, and auto-completion. Because of this, we have been exploring the use of textual notations for UFO-based conceptual modeling, which has led to the development of the Tonto (Textual Ontologies) [4] language and tooling.

Tonto inherits its core concepts from UFO in line with OntoUML. The language is accompanied by a fully-featured VS Code extension that performs syntax highlighting, auto-completion and real-time syntactic verification against UFO rules. The Tonto extension is also integrated into the OntoUML server [3], and, therefore, can benefit from full interoperability with the OntoUML ecosystem.

Tonto’s textual format facilitates its integration with tools based on Large Language Models (LLMs) [5, 6]. We explore this synergy to provide advanced modeling support for UFO-based ontologies. We combine Tonto’s rigorous constraints, aimed at ensuring ontological robustness with the flexibility of LLM-based assistance. The use of UFO concepts in Tonto allow us to provide rich context in prompts, which guide automated task execution. We leverage the fact that LLMs incorporate significant empirical linguistic knowledge acquired during their massive pre-training phase.

ER2025: Companion Proceedings of the 44th International Conference on Conceptual Modeling: Industrial Track, ER Forum, 8th SCME, Doctoral Consortium, Tutorials, Project Exhibitions, Posters and Demos, October 20-23, 2025, Poitiers, France

*Corresponding author.

✉ matheuslenke@gmail.com (M. L. Coutinho); jpalmeida@ieee.org (J. P. A. Almeida); g.guizzardi@utwente.nl (G. Guizzardi)

🆔 0009-0003-1709-6336 (M. L. Coutinho); 0000-0002-9819-3781 (J. P. A. Almeida); 0000-0002-3452-553X (G. Guizzardi)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Our ultimate goal is to understand what kinds of ontology-driven conceptual modeling tasks can be supported with pre-trained LLMs and (zero-shot or few-shot) prompting strategies. A common barrier in assessing this sort of support empirically is the lack of usable tools for experimentation. That is the object of this paper. We combine the existing Tonto extension with the AI Cursor tool (<https://cursor.com>). Cursor integrates an extensible text-based coding environment with a variety of LLMs in a seamless way. It allows us to define ‘guidance’ texts which are provided to the LLMs in tandem with the current coding artifacts. These texts provide instructions to perform specific tasks; the tool (in its ‘agent mode’) can take actions such as creating files, changing code, etc., while still maintaining the modeler in the loop. (Similar functionality has recently been implemented in VS Code natively, and hence, can also be explored with the approach discussed in this paper.)

We have defined a number of detailed UFO-based ‘guidance’ texts which are used as part of the context of prompts for the LLMs in the following tasks: (i) enhancing an existing ontology by creating new elements and relations (while adhering to UFO constraints); (ii) checking the consistency of the ontology terminology; (iii) checking the usage of UFO concepts; (iv) understanding a package or the entire ontology in a summarized explanation in natural language; (v) creating documentation for existing elements in an ontology, labels and descriptions; (vi) adding multi-lingual terminology and descriptions. Our initial exploration with the tool shows that it is capable of automating these tasks for Tonto models. We plan to employ the tool as a testbed for empirical user studies.

This paper is further structured as follows: Section 2 describes the solution we employ, briefly positioning Tonto and its editor in an overall architecture. Section 3 presents one of the examples used for demonstration, and Section 4 provides some conclusions and outlines plans for future work.

2. Solution

Tonto The Tonto language was designed to be both human-readable and expressive [4]. Its syntax is intentionally familiar to users of mainstream object-oriented languages. Tonto introduces keywords to reflect the taxonomy of types in UFO. For example, there are keywords for kind, subkind, phase, role, category, etc., reflecting the metaproperties of rigidity and sortality as well as other ontological distinctions. There are also keywords to enable the reification of relations and aspects (`relator`, `mode`, `quality`), event types (`event`) and to provide specialized semantics for associations (object participation in events, aspect inheritance, relator mediation, whole-part relations, etc.) [1].

Architectural Overview Figure 1 illustrates the basic workflow of Cursor’s ‘agent mode’ as used in this project in integration with the Tonto extension. An agent is an artificial entity that perceives its environment, makes decisions, and takes actions based on those perceptions in order to achieve specific goals. Cursor implements an Agent that is able to execute a series of activities in the Code Workspace, and it leverages an underlying Large Language Model (LLM) to operate within the code workspace, where it can execute a series of tasks such as file access, folder navigation, and semantic search to achieve specific modeling goals. The process begins when the user makes a request in the Cursor agent chat. In response, Cursor prepares the initial input for the selected LLM. This input includes the main system prompt, any custom rules defined by the user, and the actual content of any referenced Tonto files. The agent then begins a step-by-step loop to complete the request. It invokes various actions such as reading from files, writing new code, or running Tonto syntax verification. The result of each action is used to decide the next steps. To ensure transparency, the user can see this entire process—including the agent’s reasoning and tool usage—in the chat window. This interactive loop continues until the agent concludes the task is finished or a technical limit is reached (such as the maximum LLM token count). At the end of each cycle, the agent presents its proposed changes for user validation. The user retains oversight by accepting or rejecting these modifications, thereby ensuring the model evolves in alignment with their preferences. (Not unlike Cursor, VS Code also leverages an ‘agent mode’, with a similar architecture.)

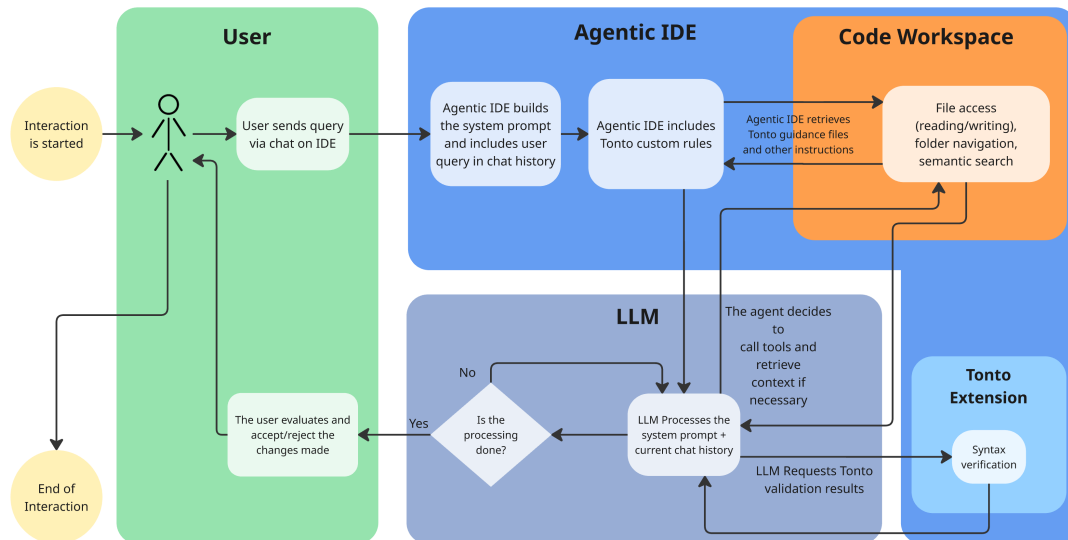


Figure 1: Architecture showing how a user query is processed by the Agent mode in Cursor

UFO-Based Task Guidance Rules are provided as markdown files to help the LLM understand and work with Tonto. The following guidance files were defined (and are publicly available in the project repository¹): (i) **Tonto Guidance**: This is the core instruction set. It provides a concise orientation for the LLM (establishing 'chains of thoughts'), covering the fundamentals of the Tonto language, its project structure, and its key capabilities. (ii) **LLM Guidance**: This document acts as a 'router'. It gives the LLM a general workflow and tells it how to use the other more specialized guidance files to handle specific user requests. (iii) **Enhancing Guidance**: This guide is for building and extending ontologies. It instructs the LLM on how to add new classes, attributes, relations, and generalization sets while maintaining the model's ontological consistency with UFO's axioms. (iv) **Terminology Guidance**: This document helps the LLM act as a vocabulary expert. It guides the analysis of the names used for classes, attributes, and relations, allowing the LLM to provide feedback for improving the ontology's clarity and precision. (v) **Understanding Guidance**: This guide helps the LLM summarize a model in natural language. It is used to generate descriptions of a single package or an entire project to help users quickly understand its contents. UFO notions are used, e.g., directing the LLM to prioritize ultimate sortals (kinds) [1]. (vi) **Documentation Guidance**: This document provides guidelines for generating high-quality documentation for elements in a Tonto ontology. It pushes the LLM to describe an element and also to explain the underlying ontological choices based on UFO principles.

3. Demonstration

In this section, we illustrate the use of the UFO-based guidance files. Given an existing ontology about the university domain, as shown in Figure 2, we have prompted the LLM agent as follows: "Your task is to help me check the terminology of my ontology and fix semantic errors. Get the errors from @Linter Errors. Take a look at all packages." We ran this test under two conditions: with our custom guidance texts, and without them, in order to establish a comparative baseline. The initial model uses the unspecific `class` keyword (as shown in the screenshot shaded in a light red background). Further, it lacks documentation in the form of labels and descriptions. The complete original and automatically modified specifications are available on the aforementioned project repository.

Without guidance, the agent struggled to produce a valid model, and it did not use the best practices from Tonto (this baseline output is also available in the project repository). The output contained significant errors, such as: incorrect use of Tonto's grammar and keywords; failure to apply appropriate

¹<https://github.com/matheuslenke/tonto-llm>

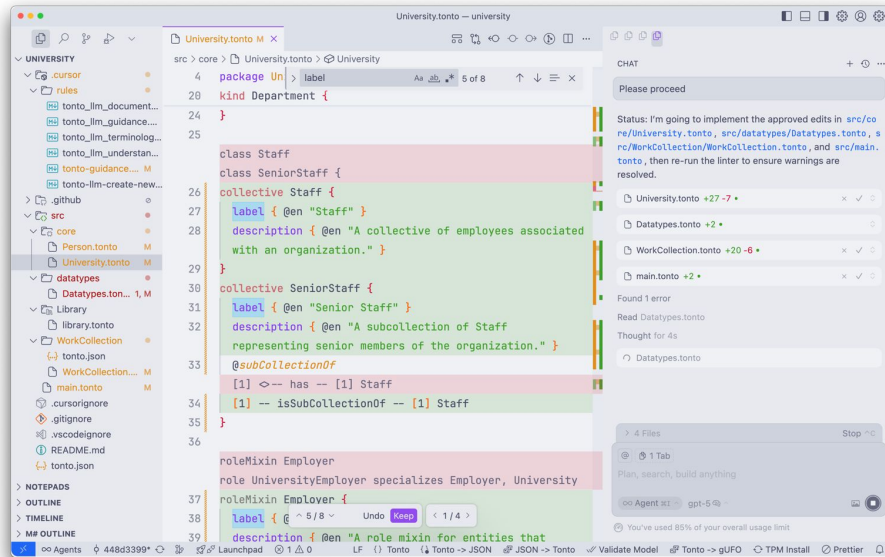


Figure 2: Cursor IDE where user prompts Tonto agent to “check terminology” and “fix semantic errors”.

UFO concepts, syntactically incorrect specialization relationships, and labels and descriptions as code comments instead of using the proper ‘label’ and ‘description’ syntax.

In stark contrast, when the same prompt was provided in the environment that was customized with the Tonto guidance files, the agent generated a significantly more robust and ontologically sound model. It correctly leveraged a wide range of UFO concepts (that correspond to Tonto keywords) and successfully implemented complex structures, such as the powertype pattern, demonstrating clear adherence to the well-founded modeling guidelines. In the screenshot shown in Figure 2, the generic `class` keyword was properly replaced by the `collective` keyword; the generic aggregation relation was properly replaced by the `subCollectionOf` relation. Space constraints prevent us from describing other examples here, covering the various supported tasks. Instructions for download of the guidance files and configuration of Cursor for use of Tonto are available in the project repository along with example prompts for other tasks.

4. Conclusions

There is a plethora of research addressing the use LLMs for different aspects of ontology engineering and conceptual modeling, as evidenced by a recent survey [7] and a discussion paper [8]. For example, LLMs have been employed to to classify domain entities using top-level categories [9], generate SWRL rules [10], support ontology development from datasets [11], etc. Here, our focus is on LLM assistance for tasks in ontology capture and formalization [12]. We employ production-ready tools and leverage foundational ontology distinctions both in guidance texts and in the language grammar verification.

This work is currently a proof of concept. While the initial results are promising, the LLM integration is a prototype that requires further development and rigorous evaluation to measure its effectiveness. Our immediate future work will focus on conducting controlled experiments to rigorously assess the impact of the LLM-powered assistant. This study will be designed to measure not only the effect of using an LLM but also the specific value added by our UFO-based guidance files. Participants will be asked to perform ontology modeling tasks in Tonto under three different conditions: (i) with no AI support, (ii) with a baseline LLM model, and (iii) with the guided assistant. We’ll measure success across three key areas: efficiency, by measuring task completion time; model quality, by evaluating the number of ontological errors and anti-pattern occurrences; and, finally, user satisfaction, measured using standard usability surveys to gauge the user experience. We also intend to assess whether possible

benefits persist in projects of different levels of complexity and involving users with different profiles.

Finally, we plan to explore more sophisticated prompt engineering techniques and expand the range of supported tasks. A long-term goal is to leverage LLMs to create natural language interfaces for querying and manipulating ontologies, making them more accessible to a wider audience of domain experts who may not be familiar with formal modeling languages. One challenge we anticipate is domain sensitivity. In particular, we expect the agent to struggle with niche domains, which are underrepresented in pre-training. Although we have observed good initial results in this exploratory phase with the models Claude 3.7 Sonnet, Gemini 2.5 Pro, and OpenAI o3, an indepth study should assess their particular merits in these tasks.

Acknowledgments

This research is funded in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, FAPES (1022/2022) and CNPq (443130/2023-0, 313412/2023-5).

Declaration on Generative AI

During the preparation of this work, the authors have employed Gemini 2.5 Pro in refining the guidance texts as well as in grammar and spelling check for the manuscript. The authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

- [1] G. Guizzardi, Ontological foundations for structural conceptual models, Centre for Telematics and Information Technology, Enschede, The Netherlands, 2005.
- [2] G. Guizzardi, et al., UFO: Unified Foundational Ontology, *Applied Ontology* 17 (2022) 167–210.
- [3] C. M. Fonseca, et al., Ontology-driven conceptual modeling as a service, in: *Proc. Joint Ontology Workshops 2021, CEUR Workshop Proceedings*, 2021. URL: <http://ceur-ws.org/Vol-2969/>.
- [4] M. L. Coutinho, J. P. A. Almeida, T. P. Sales, G. Guizzardi, A Textual Syntax and Toolset for Well-Founded Ontologies, in: *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2024.
- [5] C. Raffel, et al., Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, 2023. URL: <http://arxiv.org/abs/1910.10683>.
- [6] H. Touvron, et al., LLaMA: Open and Efficient Foundation Language Models, 2023. URL: <http://arxiv.org/abs/2302.13971>, arXiv:2302.13971 [cs].
- [7] D. Garijo, et al., LLMs for Ontology Engineering: A landscape of Tasks and Benchmarking challenges, in: *ISWC 2024 Special Session on Harmonising Generative AI and Semantic Web Technologies*, 2024.
- [8] V.C.Storey et al., Large language models for conceptual modeling: Assessment and application potential, *Data & Knowledge Engineering* 160 (2025) 102480. doi:10.1016/j.datak.2025.102480.
- [9] A. Lopes, et al., How to classify domain entities into top-level ontology concepts using large language models: A study across multiple labels, resources, and languages, *Applied Ontology* (2024) 1–29. doi:10.3233/AO-240032.
- [10] A. Soularidis, K. Kotis, M. Lamolle, Z. Mejdoul, G. Lortal, G. Vouros, LLM-Assisted Generation of SWRL Rules from Natural Language, in: *2024 International Conference on AI x Data and Knowledge Engineering (AIXDKE)*, IEEE, Tokyo, Japan, 2024, pp. 7–12.
- [11] M. Val-Calvo, et al., Ontogenix: Leveraging large language models for enhanced ontology engineering from datasets, *Information Processing & Management* 62 (2025) 104042. doi:10.1016/j.ipm.2024.104042.
- [12] R. Falbo, SABiO: Systematic approach for building ontologies, in: *Proc. 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering*, volume 1301 of *CEUR Workshop Proceedings*, 2014. URL: <https://ceur-ws.org/Vol-1301>.