An Architecture for Integrating Large Language Models into Metamodeling Platforms:

The Example of MM-AR

Gunakar Challa[1,*], Aya Gartini[1], Fabian Muff[1] and Hans-Georg Fill[1]

[1]University of Fribourg, Boulevard de Pérolles 90, 1700 Fribourg, Switzerland.

Abstract

The large-scale adoption of large language models for the integration of generative artificial intelligence capabilities is occurring across several domains. This also applies to the domain of conceptual modeling, where a number of approaches are currently being investigated for the creation and interpretation of models utilizing this technology. However, a significant number of these approaches are currently limited to a specific modeling language. Accordingly, the current paper proposes an architecture on the level of metamodeling platforms to facilitate the integration of large language models into modeling editors for triggering actions on arbitrary types of models. We describe the underlying concept and report on a first prototypical implementation of the approach for the web-based MM-AR metamodeling platform.

Keywords

Large Language Models, OpenAI, Enterprise Modeling, MM-AR

1. Introduction

LLMs can be useful in conceptual modeling, where experiments have shown the enormous potential of large language models such as ChatGPT[1] to support modeling tasks [1]. This can include the creation, modification, or analysis of conceptual models or even metamodels. However, as shown in [2], current LLMs are not yet capable of interpreting complex model data such as metamodels or model instances that must follow a predefined structure. Therefore, the integration of descriptive prompts to create or modify conceptual models based on complex metamodels is not yet possible. What is missing is an approach that is capable of creating and modifying conceptual models based on predefined metamodels, without the need for a large language model to understand the complex and generic structure of different modeling languages.

Therefore, in this paper we present a new approach for integrating LLMs into metamodeling platforms in a generic way. We will use the MM-AR metamodeling platform as an example [3] to

ER2024: Companion Proceedings of the 43rd International Conference on Conceptual Modeling: ER Forum, Special Topics, Posters and Demos, October 28-31, 2024, Pittsburgh, Pennsylvania, USA

*Corresponding author.

gunakar.challa@unifr.ch (G. Challa); aya.gartini@unifr.ch (A. Gartini); fabian.muff@unifr.ch (F. Muff); hans-georg.fill@unifr.ch (H. Fill)

http://www.unifr.ch/inf/digits (F. Muff); http://www.unifr.ch/inf/digits (H. Fill)

0009-0007-2697-0215 (G. Challa); 0009-0004-3731-5521 (A. Gartini); 0000-0002-7283-6603 (F. Muff); 0000-0001-5076-5341 (H. Fill)

[1]https://openai.com/index/chatgpt/ last accessed: 25.07.2024

convey the practical aspects of implementation. In contrast to other approaches, the LLM does not need to know the context of the entire metamodel, but only some predefined functions to call. A user's textual input is then mapped to these predefined functions that can be executed by the metamodeling platform. This reduces the complexity of the task for the LLM and restricts the possible output to a controllable set of functions, making the approach flexible and extendable.

The remainder of the paper is structured as follows. First, we discuss related work in Section 2, followed by a description of the concept of the approach in Section 3 and the prototypical implementation of the proposed methodology in Section 4. In Section 5 we illustrate the approach on two metamodels.

2. Related Work

Several approaches have been explored for the application of large language models to conceptual modeling and metamodeling. These will be briefly characterized in the following.

In initial experiments, large language models were used to create and interpret entity relationship diagrams (ERD), business process models, UML class diagrams, and Heraklit models [1] and later also BPMN process models and Petri nets [4]. However, the reported experiments were restricted to these modeling languages and did not target the level of metamodeling.

Baumann et al. [5], attempted to generate models for domain-specific languages for which an LLM had little or no training data. They used a Retrieval Augmented Generation (RAG) approach to automatically retrieve relevant examples from a knowledge base based on the user's query. This permits to a certain extent to target arbitrary modeling languages, but the approach did not consider the LLM-based manipulation of the models.

Subsequently, experiments were conducted in which an LLM was provided with a meta-metamodel, several metamodels, and instance models in JSON format created by a metamodeling platform and accompanied by natural language descriptions [2]. On this basis, the LLM was asked for various outputs that required an understanding of the connections between the different layers. The experiments showed that ChatGPT was not able to interpret the relationships between meta-metamodels, metamodels and model instances, resulting in unstable and invalid results, which was mainly due to the large size of input data.

Although previous approaches showed very well the utilization of large language models for the creation and interpretation of conceptual models, only few investigations have so far been made that can be applied to any modeling language and are integrated in a model editor.

3. Concept of the Approach

The idea of our approach is to build the final model incrementally in small steps. Each step involves running a few lines of code in the form of a function. When these functions are triggered they perform a UI action on the existing instance of the model, such as creating an instance, naming an instance, moving an instance, deleting a relation, etc. Our approach relies on these functions as the interface to the large language model - see the architecture depicted in Figure 1. The LLM is provided with all the available functions and input from the modeler. The function names, descriptions, parameter names, parameter descriptions, and dynamically

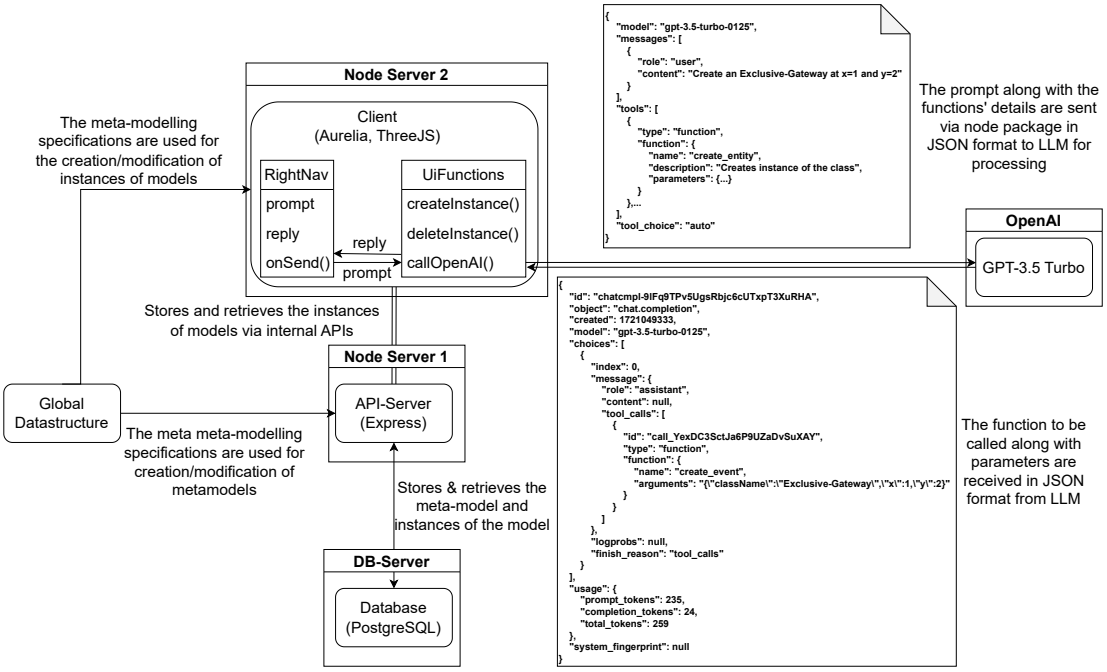

Figure 1: Architecture of LLM integration into the MM-AR Metamodeling platform.

obtained sample values of the parameters are provided to the LLM as context information. There is no need for the LLM to know specific information about the meta-metamodel or the underlying metamodels of the platform. Based on the user's input and context information provided, the LLM returns the function name to call, including the function parameters to pass to the function. For example, if the user specifies keywords like *create*, *build*, or *instantiate* in the prompt and there is a function called "create" in the list of functions provided to the LLM whose description mentions that it creates an entity, then the LLM chooses this function over other functions. This chosen function is subsequently executed by the metamodeling platform for modifying the model instance in the same way as a user would use a mouse and keyboard. In the next section, we discuss the prototypical implementation of these concepts as integration into the MM-AR metamodeling platform [3].

4. Prototypical Implementation

As basis for the first prototypical implementation, we used the MM-AR metamodeling platform [3]. It permits to define metamodels in JSON format for which it automatically creates according model editors that let users interact with the models using mouse and keyboard actions [6]. These actions are implemented as TypeScript functions on the client side of the platform. Although one can design & view the models in 3D & augumented reality in MM-AR, we will continue with 2D models. Nevertheless, the approach is also extensible to three dimensions.

As shown in Figure 1, the platform is implemented as *Node.js*[2] web server running *Aurelia2*[3] and the *JavaScript WebGL* visualization framework *THREE.js*[4]. Furthermore, the platform relies on a database server with *PostgreSQL*[5], which is accessed via an *API Server* running as *Node.js* application and *express*[6].

As an extension to traditional UI interaction methods, the platform has been enhanced with a prompt field that allows descriptive input. Upon the click of a button, the descriptive input of the desired modeling task, along with the possible set of UI functions, their parameters and corresponding descriptions, is sent to the OpenAI API using GPT 3.5 Turbo as the language model. From the user's point of view, only descriptive input is considered. The API then returns, on the basis of the input of the user the most probable function name, as well as the required parameters in a JSON format. This function is then called, and the according modeling actions are executed on the model instance. The right side of Figure 1 shows exemplary prompt data sent to the LLM (top) and received by the LLM (bottom).

5. Illustrative Scenarios

To illustrate the generic application of the new approach, two use case scenarios for (1) creating instances of model elements and (2) renaming existing model elements in the BPMN and the e3value modeling languages are presented below. Even though these two metamodels are simplified, the underlying structure is already quite complex[7].

Figure 2 shows the state of the model after sending a request with the prompt: "Create a start event" for a BPMN diagram. Figure 3 shows that attributes of instances in a model can be changed by textual input. In this example, a Boundary Element of an e3-value model is created and named by stating: "Create a Boundary Element and name it as point".

As we can see, the LLM correctly identified which function to invoke in both the cases and has correctly extracted the parameter value out of the given prompt. This way compound prompts can also be processed for executing multiple tasks sequentially. Thus, we can also build and modify large and complex models using multiple simple prompts. We can observe that this approach works on arbitrary metamodels and is in indeed platform independent. The only requirement is that UI actions must be accessible via functions, where each function performs a dedicated UI action and vice versa.

6. Conclusions and Limitations

In this paper, we introduced a light-weight approach for integrating LLMs directly into metamodeling platforms, which overcomes the difficulties of LLMs in understanding complex model data by not relying on specific model information, but using UI interaction features as a proxy. In a

[2]https://github.com/nodejs/node
[3]https://github.com/aurelia/aurelia
[4]https://github.com/mrdoob/three.js
[5]https://www.postgresql.org/docs/
[6]https://github.com/expressjs/express
[7]An illustration of the two metamodels can be found on: https://doi.org/10.5281/zenodo.12920616

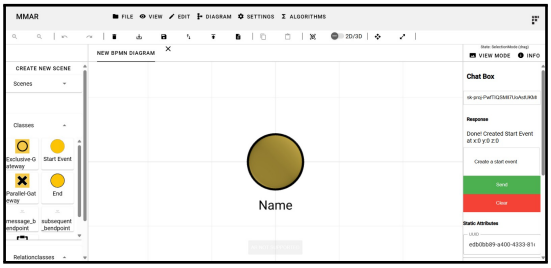

Figure 2: Example of creating a start event with a descriptive prompt in a BPMN diagram.

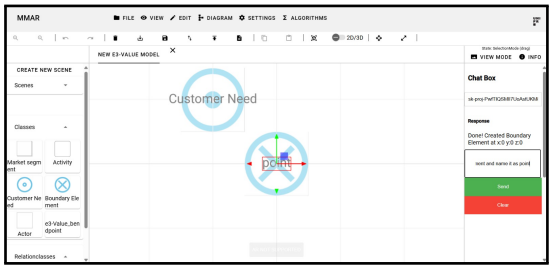

Figure 3: Example of creating a Boundary Element named 'point" in an e3-Value model with a descriptive prompt.

first prototypical implementation, we were able to show that the approach works for basic model actions, e.g., creating instances and renaming instances for two metamodels but is extensible to all other UI actions as well. Also, it is metamodel-agnostic and platform-independent.

Nevertheless, this approach is not without some limitations. The prototypical implementation is reliant upon the names instead of platform's Universally Unique Identifiers (UUIDs) to distinguish the different concepts. Consequently, it cannot be guaranteed that the correct metamodel concept is being employed when names are used. We recommend future developers and researchers to invest sufficient time in giving adequate contextual information to the LLM through descriptions.

References

[1] H. Fill, P. Fettke, J. Köpke, Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT, Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. 18 (2023) 3. doi:10.18417/EMISA.18.3.

[2] F. Muff, H. Fill, Limitations of ChatGPT in Conceptual Modeling: Insights from Experiments in Metamodeling, in: Modellierung 2024 - Workshop Proceedings, GI e.V., 2024, p. 8. doi:10.18420/MODELLIERUNG2024-WS-008.

[3] F. Muff, H. Fill, Initial Concepts for Augmented and Virtual Reality-based Enterprise Modeling, in: ER Demos and Posters 2021, volume 2958, CEUR-WS.org, 2021, pp. 49–54. URL: https://ceur-ws.org/Vol-2958/paper9.pdf.

[4] H. Kourani, A. Berti, D. Schuster, W. M. P. van der Aalst, Process Modeling with Large Language Models, in: BPMDS 2024, Springer, 2024, pp. 229–244. doi:10.1007/978-3-031-61007-3_18.

[5] N. Baumann, J. S. Diaz, J. Michael, L. Netz, H. Nqiri, J. Reimer, B. Rumpe, Combining Retrieval-Augmented Generation and Few-Shot Learning for Model Synthesis of Uncommon DSLs, in: Modellierung 2024 - Workshop Proceedings, GI e.V., 2024, p. 7. doi:10.18420/MODELLIERUNG2024-WS-007.

[6] F. Muff, H. Fill, M2AR: A Web-based Modeling Environment for the Augmented Reality Workflow Modeling Language, in: Proceedings of the MODELS Companion '24, ACM, 2024. doi:10.1145/3652620.3687779.