Generative AI with Modeling and Simulation of Activity and Flow-Based Diagrams

Abdurrahman Alshareef[1(✉)], Nicholas Keller[2], Priscilla Carbo[2], and Bernard P. Zeigler[2]

[1] Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
ashareef@ksu.edu.sa
[2] RTSync Corp., Chandler, AZ 85226, USA
{nicholas.keller,priscilla.carbo,zeigler}@rtsync.com

Abstract. In systems engineering and model-based design, the complexity and interrelationships across different system elements always demand continuous elaboration and expansion in various overlapping domains. We examine how such a phenomenon can be assisted with generative AI and benefit from large language models (LLMs), such as GPT-4. We demonstrate ways of incorporating generated text and outputs from LLMs into the modeling process. The approach can customize the GPT-4 model with an activity metamodel specified in Eclipse Ecore or predefined activity diagrams encoded in a textual format for learning from instances. Alternatively, the descriptive text from the LLM can be provided as input to a parser, resulting in an activity that can be readily transformed into a discrete event system specification (DEVS) model with simulation capability. We will discuss how the process can be enhanced in a simulation environment, thus offering the opportunity to examine a variety of scenarios and arguments for incorporating generative AI or general AI as a collaborative agent in the domain of interest. One scenario could begin with a simplified text describing a generic process, yielding an approximate representation as a starting point for further elaboration by modelers to a complex specification through a systematic, guided, and well-defined framework. We demonstrate the approach with activity and flow-based diagrams in a manner applicable to SysML, UML, and systems engineering at large.

Keywords: Activity Diagram · Generative AI · DEVS · SysML · UML · Systems Engineering · Ecore

1 Introduction

We examine how generative AI can offer valuable assistance in constructing simulation models. Such assistance can take multiple forms, from trivially generating plain textual descriptions to making critical maneuvering decisions and

J.-L. Guisado-Lizar et al. (Eds.): SIMUtools 2023, LNICST 519, pp. 95–109, 2024.
https://doi.org/10.1007/978-3-031-57523-5_8

threat recognition tasks. In the case of large language models (LLMs), it can help distinguish between boilerplate and logic infused in models. It can help in repetitive tasks that may vary in significance to different modelers with different domain backgrounds and perspectives. The assessment can also take a more profound form and facilitate discussion and experiments related to problems regarding domain-neutral and multi-domain modeling along with platform-independent modeling all the way through to Artificial General Intelligence (AGI) with advanced planning capabilities.

In systems engineering and software modeling, generative AI offers an opportunity to facilitate the creation of models in design frameworks like the System Modeling Language (SysML) and Unified Modeling Language (UML) at large. Corpora of such documents must have been subject to the aggregations of LLMs. Basic inquiries demonstrate that. However, they are contributing to the body of knowledge in these models in a collective manner. Given the scale of such models, they are susceptible to biases and specific patterns from various resources, which we plan to investigate in this paper with simulation. To address potential failures and threats, research has been initiated for safety and responsibility, such as responsible AI [9] and safe and beneficial AI [13].

For activities and activity diagrams, in particular, the task demands more effort to navigate through intricacies posed by challenging problems of interpretations, semantics, transformations, and code generation [1]. Currently, GPT-4 cannot generate executable code or real instances from an activity diagram. Moreover, different parallelization schemes and algorithmic complexity, as evident in any sophisticated activity modeling, require advanced handling and presumably a degree of reasoning and validation. However, ongoing efforts, such as in GPT-4, attempt to enhance LLMs with these capabilities. Here, we demonstrate ways they can assist simulation modeling and approximate design in the preparatory sense or earlier stages of model development, as well as in intrinsic elements of the process, such as random number generation and sampling.

The general frame of learning and perception has been subject to examination by simulation research [14]. More recent research [10,19] seeks to infuse generative adversarial network (GAN) into agent-based and traffic simulations to enhance synthetic populations and sampling. In addition, major developers [12] in such technology use simulated tests and exams to evaluate and benchmark the performance of their models where valuable insights can highlight and guide future decisions for further improvement and acquisitions. The simulation element, as a plugin, framework, or environment, can significantly contribute to the whole ecosystem and guide optimizations in data and logic with discipline and confident assessment of alternative designs.

2 Common LLMs

Commonly known as ChatGPT, the GPT-4, a member of the Generative Pretrained Transformer (GPT-n) family, is the most recently released LLM model by OpenAI [12]. The model fundamentally uses a deep neural network for natural

language understanding. It initially has a knowledge cutoff as of September 2021. GPT-4 Turbo extended its knowledge to April 2023. Thus, the training data consists of information from the internet up until that date. GPT-3 (GPT-4 predecessor) uses 175 billion parameters [8], and the number of parameters in GPT-4 has yet to be released. Simulating exams taken by humans was one of the performance benchmarks conducted to evaluate and assess such models. The model accepts prompts in different forms, such as images or texts. In this work, we primarily experimented with using the textual form for all prompts. However, we also experimented with textual forms that are handwritten and captured in pictures. We plan to extend that and examine various visual inputs, such as images, slides, diagrams, charts, or infographics, in future works.

Google developed PaLM [4], a Transformer model trained with the Pathways system, an advanced design utilizing many TPUs (Tensor Processing Unit). Bahram et al. [2] presented the Pathways design developed for distributed dataflow and computation accelerators for machine learning (ML). The novelty in such design comes primarily from a distinction between the data plane and the control plane while warranting parallel execution despite sharding. In [4], PaLM uses 540 billion parameters to train the neural network. We used PaLM 2 branded as Bard in some of the experiments.

3 Related Work

We looked for works that use ChatGPT or other LLMs to create models in specific modeling languages and for activity diagrams in particular. Some recent research uses LLMs and other technologies to automatically or semi-automatically generate code, UML, and SysML. In [3], the authors test its ability to re-create existing class diagrams through iterative prompting. They conclude that ChatGPT can handle only relatively small models, is inconsistent, and does not comprehend all UML concepts. They also observe that its performance depends highly on the specific UML syntax and is much better with OCL (Object Constraint Language). Given that the performance of ChatGPT is highly dependent upon the specific domain, we considered it worthwhile to explore its applicability to activity diagrams.

In [17], the authors use BERT-based (Bidirectional Encoder Representations from Transformers [6]) language models to generate requirements tables from U.S. federal aeronautical regulations. It does this primarily through named entity recognition and chunking sentences into linguistic elements such as noun phrases, verb phrases, and much more. This method cannot generate activity diagrams, although it can be used to aid in generating SysML requirements tables. Their technique generates requirements from documentation but cannot create that documentation or requirements tables from scratch like ChatGPT can. In [15], authors discuss the potential data curation role that could benefit AI-enabled aerospace systems along with challenges for concept development and MBSE (Model-Based System Engineering.)

In [22], the authors are concerned with extracting SysML from system documentation. Unlike [17], they use statistical methods instead of a large language

model. Their approach relies upon frequency-inverse document frequency (tf-idf) and part-of-speech tagging. They can only generate block definition diagrams, internal block diagrams, and requirement diagrams. Like [17], they rely more on existing documentation.

GitHub's copilot uses an LLM to write code from natural language prompts. We consider our work to align with this approach with application to SysML and UML. Copilot helps generate code with a considerable success rate [20]. This performance is likely possible for SysML with current models or further training. Although practical, it is prone to producing errors. However, recent research has found that experienced developers best use the copilot as an AI pair programmer [5]. Junior developers may fail to identify the errors that the copilot produces. We expect a similar dynamic with our SysML and UML generation tool once it matures while valuing human expertise, inputs, and logic. Our work also addresses some limitations in natural language requirement descriptions highlighted in [18] while relying on the DEVS formalism [21] as an underlying and core guiding principle.

4 Possible Use Cases of Generative AI for Activity Diagrams

As illustrated in Fig. 1, we consider three possibilities in which generative AI can assist in the modeling of activity and flow-based diagrams.

The first possibility is where we input examples or the metamodel to the LLM and use it to generate instances that comply with the deduced schema. The second one is where we use it to generate a generic text describing the desired activity model and then parse the resulting text according to a set of given syntactical rules using our code generation facility. We consider the third possibility for future work in which we train, and transfer learn the LLM with additional knowledge about languages and simulation modeling processes using reinforcement learning.

4.1 Customizing GPT-4 to Generate Activity Diagrams

Inserting the Ecore metamodel can be a starting prompt. Then, subsequent queries follow through with requests to generate model instances. In this case, an instance can be generic or domain-specific. A generic instance consists of various activity elements and probably has a generic naming pattern. We had to add particular keywords to create a full-fledged instance without using informal texts or pseudocode. Within the resulting file, it used an XPath-like style for referencing. Since we use Sirius for editing, we changed the referencing style to use an ordered list instead and use element indices. Moreover, since the model has constraints specified using the Acceleo query language, we had to explicitly insert instructions explaining those constraints in a natural language.

Alternatively, inserting instance example as the input string can also be the starting prompt. In this case, akin to post-training reinforcement learning,

we expect the GPT-4 to deduce the metamodel and the overall schema and infer that from one or multiple examples. It is also reasonable to expect the GPT-4 to generate a similar instance without going through deduction. The example model needs to include the various elements in the activity metamodel and our proposed activity editor with features supporting the specification of the simulation experiments and visualization options. A combinatorial design of the example with permutation coverage of all possible parameterizations of variables with finite domains would suffice in giving the GPT-4 comprehensive insights about variations.

4.2 Parsing Textual Flow and Activity Descriptions

We use the GPT-4 in its generic sense to produce a textual description of the activity of interest. We feed the resulting text as a string to the code generator equipped with basic parsing functionality that would, therefore, transform it into a full-fledged activity model. A wide range of options can be used here to implement such a process, including but not limited to the reuse of the GPT-4 itself. Preliminary, we generate activity with various nodes and flows given the basic structure of the given string, such as new lines and numbering. A drawback of this method is that it may include less transfer learning and training to the general GPT-4 model. However, this drawback might be advantageous in domains where a certain level of privacy is required. A more apparent separation exists between the generative AI and other modeling components. Figure 1 visualizes the ways we followed in this paper to produce the desired experiments through activity modeling.

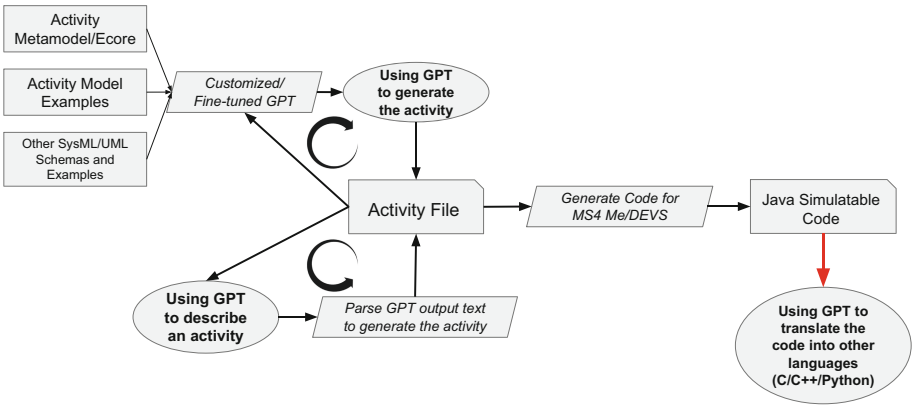


Fig. 1. GPT-4 (or any LLM) assistance in the simulation modeling development

5 Model Generation and Simulation Experiments

We design our simulation experiments while delegating some development tasks to GPT-4, as Fig. 2 explains. It is not surprising to note that the results of each approach differ. For example, the instance obtained after prompting the Ecore metamodel differs from the result obtained after actual instance model training. We used both GPT-4 and PaLM 2. However, we demonstrate the models obtained by employing GPT-4 and plan to explore further variations in future development.

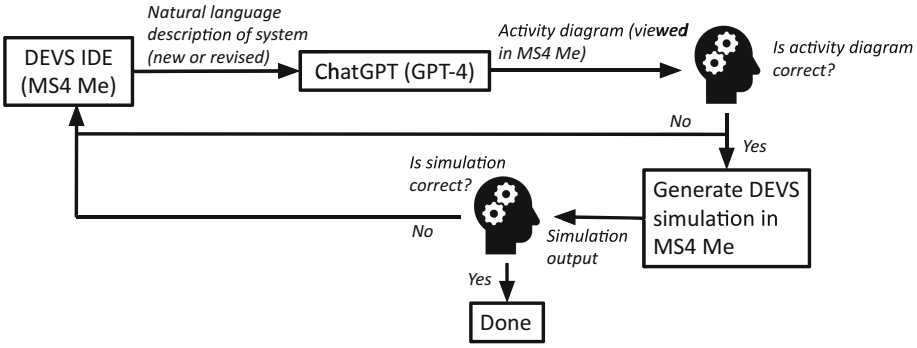


Fig. 2. An overall description of using GPT-4 with activity modeling and simulation in MS4 Me.

5.1 Prompt Engineering

After promoting the Ecore schema, it created a simple but incomplete instance. In the second prompt, it created a fully-fledged instance but with few elements, only one instance of each class, that is, one *Flow*, *Action*, *Sync*, *Select*, *Input Parameter*, and *Output Parameter* with generic names post-fixed with indices and varying value assignments for the Boolean attributes. *Action* is a type of activity node, while *Sync* is a common type for fork and join nodes, and *Select* is a common class for decision/choice and merge nodes. More details about the metamodel can be found in [1].

Additional instructional prompts were necessary to produce a correctly populated instance. More instructions were needed to create more extensive activities with more elements. There were issues with flow specifications in which identifiers for source and destination nodes did not exist. The instance initially followed an XPath-like referencing scheme. We instructed it to create an ordered list of behaviors to comply with the Eclipse Sirius [7] visual editor. Since some constraints on the metamodel were originally specified using the Acceleo query language without being part of the Ecore schema, these constraints were violated. Using natural language, we prompted the GPT-4 with such constraints.

And it complied. More prompts were given to fix specific mistakes in the activity or to add missing flows. Most of the resulting instances did not use parallel flows or hierarchical structures. Figure 3 shows the final resulting activity.

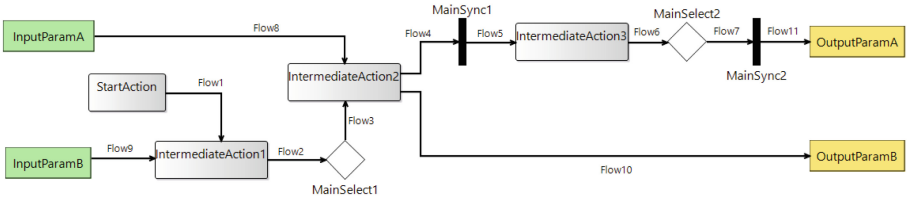


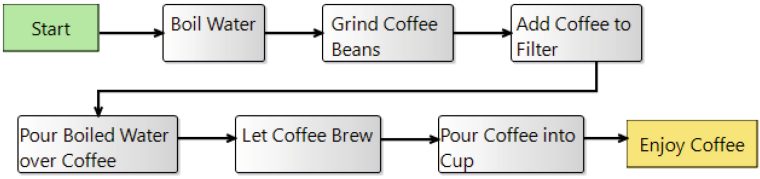
Fig. 3. The generated generic activity. Notice that some elements, such as the fork and join, have been misplaced or roughly added.

We could easily see some basic issues in the generated activity. While the activity is syntactically correct, it does not contain parallel flows after the *MainSync1* node, regardless of characterizing the node as a sync, where the node could initiate the parallel flows. Some actions receive multiple incoming flows. It would be better if such flows were processed with control nodes to describe the dictation of the desired control explicitly. Nevertheless, having such a model at the initial step and without making a substantial effort can be valuable. Automation of tedious aspects in modeling and minimizing the entry barriers can offer significant value in some domains.

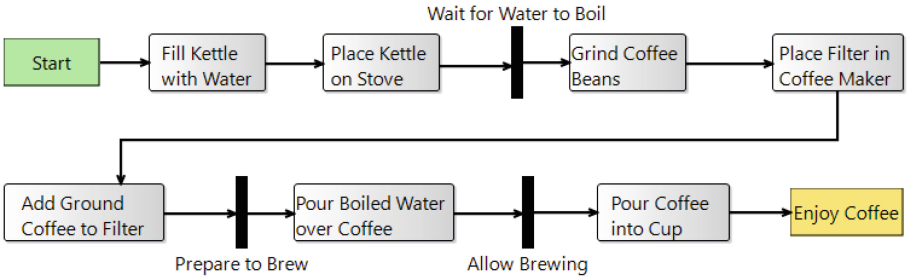
Figure 4 shows the three resulting activities after prompting the GPT-4 with an example activity. The entered example activity is for an airport check-in process and has one input and one output parameter. After entering the text, we started with a text asking the model to deduce the syntax from the entered example activity and then produce an example for making coffee. Figure 4a shows the first result. Figure 4b shows the result after promoting the GPT-4 to add sync nodes. Figure 4c shows the resulting activity after adding a prompt for adding parallel flows. Notice that the resulting activities still do not have a proper use of parallel flows with reasonable description at the example level.

5.2 Syntactical Parsing of the GPT-4 Output String

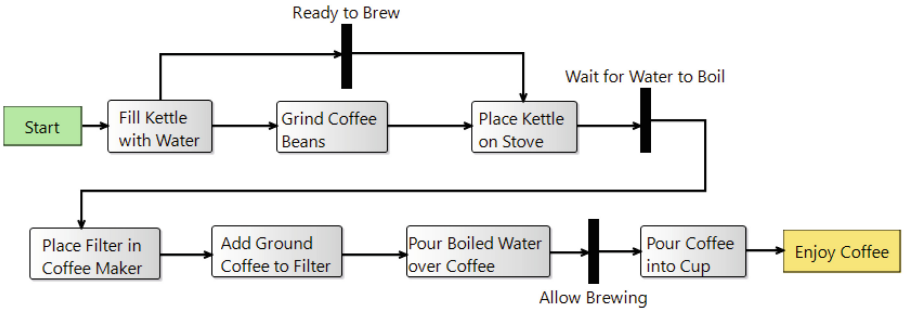
A safer option is to use GPT-4 as is and focus on producing the simpler, non-nuanced parts of the activity and flow descriptions in simple English. Then, we encoded a designated parser for the output string as part of our code generation facility. The advantage of such an approach is ensuring correctness relative to our metamodel, schema, and DEVS specification. The drawback is its limited ability



(a) Resulting activity after prompting with an example



(b) After the second prompt for adding sync nodes



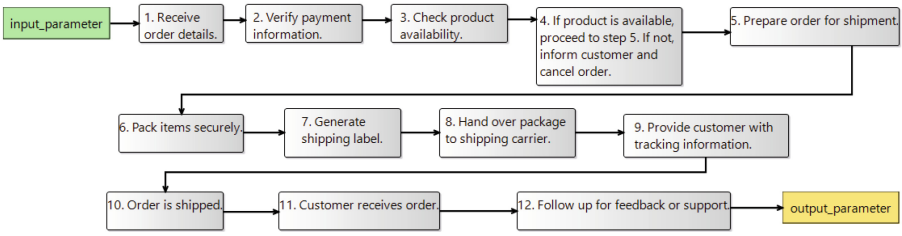
(c) After the third prompt for adding parallel flows

Fig. 4. Generating an activity after promoting GPT-4 with an example activity that is an instance of the activity Ecore metamodel. Again, notice the misplacement of some nodes.

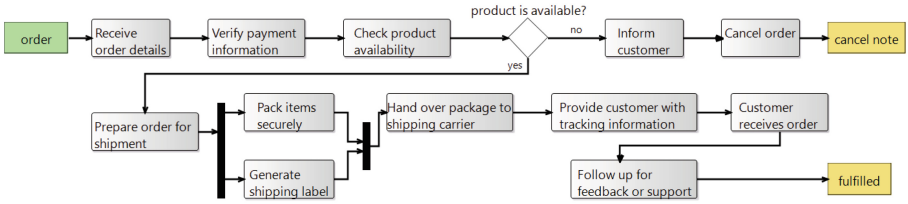
to parameterize the various parameters and stochastic specifications supported in the activity editor. How to incorporate such aspects in the parsing process needs to be clarified. However, some aspects have already been addressed in DEVS natural language and ongoing research in natural language processing and programming language development.

The approach focuses on gaining the low-hanging fruits of GPT and LLM while maintaining the rigor found in mathematical modeling. The overall perspective is to delegate parts of the model development tasks to generative AI and the language model, in particular for aspects that are akin to the LLM function like naming, helping modelers to adopt a more pragmatic perspective and shifting their focus toward what they deem more relevant to their domain of

interest such as logic, creative and critical thinking, tactical design, and reasoning. The warranted flexibility has the potential to align well with the virtues of domain generic, domain-neutral, multi-domain, and platform-independent modeling. It allows modelers to select what to delegate to GPT with some flexibility, especially in capturing requirements that might be specified in various forms and preferences and using different tools, editors, and documentation standards and techniques. Figure 5a shows the basic transformation of text into an activity where modelers can, therefore, infuse more logic, control, and flow routes using the full design experience with the activity editor in the MS4 Me environment. See the resulting activity after the enhancement in Fig. 5b.



(a) Resulting activity after parsing the resulting textual description from GPT-4



(b) The same activity after manual enhancement using the full design experience in the activity editor within MS4 Me

Fig. 5. Parsing the generated string for online order activity. Correctness can be ensured in this approach with additional supporting features.

All of the aforementioned generated activities are now readily simulatable in the MS4 Me environment and DEVS-compliant according to the earlier activity specification [1]. We plan to examine the simulation results of models constructed in such a manner. Comparing them with results from ordinary models can provide more insights into LLMs and their interrelationships with model development.

5.3 The Case for Using Generative AI to Capture Requirements in Visual Forms with Perceptive Model Generation

This case would ultimately cover various visual forms of requirements to facilitate model rapid development. We will demonstrate briefly in this paper and plan to extend it in future works and projects. The grand plan for such an endeavor aims to utilize the recent developments in GPT models to transform the current model development lifecycle in general and for systems engineering in particular. Starting by simply using the GPT model to capture the specification in a variety of forms, such as documents, figures, diagrams, and handwritten sketches or notes in audio formats. The job then is to thoroughly digest those entries in a refinement process to ensure the meaningfulness of the resulting models and simulations. We managed to obtain some promising results for small-scale experiments and plan to continue the examination with larger models at scale. The resulting experiment applies to other sections in this paper. In Fig. 6, we demonstrate the overall lifecycle of simulation models in such systems along with their counterpart experiments.

We note that addressing existing challenges pertinent to domain-specific knowledge comes as a priority in the approach of such a cycle. The envisioned AI-enabled system prioritizes the general human expertise and domain experts while enhancing their productivity and filling the technical gaps for them as a way towards a systems engineering approach with overlapping interdisciplinary subjects and domains. Our current approach covers a variety of formal and informal specifications and can extend further seamlessly to cover more elements in a wide range of subjects. The SysML acts as a collective approach for capturing system requirements and communicating mental models across multi-faceted expertise from different backgrounds. The resulting simulations and lessons from the conducted experiments can feed through the system iteratively until reaching the desired goal, where the simulation models can gradually and eventually evolve into a fully-fledged system in the production phase with optimized designs to maximize potential.

5.4 Generative AI and Simulation: Potential Benefits

The importance of simulation as a profound perspective and benchmark tool for assessing systems with AI models and components is undeniable. The idea of enhancing the generative elements in AI models has been entertained in the past [14] with competitive or cooperative reactions in contexts such as game theoretic simulation and cognizant simulation with AI. Recent generative AI development and the advancement in GAN and GPUs can benefit from tools with simulation capacity to perform necessary assessments and evaluation of such systems and with more disciplined testing and adversarial testing mechanisms. The demand for such techniques is urgent, especially when using generative AI with planning capacity in critical domains, given the evidence of such models producing harmful content in so-called hypothetical scenarios [12]. We need not take human-in-loop intervention for granted but instead develop models and conduct research to

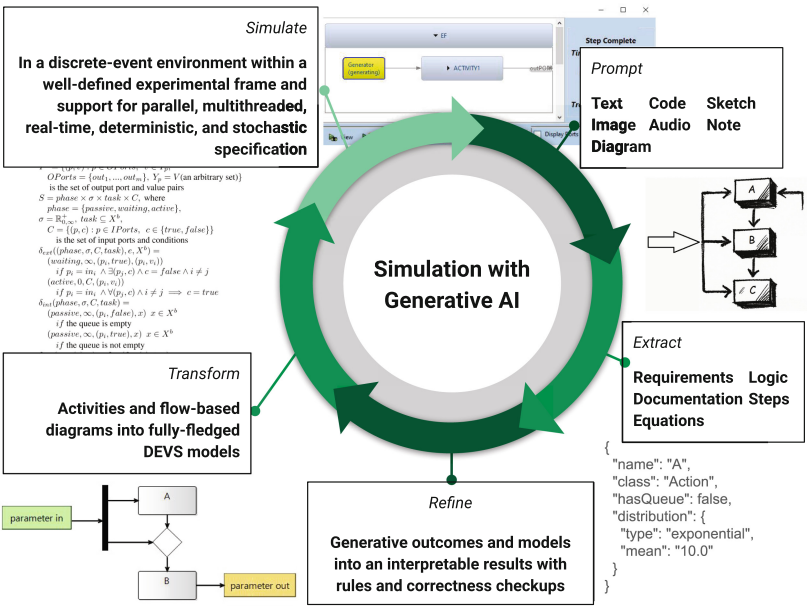


Fig. 6. An illustration of the envisioned lifecycle to capture systems in SysML, UML, Activities, DEVS, DEVS Markov, Queuing, etc., with stronger accounts for timing toward an AI-enabled system with an advanced planning capacity.

uncover more about the black box [11] in ways, no matter how small, that in no way undermine the crucial part of human experience and knowledge but rather signify it. The interrelationships amongst such an ecosystem are complementary and multi-directional. New integrative and incorporative studies are needed to produce tools and techniques to account for the recent development with rigorous simulation support.

Figures 7 and 8 visualize our current approach to the subject in this paper. We treat each element with examination to any redundant, overlapping, or unique role of each. Most of our effort so far in trying to utilize GPT-4 accounts for early stages and starting points in model development. The blue curve in Fig. 7 illustrates the effects of using GPT-4 in the early stages to stimulate critical thinking and facilitate aspects of communicating mental models in a way that could benefit the initial conceptualizations of models and systems under study. However, the curve seems to converge at some points where the LLM may not produce additional utility or even starts to produce inaccurate results. It is possible to consider the resulting models as approximations of actual systems or models that can have alternative representations using more rigorous probabilistic, deterministic, or cognitive manifestations. Figure 8 illustrates the overlapping, unique, or redundant roles we encountered throughout our prompts engineering, simulation modeling, and experimental design.

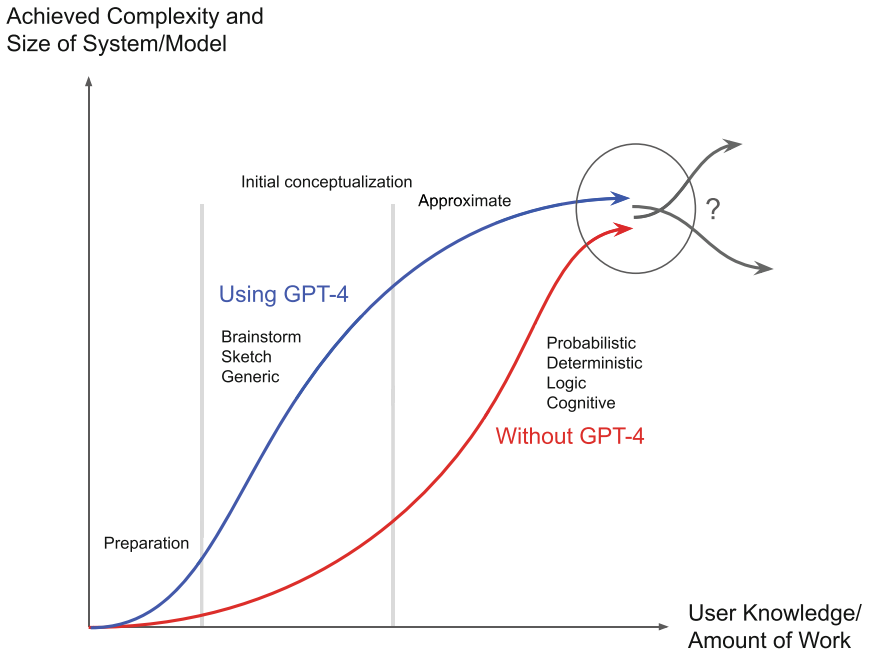


Fig. 7. Roles of using GPT and impact on learning curve

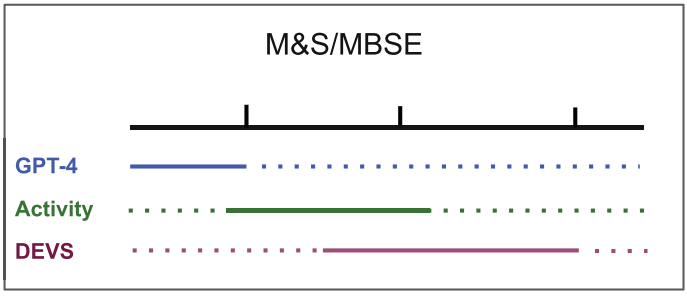


Fig. 8. Our current approach for stages of integrative GPT-4, activity, and DEVS, for M&S MBSE.

LLMs for large scale activities indicates an attractive endeavor to explore infinite variations in design with the possibility to restrain large models. Such a task could offer an ideal setting where handling activities with large amounts of elements and interconnecting components can be complicated for humans. In this setting, queries about the model metadata can be informative and constructive. In a recently submitted article that is currently under review, we tested and explored our tool and code generator for a manually developed activity with 161 actions, 118 end flow, 46 fork/join, 8 decision/merge, 18 input, and 8 output

parameters, resulting in 321 Java code files. We plan to match this capacity with LLMs to enhance the modelers' grasp over large models often encountered in system engineering domains with advanced, flexible, representative knowledge and reasoning.

6 Challenges and Limitations

Although the landscape of generative AI and the use of LLMs is rapidly changing, we attempt to observe and document some of the currently existing limitations we have seen and encountered in our experiments. Amid all the handiness that could come about in using such models, the problem of domain expertise and questions about the level of generality remains quite pressing. The issue can be better discussed in the context of a boilerplate and rules of transformation and translation. Some questions may arise in these contexts to ignite a thorough discussion about the degree of cognition and generative usefulness where both are attained to an extended level.

From the activity modeling perspective, the obtained texts and results from GPT-4 in different ways demonstrate different aspects. The resulting activities from inserting the Ecore metamodel differ from those obtained after inserting an example activity. This particular distinction mimics, in a way, the previous research efforts made to acquire domain-general and domain-neutral modeling [16]. In the Ecore case, the generated activity contains a variety of values that may not be deemed particularly useful. For example, since the seed value is a long data type, the model ignored the default value and gave a different value at each round. Some instructions had to be given in a particular manner to attain correctly specified models where the given details might be subject to the question in the simulation environment. In this kind of context, the question about the degree of customization or fine-tuning without losing generality remains relevant. Further research efforts and experiments can be beneficial and necessary for the overall model improvement.

The current results from GPT-4 are mostly better suited to serve as a starting point for an approximate representation in a manner that is also akin to bootstrapping but in a less data-dependent sense and more toward reasoning, logic, and cognition capacity while using simulation as an essential element of the whole scheme to force the cognitive perspective through it. Enhancements are often necessary to render it more inclusive and aware of the existing body of knowledge. This study's proposed integrative simulation framework could contribute to the model-assisted safety pipeline [12] and examine reliability and dependency issues, such as over-reliance on specific components of the system and potential tradeoffs in centralized schemes from a system-theoretic vantage point.

6.1 Towards Fine-Tuning and Transfer Learning

Tailoring to specific domains is currently quite costly. It demands substantial domain-specific pre-training and reinforcement learning. Accounting for such an

issue demands using sophisticated environments and architectures with a degree of maturity in concepts such as modularity, hierarchy, and iterative development.

Future work will involve fine-tuning existing models or retraining new variants. While developing a specialized model can be costly, there can be some tangible benefits regarding privacy, control, and curating. The hypothetical separation between generative and deterministic, possibly rule-based, aspects in the model can be handy in aerospace, defense, and healthcare domains. However, it is at the cost of segregating from the larger corpus. Such a decision must account for long-term benefits and data-curating techniques that could result in better collective future models in a strategic, large-scale sense with valid representatives of the body of knowledge. This concern proves to be persistently an open problem. Moreover, simulation capability helps discover and explore its dimensions and scale. The position of such capability can take different places and positions across different technical and conceptual layers and facilitate the expansion and growth of the more extensive system toward an AI and LLM with a more integrated cognitive capacity.

References

1. Alshareef, A.: Activity specification for time-based discrete event simulation models. Technical report, Arizona State University (2019). https://keep.lib.asu.edu/items/157772. Accessed 30 Sept 2023
2. Barham, P., et al.: Pathways: asynchronous distributed dataflow for ML (2022)
3. Cámara, J., Troya, J., Burgueño, L., Vallecillo, A.: On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. Softw. Syst. Model. 1–13 (2023)
4. Chowdhery, A., et al.: PaLM: scaling language modeling with pathways (2022)
5. Dakhel, A.M., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M.C., Jiang, Z.M.J.: GitHub copilot AI pair programmer: asset or liability? J. Syst. Softw. **203**, 111734 (2023)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding (2019)
7. Eclipse Foundation: Sirius (2023). https://eclipse.dev/sirius/. Accessed 30 Sept 2023
8. Floridi, L., Chiriatti, M.: GPT-3: its nature, scope, limits, and consequences. Mind. Mach. **30**, 681–694 (2020)
9. Google Research: Responsible AI. Google AI Blog (2023). https://research.google/teams/responsible-ai/
10. Kotnana, S., Han, D., Anderson, T., Züfle, A., Kavak, H.: Using generative adversarial networks to assist synthetic population creation for simulations. In: 2022 Annual Modeling and Simulation Conference (ANNSIM), pp. 1–12. IEEE (2022)
11. Editorial, N.: ChatGPT is a black box: how AI research can break it open. Nature **619**, 671–672 (2023)
12. OpenAI: GPT-4 technical report (2023)
13. OpenAI: Safety and responsibility (2023). https://openai.com/safety
14. Ören, T.I., Zeigler, B.P.: Artificial intelligence in modelling and simulation: directions to explore. Simulation **48**(4), 131–134 (1987)

15. Raz, A.K., Blasch, E.P., Guariniello, C., Mian, Z.T.: An overview of systems engineering challenges for designing AI-enabled aerospace systems. In: AIAA Scitech 2021 Forum, p. 0564 (2021)
16. Sarjoughian, H.S., Alshareef, A., Lei, Y.: Behavioral DEVS metamodeling. In: 2015 Winter Simulation Conference (WSC), pp. 2788–2799. IEEE (2015)
17. Tikayat Ray, A., Pinon-Fischer, O.J., Mavris, D.N., White, R.T., Cole, B.F.: aerobert-ner: named-entity recognition for aerospace requirements engineering using bert. In: AIAA SCITECH 2023 Forum, p. 2583 (2023)
18. Wach, P., Salado, A.: The need for semantic extension of SysML to model the problem space. In: Madni, A.M., Boehm, B., Erwin, D., Moghaddam, M., Sievers, M., Wheaton, M. (eds.) Recent Trends and Advances in Model Based Systems Engineering, pp. 279–289. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-82083-1_24
19. Wu, C., et al.: Spatiotemporal scenario generation of traffic flow based on LSTM-GAN. IEEE Access **8**, 186191–186198 (2020)
20. Yetistiren, B., Ozsoy, I., Tuzun, E.: Assessing the quality of GitHub copilot's code generation. In: Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering, pp. 62–71 (2022)
21. Zeigler, B.P.: DEVS representation of dynamical systems: event-based intelligent control. Proc. IEEE **77**(1), 72–80 (1989)
22. Zhong, S., Scarinci, A., Cicirello, A.: Natural language processing for systems engineering: automatic generation of systems modelling language diagrams. Knowl.-Based Syst. **259**, 110071 (2023)