Preliminary Results of An Experiment on Leveraging Large Language Models to Assist Modelers in Interpreting DEVS Natural Language Models

Valdemar Vicente Graciano Neto[1], Nicholas Keller[2], Doohwan DH Kim[2], Chungman Seo[2], Priscilla Carbo[2], Bernard Zeigler[2]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Goiânia – Goiás – Brazil

[2]RTSync Corp.
Chandler – Arizona — United States

valdemarneto@ufg.br, {nicholas.keller, dhkim, cseo, priscilla.carbo, zeigler}@rtsync.com

Abstract. *Discrete Event System Specification (DEVS) Natural Language (DNL) implements the DEVS simulation formalism using a natural language-like notation. However, DNL models can still be complex, involving multiple inputs/outputs, internal/external state transitions, and arbitrary Java code blocks, which steepens the learning curve and reduces the efficiency of junior modelers. Concurrently, Large Language Models (LLMs) like ChatGPT have gained popularity across various domains for their ability to answer specific questions about referenced content. If an LLM tool could reference simulation models written in DNL, it could potentially greatly increase modeler efficiency. To this end, we developed GEM DEVS Chat, a tool designed to assist developers in understanding DEVS models within a simulation project. This paper presents GEM DEVS Chat and reports on an experiment conducted during a Modeling and Simulation course for undergraduate and graduate students. The experiment involved eight students, divided into control and experimental groups. Results indicate that students assisted by the tool understood DEVS models more quickly and accurately.*

1. Introduction

DEVS Natural Language (DNL) is a domain-specific language used within the MS4 Me Integrated Development Environment (IDE). The goal of DNL is to make DEVS modeling and simulation more approachable [Zeigler et al. 2012]; however, it remains challenging for novice modelers to learn and use DNL due to its difference from the more familiar object-oriented programming of languages such as Java. Therefore, we developed GEM DEVS Chat to let modelers ask questions about DEVS atomic and coupled models in a project. We expect that GEM DEVS Chat can be used by new model developers to more effectively learn from existing sample DEVS projects and for experienced DEVS developers to get up to speed faster on modeling projects they are onboarding to. GEM DEVS Chat leverages an LLM to provide a user friendly and flexible question and answer experience.

The main contributions of this paper are twofold: (i) presenting the GEM DEVS Chat tool itself, and (ii) reporting results from a preliminary controlled experiment involving eight undergraduate and graduate students. Results show that adopting GEM DEVS Chat can increase the understanding of DNL models.

The remainder of this paper is organized as follows. Section 2 presents a brief theoretical framework along with related work. The GEM DEVS Chat tool is described in Section 3. The experiment planning, execution, results and discussion are brought in Section 4. Finally, Section 5 presents the conclusions.

2. Background

DEVS (Discrete Event System Specification) is a simulation formalism whose models use state variables and state transition functions to update in response to input events [Zeigler et al. 2012, Graciano Neto and Kassab 2023]. DEVS Natural Languages (DNL), in turn, extend the DEVS formalism to facilitate modeling and simulation (M&S), where natural language descriptions and parameters are integrated with formal DEVS constructs. DNL allows for the inclusion of qualitative descriptions and linguistic parameters into DEVS models, making them more accessible and relevant to domains such as software engineering [de França and Graciano Neto 2021, Bulcão-Neto et al. 2022], biology, ecology, economics, and social sciences [Zeigler et al. 2012, Blas et al. 2020, Alshareef and Zeigler 2020]. By incorporating natural language, DNL improves the interpretability of DEVS models, especially in contexts where human interpretation and understanding of system behavior are crucial.

Large Language Models (LLM), in turn, refer to a class of artificial intelligence models, such as OpenAI's GPT series or Google's BERT, which are trained on vast amounts of textual data to generate human-like text and perform various natural language processing tasks. LLMs have demonstrated remarkable abilities in natural language understanding, generation, translation, summarization, and more complex tasks like question answering and dialogue generation. It has become very popular, particularly to help on code generation at a broad spectrum [de Albuquerque et al. 2024].

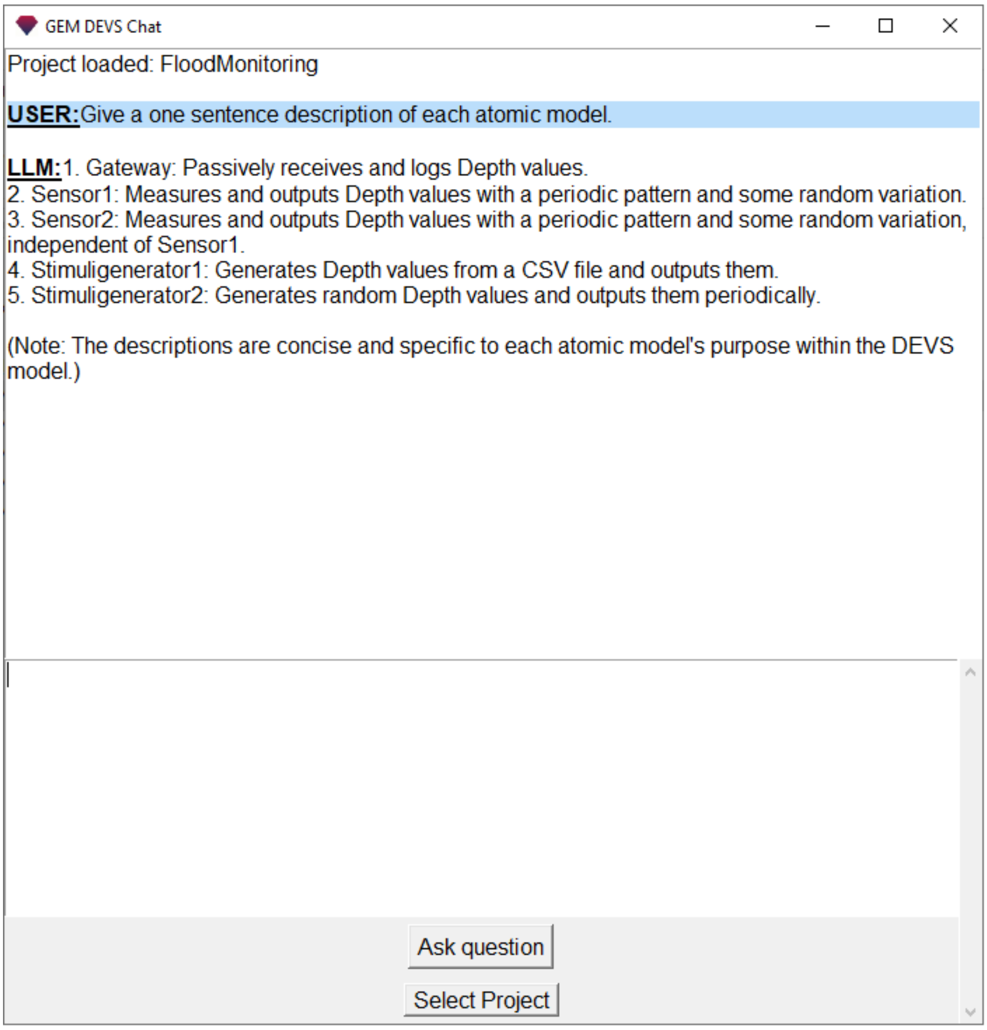
DNL and LLM intersect in their approach to incorporating natural language into formal modeling frameworks. DNL integrates natural language descriptions into DEVS models, enhancing the descriptive and interpretative power of the models in natural systems; whilst LLMs provide the capability to generate and process natural language at scale, which can be utilized in DNL to describe and analyze system behaviors or to enrich simulations with textual data.

The synergy between DEVS, DNL, and LLMs opens avenues for enhanced modeling and understanding of complex systems across disciplines. For instance, applying LLMs to generate (or understand) natural language descriptions of system behaviors can enrich DEVS models in DNL, leading to more accurate simulations and deeper insights into natural and socio-technical systems.

Related work. LLM and simulation models have emerged over the past recent years [Alshareef et al. 2024, Gao et al. 2023, Abbasiantaeb et al. 2024, Agrawal et al. 2024]. We performed an exploratory *ad hoc* literature review on Google Scholar on June, 2024 using the search strings `LLM and Simulation`, `LLM and DEVS` or `"large language models"` and `"Discrete Event Specification Systems"`. Many of those studies have simulated LLM behavior or have used LLM to simulate conversations. So far, we did not find any studies that systematically associates both topics: LLM and DEVS, particularly for querying DEVS models and assist modelers to understand them. To the best of our knowledge, this can bring that contribution to the state-of-the-art.

3. GEM DEVS Chat: A LLM tool for querying DNL models

GEM DEVS Chat was conceived to use a LLM tool to assist simulation modelers on the understanding of the code. It was designed to assist in the understandability of DEVS models written in DNL and deployed on the MS4 Me platform. Figure 1 shows the interface of the tool. The tool was written in Python and can be run in any computer that has Python installed.



Project loaded: FloodMonitoring

USER: Give a one sentence description of each atomic model.

LLM:1. Gateway: Passively receives and logs Depth values.
2. Sensor1: Measures and outputs Depth values with a periodic pattern and some random variation.
3. Sensor2: Measures and outputs Depth values with a periodic pattern and some random variation, independent of Sensor1.
4. Stimuligenerator1: Generates Depth values from a CSV file and outputs them.
5. Stimuligenerator2: Generates random Depth values and outputs them periodically.

(Note: The descriptions are concise and specific to each atomic model's purpose within the DEVS model.)

Figure 1. A screenshot of the LLM tool to query about MS4 Me Simulation projects.

First, the user selects a project using the button ‘Select Project’. The project should be a directory with the entire structure of a MS4 Me DEVS Simulation Project. Once the directory is selected, the project is loaded, and the user can ask questions about the atomic and coupled models in that project. In the case of this figure, a project of a Flood Monitoring system was loaded and the tool was required to give a one sentence description for each atomic model of the project. As for the tool manual, a sort of different types of questions can be directed to the tool, such as: *Give a paragraph description of each coupled model*, *How can I improve the accuracy of the [name] atomic model?*, *Give me an outline for an atomic model of [system to model]* and *Define the states, internal/external transitions, inputs/outputs, variables, and key equations*.

4. Experiment

This section details the research method used, structured in three well-defined steps, following the guidelines of [Wohlin et al. 2012]: (i) Planning, (ii) Experimentation, and (iii) Reporting, as follows.

4.1. Planning

We applied this protocol to conduct a controlled experiment.

The **objective** of this study was formalized based on the GQM (Goal-Question-Metric) technique [Basili 1993]: *Analyze* the DEVS code in a simulation project *for the purpose of* understandability *with respect to* the use of a LLM tool *from the point of view of* researchers *in the context of* a Modeling and Simulation training course.

From the goal, we derived the research question that expresses the objectives of this experiment:

RQ. *How effective is the LLM-based tool GEM DEVS Chat in the degree of understandability of DEVS models by students being trained in a M&S course?*

Rationale: The aim of this RQ is to investigate, through a controlled experiment, whether the students who use a LLM-based tool can achieve such a larger degree of understandability of the code of a DEVS simulation project deployed in MS4 Me environment.

From that RQ, two variables were derived to measure the results: **response time** and **score**. The students were submitted to an experiment and invited to answer an assessment test with ten questions about a given MS4 Me project. Then, the time to finish and the score in the final test were measured. Details are provided in this section later.

By following the PICO (Population, Intervention, Comparison, Outcomes) principles defined in Kitchenham and Charters [Kitchenham and Charters 2007], we can understand the experimental design of this study. As shown in Table 1, the population of this study is a group of students attending to a Modeling and Simulation elective course offered by a university during 2024 scholar year and the sample is equal to the entire population of that class. The intervention is the use of DEVS GEM Chat to support students in the understandability of the DNL code of a MS4 Me project. From that population, we extracted two samples: control group and experimental group. The control group is that one who will try to understand the same code without using that tool. Then, the comparison will be performed according to two specific metrics: time to respond to an assessment test and the achieved score. The expected outcome is a greater degree of understanding achieved by the student group using the tool. Such understandability would be expressed as shorter time to answer the test and a greater score.

The assessment test was structured with 10 closed-questions asking about five different models available in the .DNL files delivered to both groups. Each model had two questions related to it.

Table 1. PICO for this experiment.

Population	*Students in a Modeling and Simulation Elective Course.*
Intervention	*The use of a LLM-based tool to help on DEVS code inspection.*
Comparison	*Time Response and Score.*
Outcome	*Better achieved understandability of the models.*

Hence, from this experimental design, the following hypotheses can be raised: H_1*:* The adoption of DEVS GEM Chat can increase the degree of understandability of DNL models by the students. The null hypothesis then is H_0*:* The adoption of DEVS GEM Chat **does not impact** on the degree of understandability of DNL models by the students. `Understanding` is measured by two independent variables: `time` and `score`. The first is time taken to answer the test applied to both samples, and second is the score achieved for the test.

To test the null hypothesis, the following well-defined steps were planned:

Step 1. Environment Preparation and Setting. The GEM DEVS Chat tool was installed in the machines of the laboratory where the course takes place weekly. MS4 Me was already installed in those machines, as well.

Step 2. Onboarding moment. A prize draw website was used to randomly separate the students in control and experimental group. After that, the control group received a project similar to that used to the real experiment to get familiar of the logics of the code, while the experimental group received the manual of GEM DEVS tool in English to get used to the potential of the tool.

Step 3. Controlled experiment execution. After the onboarding session, the experiment commenced, measuring the time to answer the test and the scores achieved. Two groups would be separately approached: one not using the tool and being asked to understand the DEVS models in a project; and another one using the GEM DEVS Chat tool to understand the same set of models. In the former, the professor only delivers the DEVS projects, asks them to deploy in MS4 Me, and, as follows, they perform an inspection in the code towards understanding what they are visualizing. In the latter, the students are invited to use the GEM DEVS Chat tool to understand the code. At the end, both groups are asked to answer a questionnaire about the explanation of the codes.

Step 4. Results analysis and reporting. Finally, the results were analyzed. This step comprised the analysis of results obtained by comparing the manual inspection by the control group and the use of the tool to make it. The control group and the treatment group were submitted to assessment tests to measure the results. The questionnaire was used to collect the necessary data to investigate whether the objectives of the study were achieved.

The next section describes the experiment in detail.

Table 2. Control Group results.

Control Group	*Time Starting*	*Time Finishing*	*Total Time*	*Score*
Student 1	19:40	20:18	00:38	9
Student 2	19:40	20:18	00:38	10
Student 5	19:40	20:01	00:21	10
Student 8	19:40	20:31	00:51	7

Table 3. Experimental Group results.

experimental Group	*Time Starting*	*Time Finishing*	*Total Time*	*Score*
Student 3	19:40	20:06	00:26	10
Student 4	19:40	20:17	00:37	10
Student 6	19:40	20:14	00:34	10
Student 7	19:40	20:11	00:31	10

4.2. Experimentation

The experiment took place on June 19th, 2024 and involved **8 undergraduate and graduate students** enrolled in the M&S course. The graduate students were in their first year of master's and PhD courses, while the undergraduate students were in the 8th semester of Information Systems bachelor's degree. The students were numbered and labeled from Student1 to Student8 to anonymize their identities. Only Student1 was a graduate master's degree student, whislt the others were all undergraduate students from the final of the Information Systems bachelor's degree course. Control Group (CG) was formed by CG = {*Student1, Student2, Student5, Student8*} and the Experimental Group (EG) was composed of EG = {*Student3, Student4, Student6, Student7*}.

Once the environment was prepared, the onboarding took place for 15 minutes (from 7:25 pm to 7:40 pm). The CG received a project similar to that used in the experiment (about a chemical reaction). They deployed the project to MS4 Me and were able to study the code. The EG had access to the GEM DEVS Chat tool manual and the tool itself, besides the same project received by CG.

The experiment itself and timing started at 7:40pm. The delivery time was measured using the Turing platform[1], a Moodle implementation where students could submit their responses. The next section details the results.

4.3. Reporting

The experiment lasted almost one hour and the tools run in DELL OPTIPLEX 790 with 8GB RAM, Intel i3 7th generation, 1 TB HD, Windows 10. Tables 2 and 3 illustrate the results. We can observe that, considering the average, the total time to delivery the answers in the control group (37 minutes) were greater than those from the experimental group (32 minutes). Similarly, the the average score was also lower in CG than in EG, revealing that GEM DEVS Chat tool can reduce the time to understand a DNL code and also allow students to better understand the purposes of that code.

We can certainly observe an outlier in the CG (Student5), whose conclusion time

[1] https://turing.inf.ufg.br/login/index.php

was the lowest among all the students and also achieved the maximum score. However, all other students (1, 2 and 8) had conclusion times greater than all the students from EG and two of them had lower scores than the maximum. Figure 2 presents box plots for the same data[2], respectively the time and score for the control group and time for experimental group (since scores were identical for the EG, the plot was not needed). Those plots offer an alternative visualization for the same data. Time in CG varied considerably more when compared to EG and the median time was a smaller in EG than in CG. The median score in CG was also high (9.5), but there were situations of lower scores (7 and 9).

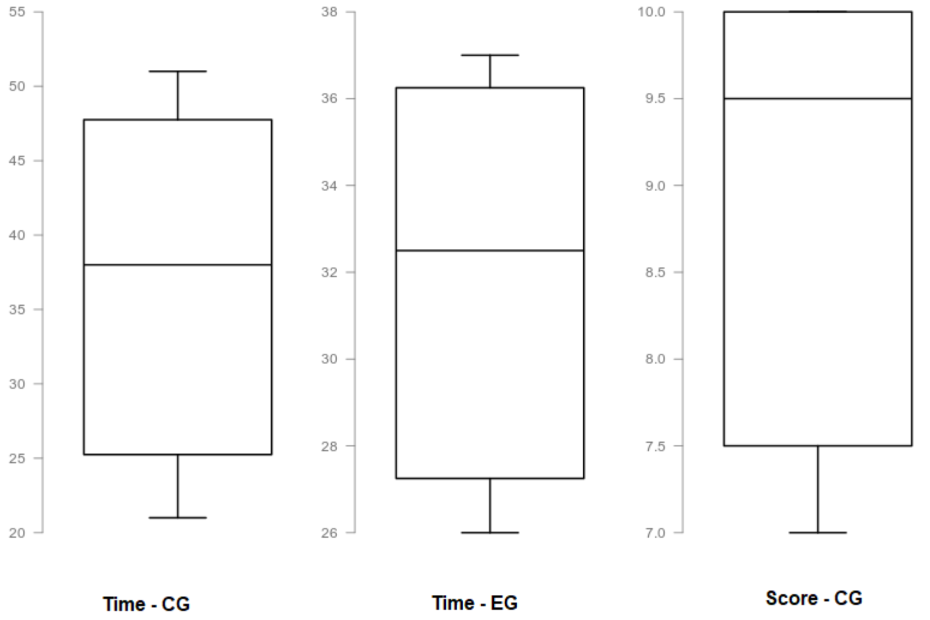


Figure 2. Box plots for the collected data.

The Shapiro-Wilk test (for samples between 2 and 51 elements)[3] showed that `time` presents a normal distribution considering the data for the eight involved students, while `score` does not present a normal distribution. In this case, we can use t-test for two samples to compare `time`[4] [Moya 2021]. A significance level $\alpha = 0.05$ was defined. After applying it, the t-value is 0.75956 and the p-value is .238147. Hence, the result was not significant at $p < .05$, and we could not refuse the null hypothesis when considering time, despite EG exhibiting lower time to respond to the questionnaire than CG.

4.4. Discussion

Regarding the experimental group, even with the instructions, some students had difficulty understanding which project the applied test was about. Furthermore, the tool only works when the project is already deployed in MS4 Me and the full address of that project directory is passed. Therefore, some students who were using the tool spent a few minutes

[2] Created using BoxPlot: http://www.alcula.com/calculators/statistics/box-plot/

[3] Applied using online service at https://www.statskingdom.com/shapiro-wilk-test-calculator.html

[4] T-test applied using https://www.socscistatistics.com/tests/studentttest/default2.aspx.

making these checks and adjustments before being able to properly interact with the tool. A student in the control group had to change machines because MS4 Me was not working on that tool. Therefore, some students in the EG even lost time compared to those who did not use the tool. If they have a better score and time, it will show that the tool was successful. There was also communication between some members of the control group, which could bias the result. A student in the EG reported that he asked the tool the same question and, similarly to ChatGPT, received two different answers. One student reported difficulty with English and also the fact that he only used .dnl to understand the second project, whereas the first (sample project) had .ses and could be executed, making understanding easier. Student 4 reported that he finished early, but that he kept reviewing his answers.

When the subject is LLM, hallucinations and wrong answers are a recurrent issue, i.e, generating wrong or absurd answers only to avoid not delivering content [Jiang et al. 2024]. In regards to GEM Chat DEVS, two actions were performed during the creation of the tool to avoid such phenomena: 1) Developers reduced the risk of hallucination by including the atomic models in the context window; and 2) The hallucinations that do occur have not been observed in GEM DEVS Chat. They are mitigated by the fact that the student can check the .dnl file itself. GEM DEVS Chat helps students get up to speed on an atomic model quickly, but they are still expected to examine the .dnl file itself. If they see an inconsistency or something that doesn't make sense, then that is a hint that the LLM may be hallucinating.

Threats to Validity and Limitations. Our controlled experiment involved 8 subjects, where a t-test did not reject the null hypothesis. By following the classical taxonomy of threats [Wohlin et al. 2012], we have threats related to: (i) **Conclusion Validity**, which concerns the degree to which conclusions about the relationship between variables are justified. With a small sample size (8 subjects), there's a risk that the study lacks sufficient statistical power to detect true effects, leading to inconclusive results, as it happened. Then, we could not conclude that the use of the tool pragmatically can increase the time of response and understandability of DEVSNL code; (ii) **Internal Validity**, which refers to the degree to which a study accurately demonstrates a causal relationship between variables, without confounding factors influencing the results. With a small sample size and potential for uncontrolled variables. For mitigating that threat, we randomly assigned subjects to experimental conditions to minimize selection bias and ensure comparability between groups. However, maybe because of the sample size, it was not possible to establish a causal relationship; (iii) **Construct Validity**, which concerns whether the measures used in the study accurately capture the theoretical constructs of interest. To mitigate that threat, we ensured that the outcome measures were valid and reliable for assessing the intended constructs, and we also used multiple measures to assess the same construct to enhance reliability and cross-validate results; and (iv) **External Validity**, which regards to the generalizability of the study findings to other populations. The small and homogeneous sample we had may have limited the ability to generalize findings beyond the specific conditions of the study.

Ethical issues. This experiment was not submitted to Research Ethics Committee. To reduce likely impacts of it, we let it explicit for students that they would not have any damage by participating or refusing to participate in that study. Moreover, their identities were anonymyzed, obeying the data protection legislation.

Replicability. For packing and reproducibility purposes and by following Open Science principles [OliveiraJr et al. 2022], all the non-proprietary material used to plan the experiment is available in an external link[5]. Moreover, MS4Me can be downloaded from RTSync website[6] and used for academic purposes, as well as the access to GEM DEVS Chat can also be consulted.

5. Final Remarks

The main contribution of this paper was (i) to present the GEM DEVS Chat tool, a Large Language Model (LLM) empowered tool that could help on the understandability of DEVS Natural Language simulation models and (ii) report the results of the conduction of a controlled experiment on the use of that tool in a class of eight undergraduate and graduate students on Modeling and Simulations. Although it was not possible to refuse the null hyphotesis (H_0*: The adoption of DEVS GEM Chat* ***does not impact*** *on the degree of understandability of DNL models by the students*), the results sounded promising, with the experimental group exhibiting better time to solve problems and a better score on average. Future work include to replicate the experiment and assure to (i) increase sample size, (ii) diversify samples, and (iii) replicate it into different settings. Expanding the use of simulation and LLM for other domains is also a possible path, such as for Information Systems [Araujo et al. 2017, Monteiro and Maciel 2020, de Oliveira et al. 2021].

Acknowledgments

We thank the students who participated of the study. By following the SBC Code of Conduct for Authors in Publications, we explicitly declare that the ChatGPT tool was used to write parts of the Theoretical Foundation of this work. As stated by the Code, Use of Generative Artificial Intelligence (AI) tools is allowed, but it must be explicitly declared. We are aware that the use of such tool does not exempt the authors from responsibility for all of their content.

References

Abbasiantaeb, Z., Yuan, Y., Kanoulas, E., and Aliannejadi, M. (2024). Let the LLMs talk: Simulating human-to-human conversational qa via zero-shot LLM-to-LLM interactions. In *Proc. of 17th ACM WSDM*, pages 8–17.

Agrawal, A., Kedia, N., Mohan, J., Panwar, A., Kwatra, N., Gulavani, B., Ramjee, R., and Tumanov, A. (2024). Vidur: A large-scale simulation framework for llm inference. *Proceedings of Machine Learning and Systems*, 6:351–366.

Alshareef, A., Keller, N., Carbo, P., and Zeigler, B. P. (2024). Generative AI with Modeling and Simulation of Activity and Flow-Based Diagrams. In Guisado-Lizar, J.-L., Riscos-Núñez, A., Morón-Fernández, M.-J., and Wainer, G., editors, *Simulation Tools and Techniques*, pages 95–109, Cham. Springer Nature Switzerland.

Alshareef, A. and Zeigler, B. P. (2020). Integration of activity specification into devs modeling & simulation development environment. In *MSSiS*, pages 46–55. SBC.

[5] https://ww2.inf.ufg.br/~valdemarneto/projects/devsLLM.html

[6] https://rtsync.com/

Araujo, R., Fornazin, M., and Pimentel, M. (2017). An analysis of the production of scientific knowledge in research published in the first 10 years of isys (2008-2017). *iSys-Brazilian Journal of Information Systems, Porto Alegre*, 10(4):45–65.

Basili, V. R. (1993). Applying the goal/question/metric paradigm in the experience factory. *Software quality assurance and measurement: A worldwide perspective*, 7(4):21–44.

Blas, M. J., Leone, H. P., and Gonnet, S. M. (2020). Building devs models from the functional design of software architecture components to estimate quality. In *II MSSiS*, pages 36–45. SBC.

Bulcão-Neto, R., Teixeira, P., Lebtag, B., Graciano-Neto, V., Macedo, A., and Zeigler, B. (2022). Simulation of IoT-oriented Fall Detection Systems Architectures for In-home Patients. *IEEE Latin America Transactions*, 21(1):16–26.

de Albuquerque, B. V. L., da Cunha, A. F. S., Souza, L., Siqueira, S. W. M., and Santos, R. (2024). Generating and reviewing programming codes with large language models: A systematic mapping study. In *Proc. of the 20th SBSI*, pages 70:1–70:10, Juiz de Fora, Brazil. ACM.

de França, B. B. N. and Graciano Neto, V. V. (2021). Opportunities for simulation in software engineering. In *Anais do III Workshop em Modelagem e Simulação de Sistemas Intensivos em Software*, pages 50–54. SBC.

de Oliveira, G. D., Porto, P. P. G., Alves, C. d. M. A., and Ralha, C. G. (2021). An agent-based model for simulating irrigated agriculture in the samambaia basin in goiás. *Revista de Informática Teórica e Aplicada*, 28(2):107–123.

Gao, C., Lan, X., Lu, Z., Mao, J., Piao, J., Wang, H., Jin, D., and Li, Y. (2023). S ˜3: Social-network simulation system with large language model-empowered agents. *arXiv preprint arXiv:2307.14984*.

Graciano Neto, V. V. and Kassab, M. (2023). *What Every Engineer Should Know About Smart Cities*. CRC Press - Taylor & Francis. 1st Edition. 254 p.

Jiang, C., Xu, H., Dong, M., Chen, J., Ye, W., Yan, M., Ye, Q., Zhang, J., Huang, F., and Zhang, S. (2024). Hallucination augmented contrastive learning for multimodal large language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27036–27046.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.

Monteiro, E. L. and Maciel, R. S. P. (2020). Maturity models architecture: A large systematic mapping. *iSys - Brazilian Journal of Information Systems*, 13(2):110–140.

Moya, C. R. (2021). *Como escolher o teste estatístico: um guia para o pesquisador iniciante*. Câmara Brasileira do Livro, São Paulo, Brazil.

OliveiraJr, E., Cordeiro, A. F. R., and Nascimento, D. (2022). Surveying the open science knowledge in a southern brazilian university. In *OpenSym 2022*, pages 1:1–1:10, Madrid, Spain. ACM.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Zeigler, B. P., Sarjoughian, H. S., Duboz, R., and Souli, J.-C. (2012). *Guide to Modeling and Simulation of Systems of Systems*. Springer.