Reversed Model Verification by Inferring Conceptual Models from Simulation Code

Rumyana Neykova[1] and Derek Groen[1]

Department of Computer Science, Brunel University London

Abstract. Extracting high-level conceptual models from simulation code can benefit model validation and verification, system optimisation, and cross-disciplinary communication. However, conceptual models are often embedded within implementation details, making them difficult to access and interpret. This paper explores the feasibility of using Large Language Models (LLMs) to infer conceptual models from simulation code. We conduct a preliminary investigation on an agent-based simulation (Flee), demonstrating how LLMs can extract key structural, behavioural, and temporal elements. Our results suggest that LLMs can generate meaningful conceptual representations that align with expert-created models, offering potential support for model verification. However, we also identify limitations such as omissions and misinterpretations, highlighting the need for human oversight. While our study is based on a single example, it provides initial insights into the role of LLMs in conceptual model inference and their potential integration into simulation validation workflows.

Keywords: Conceptual Model · LLM · Inference.

1 Introduction

Modern simulation systems rely on complex codebases that encapsulate intricate interactions and dynamic behaviors. Despite their importance, the high-level conceptual models underpinning these simulations often remain implicit within implementation details, hindering model verification, optimization efforts, and interdisciplinary collaboration. Traditionally, conceptual models are either manually extracted by experts—a time-consuming and inconsistency-prone process—or neglected entirely, leaving simulations unverified against their conceptual foundations. Recent advancements in Large Language Models (LLMs) offer new opportunities for automating conceptual model inference from simulation code. This paper investigates whether LLMs can extract accurate conceptual models from simulation code, what types of insights these models can capture, and what limitations need to be addressed for effective integration into validation workflows.

To explore these questions, we apply LLM analysis to infer conceptual models from a real-world agent-based simulation (Flee), focusing on temporal structure,

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_43

entity behavior, and system architecture. We choose Flee because of its wide uptake, and because the authors understand the code deeply, allowing for manual evaluation of results. We evaluate alignment between inferred models and expert-created representations to assess their applicability for model verification. Our findings suggest that LLMs can extract meaningful conceptual elements that support validation, though with limitations requiring human oversight. While based on a single case, this work demonstrates the potential of LLMs as tools for conceptual model inference and outlines directions for integrating these approaches into simulation validation workflows.

2 Related Work

Conceptual Modeling in Simulation Conceptual modeling constitutes the process of abstracting a model from the real world [12]. Simulations represent implemented conceptual models, typically through computer programs, though exceptions exist [5]. Conceptual models are popular because they are easy to understand and interpret for non-developing researchers [7] and because they are essential in the early stages of simulation development [4].

The various conceptual model diagrams found in literature differ in scope and design, often focusing on specific aspects such as time loop dynamics [3, 14], behavioral elements [10, 15], or system architecture [6, 8], yet these are typically created manually rather than extracted from implementation code. Model verification—comparing the conceptual model with its implementation—commonly relies on targeted tests that ensure specific mechanisms adhere to established rules [13]. For example, one might check whether floating point representations in computer code could lead to unintended instabilities in a fluid dynamics algorithm. However, this approach is limited in scope, as many conceptual models encompass multiple mechanisms, while academic literature frequently presents diagrams representing substantial portions of the overall model (e.g., Figure one in Mehrab et al. [9]).

Within this work we attempt to extract these higher level conceptual models directly from simulation code using a LLM-based approach. To our knowledge, we are the first to do so.

LLMs for Software Analysis Recent advances in large language models have led to innovative approaches in automated code analysis and conceptual modeling. Ali et al. [1] demonstrated how LLMs support conceptual modeling by creating, updating, and visualizing UML diagrams through natural language interactions. Similarly, Nicola et al. [2] explored how LLMs can assist business process modeling by extracting process models from text and identifying interaction patterns that inform modeling best practices. Nam et al. [11] showed that LLM-based tools can significantly improve code understanding through contextual explanations of unfamiliar code.

While these studies show promising results, they have not been applied to simulation modeling specifically, nor do they focus on autonomous extraction from existing code. Most prior work targets general code documentation or sup-

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_43

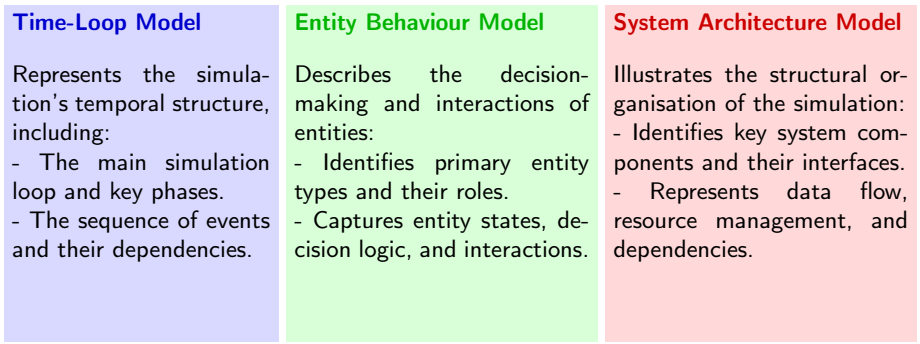

Fig. 1: Types of Conceptual Models Extracted from Simulation Code.

ports human modelers through interactive workflows rather than direct model inference. The unique challenges of simulation code—with its complex temporal structures, entity behaviors, and system interactions—remain unaddressed by current LLM applications.

3 Conceptual Model Extraction: Approach and Case Study

To structure conceptual model inference, we define three conceptual model types that capture distinct aspects of a simulation (Fig. 1). Temporal structure focuses on the time-loop mechanics of the simulation, capturing key phases, event sequences, and dependencies. Understanding the temporal structure is important for validating whether system processes align with expected time-driven behaviours. Behavioural modelling examines entity decision-making and interactions, identifying key agent types, their roles, and the logic governing their state transitions. Behavioural models are particularly relevant for agent-based simulations, where emergent behaviour depends on individual decision rules. System architecture analyses the structural organisation of the simulation, identifying key components, their interfaces, and data flows. Architectural models provide a high-level view of system composition and modular interactions, which is needed for understanding dependencies and debugging. These three dimensions align with established practices in conceptual modelling and provide complementary perspectives on a system's operation. By extracting models across these levels, we assess the extent to which LLMs can infer useful abstractions from raw simulation code.

Case Study: Inferring Conceptual Models from the Flee Simulation. To evaluate LLM-based conceptual model inference, we conducted experiments on Flee [3], an agent-based simulation framework designed to model forced migration patterns. The simulation uses discrete time-stepping to track population movements between locations based on conflict dynamics, decision-making heuristics, and resource availability.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_43

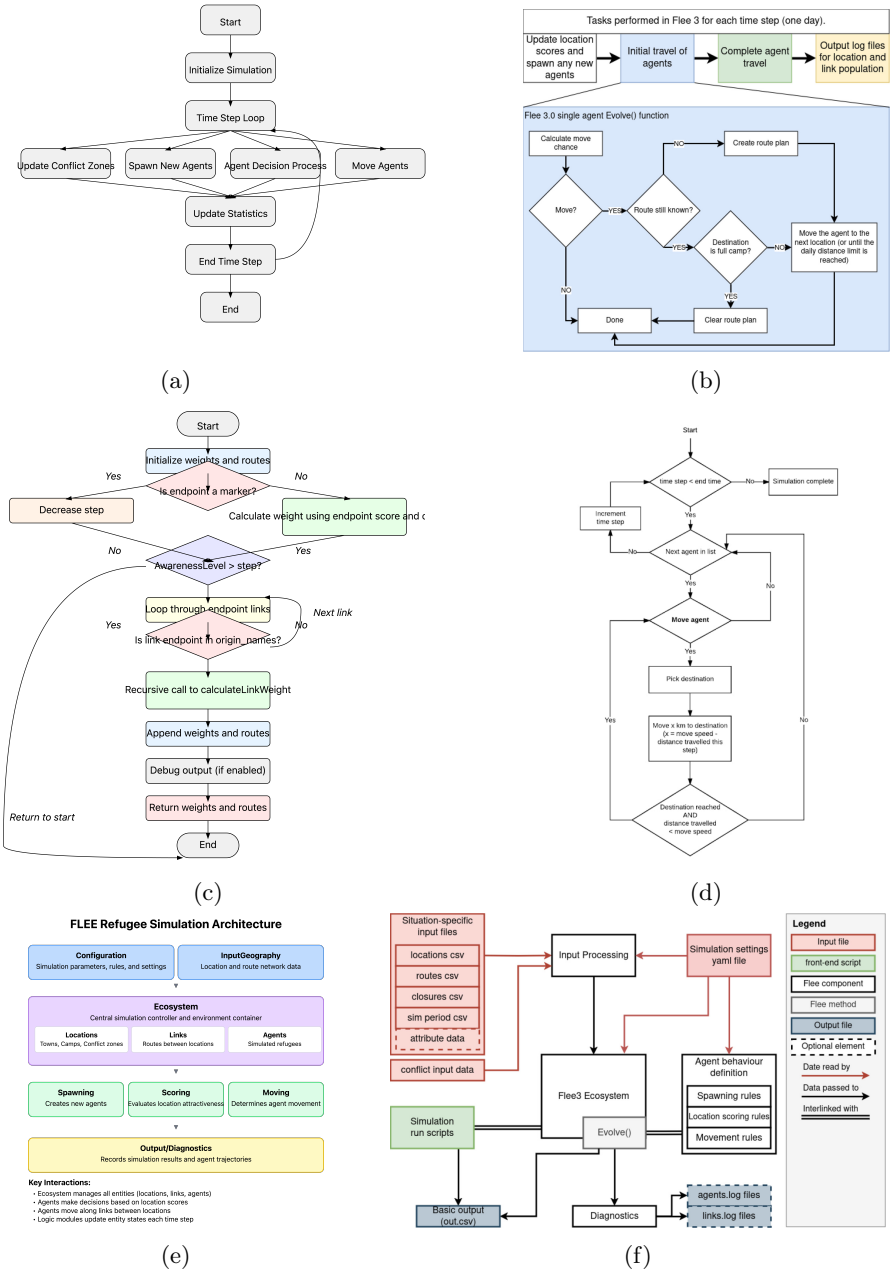

Fig. 2: Comparison of LLM-generated diagrams with literature: (a) LLM-generated time-loop process diagram, (b) time loop process detail from literature [3], (c) LLM-generated agent decision process for calculating link weights, (d) closest corresponding Flee flowchart from literature [15], (e) LLM-generated architectural diagram, and (f) its equivalent in recent literature [3].

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_43

Our analysis was conducted on an 18-file, 5,000-line code base, processed using Claude Sonnet 3.5. The entire source code was provided to the LLM as project knowledge, enabling a holistic understanding of the system. The outputs were generated through zero-shot prompting, explicitly requesting simplified high-level overviews. In the case of Fig. 2e, the initial response contained excessive detail, prompting us to refine our request for a more abstract representation. Additionally, we specified visualization in React.js format rather than Mermaid, both being supported by Claude. To extract conceptual models, we employed structured queries to guide the LLM's analysis. For instance, to obtain the system architecture model shown in Fig. 2e, we used the following initial prompt: *Visualise a high-level architectural diagram of the simulation code in your project knowledge. Showing the main components and their interactions.*

The results are summarised in Fig. 2, which presents examples of inferred time-loop, entity behaviour, and system architecture models. Section 4 evaluates these models by comparing them with the original simulation code and conceptual models from the literature.

4 Evaluation

Within this section, we present three LLM generated diagram examples, and provide a brief reflection on their quality. Our evaluation assesses the LLM-generated conceptual models based on accuracy (alignment with code), completeness (coverage of key elements), and interpretability (clarity of presentation). Table 1 summarizes the quantitative assessment of shortcomings across all three model types.

We present two time loop representations from the Flee ABM in Fig. 2a–2b. The LLM-generated diagram effectively captures most conceptual elements with correct terminology, but shows four specific inconsistencies: (i) omitted score updating for non-conflict zones, (ii) mislabeled "update statistics" instead of output writing, (iii) incomplete agent decision process missing the finish travel task, and (iv) incorrect sequencing of conflict zone updates. Table 1 indicates only one missing concept element, suggesting high fidelity. Both LLM and literature diagrams show strong consistency, with the former highlighting the flow from beginning to end (the literature diagram focuses only on the loop iterations), and the latter providing more zoom-in detail on the agent decision-making process.

We present agent decision representations in Fig. 2c–2d. The LLM diagram captures the link weight calculation algorithm comprehensively despite formatting issues, with three specific shortcomings: (i) the loop arrow is inverted, (ii) the "Yes" part in the endpoint link loop seems to be placed one arrow too high, (iii) the arrows around the "is link endpoint in origin_names" are incorrectly arranged. Table 1 shows arrow labeling as the most significant issue, with four incorrect labels. Interestingly, the closest conceptual diagram we could locate in the literature is not similar at all, as all the logic in the LLM-generated figure is only represented by a single box ("pick destination"). Therefore, despite its shortcomings, the LLM-generated diagram exposes important conceptual el-

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_43

ements of the Flee code, particularly because the generation and weighting of possible agent routes is a fundamental aspect of the agent decision-making. In addition, the LLM has successfully picked up the concept and functionality of marker locations (including the need to adjust step numbers), which is something that has not been clearly covered in any of the existing literature.

Fig.2f shows architectural diagrams of Flee. The LLM version depicted in Fig.2e identifies essential components—inputs (top), ecosystem objects (top middle), dynamical components (middle bottom), and outputs (bottom)—connected via triangular indicators. The literature diagram (Fig.2f) offers more detailed relations and explicit file type names, though both share clear commonalities in ecosystem and agent behavior modules. Key shortcomings in the LLM diagram include: (i) missing backlinks from behavioral definitions to Ecosystem objects and (ii) vague configuration elements. Table 1 shows this diagram contains fewer issues than the other conceptual models.

Across all three model types, we observe that LLMs excel at identifying main components and general flows, but struggle with precise relationship details (particularly directional relationships) and specific technical terminology. The time loop model exhibits the highest overall accuracy, likely due to its explicit sequential structure in the code, while the agent decision model presents the most challenges in accurately representing complex conditional logic.

5 Discussion and Conclusions

Conceptual model extraction has not been widely explored for model verification because manual extraction is often tedious and sometimes intractable. The ease with which LLMs can generate various levels of conceptual model details from implementations urges revisiting the role these extractions can play in understanding and verifying simulation behavior. What was previously considered impractical due to the effort involved can now potentially become an integral part of simulation development and validation workflows.

Our preliminary investigation demonstrates that LLMs offer potential for conceptual model inference across multiple dimensions of simulation systems. This allows for simulation verification at multiple levels: from time-stepping

Table 1: Evaluation overview of LLM-extracted conceptual diagrams

Shortcoming	# of occurrences for each diagram		
	time_loop	agent_decision	architecture
Wrong arrows	1	2	1
Wrong arrow labels	0	4	0
Non-diagram elements	0	0	4
Missing relevant concept element	1	0	0
Overly vague concept element	2	0	1

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_43

mechanics to agent decision logic to system composition. LLMs can identify relationships that would otherwise require manual analysis, provide readable abstractions for domain experts without programming knowledge, and surface inconsistencies between implementation and intended behavior. We have demonstrated that it is possible to create high-fidelity conceptual models automatically. The quality of the extracted models in our case study suggests that the potential benefits substantially outweigh the current limitations, warranting further exploration of this approach.

Our study raises important questions about how code commenting style, documentation quality, and naming conventions influence conceptual model extraction. While we haven't explicitly tested this influence, we suspect that well-structured comments and semantically meaningful filenames likely enhance LLM extraction performance significantly. For simulation developers, this suggests an opportunity to strategically instrument code with conceptual markers, standardized documentation patterns, or explicit annotations that could guide more accurate model extraction. The optimal format and extent of such instrumentation remains unexplored, presenting a promising direction for developing standards that maximize verification benefits while minimizing documentation overhead.

In terms of limitations, we have shown that inferred models can exhibit imprecise event sequencing, oversimplified interactions, and miss subtle conceptual elements. Expert validation therefore remains necessary to refine the LLM outputs. In addition, our explorative study examines one simulation code base (Flee), so assessing broader applicability requires further research. In general, reproducibility presents a significant challenge for LLMs. As their outputs can vary between runs and across different model versions, the conceptual models generated may not be consistent.

In recent times, developers increasingly rely on LLMs, instead of programming skills, for application development. Our approach could help these developers to check whether LLM-guided implementations align with their conceptual vision. Furthermore, our findings suggest several integration directions: implementing regular conceptual extraction during development, analyzing models from different implementations of the same system, linking conceptual elements to specific code segments, and creating interfaces for expert review.

Rather than a complete solution, our work serves as a call to action for the simulation community to explore these capabilities further. By investigating across diverse simulation domains and developing specific methodologies for extraction and verification, we can improve the reliability and transparency of simulation-based research.

Acknowledgments. This work has been supported by the SEAVEA ExCALIBUR project, which has received funding from EPSRC under grant agreement EP/W00771/1.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_43

References

1. Ali, S.J., Reinhartz-Berger, I., Bork, D.: How are LLMs Used for Conceptual Modeling? An Exploratory Study on Interaction Behavior and User Perception. In: Maass, W., Han, H., Yasar, H., Multari, N. (eds.) Conceptual Modeling, pp. 257–275. Springer Nature Switzerland (2025). https://doi.org/10.1007/978-3-031-75872-0_14
2. De Nicola, A., Formica, A., Mele, I., Missikoff, M., Taglino, F.: A comparative study of LLMs and NLP approaches for supporting business process analysis. Enterprise Information Systems **18**(10), 2415578 (2024). https://doi.org/10.1080/17517575.2024.2415578
3. Ghorbani, M., Suleimenova, D., Jahani, A., Saha, A., Xue, Y., Mintram, K., Anagnostou, A., Tas, A., Low, W., Taylor, S.J., et al.: Flee 3: Flexible agent-based simulation for forced migration. Journal of Computational Science **81**, 102371 (2024)
4. Groen, D., Suleimenova, D., Jahani, A., Xue, Y.: Facilitating simulation development for global challenge response and anticipation in a timely way. Journal of Computational Science **72**, 102107 (2023)
5. Holmberg, E.: On the clustering tendencies among the nebulae. ii. a study of encounters between laboratory models of stellar systems by a new integration procedure. Astrophysical Journal, vol. 94, p. 385 **94**, 385 (1941)
6. Hussan, J.R., Hunter, P.J.: Comfort simulator: A software tool to model thermoregulation and perception of comfort. Journal of Open Research Software **8**(1), 16 (2020). https://doi.org/10.5334/jors.288
7. Liu, J., Yu, Y., Zhang, L., Nie, C.: An overview of conceptual model for simulation and its validation. Procedia engineering **24**, 152–158 (2011)
8. Luo, L., Zhou, S., Cai, W., Low, M.Y.H., Tian, F., Wang, Y., Xiao, X., Chen, D.: Agent-based human behavior modeling for crowd simulation. Computer Animation and Virtual Worlds **19**(3-4), 271–281 (2008). https://doi.org/10.1002/cav.238
9. Mehrab, Z., Stundal, L., Venkatramanan, S., Swarup, S., Lewis, B., Mortveit, H.S., Barrett, C.L., Pandey, A., Wells, C.R., Galvani, A.P., et al.: An agent-based framework to study forced migration: A case study of ukraine. PNAS nexus **3**(3), pgae080 (2024)
10. Molina, T., Ortega, J., Muñoz, J.: HELMpy, open source package of power flow solvers. Journal of Open Research Software **9**(1), 23 (2021). https://doi.org/10.5334/jors.310
11. Nam, D., Macvean, A., Hellendoorn, V., Vasilescu, B., Myers, B.: Using an LLM to Help With Code Understanding. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. pp. 1–13. ACM (2024). https://doi.org/10.1145/3597503.3639187
12. Robinson, S.: Conceptual modeling for simulation. In: 2013 Winter Simulations Conference (WSC). pp. 377–388. IEEE (2013)
13. Sargent, R.G.: Verification and validation of simulation models. In: Proceedings of the 2010 winter simulation conference. pp. 166–183. IEEE (2010)
14. Schmieschek, S., Shamardin, L., Frijters, S., Krüger, T., Schiller, U., Harting, J., Coveney, P.: LB3D: A parallel implementation of the Lattice-Boltzmann method for simulation of interacting amphiphilic fluids. Computer Physics Communications **217**, 149–161 (2017). https://doi.org/10.1016/j.cpc.2017.03.013
15. Suleimenova, D., Bell, D., Groen, D.: A generalized simulation development approach for predicting refugee destinations. Scientific reports **7**(1), 13377 (2017)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97635-3_43