Automatic generation of ABM Narratives using Simulation Traces and LLM

Zenith ARNEJO[1,2][0000−0002−9795−1974], Benoit GAUDOU[1][0000−0002−9005−3004], Mehdi SAQALLI[3][0000−0001−5405−466X], and Nathaniel BANTAYAN[4][0000−0002−2398−7720]

[1] UMR 5055, IRIT, Université Toulouse Capitole, Toulouse, 31000, France
[2] Institute of Computer Science, University of the Philippines Los Baños, Laguna, 4030, Philippines
[3] UMR 5602, GEODES, CNRS, Université Toulouse Jean Jaures, Toulouse, 31000, France
[4] Institute of Renewable Natural Resources, University of the Philippines Los Baños, Laguna, 4030, Philippines
zenith.arnejo@ut-capitole.fr

Abstract. Effective communication of agent-based models is essential for ensuring their usability and transparency. However, conventional documentation approaches often struggle to capture the dynamic execution details of simulations, making it challenging to convey complex processes clearly and accessibly. Moreover, ABMs frequently exhibit emergent and unexpected behaviors resulting from multiple agent interactions—dynamics that static model descriptions does not fully capture. This paper presents a novel methodology for generating execution-based narratives for ABMs using simulation logs and large language models. By integrating process mining, Business Process Modeling Notation, and automated narrative generation, the approach transforms raw simulation data into coherent visual and textual artifacts that faithfully reflect the model's dynamic execution. Unlike conventional documentation—which often relies on subjective assessments and demands significant effort from modelers—this methodology minimizes subjectivity and reduces the effort required from modelers while promoting a more accessible approach to model communication. To demonstrate its expressivity, we applied the methodology to the Luneray Flu Model and successfully produced artifacts such as process maps, business process diagrams, and narrative explanations. This work offers a step toward improving transparency and accessibility in ABM verification and communication.

Keywords: ABM Narratives · Communication Support · LLM

1 Introduction

Effective communication among model developers and stakeholders is essential in modeling and simulation [11]. This is particularly true for agent-based models (ABMs), where a model's value depends on the clarity with which its design and results are described to stakeholders [14]. Clear communication fosters

better understanding, supports validation efforts, and enhances model reproducibility [10]. However, the diverse and inconsistent methods used to describe models—and the lack of assurance that the implementation, simulation, and results align with the description—hinder stakeholder understanding and raise concerns about model transparency [10]. Importantly, ABMs are characterized by complex interactions among agents that often give rise to emergent behaviors and unexpected outcomes [5]. These dynamic phenomena can remain hidden or only partially described in traditional static documentation frameworks, such as the ODD framework, which relies exclusively on exhaustive textual descriptions [9]. Consequently, static descriptions may overlook critical aspects of a model's capabilities that only become apparent during execution.

Given these challenges, there is a need for innovative methods that align model descriptions with the dynamic nature of model execution while minimizing modelers' subjective bias. This paper presents a novel methodology for generating execution-based narratives for ABMs using simulation logs and large language models (LLMs). The approach combines process mining to discover agent workflows, visualization through Business Process Modeling Notation (BPMN), and automated narrative generation with LLMs. This integration transforms raw simulation data into coherent visual and textual artifacts that closely mirror the model's execution dynamics. By capturing the behaviors intrinsic to ABMs, our methodology not only improves transparency but also supports model verification and stakeholder engagement by providing accessible and accurate representations of simulation processes. The remainder of the paper is structured as follows: the Related Work section reviews existing contributions to model descriptions in the context of model communication; the Methodology section outlines the pipeline for automatically generating model descriptions using simulation traces and LLMs; the Application section demonstrates the methodology with a toy model; and the Discussion section analyzes the findings, identifies limitations, and summarizes the contributions.

2 Related work

Model descriptions are vital artifacts in any modeling project, serving multiple purposes: they facilitate communication, deepen comprehension, enable replication, and allow comparisons with other models [16]. The quality of these descriptions is critical to the success of a project and the usability of its outputs. However, one persistent challenge is balancing detail with readability. Comprehensive descriptions necessary for replication can be overwhelming for complex models, while overly general descriptions risk omitting critical information or creating inconsistencies with real-world systems. This balancing act remains a significant barrier to improving model descriptions for ABMs.

One promising approach to address this issue is the use of narrative explanations. In the work of [15], narrative explanations were employed to clarify the results of generative simulation models, providing detailed accounts of key events and processes during simulations. These narratives act as intermediaries

between high-level summaries of system-level patterns and formal descriptions of model structures, making them particularly useful for participatory modeling projects. Similarly, [3] demonstrated the effectiveness of narrative explanations (or scientific storytelling) in communicating complex human-environment interactions within a transdisciplinary ABM. While these approaches show promise, current methods rely heavily on manual efforts, which are time-consuming and prone to bias and inconsistencies.

A persistent criticism of ABMs is the perception that they function as "black box" systems, leading to hesitancy among researchers and stakeholders to adopt them [6, 21]. While model documentations and narrative explanations are crucial in addressing this issue, they often fall short due to the time-intensive nature of documentation and the risk of oversimplifying complex dynamics. This underscores the need for automated approaches to create descriptions that are transparent, accessible to non-experts, and faithful to the model's complexity, while minimizing the effort required for their production.

3 Methodology

In this work, we present a three-part methodology for the automatic generation of model narratives for any given discrete-time simulation.

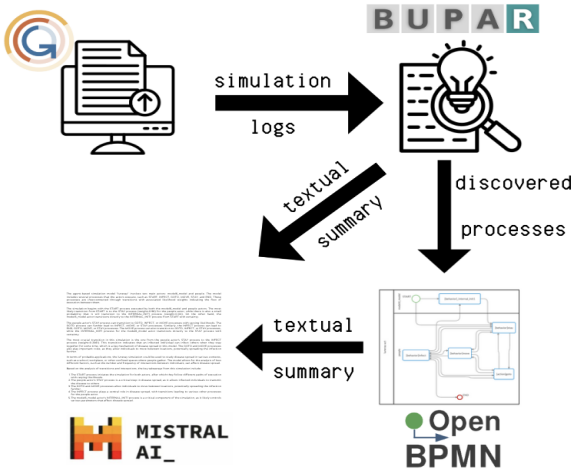

Fig. 1. Methodology for automatically generating ABM narratives

Figure 1 illustrates the proposed methodology. The primary objective is to reveal the inner workings of the model without relying on its documentation. To achieve this, we focused on extracting information directly from simulation traces or logs using "process mining" [2]. We employed automated process discovery, a key technique in process mining, to derive processes directly from these logs and present them in the form of a process map. The discovered workflow

is then further visualized using Business Process Model and Notation (BPMN) to organize the process according to the agents that perform it. Subsequently, a textual summary of the diagrams is generated and used as input for querying the selected large language model (LLM). The methodology produces two key artifacts: (1) process diagrams, including process maps (PM) and business process diagrams (BPD), and (2) a simulation narrative.

3.1 Simulation Logging

The methodology begins by capturing simulation logs at each simulation step during model execution, focusing on two types of information: (1) variables whose values change during execution and (2) methods that are executed. Logging the variables whose values change provides insights into which properties of the agent evolve throughout the simulation, while capturing executed methods reveals the agent's actions during the process. These elements were chosen for logging because they directly reflect the simulation's dynamic state transitions and decision points, providing a comprehensive view of agent behavior. This approach is not limited to a specific simulation platform; it can be applied to any platform that supports discrete-time simulation. Since discrete-time simulations inherently progress through distinct time steps, the logging of variable changes and method executions can be consistently implemented, regardless of the underlying architecture. In this study, we utilized the GAMA platform [22] to maximize the use of readily available models and then integrated logging capabilities within the platform.

case_id	activity	time_stamp	value	resource
2	[action]description.segregation_model	01/01/1970 01:00	nil	nil
2	nil	01/01/1970 01:00	27846	[Variable]rng_usage.segregation_model
2	nil	01/01/1970 01:00	1601	[Variable]all_places.segregation_model
2	nil	01/01/1970 01:00	1601	[Variable]free_places.segregation_model
2	[action]initialize_places.segregation_model	01/01/1970 01:00	nil	nil
2	nil	01/01/1970 01:00	1120	[Variable]number_of_people.segregation_model
2	[behavior]_internal_init49.people	01/01/1970 01:00	nil	nil

Fig. 2. Example simulation log

Figure 2 presents sample entries from the simulation log. The column details are as follows: (1) "case_id" field indicates the execution cycle for a single experiment or the simulation replication for a batch experiment.; (2) "activity" specifies the executed method in the format [<type>]<method name>.<agent type>; (3) "time_stamp" records the simulation clock; (4) "value" represents the current value of the variable; and (5) "resource" provides information on the variable that changed, formatted as [Variable]<variable name>.<agent type>. For the activity type, in the context of the GAMA Modeling Language (GAML), this can be either a behavior (referred to as a "reflex" in GAML) or an action. Moreover, for log entries corresponding to methods, the 'value' and 'resource' fields are assigned 'nil', whereas for log entries corresponding to variables, the 'activity' field is assigned 'nil'. The organization of the simulation log is based on the minimum information required to describe an 'event,' which, according to [2], includes a case identifier, an activity name, and a timestamp. This information is necessary to proceed to the next step of the methodology.

3.2 Process Visualization

The subsequent step in the methodology focuses on generating process visualizations. This step begins with applying process discovery techniques to simulation logs to derive a process model that accurately represents the observed process behavior [1]. By treating simulation logs as event logs, the underlying simulation processes are visualized as process maps (PM). These PMs are constructed using the Directly-Follows Graph (DFG) algorithm, a widely supported process discovery approach in process mining tools [1]. DFG represents processes as a directed graph, where each node denotes an activity, and each directed edge captures the sequence in which activities occur based on the order of entries in the event log. Additional information captured includes transition likelihoods on edges, computed based on the frequency of transitions, and node proportions, which indicate the occurrence frequency of specific activities relative to the total log entries.

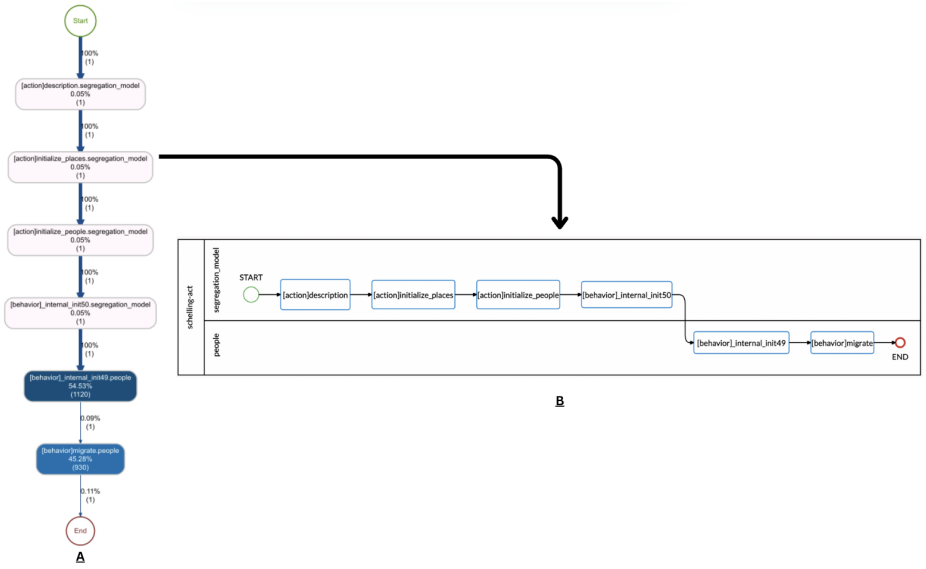

Fig. 3. Example PM (A) and its corresponding BPD (B)

To generate separate PMs for methods and variables, we first filtered the logs and then conducted two distinct process discovery analyses. For the methods map, we retained only log entries where the activity field was not nil. Conversely, for the variables map, we retained only entries where the resource field was not nil. Using the DFG algorithm, the filtered logs were processed sequentially: for each pair of consecutive, distinct log entries, an edge was created, while consecutive identical entries resulted in an incremented edge weight. For example, as shown in Figure 2, after filtering out nil entries for the methods map, the log entry [action]description.segregation_model is followed by [action]initialize_places.segregation_model. Accordingly, Figure 3A displays an edge connecting the nodes corresponding to these log entries. Additionally, be-

cause the log file begins with [action]description.segregation_model, an edge from the START node to this entry was also created. Lastly, to enhance the readability of the generated process map, we filtered out loops and retained only edges with a weight of at least 0.05%.

In addition to the PM, we visualized the extracted process using Business Process Model and Notation (BPMN)—the de facto standard for representing business processes in a highly expressive graphical format [7]. BPMN provides elements such as pools and lanes to organize processes according to the agents that perform them. Using this notation, the resulting diagram features pools that encompass all logged aspects of the simulation, while the lanes correspond to each involved agent type. An example business process diagram (BPD) is provided in Figure 3B. In this example, the pool is named "schelling-act," representing the discovered process based on the action logs of the Schelling simulation, while the lanes include a "people" lane—representing the sole agent type—and a "segregation_model" lane, which corresponds to the simulation environment. This visualization aims to enhance the understanding of process interactions and agent roles. The use of BPMN for communicating agent-based models (ABM) to business users has also been explored in [17] and [18].

To achieve these transformations, we utilized the tools from the bupaR ecosystem [12], a business process analysis toolkit for R, for process discovery and visualization. The BPMN version of the PM was generated using the BPMN 2.0 metamodel available in OpenBPMN [20], while Cardanit [8] was used to generate a BPD that has been automatically layouted.

3.3 Narrative Generation

The final step in the methodology is narrative generation. The creation of this artifact is motivated by the widely held notion that narratives enhance human understanding by structuring events into narratives [4]. According to [15], a narrative is defined as a means of explaining the sequences of events and interactions within a system, integrating them into a coherent account to illustrate how patterns emerge from those sequences. In our case, since the focus of our work is on exposing the processes of a given ABM, we limit our definition of narrative to textual descriptions that present the sequences of events and interactions in a simulation model, describing its most important processes and variables.

To achieve this, we begin by automatically generating textual summaries of the diagrams created in the previous step. These summaries serve as the foundation for the narrative explanation and are produced with the context length limit in mind—the maximum number of tokens that a large language model (LLM) can process at one time. Next, we incorporate the LLM into the methodology to generate narrative explanations based on these summaries. A key requirement for producing a coherent narrative is that the method and variable names in the ABM's implementation code align with real-world terms that accurately reflect what each represents, as this information is captured in the simulation log and, in turn, reflected in the textual summaries. In this study,

we employed the 7-billion-parameter language model developed by Mistral AI [13] to generate the narrative explanation for the ABM.

Prompt Creation. To ensure that the LLM generates a coherent narrative, we conducted prompt engineering to identify the most effective prompt. Since the previous step produces four diagrams, resulting in four textual summaries, we began with three user prompts. For the first prompt, we described the task of creating narratives based on descriptions of the variables and processes of the same simulation model that would be provided. We also specified the types of descriptions expected: the first being a textual summary based on a directed weighted graph, and the second based on a business process diagram. This approach ensured that the LLM had all the necessary information before generating a narrative, while also addressing the fact that this information could not be combined into a single prompt due to the LLM's context length limit. However, after several tests, we noticed that the LLM struggled to retain the information provided and generated from previous prompts, highlighting the limitation of current LLMs in maintaining long-term coherence [4].

Action Log		Variable Log		Emphasis
PM	BPD	PM	BPD	
x	x			Actions executed in the simulation and the connections between agent actions
		x	x	Agent variables that changed during the simulation and interactions between agents' attributes
x		x		Transitions in the executed methods and agent variables
	x		x	Structure of the simulation based on agent attributes and behavior

Table 1. Combinations of textual summaries with corresponding narrative emphasis

Finally, we decided to consolidate all prompts into a single user prompt for analyzing combinations of textual summaries (see Table 1). Simultaneously, we switched from using a user prompt to a system prompt to optimize model performance. Lastly, we appended the textual summaries to the prompt. Below is the system prompt used to query the chosen LLM.

"You are provided with detailed descriptions of a complete simulation execution for an agent-based model. Your task is to: (1) Analyze interactions through careful examination of the interactions between processes and agents within the model; (2) Identify key transitions by highlighting crucial transitions and important paths within the simulation; and (3) Develop a comprehensive narrative that explains the overall purpose and mechanism of the agent-based model, the probable applications of the model in real-world or theoretical contexts, and the key insights and takeaways derived from the analysis of interactions and transitions. Below are the descriptions: "

4 Application

To demonstrate the methodology, we applied it to one of the toy and pedagogical models available on the GAMA platform—a simple susceptible-infected (SI) model simulating the spread of flu in the city of Luneray, Normandy, France [22]. In this model, people agents move from building to building via the road network, and infected individuals can transmit the flu to their neighboring agents. According to the model's description, five key modeling choices guide its implementation: (1) People move along the roads from building to building; (2) People use the shortest path to travel between buildings; (3) All individuals move at a constant speed; (4) Upon arriving at a building, people stay there for a specific period of time; and (5) Infected individuals are never cured.

Results. To collect the simulation logs, we conducted a batch simulation consisting of six runs, each comprising 600 cycles. Each run involved 2,147 people agents and was executed with identical parameter settings, with log files recorded at each cycle throughout the simulation. This approach ensures that the resulting process represents a general overview rather than the outcome of a single simulation. In total, we obtained six log files, each averaging 150 MB. After applying automated process discovery, we generated a process map (PM) from the action log files, as shown in Figure 4A, and the corresponding business process diagram (BPD) is presented in Figure 4B.

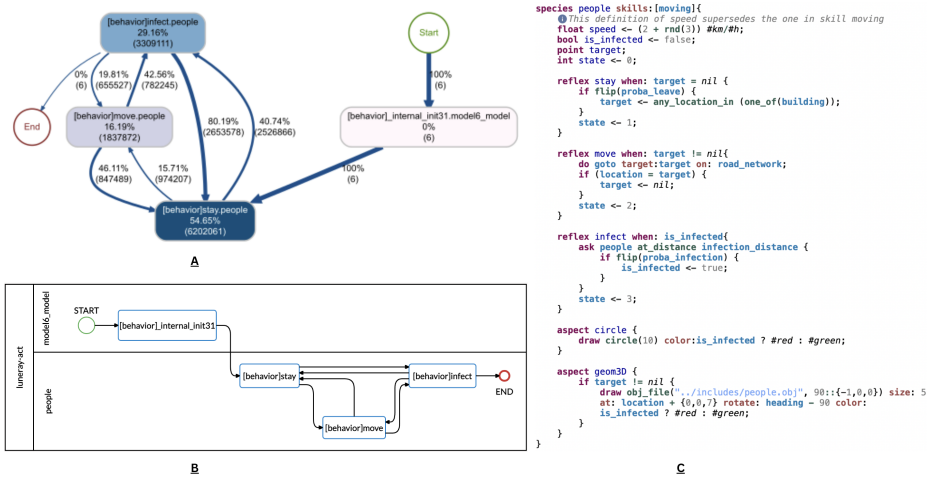

Fig. 4. (A) PM derived from simulation action logs; (B) Corresponding BPD of A; and (C) Actual implementation code for the "people" agent type

.

Since there is only one main agent type in the simulation — the people agent — the generated BPD displays two lanes (see Figure 4B). The first lane, labeled model6_model, represents the simulation environment, corresponding to the simulation model's actual name ("model6") as defined in the implementation code. The second lane, labeled people, represents the only agent type

present in the simulation. In the diagram, the START node connects to the [behavior]_internal_init31 method in the model6_model lane, which is responsible for instantiating the agents. Following instantiation, the next method is [behavior]stay, indicating that the first action executed by the agents is "stay." Moreover, the infect method is directly connected to the END node, as the log file shows that the final recorded event before termination is an infect method. To further clarify the interconnections depicted in the diagram, Figure 4C presents the implementation code for the people agent type.

Afterward, textual summaries of the diagrams were automatically generated and used to complete the prompt for querying the LLM to produce a narrative. As discussed in the previous section, multiple narratives can be generated based on the textual summaries. In this paper, we present the narrative derived from the textual summaries based on the BPD and PM of the action logs, shown in Figure 5.

Comprehensive Narrative
Overall Purpose and Mechanism
The agent-based model appears to simulate the behavior of individuals (people) in response to an infectious process. The simulation begins with the START event, which triggers the _internal_init31 task executed by model6_model. This initialization leads directly to the stay state for the people actor.

From the stay state, individuals can transition to either the infect or move states. The infect state represents a scenario where an individual becomes infected and has a high likelihood (0.8013) of staying in that state. However, there is also a low likelihood (0.1987) of moving from the infect state to the move state.

The move state indicates that individuals are mobile and can transition back to the infect state with a moderate likelihood (0.4369) or stay in the stay state with a similar likelihood (0.4517). The stay state allows for transitions back to either the infect or move states, with moderate and low likelihoods respectively.

Probable Applications
This model can be applied in various real-world and theoretical contexts:

- *Epidemiology : Simulating the spread of infectious diseases within a population.*
- *Urban Planning : Understanding how mobility patterns affect disease transmission.*
- *Public Health Policy : Evaluating the effectiveness of interventions that encourage staying at home versus moving around.*

Key Insights and Takeaways
1. *Initialization Phase : The model starts with an initialization phase (_internal_init31) that sets individuals to a stay state, suggesting a controlled or quarantined starting condition.*
2. *Infection Dynamics : The high likelihood of staying in the infect state indicates that once infected, individuals are likely to remain so for some time.*
3. *Mobility Impact : The moderate transition probabilities between move, stay, and infect states highlight the complex interplay between mobility and infection risk.*
4. *End State : The absence of a direct transition from infect to END (weight = 0.0) suggests that the model does not account for recovery or removal from the infectious state, which could be a limitation.*

Overall, this agent-based model provides a framework for understanding how individual behaviors and transitions between states can influence the spread of an infection within a population. The insights gained can inform strategies to mitigate disease transmission by adjusting mobility patterns and intervention measures.

Fig. 5. Generated narrative for the Luneray Flu model

Verification. To verify the consistency of the visualized processes, we manually compared the PM with the implementation code, as shown in Figure 4C. Since the GAMA platform typically executes an agent's code in a procedural manner, we observe that the first action for most agents is the "stay" reflex. However, if the condition $target = nil$ is not met, the agent proceeds to execute the next reflex. The subsequent reflexes are the "move" reflex, followed by the "infect" reflex, provided that the conditions for each reflex are satisfied. As the generated diagram represents the entire simulation execution, internal connections between the various actions are observed. For instance, one agent might end its cycle with the "move" reflex, while another agent of the same type (or species) may begin its next cycle with the "infect" reflex. These internal connections reflect the different processes that run concurrently during the simulation. However, since "aspect" functions in the GAMA platform do not represent an agent's behavior

but rather describe how the agent is displayed, they are neither reflected in the visualization nor stored in the simulation log.

To evaluate the integrity of the narrative, we compared the differences between the source text and the LLM-generated text using Jaccard Similarity measure [19]. This metric has previously been used in [4] to evaluate the narrative generation capabilities of LLMs. For the Jaccard coefficient, we determined a 53.60% similarity, indicating that the input and generated narratives share approximately half of their elements. This result is ideal, as we aim for the narrative not only to provide simulation explanations but also to make inferences that provide insights into the simulation.

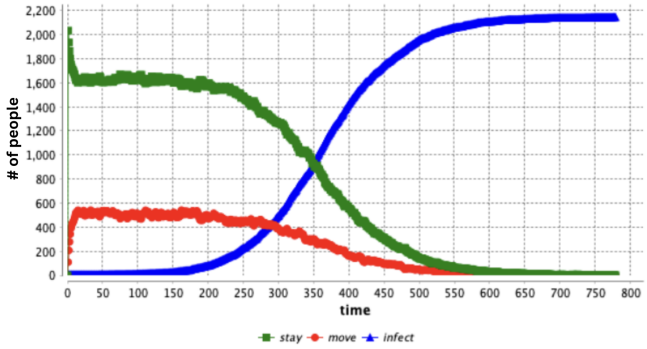

Fig. 6. Distribution of the people agents executing specific methods over time

To assess the fidelity of the generated narrative, we examined whether its insights aligned with simulation observations. According to the narrative, during the initialization phase, people agents are more likely to execute the "stay" method simultaneously. This observation is supported by the visualizations from a single execution of the Luneray Flu model, as shown in Figure 6. At the beginning of the simulation, all people agents execute the "stay" method, and this behavior gradually declines until around the 30th cycle, as indicated by the green trend. Regarding infection dynamics, the narrative suggests that once a people agent becomes infected, it is likely to remain infected. This trend is also evident in the simulation results—Figure 6 shows that once an agent executes the infect method, it continues to do so in subsequent cycles. Additionally, the narrative correctly identifies that the model does not account for recovery or removal from the infectious state. This aligns with the modeling assumption stated during implementation: "Infected individuals are never cured." These observations confirm that the generated narrative provides valid insights, even though the input data does not explicitly specify these trends and constraints.

5 Discussion

The example provided in the previous section illustrates how the methodology—using simulation logs to automatically generate ABM narratives—can offer a more transparent and accessible description of an agent-based model. This

methodology produces three key artifacts: process map (PM), business process diagrams (BPD), and simulation narratives.

PMs provide an overview of how processes are executed during simulation by revealing critical methods and variables. Although clutter and information overload can be expected in visualizations of large, complex models, this issue can be mitigated by filtering nodes to include only those with weights within specific thresholds. Similarly, BPDs, derived from PMs, provide a structured view of agent interactions and relationships, aligning with the agent-based modeling paradigm. In addition to their role as standalone visual artifacts, BPDs and PMs are crucial for creating an effective prompt for the LLM. The structured and filtered representations provided by the PMs and BPDs distill the essential elements of the simulation—such as event sequences, critical transitions, and inter-agent interactions—into a concise and context-rich format. The generated narrative, which is grounded in both the implementation code and the simulation execution, complements these visualizations by offering an accessible description of the model. It helps viewers interpret the visualizations when they become overwhelming, thus ensuring a comprehensive understanding of the simulation. Together, the visualizations and the narrative form a robust foundation for a detailed description of the agent-based model, with the fully automated process significantly minimizing the effort required to produce these artifacts.

These artifacts enhance the understanding of simulation models across diverse audiences by serving as communication tools for stakeholders, complementary documentation for modelers, and structured narratives for researchers. While currently limited to representing agent workflows based on modified variables and executed functions, future improvements—such as logging critical input parameters and linking them to observable outputs—could further refine their utility. Nonetheless, the integration of BPDs, PMs, and generated narratives significantly contributes to better model communication by providing a structured overview of processes, inter-agent interactions, and key variables, thereby enhancing model's transparency and accessibility.

References

1. van der Aalst, W.M.P.: Foundations of Process Discovery, pp. 37–75. Springer International Publishing, Cham (2022)
2. van der Aalst, W.M.P.: Process mining: A 360 degree overview. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook, Lecture Notes in Business Information Processing, vol. 448, pp. 3–34. Springer (2022)
3. Allison, A., Dickson, M., Fisher, K., Thrush, S.: Communicating Drivers of Environmental Change Through Transdisciplinary Human-Environment Modeling. Earths Future **9**(9) (Sep 2021)
4. Bartalesi, V., Lenzi, E., De Martino, C.: Using large language models to create narrative events. PeerJ Computer Science **10**, e2242 (Oct 2024)
5. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences of the United States of America **99**(10), 7280–7287 (2002)

6. Cartwright, S., Bowgen, K., Collop, C., Hyder, K., Nabe-Nielsen, J., Stafford, R., Stillman, R., Thorpe, R., Sibly, R.: Communicating complex ecological models to non-scientist end users. Ecological Modelling **338**, 51–59 (Oct 2016)
7. Chinosi, M., Trombetta, A.: Bpmn: An introduction to the standard. Computer Standards & Interfaces **34**(1), 124–134 (2012)
8. ESTECO SpA: Cardanit (10-18-2024), https://www.cardanit.com/
9. Grimm, V., Railsback, S.F., Vincenot, C.E., Berger, U., Gallagher, C., DeAngelis, D.L., Edmonds, B., Ge, J., Giske, J., Groeneveld, J., Johnston, A.S., Milles, A., Nabe-Nielsen, J., Polhill, J.G., Radchuk, V., Rohwäder, M.S., Stillman, R.A., Thiele, J.C., Ayllón, D.: The odd protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism. Journal of Artificial Societies and Social Simulation **23**(2), 7 (2020)
10. Grueau, C.: Towards a domain specific modeling language for agent-based modeling of land use/cover change. In: Parsons, J., Chiu, D. (eds.) Advances in Conceptual Modeling. pp. 267–276. Springer International Publishing, Cham (2014)
11. Haveman, S., Bonnema, G.: Communication of simulation and modelling activities in early systems engineering. Procedia Computer Science **44** (03 2015)
12. Janssenswillen, G., Depaire, B., Swennen, M., Jans, M.J., Vanhoof, K.: bupaR: Enabling reproducible business process analysis. Knowledge-Based Systems **163**, 1857 (2019)
13. Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D.d.l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al.: Mistral 7b. arXiv preprint arXiv:2310.06825 (2023)
14. Lee, J.S., Filatova, T., Ligmann-Zielinska, A., Hassani-Mahmooei, B., Stonedahl, F., Lorscheid, I., Voinov, A., Polhill, G., Sun, Z., Parker, D.: The complexities of agent-based modeling output analysis. JASSS **18**(4) (2015)
15. Millington, J.D., O'Sullivan, D., Perry, G.L.: Model histories: Narrative explanation in generative simulation modelling. Geoforum **43**(6), 1025–1034 (Nov 2012)
16. Müller, B., Balbi, S., Buchmann, C.M., de Sousa, L., Dressler, G., Groeneveld, J., Klassert, C.J., Le, Q.B., Millington, J.D., Nolzen, H., Parker, D.C., Polhill, J.G., Schlüter, M., Schulze, J., Schwarz, N., Sun, Z., Taillandier, P., Weise, H.: Standardised and transparent model descriptions for agent-based models: Current status and prospects. Environmental Modelling & Software **55**, 156–163 (2014)
17. Onggo, B.S.S.: Bpmn pattern for agent-based simulation model representation. In: Proceedings of the 2012 Winter Simulation Conference (WSC). pp. 1–10 (2012)
18. Onggo, B.S.S., Karpat, O.: Agent-based conceptual model representation using bpmn. In: Proceedings of the 2011 Winter Simulation Conference (WSC). pp. 671–682 (2011)
19. Real, R., Vargas, J.M.: The probabilistic basis of jaccard's index of similarity. Systematic Biology **45**(3), 380–385 (09 1996)
20. Soika, R., Ortmayr, T.: OpenBPMN (2023), https://github.com/imixs/openbpmn?tab=License-1-ov-file
21. Taghikhah, F., Voinov, A., Filatova, T., Polhill, J.: Machine-assisted agent-based modeling: Opening the black box. Journal of Computational Science **64** (2022)
22. Taillandier, P., Gaudou, B., Grignard, A., Huynh, Q.N., Marilleau, N., Caillou, P., Philippon, D., Drogoul, A.: Building, composing and experimenting complex spatial models with the gama platform. GeoInformatica **23**(2), 299–322 (Apr 2019)