RCR Report for the Paper: "On the Modelling of Aggregated Behaviour for Simulation: An Event–Based Architecture"

Justin Noah Kreikemeyer
Institute for Visual and Analytic Computing
University of Rostock
Rostock, Germany
justin.kreikemeyer@uni-rostock.de

Abstract

This report evaluates the artifacts associated with the conference paper "On the Modelling of Aggregated Behaviour for Simulation: An Event–Based Architecture." The authors of the aforementioned paper provided well-documented code on a permanent repository hosted at Zenodo. Following the provided instructions and using the provided code, the results could be independently reproduced and conform with the main statements made in the paper. Thus, the article receives the ACM Artifact Review (v1.1) badges *Artifacts Available*, *Artifacts Evaluated—Reusable*, and *Results Validated—Results Reproduced*.

CCS Concepts

• **General and reference** → **Evaluation**; • **Computing methodologies** → **Modeling and simulation**.

Keywords

ACM SGISIM PADS, artifact evaluation, reproducibility

ACM Reference Format:
Justin Noah Kreikemeyer. 2026. RCR Report for the Paper: "On the Modelling of Aggregated Behaviour for Simulation: An Event–Based Architecture". In *40th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '26), June 24–26, 2026, Vienna, Austria.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3806789.3816102

1 Introduction

The evaluated paper [1] compares notations and formalisms used to model agent behavior, in particular planning, in complex systems using discrete-event simulation. A novel discrete event simulation architecture is proposed, which is used in the evaluated paper to obtain quantitative data on runtimes and rejection statistics for comparing different notations and modeling strategies. The seven notations of agent-behavior compared are: Discrete EVent System Specification (DEVS), Behavior Trees (BT), Hierarchical Task Networks (HTN), Decision Model and Notation (DMN), Business Process Model and Notation (BPMN), Data Petri Nets (DPN), and Monte Carlo Tree Search (MCTS).

SIGSIM-PADS '26, Vienna, Austria

ACM ISBN 979-8-4007-2648-4/26/06
https://doi.org/10.1145/3806789.3816102

For the evaluation, the authors instantiate and test their simulator with the strategy game "Risk"[1], a board game about the conquest of territories. In their simulation, automated agents take the role of the players, and plan where to place, move, and attack with their troops. They implement three different strategies in each notation: a random, defensive, and an aggressive strategy. All $3^7 = 2,187$ permutations of notations and strategies are then tested according to several criteria.

Each permutation is run for 100 turns of Risk and the average score, average runtime of the planning phase, number of elements required in each notation, and average code complexity of the files required to implement the strategy are reported (cf. Table 1). Further, the number of rejection events is measured. These occur when an agent's intention cannot be granted, e.g., as it is inconsistent with the current world state. The authors distinguish three types of rejections for their Risk implementation: attacks on owned territories (Type 1), attacks with insufficient troops (Type 2), and attacks requested from unowned territories (Type 3). Rejections only occur for the attack intention, not for placement or movement intentions (cf. Table 2).

2 Artifact Evaluation

This section details the procedure for evaluating the artifacts [2] associated with the paper [1] and reproducing the results presented therein.

2.1 Quality of the Artifact

The artifacts provided are comprehensive, well-documented, and accessible in the long-term. A `README.md` file contains authors and a correspondence contact, the hardware configuration used by the authors, as well as easy-to-follow set up, quick start, and reproduction instructions. It further documents additional features of the implemented simulator and its architecture. The source code is commented sufficiently for others to comprehend. All artifacts have been archived on Zenodo [2] and assigned a digital object identifier (DOI), meeting the standard requirements for long term accessibility.

2.2 Environment, Software, and Installation

The artifact evaluation was carried out on a machine equipped with an AMD EPYC 9754 128-Core Processor and 768 GiB RAM running at 4800 MHz. As operating system, `Ubuntu 24.04.4` with Linux kernel `6.8.0-107-generic` was used. Python version `3.12.3` was employed, which is lower than the version tested by the authors

[1] https://en.wikipedia.org/wiki/Risk_(game), last accessed 2026-04-27

Table 1: Comparison of experimentation results reporting a demonstrative run over 100 turns with 2187 combinations between original publication and reproduction (format: original/reproduction). The original experiment took a wall-clock time of 18.82 hours, averaging 11.74 minutes per simulation. The reproduction experiment took a wall-clock time of 2.30 hours, averaging 7.65 minutes per simulation. The difference in wall-clock times is due to the difference in hardware. Values in the columns # and Complexity were calculated separately by the authors and not part of the reproduction process.

	Strategy	#	Complexity	Runtime (std)	Score (std)
Base					
	random	n/a	4.4	**0.06 (±0.02)**/0.04 (±0.01)	**51.67 (±22.92)**/60.62 (±26.16)
	defensive	n/a	6.6	**0.18 (±0.04)**/0.03 (±0.00)	**132.21 (±31.09)**/101.52 (±25.53)
	aggressive	n/a	7.0	**0.36 (±0.13)**/0.28 (±0.11)	**34.84 (±15.21)**/40.54 (±14.21)
DEVS					
	random	16	2.06	**0.56 (±0.15)**/0.40 (±0.10)	**58.78 (±26.58)**/71.89 (±28.96)
	defensive	32	2.10	**1.74 (±0.20)**/1.07 (±0.14)	**140.70 (±32.13)**/166.72 (±32.03)
	aggressive	32	2.21	**1.36 (±0.22)**/0.88 (±0.15)	**48.45 (±14.87)**/52.86 (±15.16)
BT					
	random	30	1.71	**16.48 (±3.78)**/8.44 (±1.57)	**55.29 (±25.67)**/64.36 (±26.30)
	defensive	35	1.81	**56.76 (±11.29)**/25.39 (±4.04)	**135.66 (±33.39)**/143.86 (±31.03)
	aggressive	49	1.91	**35.50 (±15.56)**/18.03 (±6.56)	**33.01 (±13.81)**/38.49 (±14.33)
HTN					
	random	25	2.13	**0.54 (±0.14)**/0.45 (±0.11)	**59.28 (±27.70)**/77.86 (±28.93)
	defensive	30	2.50	**36.17 (±15.70)**/33.75 (±13.00)	**138.48 (±33.16)**/161.10 (±35.98)
	aggressive	34	2.20	**13.44 (±4.67)**/14.12 (±3.86)	**29.77 (±13.02)**/39.29 (±13.17)
MCTS					
	random	6	2.29	**29.91 (±5.35)**/23.97 (±5.19)	**73.97 (±32.00)**/67.23 (±27.68)
	defensive	8	2.16	**25.16 (±5.85)**/20.91 (±2.17)	**158.83 (±34.83)**/142.39 (±30.58)
	aggressive	8	2.25	**14.03 (±1.95)**/14.04 (±1.60)	**44.29 (±11.22)**/50.12 (±10.58)
BPMN					
	random	24	3.17	**0.46 (±0.10)**/0.26 (±0.05)	**83.68 (±36.56)**/71.36 (±29.99)
	defensive	32	2.35	**12.54 (±2.01)**/7.30 (±1.11)	**174.67 (±35.70)**/166.95 (±32.25)
	aggressive	28	2.45	**5.38 (±0.82)**/3.52 (±0.54)	**67.01 (±20.60)**/69.20 (±18.37)
DPN					
	random	10	3.52	**0.39 (±0.12)**/0.27 (±0.08)	**63.51 (±30.13)**/64.52 (±27.35)
	defensive	14	3.05	**5.28 (±3.31)**/2.69 (±1.49)	**159.14 (±36.97)**/150.62 (±32.26)
	aggressive	16	2.96	**1.29 (±0.45)**/0.78 (±0.26)	**27.63 (±11.70)**/28.31 (±11.39)

(`3.13.5`). To account for this, the `Pipfile` was altered by the reviewer to point to the slightly older Python version.

Further, instead of the `py` command suggested by the README, which does not come with the standard Python installation on Ubuntu, the command `python3` was used. However, in this case, they should behave exactly the same, as using `py` without arguments points to the default python version. Otherwise, the README was followed exactly. In particular, the following commands were run to set up the environment and run the experiments:

```
1 pipenv sync
2 pipenv shell
3 (unbuffer time python run_eval.py --turns 100 2>&1)
    | tee -a log-run_eval_turns_100.txt
4 (unbuffer time python write_results.py\
    --path eval/eval_runs_0001 --collect 2>&1)\
    | tee -a log-write_results.txt
5 (unbuffer time python write_rejects.py\
    --path eval/eval_runs_0001 --collect 2>&1)\
    | tee -a log-write_rejects.txt
```

The tools `unbuffer`, `time`, and `tee` were used to preserve any output for later reference or debugging and to have an additional execution time reference. The actual commands run are the same as provided in the file `README.md`.

2.3 Reproduction Results

Running the above commands took 2.3 wall-clock hours, which is much shorter than the 18.82 wall-clock hours reported by the authors. This is because the simulator could benefit from the many additional cores of the hardware used for reproduction.

The main quantitative results of the paper are summarized in two tables, one for the runtime measurements and one comparing rejection counts. Table 1 and Table 2 compare the reproduced quantities with the ones reported in the paper. As noted by the authors, differences are expected among different hardware configurations and due to the stochastic simulation. However, the reproduction shows the same trends as reported by the authors, i.e., relations between the compared approaches match or slightly differ in a manner to be expected by the obtained standard deviations.

Table 2: Breakdown of rejected events across simulation runs. Comparison between results from original article and results from reproduction (format: original/reproduced). It is evident that the orders of magnitudes match between the original and reproduced values and often the values are also within an acceptable relative margin of each other.

	Strategy	place	attack			move
			Type 1	Type 2	Type 3	
Base						
	random	0/0	**4,140**/5,235	0/0	0/0	0/0
	defensive	0/0	**49,519**/10,946	0/0	0/0	0/0
	aggressive	0/0	**97,323**/92,368	**10,290**/11,991	**44,182**/50,321	0/0
DEVS						
	random	0/0	**6,441**/6,129	0/0	0/0	0/0
	defensive	0/0	**26,058**/32,820	0/0	0/0	0/0
	aggressive	0/0	**127,524**/143,895	**19,927**/25,427	**51,811**/63,006	0/0
BT						
	random	0/0	**5,111**/5,321	0/0	0/0	0/0
	defensive	0/0	**39,843**/38,890	0/0	0/0	0/0
	aggressive	0/0	0/0	**21,746**/23,646	**54,034**/58,235	0/0
HTN						
	random	0/0	**4,488**/6,601	0/0	0/0	0/0
	defensive	0/0	**41,808**/53,934	0/0	0/0	0/0
	aggressive	0/0	0/0	**26,318**/31,757	**138,602**/165,479	0/0
BPMN						
	random	0/0	**10,551**/6,999	0/0	0/0	0/0
	defensive	0/0	**66,092**/58,438	0/0	0/0	0/0
	aggressive	0/0	0/0	**1,483**/1,356	**271**/208	0/0
DPN						
	random	0/0	**5,997**/5,659	0/0	0/0	0/0
	defensive	0/0	**59,685**/49,778	0/0	0/0	0/0
	aggressive	0/0	**5,809**/5,801	**7571**/7535	**38710**/36310	0/0
MCTS						
	random	0/0	**8,398**/6,346	0/0	0/0	0/0
	defensive	0/0	0/0	0/0	0/0	0/0
	aggressive	0/0	0/0	**5,851**/5,713	**18,956**/18,933	0/0

In particular, the following statements are made in the paper [1] and are confirmed by the reproduced results in Table 1 and Table 2:

- *"[...] we see that notations with execution semantics outperformed the others, but all notations were slower than the baseline. The implementations for BT, HTN, and MCTS all experienced longer runtimes between simulations."* (page 10). This can be seen in the Runtime column of Table 1.
- *"The DEVS implementation had the least runtime between simulations, with BPMN and DPN as the second-best choices."* (page 10). DEVS, BPMN, and DPN are the fastest strategies and fairly close, but DEVS is much faster in the defensive strategy as well (cf. Table 1).
- *"The defensive strategy had longer runtimes [...]"* (page 10). In the reproduced results in Table 1, the defensive strategy almost consistently had a longer runtime.
- *"without an interaction life-cycle as in aggregated behaviour, conditional and situational behaviours like the agg. strategy require continuous replanning to avoid rejections."* (page 10). This general statement about the results of the rejection experiment is supported by the very similar reproduced results in Table 2. It can be seen that especially the open-ended aggressive strategy leads to rejections with the implemented primitive and compound behavior.

3 Conclusion

All artifacts required to reproduce the main results of the paper [1] have been archived by the authors in an accessible and long-term manner. Their documentation is detailed, enables reproduction, and even facilitates their re-use through details on the simulation architecture, description of additional runners, and well-commented source code. The results obtained based in the instructions provided by the authors are in line with the paper. Thus, the evaluation of the artifacts above supports assigning the ACM Artifact Review v1.1 badges, *Artifacts Available*, *Artifacts Evaluated—Reusable*, and *Results Validated—Results Reproduced*.

Acknowledgments

The reproduction reported on was supported by the computational resources of the Modeling and Simulation Chair at the University of Rostock, Institute for Visual and Analytic Computing.

References

[1] Adam Banham, Claudia Szabo, and Ryan Beruldsen. 2026. On the Modelling of Aggregated Behaviour for Simulation: An Event– Based Architecture. In *Proceedings of ACM SIGSIM International Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '26)*. ACM, New York, NY, USA. doi:10.1145/3806789.3810253

[2] Adam Banham, Claudia Szabo, and Ryan Beruldsen. 2026. *On the Modelling of Aggregated Behaviour for Simulation: An Event–Based Architecture.* doi:10.5281/zenodo.19361252