# Yanapay: A Simulation Toolkit for Evaluating Human-AI Collaboration in Emergency Evacuations

Alexandros Kangkelidis*, Carlos Gavidia-Calderon*†, Amel Bennaceur*, Bashar Nuseibeh*
*The Open University, UK. †The Alan Turing Institute, UK

***Abstract*—Evaluating Human-AI collaboration in safety critical situations, such as emergency evacuations, requires an evaluation framework that accounts for the characteristics of both human and AI behaviour. This paper presents the Yanapay toolkit, which applies social psychology principles to simulate human decision making and Human-AI collaboration in emergency evacuations. Researchers can use the Yanapay toolkit to design and evaluate strategies for AI agents collaborating with humans during emergency evacuations. The toolkit offers configurable scenarios, enabling the simulation of different evacuation conditions. Through agent-based simulations, researchers can understand how AI agents can leverage sociocultural factors and determine the most effective strategies for collaboration in emergency evacuations. Yanapay automates and streamlines the simulation workflow in order to evaluate, analyse, and compare strategies under different simulation scenarios. We invite researchers to extend and experiment with Yanapay to advance the state of software engineering research in Human-AI collaboration. A video demonstrating the tool is available at https://youtu.be/iksSM_eW6hI.**

## I. Introduction

Artificial Intelligence (AI) systems have increasingly been used in emergencies to assist in evacuation efforts [1]. By collaborating with humans on the scene, these systems can significantly enhance the speed and safety of evacuations [2]. The development and validation of AI systems for emergency evacuations highlight unique software engineering challenges related to modelling and reasoning about collaboration with humans. These systems must be rigorously tested across diverse scenarios while accounting for complex human behavioural factors. While existing simulation frameworks focus either on technical system validation or human behaviour modelling, there is a gap in software engineering tools that support the systematic evaluation of Human-AI collaboration.

Agent-based simulations can help address the complexities of modelling human behaviour. They are made up of autonomous entities interacting with each other and their environment. They have been widely applied to model individual decision making and social behaviour [3]. However, the process of setting up these simulations, running experiments across diverse conditions, and analysing the results can be a time-consuming process. Researchers often need to test numerous parameters systematically, which may require good knowledge of the simulation framework and can make the workflow cumbersome. There is an increasing need for artifacts in Software Engineering, particularly in safety critical domains like emergency evacuations, to validate Human-AI collaboration and offer platforms for reproducibility and benchmarking. To address this gap, this paper introduces Yanapay[1], a simulation toolkit designed to evaluate Human-AI collaboration in emergency evacuations.

Yanapay builds upon the IMPACT+ model [4], an extension of the original, empirically validated, IMPACT model developed by Van der Wal *et al.* [5]. The IMPACT model simulates evacuations from a transport hub and incorporates social, cultural, cognitive, and emotional factors derived from real-world evacuation drills. The key addition of IMPACT+ is the introduction of a Search and Rescue (SAR) robot. When encountering a fallen victim, the SAR robot must decide whether to ask for help from a nearby victim, also called *zero-responder* or from a member of staff, also called *first-responder*. The strategy that researchers aim to validate through the toolkit determines this decision.

From a software engineering perspective, Yanapay provides a modular architecture that separates the simulation engine, strategy implementation, and results analysis. The toolkit offers automated workflow management and configurable parameters specified through a configuration file. Through seed setting, which controls the initial conditions of random processes (e.g., an agent's direction of travel), Yanapay supports reproducible results across multiple simulation runs and by running simulations in parallel, it reduces the time needed to collect results from multiple experiments. Yanapay allows researchers to develop and test various strategies that define the actions of the SAR robot, providing a benchmarking platform to compare different approaches to autonomous decision making when collaborating with humans, in evacuation scenarios.

The structure of the paper is as follows. Section II introduces the Yanapay toolkit and its main entities, including its incorporated agent-based simulation model (in subsection II-C). Section III briefly reviews related work. Section IV discusses the limitations of the toolkit, and we conclude in Section V.

## II. The Yanapay Toolkit

In this section we introduce the Yanapay toolkit in more detail. We present its architecture and its main entities, the underlying agent-based simulation model, an explanation on how to use the toolkit and an overview of the type of the results it produces.

### A. Architecture Overview

The Yanapay toolkit takes as input a) a set of parameters for any number of simulation scenarios and b) a set of

[1]"Yanapay" means "to help" in Quechua, an indigenous language of Peru

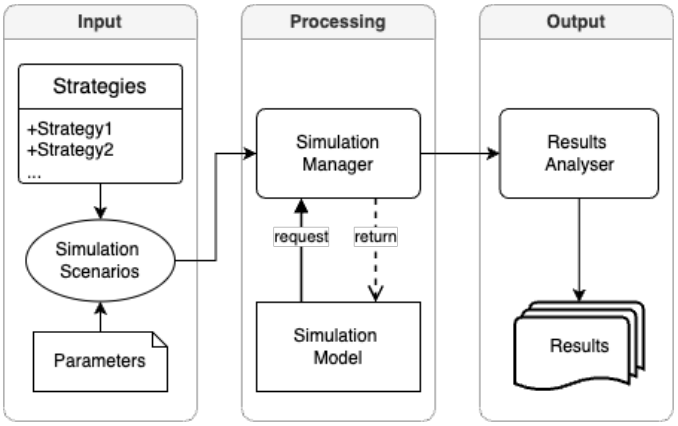


Fig. 1. Main entities of the Yanapay toolkit.

strategies that the SAR robot employs during the simulations. The toolkit then automatically generates and executes multiple simulations based on the configured scenarios. Finally, it processes and analyses the outcomes of the experiments, providing researchers with detailed results that compare the performance of the provided strategies.

The Yanapay toolkit (Fig. 1), consists of the following main entities:

- Simulation parameters and scenarios: User-defined parameters for simulation scenarios (e.g. environmental setup, number of agents).
- Strategies: a collection of user-defined strategies that will determine the actions of the SAR robot (i.e. ask help from a zero or a first responder).
- Simulation Manager: central entity responsible for setting up, managing, and executing multiple simulations. Handles communication between the simulation model and the scenario strategy, when the SAR robot encounters a situation requiring a decision to be made.
- Simulation Model: executes simulations for each of the specified evacuation scenarios.
- Results Analyser: processes the simulations' output data to provide visualisations (plots, videos) and statistical comparisons of strategies. Refer to II-E for more details.

Algorithm 1 describes the process of running an experiment. A list of scenario objects is created based on the parameters defined in the configuration file (lines 1-3). Each scenario runs *n* times. Then the simulations run in parallel (line 4) until termination (line 12). After the experiment is concluded, Yanapay saves the results and analyses them

### B. Implementation details

The Yanapay toolkit is designed for cross-platform compatibility by running within a Docker container. Docker was chosen for its ability to provide a consistent environment across different operating systems, ensuring that the toolkit can be deployed and run on a variety of platforms with minimal setup effort. The simulation model is implemented

**Algorithm 1** Run Experiment

1: $configurations \leftarrow$loadConfig("config.json")
2: $scenarios \leftarrow$buildScenarios($configurations$)
3: $simulations \leftarrow$buildSimulations($scenarios$)
4: **repeat**
5: **for all** $simulation \in simulations$ **do**
6: $result \leftarrow$runSimulation($simulation$)
7: **if** $result.success$ **then**
8: $simulations \leftarrow simulations \setminus simulation$
9: saveResults($result$)
10: **end if**
11: **end for**
12: **until** $Simulations \equiv \emptyset$

in NetLogo [6], a widely used agent-based simulation environment that runs on a Java Virtual Machine (JVM). Communication between the Python environment and the JVM is handled by PyNetLogo [7], an interface that enables Python to interact with NetLogo scripts. Additionally, a Flask [8] server running locally handles communication between the strategies defined in Python and the simulation environment through shell commands.

While the simulation model runs in a JVM using NetLogo, Yanapay allows researchers to define the SAR robot's strategy in Python. Python was selected due to its wide adoption within the scientific community. Simulation scenarios parameters—such as the seed and number of samples—are configured using a JSON [9] file. JSON was also chosen for its status as a widely adopted standard.

### C. The agent-based simulation model

The IMPACT model, on which Yanapay is based, is an empirically validated agent-based model designed to simulate human behaviour during emergency evacuations. In the simulation, agents represent individuals in a transport hub and are placed either in groups or as individuals, each with specific characteristics such as age, gender, cultural group, familiarity, and compliance. These characteristics influence how agents interact with their environment and other agents, as well as their decisions during the evacuation process.

At the start of the simulation, a fire is randomly placed within the environment. Each agent has a belief about the danger based on factors like the presence of fear, whether the fire alarm has sounded, and their individual traits. This belief influences their desire to evacuate. If they do not decide to evacuate, agents move randomly through the space. Within each group, a leader guides the rest of the members; once the leader decides to evacuate, the entire group follows. Agents representing first-responders move randomly and are tasked with convincing others to evacuate.

When there is a high concentration of agents in an area there is an increased likelihood that an agent will fall. The probability of a fall is controlled by the `fallChance` parameter. Once an agent falls, it remains stationary for a predefined duration, defined by the `fallLength` parameter. The duration can be shortened if nearby group leaders, first-responders or zero-responders asked by the SAR robot decide

to assist the fallen agent, thereby expediting their recovery and shortening the total evacuation time.

**Algorithm 2** SAR Robot

1: $didNotHelp \leftarrow \emptyset$
2: **repeat**
3: $randomWalk(robot)$
4: $V_{visible} \leftarrow V_{all} \setminus \{v \mid dist(robot, v) \geq threshold\}$
5: $V_{fallen} \leftarrow$nearest$(\{v \mid v \in V_{visible} \wedge v.fallen\})$
6: $V_{visible} \leftarrow V_{visible} \setminus (V_{fallen} \cup didNotHelp)$
7: **if** $V_{fallen} \neq \emptyset$ **then**
8: **while true do**
9: $V_{helper} \leftarrow$nearest$(V_{visible})$
10: $decision \leftarrow$strategy$(V_{helper}, V_{fallen})$
11: **if** $decision \equiv ask\_help$ **then**
12: $reply \leftarrow$askHelp$(V_{helper}, V_{fallen})$
13: **if** $reply$ **then**
14: speedUpRecovery$(V_{fallen}, boost_p)$
15: **break**
16: **else**
17: $didNotHelp \leftarrow didNotHelp \cup V_{helper}$
18: **end if**
19: **else if** $decision \equiv call\_staff$ **then**
20: callStaff$(V_{fallen})$
21: speedUpRecovery$(V_{fallen}, boost_s)$
22: **break**
23: **end if**
24: **end while**
25: **end if**
26: **until** $simulationFinished$

As shown in Algorithm 2, the SAR robot moves randomly within the environment (line 3) in search for a fallen victim. It keeps track of those agents the robot has attempted to get assistance from without success in the `didNotHelp` set. It identifies agents within its visual range (`V_visible`) and determines the nearest fallen agents, if they exist (`V_fallen`) (lines 4-5). The algorithm then filters the visible individuals to exclude those previously denied offering help (line 6). If there are fallen individuals, the robot enters a decision-making loop to engage the nearest potential helper (`V_helper`). Depending on the strategy implemented by the user, the robot may either ask for assistance from `V_helper` (lines 11-18) or call a first-responder (lines 19-22). If `V_helper` agrees to assist, the simulation expedites the recovery of the fallen agent by a predetermined amount; if not, `V_helper` is added to the `didNotHelp` set and the process is repeated (lines 16-17).

The SAR robot's strategy, implemented by the researchers using the toolkit, determines its decision-making. Based on psychological research indicating that people with similar characteristics are more likely to help each other, during emergencies [10], each strategy may use the age, cultural group and gender of the potential helper and of the victim.

Researchers can define strategies for the SAR robot agents and adjust simulation parameters to evaluate them under different conditions, without needing to modify the underlying model. Yanapay automates the simulation setup, execution, and result analysis, enabling researchers to configure and run multiple evacuation scenarios, to evaluate strategies for Human-AI collaboration.

### D. How to use the Yanapay toolkit

The Yanapay toolkit can be downloaded as a pre-built Docker image from Docker Hub *https://hub.docker.com/r/alexandroskangkelidis/robot-assisted-evacuation* . Alternately, the source code can be downloaded from the GitHub repository at *https://github.com/kangkelidis/robot-assisted-evacuation* and the image can be built locally. A configuration file is used to define simulation scenarios with distinct parameters. The toolkit is flexible in how parameter values are provided and it includes eight room types for the transportation hub, that can be used in the evacuation simulations. Each room has a different size and a distinct layout of exit doors and obstacle's locations. The Yanapay toolkit provides a foundation upon which users can built and validate their strategies for the SAR robot. In this context, AI refers to any autonomous decision making by the robot rather than a machine learning-based system.

An example for creating a new strategy by extending the built-in `AdaptationStrategy` base class is shown below in Listing 1. In this example, the `get_robot_action` method is overridden to determine the robot's action based on the distance between the helper and the victim. If the potential helper is closer to the victim than the first responder, the robot requests assistance from the helper; otherwise, it calls for the first responder.

```
class NewStrategy(AdaptationStrategy):
    def get_robot_action(self, simulation_id,
            candidate_helper, victim,
            helper_victim_distance,
            first_responder_victim_distance):

        # Decide based on distance
        if helper_victim_distance < \
            first_responder_victim_distance:
            return self.ASK_FOR_HELP_ROBOT_ACTION
        else:
            return self.CALL_STAFF_ROBOT_ACTION
```

Listing 1. Creating a new strategy

### E. Simulation Results

When the program terminates, Yanapay creates a folder containing the results of the experiments. The folder includes a file with the simulation parameters, tabulated results of the simulations with evacuation times, a video for the simulations if that option is enabled, various plots and results of a statistical analysis of the data. The plots produced include a violin plot, see Fig. 2, that shows the distribution of evacuation times for each scenario and for each strategy, a bar chart showing the robot actions for each scenario, and various line-chart plots of each parameter combination against the evacuation time. If the experiment includes multiple scenarios, the *Kruskal-Wallis H* test [11] is performed for a specified scenario, to determine if the median evacuation times are significantly different from the other scenarios.

## III. Related Work

Simulations are extensively used as a research tool in software engineering [12]. Due to the nature of emergencies,

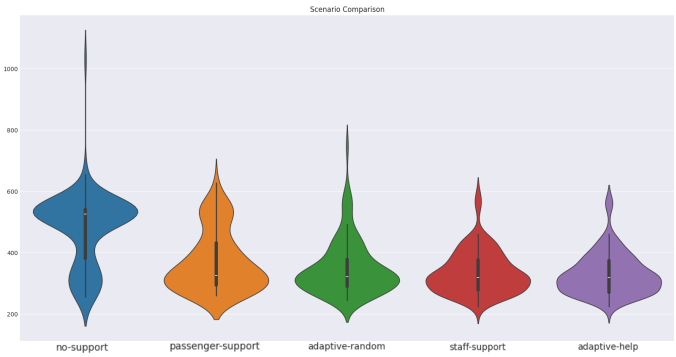


Fig. 2. Violin plot of predefined scenarios included in the toolkit, showing evacuation times for each scenario. Rank by median values, scenarios from left to right: no SAR robot, always asking for help, random decision, always calling for help, predicts the chance of a passenger accepting to help.

using simulations allows us to explore multiple configurations and scenarios safely, ethically, and in a cost-effective way [13]. On the one hand, simulation tools for emergencies using autonomous systems, e.g., Dronology [14] focus on simulating the software system itself or a single human participant rather than group behaviour. On the other hand, social psychology researchers use agent-based models to capture people's behaviour [5] but without representing autonomous systems. Mass casualty incident simulators primarily address medical or operational aspects, such as triage [15], rather than focusing on Human-AI collaboration. Yanapay combines these streams by introducing a platform for simulating collaboration between humans and autonomous systems, in emergency evacuations. Unlike other tools, Yanapay incorporates socio-cultural aspects of human behaviour alongside decision-making strategies for SAR robots, providing a novel framework for studying Human-AI collaboration.

## IV. Limitations

The current implementation of the simulation model is constrained by the limitations inherent in the original IMPACT model, such as its specific assumptions and simplifications about human behaviour during evacuations. Moreover, we recognise the opportunity for further improvement to the actions and information available to the SAR robot during decision-making. For instance, rather than considering only the nearest agent as a candidate to offer help, the robot could generate a prioritised list of agents within range, ranked by their likelihood of accepting to help. The strategy could then define a threshold, after which the robot would call a first responder. The robot would request assistance from the highest-ranked agent, moving down the list, if necessary, until help is offered or the threshold is met.

Another limitation involves the modelling of congestion during evacuation through exit doors. If all individuals rush toward exits simultaneously, the total evacuation time may increase. The delays caused by fallen agents may help reduce congestion at the exits, which, in some cases, can decrease the overall evacuation times. In addition, we plan to conduct further usability studies to gather feedback for improvement.

## V. Conclusion

In this paper, we introduced the Yanapay toolkit designed to evaluate Human-AI collaboration in emergency evacuations. By incorporating social psychology research, Yanapay addresses the challenge of simulating human behaviour through an agent-based model where a SAR robot collaborates with human agents to minimise the total evacuation time. The toolkit automates the workflow for setting up, executing, and analysing simulations, enabling researchers to develop different strategies for autonomous decision-making during interactions with humans. Future work could extend these capabilities through enhanced AI behaviour and strategy evaluation metrics. We encourage researchers to build upon this foundation to advance software engineering approaches for developing and validating collaborative AI systems in safety-critical domains.

## Acknowledgment

This work was supported by EPSRC grants EP/V026747/1 and EP/R013144/1.

## References

[1] A. Birk and S. Carpin, "Rescue robotics—a crucial milestone on the road to autonomous systems," *Advanced Robotics*, Jan. 2006.
[2] C. Van Tilburg, "First Report of Using Portable Unmanned Aircraft Systems (Drones) for Search and Rescue," *Wilderness & Environmental Medicine*, June 2017.
[3] C. Macal and M. North, "Introductory tutorial: Agent-based modeling and simulation," in *Proc. of the Winter Simulation Conf.*, Dec. 2014.
[4] C. Gavidia-Calderon, A. Kordoni, A. Bennaceur, M. Levine, and B. Nuseibeh, "The IDEA of Us: An Identity-Aware Architecture for Autonomous Systems," *ACM Trans. on Soft. Eng. and Meth.*, Mar. 2024.
[5] C. N. Van Der Wal, D. Formolo, M. A. Robinson, M. Minkov, and T. Bosse, "Simulating Crowd Evacuation with Socio-Cultural, Cognitive, and Emotional Elements," 2017.
[6] Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/, Northwestern University, Evanston, IL.
[7] M. Jaxa-Rozen and J. H. Kwakkel, "PyNetLogo: Linking NetLogo with Python," *Journal of Artificial Societies and Social Simulation*, 2018.
[8] M. Grinberg, *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.
[9] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, "Foundations of json schema," in *Proc. of the 25th International Conference on WWW*, International World Wide Web Conferences Steering Committee, 2016.
[10] J. Drury, C. Cocking, and S. Reicher, "Everyone for themselves? A comparative study of crowd solidarity among emergency survivors," *British Journal of Social Psychology*, Sept. 2009.
[11] W. H. Kruskal and W. A. Wallis, "Use of Ranks in One-Criterion Variance Analysis," *Journal of the American Statistical Assoc.*, 1952.
[12] B. B. N. de França and N. B. Ali, "The role of simulation-based studies in software engineering research," in *Contemporary Empirical Methods in Soft. Eng.*, Springer, 2020.
[13] S. Purandare, U. Sinha, M. N. A. Islam, J. Cleland-Huang, and M. B. Cohen, "Self-adaptive mechanisms for misconfigurations in small uncrewed aerial systems," in *(SEAMS)*, 2023.
[14] J. Cleland-Huang, M. Vierhauser, and S. Bayley, "Dronology: an incubator for cyber-physical systems research," in *ICSE*, ACM, 2018.
[15] R. De Rouck, M. Debacker, I. Hubloue, S. Koghee, F. Van Utterbeeck, and E. Dhondt, "Simedis 2.0 : On the road toward a comprehensive mass casualty incident medical management simulator," 2018.