A Novel Approach for Developing Emergency Evacuation Plans

Hanan Daher, Samaneh HoseinDoost, Bahman Zamani, Afsaneh Fatemi
MDSE Research Group, Faculty of Computer Engineering
University of Isfahan
Isfahan, Iran
{h_daher, s.hoseindoost, zamani, a_fatemi}@eng.ui.ac.ir

Abstract—In case of a disaster, planning for pedestrian evacuation from buildings is a major issue since it threatens human lives. To cope with this problem, evacuation plans are developed to ensure efficient evacuation in minimum time. These plans can be very sophisticated according to the complexity of the evacuation environment. This advocates the use of architectures such as Multi-Agent Systems (MAS) to develop the evacuation plans before happening of a real accident. Since developing an evacuation plan using MAS requires considerable effort, finding more efficient approaches is still an open problem. This paper introduces a new approach, based on the model-driven principles, to support developing evacuation plans. The approach includes utilizing a graphical editor for designing evacuation models, automatic generation of the evacuation plan code, as well as running the generated code on a MAS platform. We evaluated our approach using a case study. The results show that our approach provides elevated speed, less effort, high abstraction level, and more flexibility and productivity in developing emergency evacuation plans.

Keywords—Model-Driven Development (MDD), Multi-Agent System (MAS), Emergency Evacuation, Domain-Specific Modeling Language (DSML)

I. INTRODUCTION

Emergency evacuation can be defined as keeping people away from the scene of the incident to save their lives [1]. To ensure safely evacuation of pedestrians from a building, there must be a clearly defined evacuation plan. Therefore, evacuation plans have been developed to be effective, ideal, and accurate in time to help evacuate the residents in minimum time [2]. An evacuation plan consists of several steps that the evacuees must follow to reach the target (accessing the exit door in minimum time) [3]. Developing evacuation plans in buildings is considered as an important aspect in several studies. Following are some examples of the most investigated evacuation plans: tracking the walls or obstacles, sheltering in a place, following evacuation signs, following others, following leaders, using the shortest path, and random fleeing [3], [4], [5].

Since the evacuation drills are costly and cannot be held regularly, simulation modeling becomes the main method applied to investigate evacuation systems. However, the emergency evacuation environment is considered as a complex environment according to its features such as dynamicity (it changes spontaneously), uncertainty (the next state of the environment, is not necessarily dependent on the current state, and the environment changes unpredictably), openness (components of the environment are not constant), and heterogeneous (members of the environment have different behavior and structure). Multi-Agent Systems (MAS) have the power to deal with the complexity of such systems, as they can represent the individuals and their interactions in an evacuation environment [6], [7]. To simulate such systems, agent-based tools can be used in this context, GAMA[1], Any Logic[2], Netlogo[3], and Repast Simphony[4], to name a few.

Unfortunately, simulating evacuation plans using such tools requires writing a lot of code and may cause inconsistency between the simulation model designed by computer simulation engineers and the conceptual model created by domain experts [8].

To address the aforementioned problem, we propose an approach that exploits Model-Driven Development (MDD) principles to automate and ease the development of evacuation plans. By automatic generation of code from the models, MDD helps implement software effectively and quickly [9], [10], [11]. Using the proposed approach, a developer can build a design model for an evacuation scenario at a higher level of abstraction, and then the model can be transformed into the simulation code, automatically. To enhance the generated code and fill in the gaps, the developer can complete the code and run it on one of the MAS platforms.

In order to evaluate the proposed approach, we used the Kermovo shopping mall[5] scenario as a case study. We have implemented this case study in two ways, using our approach, and using GAML[6]. GAML is an agent-oriented language dedicated to the definition of agent-based simulations [8]. Two metrics have been used for the evaluation process: the development time and the Lines of Codes (LoC). The evaluation results show that our approach reduces the development effort for agent-based simulation of evacuation plans.

[1] https://gama-platform.github.io/
[2] https://www.anylogic.com/
[3] https://ccl.northwestern.edu/netlogo/
[4] https://repast.github.io/
[5] https://en.wikipedia.org/wiki/2018_Kemerovo_fire
[6] https://gama-platform.github.io/wiki/GamlLanguage

978-1-7281-8566-8/20/$31.00 ©2020 IEEE

To summarize, the main contribution of this paper is presenting a model-driven approach for the automatic generation of emergency evacuation plans. To realize this approach, we have designed a domain-specific modeling language for the domain of evacuation plans, and we have built a graphical editor that enables the designer to use this language. Also, we have defined several Model to Code (M2C) transformations which are used for the automatic generation of simulation code from the designed model.

The remainder of this paper is organized as follows. Section II introduces the related work. Section III describes the proposed approach. Section IV is dedicated to the evaluation, and finally, the conclusion is presented in Section V.

II. Related Work

Due to the importance of modeling and simulation of evacuation plans using MAS, several studies are conducted in this area. Nguyen et al. [4] proposed a new agent-based model for optimizing evacuation plans by considering the smoke effect on the pedestrians' vision inside the buildings. Kasereka et al. [12] proposed an agent-based model that enables modeling and simulation of evacuation plans from a building on fire by considering four main parameters which are: evacuee, alarm, fire, and smoke. It should be noted that, since human behavior is an important factor when modeling evacuation plans, scientists have paid more attention to this aspect in their research. In this regard, Valette et al. [5] presented an evacuation model including the effects of social relationships on the effectiveness of the evacuation plans. Trivedi and Rao [13] presented an agent-based model that considers psychological factors that cause panic in such emergencies. Similarly, Rozo et al. [1] designed an evacuation plan using agent-based simulation considering the pedestrian behavior in complex buildings.

However, in all of the mentioned research, the evacuation model was developed using a programming language that causes the developing and simulating of evacuation plans to be a difficult and time-consuming task, in particular, for domain experts who are not familiar with programming languages. On the other hand, if a programmer develops the simulation model based on his/her knowledge, this may cause an inconsistency between the simulation model established by the developer of evacuation plans and the conceptual model determined by domain experts [8]. In other researches, the evacuation models are developed utilizing a model-driven approach. Wang et al. [14] used a model-driven approach to design a scalable modeling and simulation framework for building evacuation plans. The weakness of this research is that it did not consider the behavioral concepts of people which may significantly affect the evacuation process. Gianni [15] used informal specifications of building evacuation scenarios to introduce a model-driven method to develop building evacuation simulators. This work has adopted a model-driven approach to carry-out the evacuation analysis of preliminary building design. It did not consider changing the layout of the building which can affect the efficiency of the evacuation process, as well as it did not consider the emotional status for the evacuees.

As a result, we believe that our approach has superiority over the mentioned researches as long as it can handle the shortcomings of the aforementioned research. Table I shows a comparison between our approach and the aforementioned research.

TABLE I. Related Work Comparison

Research	Year	Used MAS	Used MDD	Evacuation		Simulation	
				behavioral	structural	Static	Dynamic
Nguyen [4]	2013	●	○	○	●	●	◐
Kasereka [12]	2018	●	○	◐	●	●	●
Valette [5]	2018	●	○	●	●	●	○
Rao [13]	2018	●	○	●	◐	●	○
Rozo [1]	2019	●	○	●	◐	●	○
Wang [14]	2013	○	●	○	●	●	○
Gianni [15]	2015	○	●	○	◐	●	○
Our research	2020	●	●	●	●	●	●

Legend: ● Fully supported ◐ partially supported ○ Not supported.

III. The Proposed Approach

In this research, a model-driven approach for simulating emergency evacuation plans is proposed. Figure 1 shows different stages of the approach that a user[7] should follow for simulating an emergency evacuation plan. The rounded rectangle boxes represent the stages and the rectangle boxes are artifacts. First of all, in the "Create Model" stage, the user should create an evacuation model using the proposed evacuation modeling language, and then, in the "Run UI Launcher" stage, by running the transformation code that is designed as a context menu on the model file, the user can generate evacuation simulation code automatically. After getting the code, in the "Adjust the Output Code" stage, the user can adjust the output code by adding the missing part of the code related to the evacuation plan. Finally, in the "Run Code on MAS Platform" stage, the resultant code can be run on one of the MAS platforms.

As mentioned earlier, the proposed approach is based on a domain-specific modeling language and several transformation code. In the following sections, we describe the evacuation meta-model, which is the abstract syntax of our modeling language, as well as the graphical editor tool support that is provided for creating an evacuation model. Then, we introduce the model to code transformations that are defined for the automatic generation of simulation code from the designed model.

[7] The user of our approach is the person who designs an evacuation plan

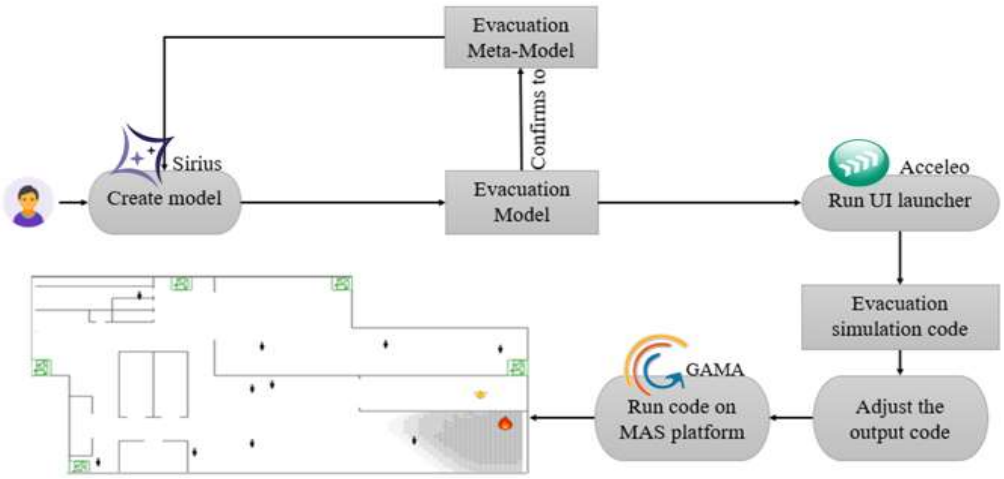

Fig. 1. The proposed approach

A. Evacuation metamodel

This section describes the proposed evacuation meta-model that was designed using the Eclipse Modeling Framework[8] (EMF). Figure 2 shows the evacuation meta-model that includes 14 main concepts. These concepts are derived from several research studies in the domain of evacuation plans [1], [4], [5], [12], [13], [14], [15]. The concepts of the meta-model are described in the following:

Evacuation System: Evacuation System is the root class that holds all other classes together.

Floor: Floor represents each floor in multi-floor buildings and it can have many entities as well as many disasters. Its attributes are:

- width: The number of cells that constitute the width of the environment grid.
- height: The number of cells that constitute the height of the environment grid.
- number of pedestrians: The total number of people inside the building.

Entity: Entity represents the agents (pedestrian) and objects in our system.

Pedestrian: Pedestrian is an agent trapped in the building, and has two types: staff and evacuee. The agent can see the fire/smoke, follow the signs, hear the alarm, escape to one of the exits, and avoid the obstacles. Each pedestrian exists on one floor and has the following attributes:

- speed: The actual speed of the agent.
- life points: The level of the agent's health at each time. An agent has a maximum 100 life points.
- is injured: If an agent loses his life points, he becomes an injured agent, and the flag returns true.
- is dead: If the life points of an agent are equal to zero, the flag returns true.
- is survived: If an agent is outside the building and has life points greater than zero, the flag returns true.
- perceived distance: The maximum distance that an agent can perceive the surrounding environment.
- Smoke quantity: The amount of smoke in the cell occupied with an agent.

Staff: Staff is a pedestrian agent that plays a rescuer crew role and helps the victims to evacuate and has two main features:

- charisma degree: Determines the competence of staff to be a leader in a group of rescuers
- emergency awareness: Refers to the average of the response time that a rescuer takes when it perceives a fire smoke.

Evacuee: Evacuee is a pedestrian agent. He/She can be a victim or can help his/her friends. Each Evacuee has the following attributes:

- has leader: Indicates whether the evacuee has a leader or not
- need help: Returns a boolean value, indicates whether the evacuee knows where the exit is, or not.
- mean know exit door: The mean value of the number of evacuees who know where the exit door is.

Some of the evacuee's attributes are related to the Belief-Desire-Intention (BDI) paradigm which is used to formalize the internal architecture of the evacuees in order to design more realistic agents. These attributes are:

- use emotion architecture: If the flag is set to true, the automatic emotion generation process is activated.
- use social architecture: Returns a boolean value that indicates whether social relations are automatically computed or not.
- use personality: Returns float value that indicates the degree of collaboration between the evacuees.

[8] https://www.eclipse.org/modeling/emf/

- openness: Indicates the openness degree of an evacuee (open-minded/narrow-minded).
- consciousness: Indicates the consciousness degree of the evacuee (act with preparations/impulsive).
- extroversion: Indicates the extroversion degree of the evacuee (extrovert/shy).
- agreeableness: Indicates the agreeableness degree of the evacuee (friendly/hostile).
- neurotic: Indicates the neurotic degree of the evacuee (calm/neurotic).
- solidarity: Represents the degree of similarity of desires, beliefs, and uncertainties between two agents.

Object: Object represents the immovable entities in our environment, each object is either sign, alarm, exit, wall, or shelter.

Alarm: Alarm represents a fire alarm.

- detection range: Time range in which the alarm can detect fire smoke.
- ringing duration: Amount of time in which alarm rings.

Sign: Sign is an object that guides evacuees to the emergency exits. A set of signs represents an evacuation path.

- direction: Shows the direction to the emergency exits.

Shelter: The place that the evacuees will refuge to it.

Wall: Wall is the obstacle inside a building. Walls obstruct people and fire from transporting to other cells.

Exit: Exit is the way to go out of a building.

Disaster: A disaster is an event that occurs in a building and causes damages.

- x_axis and y_axis: Represent the location of the disaster.
- type: Returns the kind of disaster (fire, gas pollution, or other types of disasters).

Fire: Fire is a disaster that may happen in a building. It could propagate within the building floor. Also, it affects the alarm and makes it start ringing.

- fire neighbor number: Refers to the number of cells which the fire will propagate to.

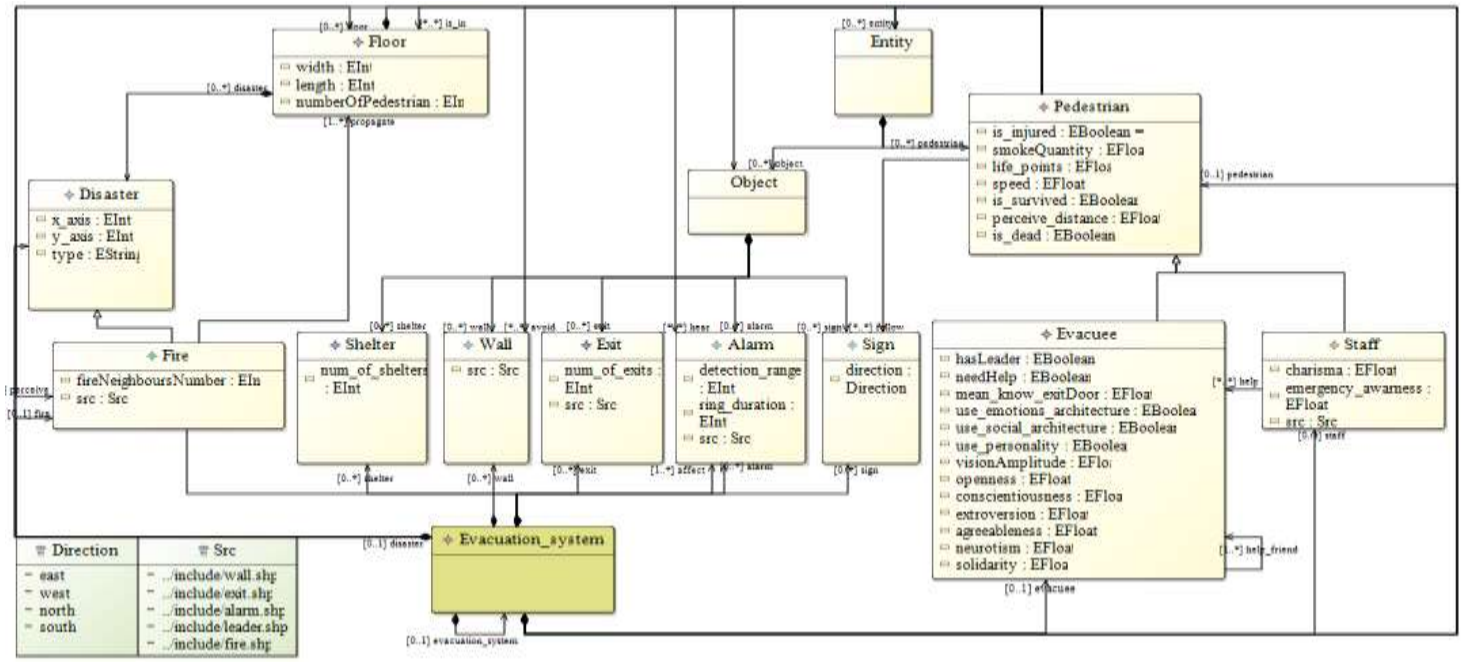

Fig. 2. The evacuation metamodel

Alarm, Wall, Exit, and Fire concepts have the src attribute for getting their shapefiles[9] from external programs (e.g. AutoCAD). The two enumerations that we have in our meta-model are:

Direction: For specifying the sign direction.

Src: For selecting the URL for the external shapefiles that will be required in our simulation.

[9] Shape files contain all the information linked to the format of spatial data.

B. The Graphical Editor

To facilitate modeling of an evacuation scenario, we provide a graphical modeling editor that graphically supports the evacuation meta-model concepts. For building the editor, we use Sirius[10] that is an Eclipse project for the easy creation of graphical modeling workbenches. An overview of the notations used in our graphical editor is presented in Figure 3.

[10] https://www.eclipse.org/sirius/

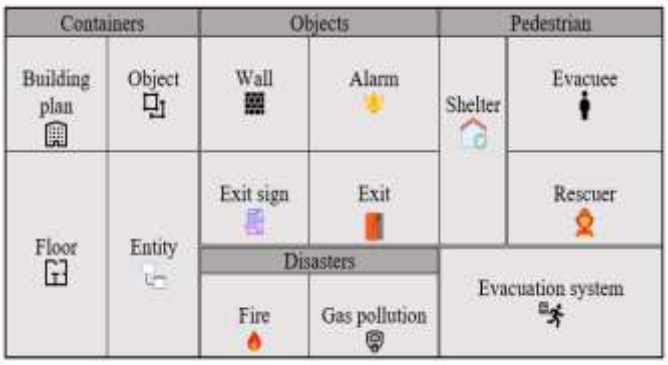

Fig. 3. The notations used in our graphical editor

Table II shows these relationships with their sources and targets.

TABLE II. RELATIONSHIPS IN THE GRAPHICAL EDITOR

Relation name	Relation source	Relation target
follow	pedestrian	sign
hear	pedestrian	alarm
avoid	pedestrian	wall
perceive	pedestrian	fire
is_in	pedestrian	floor
propagate	fire	floor
affect	fire	alarm
help	staff	evacuee
help_friend	evacuee	evacuee

Figure 4 shows the graphical environment for modeling an evacuation scenario. A modeler can drag and drop the items from the palette to the modeling area corresponding to the case study requirements. Then, he/she can modify the properties of each item through its properties window. When the model is prepared, it can be automatically transformed into the simulation code as described in the next section.

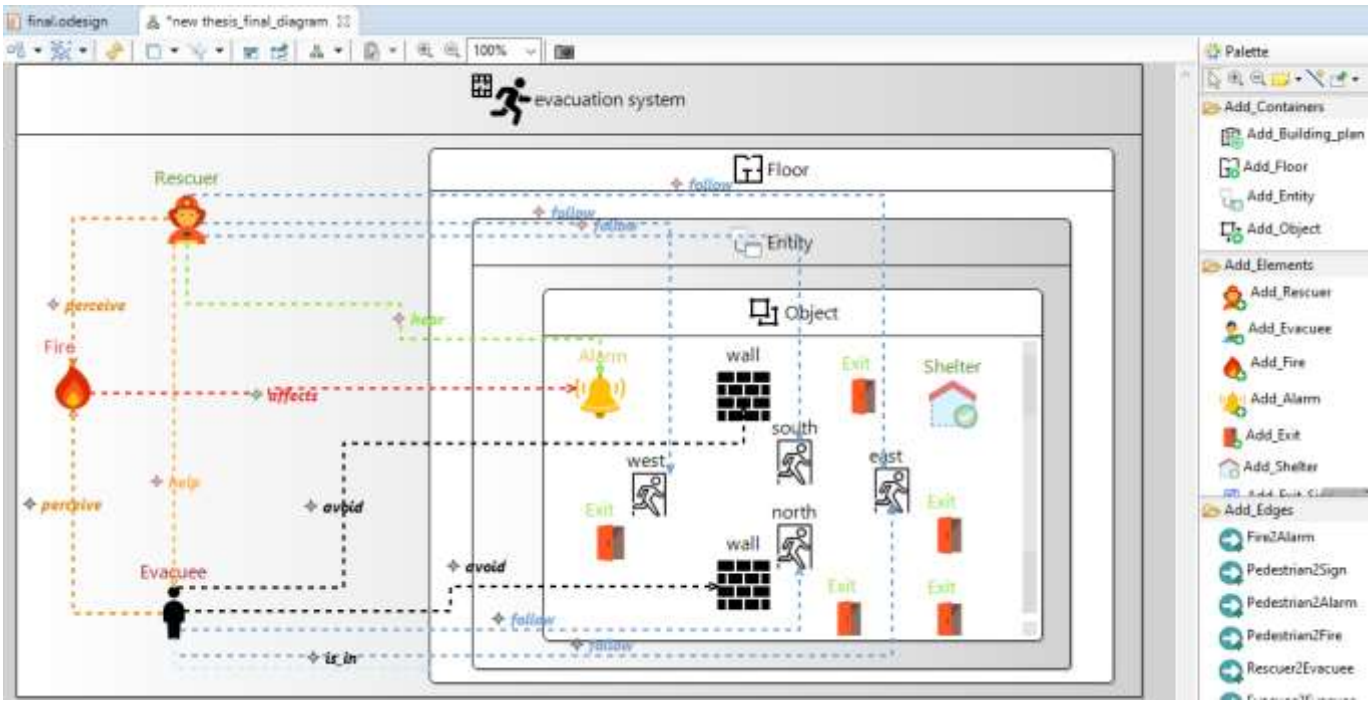

Fig. 4. The graphical editor environment

C. Model to Code Transformations

Model-to-code (M2C) transformations are also known as "code generators" and are used to generate code from a source model. There are two types of M2C transformations: visitor-based, and template-based. Acceleo[11] is a template-based transformation that can be used for generating simulation code from the EMF models. For automatic code generation for simulating evacuation plans, we wrote 522 lines of code in Acceleo. Figure 5 shows a piece of transformation code in Acceleo Query Language (AQL). In line 4, the name of the output file ('out.gaml') is specified, and then the name of the root class is used as the name of the GAML model in line 5. After that, the values of the global variables are invoked (line 7-15) for using in the reminder transformation code.

[11] https://www.eclipse.org/acceleo/

```
1 [module generate('http://www.final_thesis.org')]
2 [template public generateElement(anEvacuation_system : Evacuation_system)]
3 [comment @main']
4 [file ('out.gaml', false, 'UTF-8')]
5 model [anEvacuation_system.eClass().name.toLowerCase()/]
6 global{
7       file [anEvacuation_system.wall.eClass().name.toLowerCase()/]_shapefile <-
8                                   shape_file("[anEvacuation_system.wall.src/]");
9       file [anEvacuation_system.exit.eClass().name.toLowerCase()/]_shapefile <-
10                                  shape_file("[anEvacuation_system.exit.src/]");
11      geometry shape <-
12        envelope([anEvacuation_system.wall.eClass().name.toLowerCase()/]_shapefile);
13      int nb_cols <- [anEvacuation_system.floor.length/];
14      int nb_rows <- [anEvacuation_system.floor.length/];
15      int numberOfPedestrian <- [anEvacuation_system.floor.numberOfPedestrian/];
```

Fig. 5. Partial Acceleo code

As mentioned in the previous section, after designing the evacuation model, it can automatically be transformed into the simulation code. This is done by right-clicking on the model and selecting the "Acceleo Model to Text → generate evacuation simulation code" option as shown in Figure 6.

Fig. 6. Running the Acceleo code as a context menu on the model file

IV. Evaluation

To evaluate our approach, we used a case study implemented with our approach compared with implementing the same case study with GAML language which is dedicated for using in GAMA platform and widely used for developing agent-based evacuation plans [4][5].

Two important metrics, the development time and Lines of Code (LoC) were considered in the evaluation process [16]. In the following sections, the case study and the experiment results are discussed.

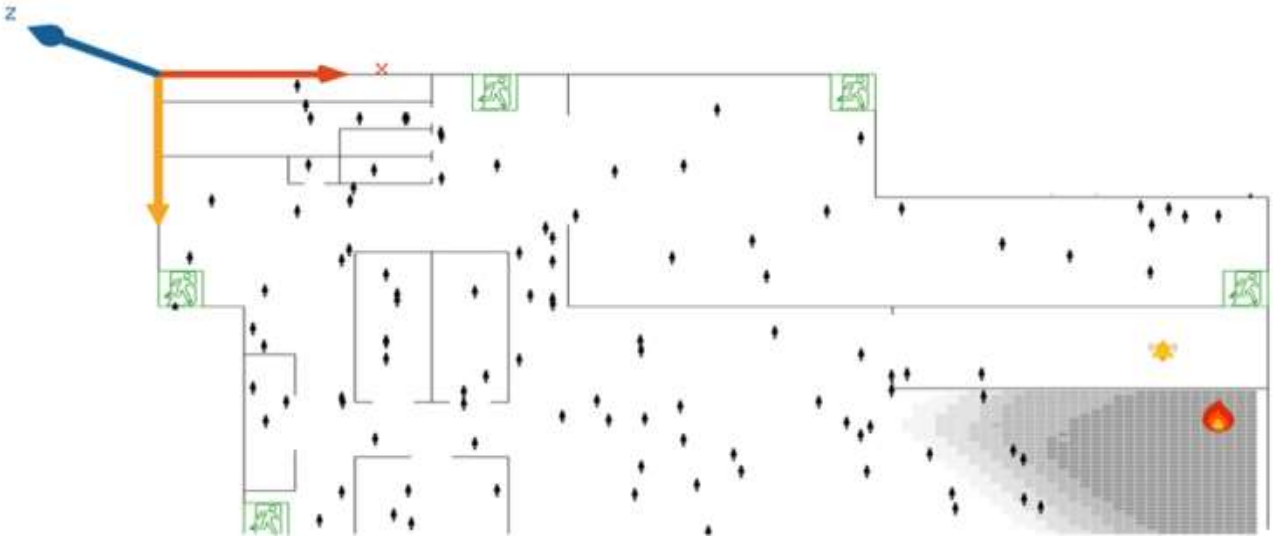

Fig. 7. Simulation of the pedestrian movement in the shopping mall

A. Case Study: Simulation with the Kemerovo Fire

In this section, we utilized a case study of the fire evacuation in the Winter Cherry shopping mall in Kemerovo, Russia. On 25 March 2018, a fire engulfed on the fourth floor and it killed at least 60 people according to the BBC news[12]. Our objective is to facilitate the developer's work when developing new evacuation plans. We implemented this case study in two ways, by utilizing our proposed approach (see Figure1), then using the GAML language. In both situations, we implemented four evacuation plans which are: (I) the shortest path; where an evacuee choose the closest door, (II) random moving; where an evacuee moving to a random location until he/she gets an exit door, (III) follow the leader; where an evacuee follow a leader and (IV) follow others; when an evacuee sees people are escaping, he/she will start escaping in turn in the same direction. The results of the evacuation simulation can be viewed in Figure 7.

B. Experiment Results

Table III shows the experiment results. The results were conducted based on LoC which refers to the number of code lines that have to be written manually for each of the aforementioned evacuation plans, and the development time required in case of utilizing our approach versus coding the whole four plans from scratch with GAML. Development time is the time required by a developer to implement one evacuation plan and it is an important metric for measuring

[12] https://www.bbc.com/news/world-europe-43531684

software functionality according to the ISO/IEC 9126[13] quality model [17], [18].

TABLE III. EXPERIMENT RESULTS

		Shortest path	Follow others	Follow leader	Random moving
Using our approach	LoC_a	6	34	16	15
	Development time (minutes)	30	80	55	43
Using GAML	LoC_g	407	485	467	419
	Development time (hours)	5	13	9	7
$PALoC_a$		98.5	92.99	96.6	96.4

$PALoC_a$ represents the percentage of automated lines of code generated using our approach for each plan. $PALoC_a$ is calculated as illustrated in (1).

$$PALoC_a = \frac{LoC_g - LoC_a}{LoC_g} * 100 \quad (1)$$

Where LoC_a represents the number of lines of code manually written when using our approach, and LoC_g is the number lines of code to be written when developing an evacuation plan using GAML.

This experience results a ratio varying from 93% to 99% for the average of automatically generated lines of code using our approach. Moreover, our proposed approach completed the whole development process about 10 times faster than coding from scratch. Hence, the evaluation results approved that using our approach for developing evacuation plans reduces the required time for the development process. Furthermore, since more than 93% of the code was generated automatically, developers' effort will be saved, and their productivity will be improved.

V. CONCLUSION

This paper presents a new MDD-based approach that helps for the automatic generation of evacuation plans. This approach begins with creating models as a starting point and ends with implementing the evacuation simulation code on a MAS platform. In order to realize our proposed approach, an evacuation meta-model is introduced, a graphical editor is built, and M2C transformation is implemented. Then, the whole process was presented as a supporting tool for the developers of evacuation plans. For the evaluation, a case study of the evacuation scenario of the Kemerovo shopping mall is discussed. This case study shows that using our approach makes developers work much easier and faster. However, this paper elaborates on building evacuation which is considered as a small-scale evacuation. As a relevant future work, we are going to apply our approach for large-scale evacuations such as evacuating a city.

[13] The international standard for the evaluation of software

REFERENCES

[1] K. Rozo, J. Arellana, A. Mercado, and M. Diaz, "Modelling building emergency evacuation plans considering the dynamic behaviour of pedestrians using agent-based simulation," Saftey Science, vol.113, pp.276–284, 2019.

[2] S. Sharma, K. Ogunlana, D. Scribner, and J. Grynovicki, "Modeling human behavior during emergency evacuation using intelligent agents: A multi-agent simulation approach," Information Systems Frontiers, vol.20, pp.741–757, 2018.

[3] C. Adam, C. Garbay, and J. Dugdale, "Multi-factor model and simulation of social cohesion and its effect on evacuation," in Hawaii International Conference on System Sciences (HICSS), 2019.

[4] M. Nguyen, T. Ho, and J. Zucker, "Integration of Smoke Effect and Blind Evacuation Strategy (SEBES) within fire evacuation simulation," Simulation Modelling Practice and Theory, vol.36, pp.44–59, 2013.

[5] M. Valette, B. Gaudou, D, Longin, P. Taillandier, "Modeling a real-case situation of egress using bdi agents with emotions and social skills," in International Conference on Principles and Practice of Multi-Agent Systems, vol. 2, pp. 423–430, 2018.

[6] J. Almeida, R. Rosseti, and A. Coelho, "Crowd simulation modeling applied to emergency and evacuation simulations using multi-agent systems," in 6th doctoral symposium on informatics engineering, 2013.

[7] H. Na and A. Banerjee, "Agent-based discrete-event simulation model for no-notice natural disaster evacuation planning," Computers & Industrial Engineering, vol. 29, pp.44–55, 2019.

[8] F. Michel, J. Ferber, A. Drogoul, "Multi-agent systems and simulations: a survey from the agent community's perspective," Multi-Agent Systems, vol 1. Pp. 3–52, A. Uhrmacher, D. Weyns, CRC Press: Boca Raton, pp. 3–52, 2018.

[9] S. Jacome, M. Ferreira, and A. Corral, "Software development tools in model-driven engineering," in International Conference in Software Engineering Research and Innovation (CONISOFT'17), 2018, pp. 140–148.

[10] D. Cetinkaya, A. Verbraeck, and M. Seck, "MDD4MS: a model driven development framework for modeling and simulation," in Summer Computer Simulation Conference, 2010.

[11] M. Brambilla, C. Jordi, and M. Wimmer, "Model-driven software engineering in practice," Synthesis lectures on software engineering, vol. 3, no. 1, 2017.

[12] S. Kasereka, N. Kasoro, K. Kyamakya, E. Doungmo, A. Chokki, and M. Yengo, "Agent-based modelling and simulation for evacuation of people from a building in case of fire," Procedia Computer Science, vol. 130, pp. 10–17, 2018.

[13] A. Trivedi, and S. Rao, "Agent-based modeling of emergency evacuations considering human panic behavior," IEEE Transactions on Computational Social Systems, vol. 5, pp. 277–288, 2018.

[14] S. Wang, G. Wainer, R. Goldstein, and A. Khan, "Solutions for scalability in building information modeling and simulation-based design," in Symposium on Simulation for Architecture & Urban Design, 2013.

[15] D. Gianni, P. Bocciarelli, A. Ambrogio and G. Iazeolla, "A model-driven and simulation-based method to analyze building evacuation plans," in Winter Simulation Conference (WSC), 2015.

[16] M. Challenger, G. Kardas, and B. Tekinerdogan, "A systematic approach to evaluating domain specific modeling language environments for multi-agent systems," Software Quality Journal, 2016.

[17] H. W. Jung, S. G. Kim and C. S. Chung, "Measuring software product quality: a survey of ISO/IEC 9126" IEEE Software, vol. 21, no. 5, pp. 88-92, 2004.

[18] D. Kung, Object-oriented software engineering: an agile unified methodology. McGraw-Hill Higher Education, 2013.