Future Generation Computer Systems 79 (2018) 155–165

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Multimodel agent-based simulation environment for mass-gatherings and pedestrian dynamics

Vladislav Karbovskii[a,*], Daniil Voloshin[a], Andrey Karsakov[a], Alexey Bezgodov[a], Carlos Gershenson[a,b,c,d]

[a] ITMO University, Saint Petersburg, Russia
[b] Universidad Nacional Autónoma de México, Mexico
[c] MA Institute of Technology, Cambridge, USA
[d] Northeastern University, Boston, USA

HIGHLIGHTS

- A multimodel agent-based simulation environment (PULSE) is presented.
- Model integration techniques suggested: common space and commonly controlled agents.
- Crowd pressure metrics for simulating crushing and asphyxia in crowds are proposed.
- Simulations of evacuation from cinema building to the city streets are carried out.

ARTICLE INFO

Article history:
Received 2 May 2016
Received in revised form
28 September 2016
Accepted 1 October 2016
Available online 7 October 2016

Keywords:
Multimodel simulation
Model integration
Agent-based modeling
Emergency evacuation
Urgent computing

ABSTRACT

The increasing interest in complex phenomena, especially in crowd and pedestrian dynamics, has conditioned the demand not only for more sophisticated autonomous models but also for mechanisms that would bring these models together. This paper presents a multimodel agent-based simulation technique based on the incorporation of multiple modules. Two key principles are presented to guide this integration: a common abstract space where entities of different models interact, and commonly controlled agents—abstract actors operating in the common space, which can be handled by different agent-based models. In order to test the proposed methodology, we run a set of simulations of cinema building evacuation using the general-purpose PULSE simulation environment. In this paper we utilize crowd pressure as a metric to estimate the capacity of different emergent conditions to traumatically affect pedestrians in the crowd. The proposed approach is evaluated through a series of experiments simulating the emergency evacuation from a cinema building to the city streets, where building and street levels are reproduced in heterogeneous models. This approach paves the way for modeling realistic city-wide evacuations.

1. Introduction

Large mass gatherings attract millions of people around the world annually. Some of these are spontaneous or non-administered, whereas majority is run by teams of managers who are in charge of establishing safe and comfortable conditions for visitors. But in order to do this, emergency planning specialists and organizers of mass gatherings have to take into account a wide variety of potential threats, such as natural and technological disasters, and acts of violence. This in turn leads to a multitude of scenarios, each of which incorporates numerous objects or actors into either stochastic processes, where a similar set of initial conditions and parameter values lead to different outputs, or deterministic processes, where initial conditions and parameter values fully determine the output.

The question is—how can we study and understand such complex phenomena? One way to do this is by describing the core features of the constituting elements mathematically using computer modeling. Models imitate real systems and processes and can vary from simple projections to instances exposing high levels of complexity. Modeling is used to fulfill a large variety of tasks; from

* Corresponding author.
E-mail address: vladislav.k.work@gmail.com (V. Karbovskii).

http://dx.doi.org/10.1016/j.future.2016.10.002
0167-739X/© 2016 Elsevier B.V. All rights reserved.

emulation of emergency situations and crisis events that are used to inform decision-making, to evaluating physical capacities of the premises where events are hosted (used in planning).

A significant number of pedestrian models has treated modeled space as an isolated system that either deals with a predefined number of agents or generates (and terminates) these agents at its boundaries. These models are convenient to use especially when simulating the behavior of pedestrians in a confined space where communication with other spaces can be neglected; this is true about building-scale models, for instance. But on larger scales, the use of "sealed" models becomes questionable as these models consequently need to be expanded and simplified in order to improve their comprehensiveness and to compensate for increasing computational demands. A potential side-effect is compromising precision, which is the specific advantage of models narrowly focused on smaller scales.

In this paper we explore the possibilities of solving this issue through coupling heterogeneous models without disrupting their integrity. More specifically, we model the behavior of agents on two scales with the use of agent-based models interacting in the proposed agent-based simulation environment – PULSE [1]. We couple two modeling spaces – city district and the confined area inside the cinema, to see how interknitted abstractions representing one phenomenon (district-level evacuation process) can effectively merge. There are several factors that had been taken into account when designing integration mechanism: (a) differences in formatting among the models; (b) software-wise differences; (c) differences in data usage.

The environment described in this paper is composed of methods and technologies that are capable of aggregating into multi-model pedestrian simulation systems. This allows the user to easily design models given characteristics of space and agents' behavior and instantly process the input with the help of the analytical unit provided within the environment. Though the area of application is not limited to the simulation of crowd motion in mass gatherings, corresponding scenarios have been used as test cases on the early stages of development. In other words, the choice of models, modeling components and setting used in this paper have been chosen to demonstrate capabilities of PULSE in evacuation modeling. Here we illuminate the features of PULSE through multi-model simulation of two scales of evacuation: building- and district-level egress. To illustrate how these two levels can be merged together in the actual scenario, we addressed flood-triggered evacuation from the mass gathering venue in one of the isolated areas of Saint-Petersburg, Russia—Vasilyevsky Island.

Majority of injuries that participants sustain during mass gathering and evacuation disasters is related to crushing and asphyxiation. With this in mind, we introduce a feature that allows calculating the pressure that emerges between agents and obstacles in dense congregations. Most pedestrian models operate with density and flow characteristics that have empirically been proven to be less reliable than crowd pressure at indicating trauma in crowds [2].

The paper is structured as follows: Section 2 presents background information and related work on coupling models in computational science. An outline of the proposed methodology is presented in Section 3. Section 4 highlights two levels of models used in the case study. Section 5 is dedicated to listing the experimental setting for the empirical study. Finally, we discuss our results together with the conclusion and prospective improvements of the described methodology.

2. Related work

Though the use of multimodel agent-based simulations is not limited to the domains of computational social science, evacuation behavior simulation, urban planning, and pedestrian modeling, the present paper is focused primarily on the models, frameworks and environments that address the issues topical for these spheres of scientific investigation. Analytical survey of existing approaches, typical issues, and theoretical basics of the multi-level simulation are discussed by Morvan [3], Gil-Quijano et al. [4], and Gilbert and Troitzsch [5]. Multi-model, as well as multiscale and multilevel simulations, have received significant interest from researchers in different fields [4]. However, the terms multi-model or multi-level simulation seem ambiguous. It is due to the fact that comprehensive conceptual papers remain few and they leave the choice of terminology to the discretion of the researcher. Existing approaches in multi-model modeling lack ready-to-use solutions and languages, while tending towards context- or domain- specific systems [6]. In particular, as [7] claims, currently-utilized models do not exhibit simultaneous representation of agents, and even interactions between agents are not properly articulated—these are points which we partially address in this paper.

The most widely-accepted approach in emergency evacuation research simulations is the coupling of human behavior-related models with the ones that represent the physical properties of the hazard. For instance, Jalali et al. [8] propose a reflective middleware, or a meta-model, that links and synchronizes existing heterogeneous models through loose coupling, which helps overcome the limitations of existing approaches to model integration (represented by the standards that allow the coupling of the freshly-developed simulators). As a result, a multi-level emergency simulator that is capable of reproducing the interaction of factors, such as spread of fire and its by-products, evacuation behavior, and dynamics of communication systems, is obtained. Camus et al. [9] also propose to integrate models into common systems through the use of a meta-model. However, the domain for this implementation is specific—reproduce behavior of evacuees in the case of a tsunami-triggered evacuation. It is also worth mentioning that the absence of post-productive integration of models is widely discussed in the domain of multi-level and multi-model simulations since the previously-presented approaches are built around highly abstract languages like MIMOSE and SmallTalk [10], and do not provide this capability, which might strongly affect the effectiveness of the developmental process in a negative way. Korhonen et al. [11] look into a problem analogous to the one described in [8], yet they consider only two components to be integrated into the simulator—the already-existing fire dynamics simulator extended with the evacuation behavior module. In [12], authors go even further in their attempt to specify the approach for integrating models that represent physical space of a tall building, fire hazard dynamics, and the complex evacuation behavior of intelligent agents, and propose an augmented reality-based solution that allows the processing of real-world sensor data. It may be claimed that the ambition of creating multi-level models for single-hazard or single-case coexists with attempts to produce complex interactive "serious game" architectures and training environments that consider a large variety of scenarios, levels, and scales aimed to meet the demands formulated by military and civic authorities [13].

To position this paper in the multi-level and multimodel discourse, a comparison with conceptual elaborations of the approach in the field is presented. For instance, in the work of Gilbert and Troitzsch [5], the authors characterize multi-level models as being able to reproduce more than two levels of abstraction, with each of them inhabited by complex, but not necessarily numerous, communicating agents. The model described here may be considered complying partially with this description since the number of simulated agents is significant and the communication between the agents of different levels might be described as replication of agents leaving one level of the model to another one. As

Morvan claims in his extensive research on current approaches to multi-level simulations [3], there are three predominating problems that multi-level simulation seeks to address: (a) modeling cross-level interactions, (b) heterogeneous models coupling, and (c) contextual adaptation of different levels of detail in order to optimize computational resources. Nevertheless, the question of defining multi-level modeling and conceptualizing the corresponding approaches is still open. Gil-Quijano et al. [4] claim that many multi-level models are in fact focused on the reproduction of a predefined single level, whereas other perspectives emerge in the observers' view, which may be as well interpreted as a criterion. Gil-Quijano et al. also claim that the "emergentist" approach is not reasonable on a large scale. Contrary to this statement, however, it is assumed in this paper that there are cases that require and allow detailed simulations of different levels that consider agents as the primary simulated entities, as opposed to flows and aggregated groups. With regard to weak and strong integrations that are outlined in [3], we propose a hybrid option—an environment that brings together models that share agents, which is a requirement for strong integration. At the same time allowing a single type of environmental objects or connectors between models, which does not allow it to fully ascribe to either forms of integration.

3. Methodology

The PULSE environment is divided into three systems namely: *agents' behavior module*, *model integration*, and *analytical unit*. Each of the systems is flexible enough to incorporate a number of subsystems—both native (developed in PULSE directly by its agent-based framework) and external (already-existing models, with or with no available source code). Apart from various domain-specific models and analytical tools, the environment also supports assimilation of data from various sources, such as fractional video feed data and mobile device tracking information. All of the elements that can be potentially incorporated according to a specific task are bound by the composite environment. Here we draft the setup that is designed specifically for the evacuation simulation experiments that will be discussed in the Section 5 of this paper. Specifically for this setting, two integrated agent behavior modules generate the input for the crowd pressure calculator that is plugged into the analytical module.

3.1. Agents' behavior module

In agent-based modeling of pedestrian movement, agents are autonomous entities capable of operating within a given environment, interacting with other agents, and adjusting their behavior accordingly. We use discrete event agent-based modeling [14] with fixed, but configurable time step. The agents' behavior is defined by rules, which are formal conditional constructions that provide decision alternatives assigned to particular circumstances. The behavior of pedestrian agents, which are based on available data, users' preferences or relevant tasks, can be tuned through a number of subsystems and these are described as follows:

1. WorldKnowledge is a filter-like module that provides information about the surroundings that is available to the agent. This information includes the features of the modeled space to which access can be discretionally restricted according to the perception range and role of agent.
2. PlanningSystem is responsible for defining high-level activities of agents. By default, we use probabilistic queuing that attributes each action to a certain probability.
3. BehaviorSystem represents the detailed behavior of an agent for all cases within the model (i.e. doing an ongoing activity). We use our own implementation of Marzinotto et al. [15] behavior tree, which is flexible behavior mechanisms, providing modularity [16].
4. NavigationSystem helps agents find their paths towards objects. This is one of the subsystems that can be implemented in different ways and thus are easy to replace. We use simple Euclidean movement for simple cases, shortest path navigation on graph with A* algorithm, and our implementation of navigation fields [17,18] if we need to reproduce more detailed movements. These mechanisms cover most required cases, however the subsystem could also be overloaded by other implementations.
5. MovementSystem is used for local movements of agents and can include collision avoidance in those cases where movement of agents can be significantly altered by impenetrable obstacles and other agents (which is common in pedestrian simulations). For some cases, when we need Euclidean movement collision avoidance is also not necessary. In accordance with evaluations made in previous surveys of models [19], we have chosen and implemented RVO [20] and Social Force [21] models.

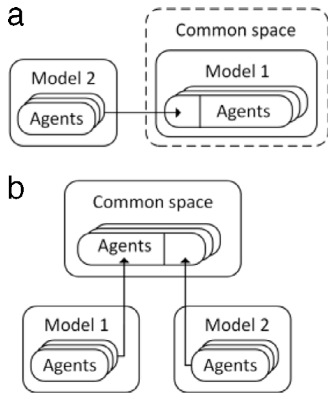


Fig. 1. Two forms of integration through common space.

The subsystems presented above have been labeled as for the sake of convenience and represent abstract features that an agent in the environment can possess. Unlike subsystem 1 that can only determine what amount of information is available to an agent at a given time, subsystems 2–5 can be replaced equivalently, extended, used in an ensemble [22], or removed altogether in some cases.

3.2. Model integration

Model coupling in PULSE is maintained through the following constructions: (a) common space and (b) abstract commonly controlled agents. These have been developed as to meet specific demands that might arise in crowd-related research. For instance, in some cases, a user, who possesses the default pedestrian model, may need to simulate movement inside a building. Hence, expansion of the initial functionality of the model is only necessary. Fig. 2 depicts the general scheme of multimodel approach for building-district case. Generally, there are two key options: (a) implement the required logic in the basic model, (b) take an already-existing model or develop a new model and integrate it with the primary model. The second option, however, has better opportunities for model reusability.

Fig. 1 presents two forms of common-space integration implemented in PULSE: (a) inside the model, that model called primary (b) and outside. Models used in PULSE can be classified into native (developed in PULSE directly by its agent-based framework) and external (already-existing models, with or with no available source code). Model 1 in the first type of integration

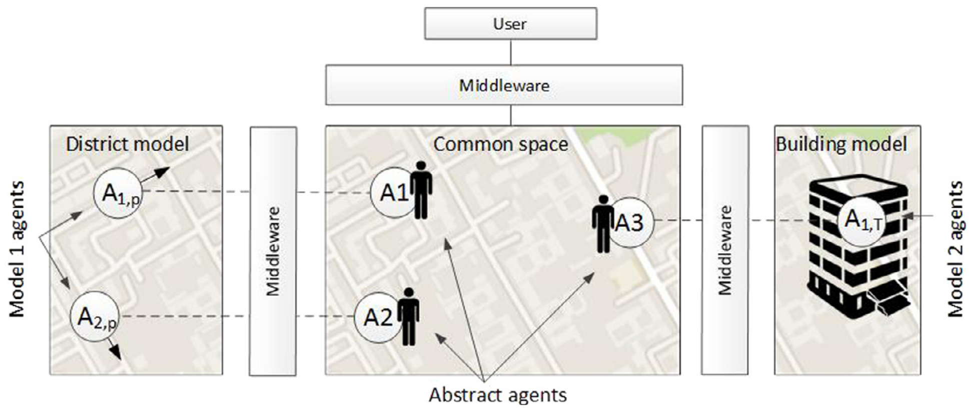


Fig. 2. Spatial integration by agents tested on two models: district and building models. All entities are mapped as abstract agents in the common space regardless of the model they belong to. One model (district model) can handle abstract agents on the first iteration while another model (building model) can administer more actions in succeeding iterations. The common space still treats them as similar agents all throughout the iterations. With additional middleware, the common space can be represented as a single model for the user.

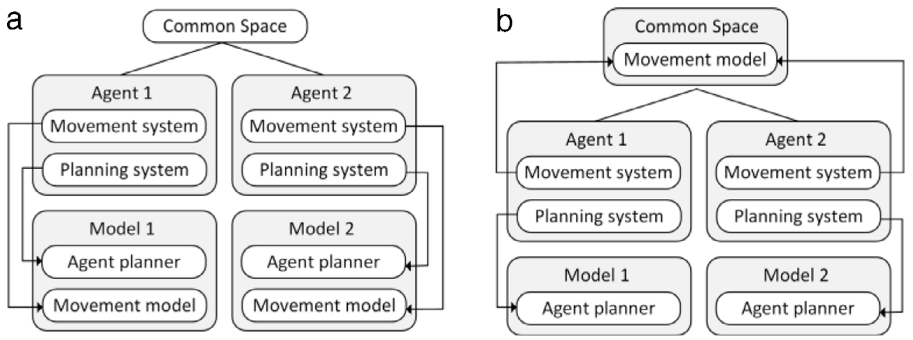


Fig. 3. Two examples of integration through the agent constructor module (a) autonomous models to common space; (b) two models with different planning system but with shared movement system.

can only be native, whereas additional models are allowed to be external.

In order to support the exchange of data between models in real time execution—a feature that is important for time-stepped models, such as agent-based models, it is necessary to implement additional methods of interaction. With this in mind, we introduce the concept of abstract commonly controlled agents (abstract agents, Fig. 2), whose key features are agent representation in an abstract model-independent space (common space), and agent management by a number of models. All entities are mapped in the common space, regardless of the model they actually exist in. Entities can then interact with each other during the simulation given that their actual models support the behavior. An abstract agent can be handled by one model (e.g. district model) on the first iteration, but after a few iterations the actions of the same agent can be administered by another model (e.g. building model) say if the abstract agent enters a building. In the perspective of the common space, however, this is still the same agent. With additional middleware, the common space can be represented as a single model for the user, despite having hidden internal models. The two models (district and building models) used in testing our platform as shown in Fig. 2 will be discussed in detail in Section 4.

Abstract commonly controlled agents (abstract agents) are agents operating in the common space. The rules followed by abstract agents form a union of rules for all used models. For example, the rule *movement* allows the agent to move from starting point to destination. Mobilization of the abstract agent can be handled by the district model on the first iteration, but after a few iterations, the relocations of the same agent can be administered by the building model. The decision of the agent to move from the district level to the building is made within the planning system. Consequently, agents' spatial relocations are processed in the movement system. The key principles of this are outlined in Fig. 3. There are two types of integration that can be implemented through the agent constructor module. See Fig. 3. The first setting allows autonomous models to share space, but not interact with each other (which is suitable for cases where objects are isolated—i.e. modeling buildings). The second setting facilitates interactions between agents that are driven by the relevant submodules. These submodules in turn allow the simulations with agents to run. These are driven by different models, but are positioned in the same space. The synchronization of agents can be described by a simple algorithm. See listing 1. When transferring from one model to another, an agent is removed from the model that has governed its behavior previously. In order to maintain the agents' state throughout the transition, additional data, apart from identifiers, mass, gender etc., is handled.

```
foreach A in AbstractAgentList:
        if (isTimeToChangeModel)
                A.ChangeControlModel(requiredModel)

        A.step()
```

Listing 1: Integration by agents, control loop

The synchronization of models is described by a simple sequence (Fig. 4). Different models have their own internal representation of time and these time representations can either be absolute or relative. Accordingly, time steps can be regular (time-stepped models) or irregular (models, providing the set of events). If necessary, it is possible to use the offset, which is essentially the multiplier of time. All these factors contribute to the value of variable isTimeForStep for each model at a given step. The step incorporates receiving data from the model, sending new data

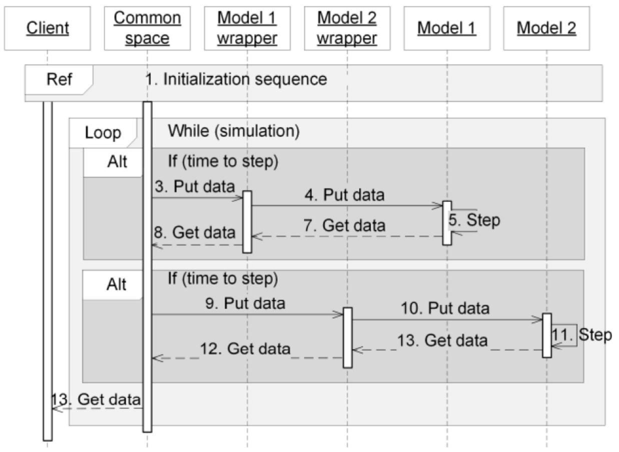


Fig. 4. Sequence diagram of data exchange.

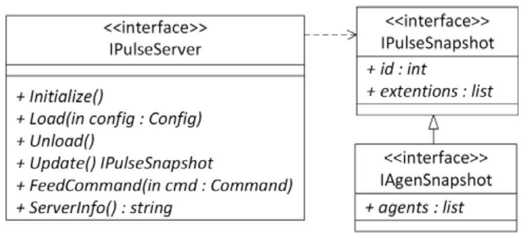


Fig. 5. Model integration and data exchange interfaces.

to the model, and the actual execution of the step. All the models are initialized instantly, which is an interim solution. However, we plan to improve this behavior in the future in such a way that the models will be loaded and unloaded on demand. For instance, it is only when the agent enters an area, as administered by a model, that the latter is loaded. Hence, triggered by a particular event.

In our implementation of common space, objects exist in the geographical coordinate system (latitude, longitude), while coordinates of the two models are in geographical Cartesian. In this case, space synchronization is only necessary. This is implemented through the transformation of the spatial coordinates via three steps: (1) mapping the origin of the coordinate system of model to the required geographical point, (2) multiplying model distance units to convert it to meters if necessary, and (3) converting meters to latitude and longitude.

3.3. *Technical features of integration*

Fig. 5 shows the diagram of the key integration interfaces. IPulseServer interface implementation is necessary for model integration. It follows a set of requirements. First off, it should be capable of initializing, loading and unloading the model, updating the state, managing commands and data. Data exchange is performed through snapshots, containing an identifier and a list of agents (in cases where they are used). Particular forms of snapshots should be implemented based on the type of data that need to be transferred.

Another major part of integration is data transmission between models, which perfectly fits the client–server architecture. Here we choose the transport layer for data transmission. Although Transmission Control Protocol (TCP) ensures data delivery, this type of protocol is not suitable for intensive data exchange (with low latency). 1%–5% of all packets are lost [23]. For this reason, TCP waits for acknowledgment packets to provide reliable delivery and resends packets in cases when it suspects that the packets are lost. Such an approach is robust, but this introduces a latency of up to several seconds. Real-time simulation strongly suffers from latency and waiting for acknowledgment is not acceptable. At the same time, real-time visualization is tolerate to some data loss, thus we choose UDP as the main transport layer for our system. To build up the protocol on top of UDP, we choose the following data types for transmission:

1. Snapshots are binary results of the modeling process containing all data required for data exchange and visualization.
2. Commands are binary chunks of data that represent instant activity, such as pressed buttons, mouse position, etc. Their main use is to control the simulation from the user interface. The size of user commands does not exceed several bytes.
3. Reliable Requests and Responses are data used for establishing virtual connection, reliable simulation control and data exchange. Reliable requests and responses implemented using acknowledgment packets with resending if protocol suspect packet loss.

The previous paragraph may raise the question on whether the use of both UDP for time-critical data and TCP for reliable data is necessary. The answer lies in the fact that both protocols are built on top of protocol IP. Internal implementation of TCP tends to induce UDP packet loss in mixed network traffic [24]. To avoid such effect, we opt at implementing reliable data transmission over UDP.

We use temporal coherence to compress snapshots data. Each snapshot contains its sequence number. When a client receives a snapshot, it adds acknowledgment to the user command. When the server receives a user command with acknowledgment, it then detects that a particular snapshot has been delivered. The server could now send compressed binary delta between the acknowledged snapshot and the snapshot that is being sent. If the event compressed snapshot is too big and does not fit into the maximum transmission unit (MTU) of communication protocol (1500 bytes for common networks), the servers then split the compressed snapshots into chunks. The probability of snapshot loss grows exponentially with increasing number of chunks. To prevent heavy snapshot loss, we implement a reliability procedure for snapshot chunks. Hence, if a new snapshot is not ready and the servers detect lost chunks, the server resends them.

3.4. *Analytical unit*

Classes that embody the code of the analytical unit are used to gather, record, and chart the data. Using these classes, a modeler defines data sources and hooks up recording or charting classes to these sources. Data can then be easily recorded in a tabular or customized format and charted in a sequence graph, histogram or user-defined plot. Most accidents pertaining to human crowds in the last decades show that fatal cases are often related to asphyxia because of crushing [25]. Pressure waves modeling, with the help of fundamental diagrams [26], which are usually used for quantitative assessment when simulating crowd movement, does not fully meet the demands of estimating the traumatic effects of crowd motion. Cases associated with static or almost static crowd fundamental diagram values (speed, density and flow) do not always give us information about possible dangerous zones. In order to collect this type of data, we introduce a metric called crowd pressure. Crowd pressure has been previously identified as more suitable for detecting dangerous zones in the crowd compared to density alone [2,27]. As suggested by the authors, pedestrians push each other in reaction to the increasing density of the crowd. This consequently increases the pressure. Note that the density value can remain the same in this case. Model-wise, this type of behavior can be reproduced through the use of agent-based modeling (flow- and particle-based models do not

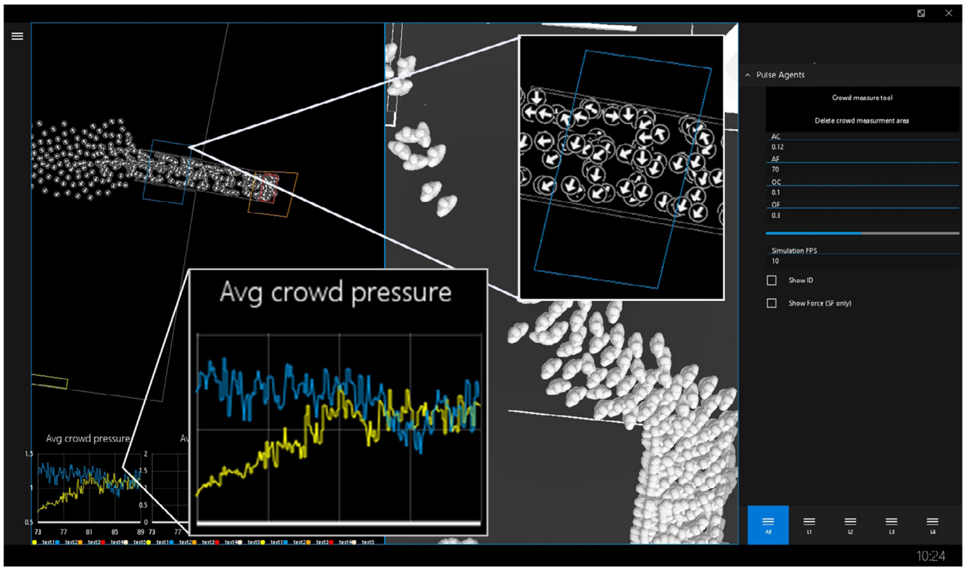


Fig. 6. Snapshot of our application highlighting the average crowd pressure feature. Average crowd pressure is calculated and plotted in real time for selected zones. Zones can be selected manually in required places.

provide necessary level of detailization of pedestrian behavior) so that agents assess crowd conditions and adapt their behavior accordingly (i.e. increase velocity as to escape the point of critical pressure). An example of crowd pressure calculation is highlighted in a sample snapshot of the application in Fig. 6.

Lee and Hughes in [28] used standard forward–backward autoregressive modeling to predict crowd pressure. For agent-based models, however, this metric can be achieved through the use of force-based models. In this paper, we use our implementation of social force model to calculate the crowd pressure [21]. However, there are existing empirical studies of pressure in crowds that describe thresholds of time and force in newtons. Evans and Hayden in [29] performed tests on live subjects to find push tolerance for males and females. Moreover, Hopkins et al. [30] estimated thresholds for crush asphyxia as 6227N in 15 s or 1112N in 4–6 min. These values characterize the moments of onset of asphyxia that is the main risk associated with being in a dense crowd. Despite these developments in research, it is still difficult to connect the current pressure model and the real pressure values due to lack of data. Additional research needs to be done, which includes pressure measurement and calibration of agent-based model to data at the same time.

Our crowd pressure model gives a relative value that indicates a potentially dangerous area in which the relative pressure is abnormally high. To calculate pressure, we use internals of social force model. Note that it is also possible to use any force-based model. The current pressure on the agent is calculated according to the following formula:

$$p = \sum_{\beta} \vec{F}_{\alpha\beta}(\vec{e}_{\alpha}, \vec{r}_{\alpha} - \vec{r}_{\beta}). \tag{1}$$

In this expression p refers to the total repulsive effect of other agents (i.e. how other pedestrians β affect the current one) to a particular agent α. $\vec{F}_{\alpha\beta}$ is repulsion force between agents α and β, $\vec{r}$ is actual position and $\vec{e}$ is desired direction of agent.

4. Models

Two agent-based models—district-scale and building-scale models, are used for prototyping the multi-level environment for evacuation using multi-agent simulations. The building-scale model, which was built "from the ground up", simulates residential dynamics in buildings and other closed spaces while the district-scale model, a modified version of the authors' previous work [31], aims at a higher level of representing pedestrian dynamics. These models are both multi-agent, use the activity approach to path planning, and have common mechanisms of representing the heterogeneity of physical profiles of agents. Despite having similarities, these models also differ in terms of the variance in scales of representation of corresponding simulated environments. For instance, the district-scale model does not presuppose the level of detailization of agents' pedestrian behavior that cannot be neglected in the building evacuation model. The coupling of the models in question has been performed using PULSE—the multi-model integration framework. The following subsections describe the models used for simulating the case investigated in the present paper and the tool used for their interaction and integration.

4.1. Building model

The building model environment E is presented as a pseudo-3D continuous space, i.e. a set of interconnected two-dimensional levels. It is composed of the following elements: (a) points of interest, (b) navigation route graph that links the points of interest, (c) impenetrable obstacles and (d) portals (uni- and bidirectional). Points of interest are objects that attract agents and imply potential interactions that can take various amounts of time. Examples of points of interests are seats, bar counters, tables at the waiting area or ticket office in the cinema. The route graph facilitates the connection of the points of interest that reflect the most common relocation patterns and sequences. Obstacles are solid objects such as walls through which agents cannot pass through. Portals allow agents to move between two-dimensional levels (using the stairs and elevators), around the same level or out of the building.

The model uses the following types of input data:

1. Building floorplans (composition of levels in a building). Each floorplan includes information on the walls, obstacles, points of interest and waypoint-based route graph.
2. Cinema movie sessions schedule.
3. Description of the agent role structure (cinema activities for visitors and staff).

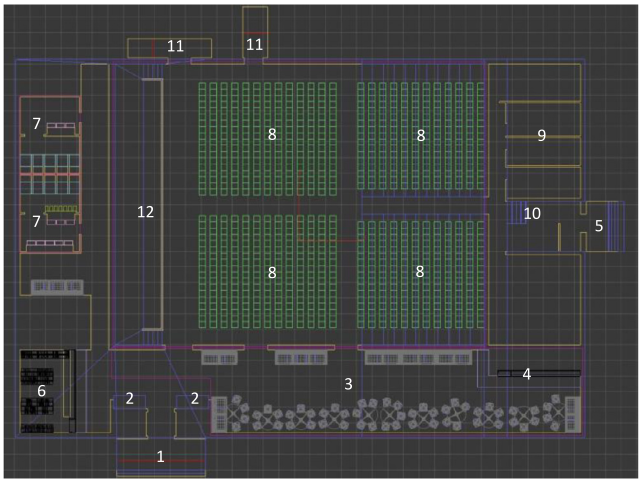


Fig. 7. Cinema plan with the following key objects: 1- main entrance, 2-ticket offices, 3-waiting area, 4-bar, 6-cloakroom, 5-service entrance, 7-toilets, 8-seat places, 9-staff rooms, 10-s floor stairs (staff only), 11-emergency exits, 12-screen.

4. Description of the physical classes of agents represented in the simulation (which lays the foundations for the estimation of the maximum speed and other movement attributes).
5. Data on number and characteristics of agents entering the building.

It is worth mentioning that the datasets matching the types of data 1–4 are static and thus configured as separate files, whereas the data on agents entering the building can be both static and dynamic. In the case when the datasets are dynamic, the generation of agents on a particular level is dependent on the input from another model (for instance, an agent leaving the district-scale modeled space through the portal that marks the entrance to the cinema is transferred to the cinema-scale model). The example of the cinema building floorplan is presented in Fig. 7 and is composed of the following key objects: 1- main entrance, 2-ticket offices, 3-waiting area, 4-bar, 6-cloakroom, 5-service entrance, 7-toilets, 8-seat places, 9-staff rooms, 10-s floor stairs (staff only), 11-emergency exits, 12-screen.

Agents operating within the environment of the building scale model are virtual "staff" and "visitors" corresponding to the physical space being modeled: $A_b = \left\{A_{1,b},\ A_{2,b}, \ldots A_{n,b}\right\} \subseteq A$. Each agent is described through a combination of internal features (current position, nearest desired position or point of interest, present speed, maximum speed, performed and scheduled activities) and behavioral rules. The latter are linked to the functional roles (representing the generalized "purposes" of agents' presence and aims of the corresponding relocations within the modeled environment) allocated to the agents' population. Each agent is allocated strictly to a single role at a time. For the cinema case, a function-based composition of the population of agents has been developed with two major classes: visitors and cinema staff members, defining the attributed mandatory primary points that set the basic route. For instance, in the case of a cinema visitor, these points are the entrance, seat location, and entrance or exit portals. Complementary to them, a number of subdivisions based on gender, intention to visit the bar, necessity and purpose of visiting the toilets, type of ticket, etc. has been elaborated to demarcate the probabilistic alternatives for secondary activities; after the movie is over, an agent may either visit the toilet and the cloakroom or proceed straight to the exit.

In order to bind the behavioral patterns of the simulated agents to the physical environment that they are operating in, a set of navigational and path-planning mechanisms has been composed based upon the adaptation of the solutions presented in [18]. The resulting multilayer navigation approach is built from the activity scheduling, path planning and micro-level components. Since each agent has a functional role and an appropriate schedule, his path is merged from the sequences of primary and secondary activity points that are thus linked by the operational waypoints of the navigation graph. Because two waypoints can communicate in a number of ways, in order to facilitate optimal relocations, agents' travel along the graph vertices is informed by the A* algorithm. Potential collisions with stationary objects and other agents are solved with the use of the modified Social Force approach first presented in [21]. This is force-based model, which means that all the interactions related to agent movements are regulated by the forces, for example attraction force between agent and his goal, repulsion force between agents or repulsive force between agents and obstacles. The calibration of model is based on empirical video data of pedestrian flow. The method is presented in [32].

R_b—Building scale model rules:

$W_b \subset R_b$—In-building warning behavior rules. Rules whose execution is triggered as soon as an agent receives the warning message urging it to leave the building immediately.

$C_b \subset R_b$—In-cinema general behavior rules. The execution of this set of rules flows from the very moment an agent enters the cinema building. Each visiting agent is obliged to buy a ticket (or to collect a pre-booked one) and proceed to the designated seat. With a certain probability that is imposed imperatively, an agent can use the cloakroom, go to the bar, waiting area or the toilet.

$L_b \subset R_b$—Leaving-the-cinema rules. The rule is brought into execution when the agent leaves the building. Agent only uses the usual exits. Further regulations of the agent as well as his coordinates are delegated to the district model.

$E_b \subset R_b$—Evacuate-the-cinema rules. Similar to L_b rules, but agent can use emergency exit. This rule is active only after evacuation notification.

4.2. District model

The district/city scale model is an elaborated version of the multi-agent model described in [31]. In its current state, the model has not only received a significant degree of autonomy but has served as the cornerstone for the integration of other models within the PULSE environment. The inputs for this model are:

1. City infrastructure data. This includes information about walls, obstacles, buildings, points of interest and road graph.
2. Cinema movie sessions schedule.
3. Description of the agent role structure (activity chains for social-economy classes).
4. Description of physical classes of agents represented in the simulation.
5. Initial distribution of agents in the modeled area (350k agents for Vasilyevsky Island, St. Petersburg).

$A_d = \left\{A_{1,d},\ A_{2,d}, \ldots A_{n,d}\right\} \subseteq A$ – District scale model agents, R_d – district scale model rules:

$D_b \subset R_d$—Daily behavior rules. Following these rules, agents visit only points of activity that correlate with their socio-economic status (for instance, office buildings, universities, specific shops, leisure facilities etc.).

$B_d \subset R_d$—Simple building rules. When agent enter in the building, which is not represented by separate model, he just wait there required amount of time, without special simulation of his activities or movements inside this building.

$C_d \subset R_d$—Building-entering rules. The described type of rules serves as basis for the regulation of the behavior of agents entering the cinema building. Having entered the building, agents' properties (such as coordinates) migrate to the building-scale model.

The example of agent's daily schedule and visualization of current agent density are represented on Fig. 8.

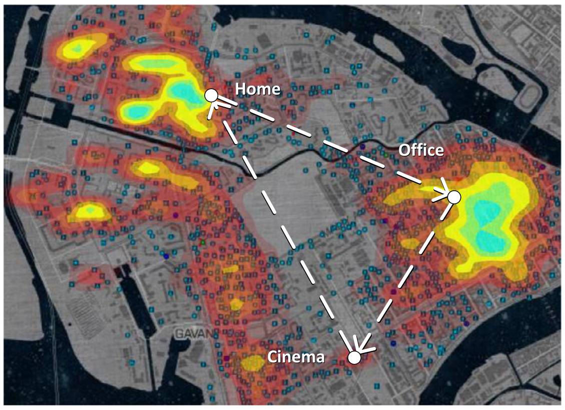


Fig. 8. Example of agent's daily schedule.

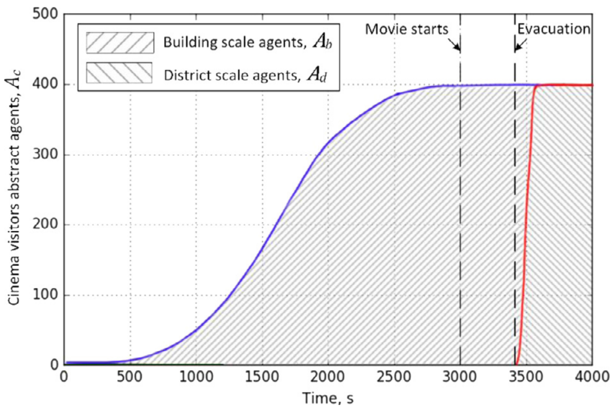


Fig. 10. Spatial integration by agents (agent migration). Blue line represents the number of agents controlled by the building-scale model, while the red one stands for the number of agents handled by the district-scale model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Multimodel evacuation simulation: Case study

The aim of this section is to illustrate the idea that a wide variety of research can be carried out by using a set of integrated models where specific scales and levels (as well as points of view) are represented by separate models. For instance, it can be applied to study the dynamics of evacuation in a city area struck by a flood [33]. Flood, in this case, serves as a trigger for evacuation and a factor that can affect its dynamics.

The district-scale model provides a common space here, whereas the building-scale model serves as a secondary model. A method of commonly controlled agents is applied to ensure the integration of the agents used in both models. Moreover, it provides models with a capacity to exchange data in real time. With regard to spatial integration, agents that are operating within the secondary model (building-scale model) are also mapped on the common space (in this case—the one offered by the district-scale model), but their behavior is governed by the subsystems from the former model. In other words, from the users' perspective, there is no difference where particular agent is modeled, since it is mapped on the common space. Fig. 9 shows the example of integration of the two models through common space and PULSE.

We note that the results of the original research for the emergency evacuation dynamics is not among the prime objectives of this paper. In other words, the setting that we use has been extended with a simple model of emergency evacuation precisely for the purpose of testing the concept.

The following is the description of the proposed simulation scenario. First, we reproduce the daily commute and shopping/leisure travel dynamics on Vasilyevsky Island (St. Petersburg, Russian Federation) using the district-scale model. It is assumed that some agents attend the cinema as a form of social and recreational activity. From the district-scale model, a set of agents is derived and handled (as soon as agents arrive to the cinema prior to the start of the session) to the lower level represented by the model of pedestrian indoor behavior. After entering the building, agents perform a series of activities that are built from obligatory (activities that cannot be skipped such as entering the building, taking seat, and evacuating the building) and optional (visiting the café, restrooms, etc.) activities. Within the time frame of the session, agents receive alarm notification urging them to evacuate immediately from the building. Agents (both representing staff and visitors) attempt to leave the building using the shortest available route.

The global set of agents (A) consists of two subsets, representing the agents governed by separate models: $A = A_b \cup A_d$. Thus $A_c \subseteq A$—agents visiting cinema. For the described experiment, it is true that each agent belonging to A_c keeps his/her affiliation with the set even after leaving the cinema building. The results of the experiments are presented in Fig. 10. Agents' arrival to the cinema is suggested as a non-random process, which can be inferred from the plot. It is determined by district/city-scale model and the distribution of agents' daily activities. The service time is deterministic given the limited number of services (only one—the cinema). After the emergency notification is disseminated, all agents quickly leave the building. In simulation terms, when an agent leaves the cinema, he/she loses the connection with the building model and becomes linked to the city/district-scale model.

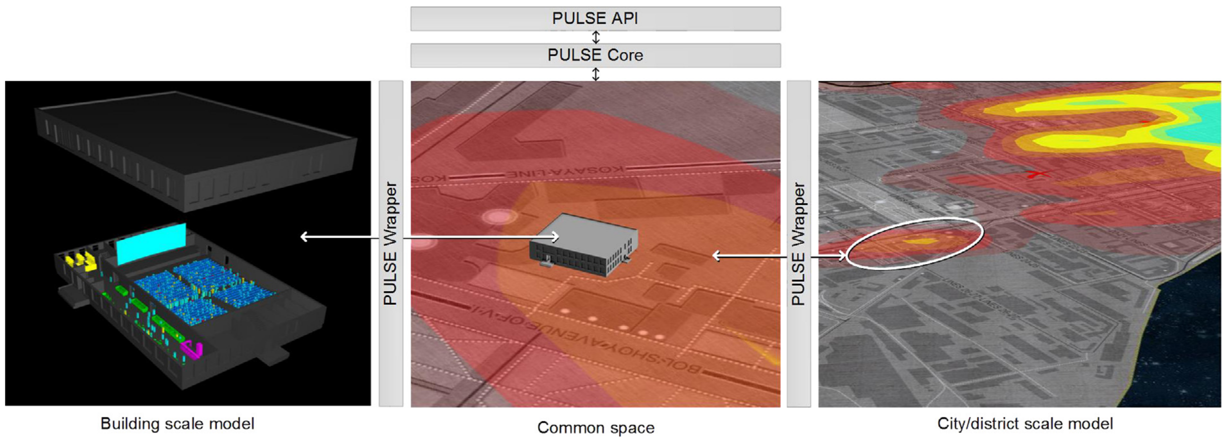


Fig. 9. Integration of two models through common space and PULSE.

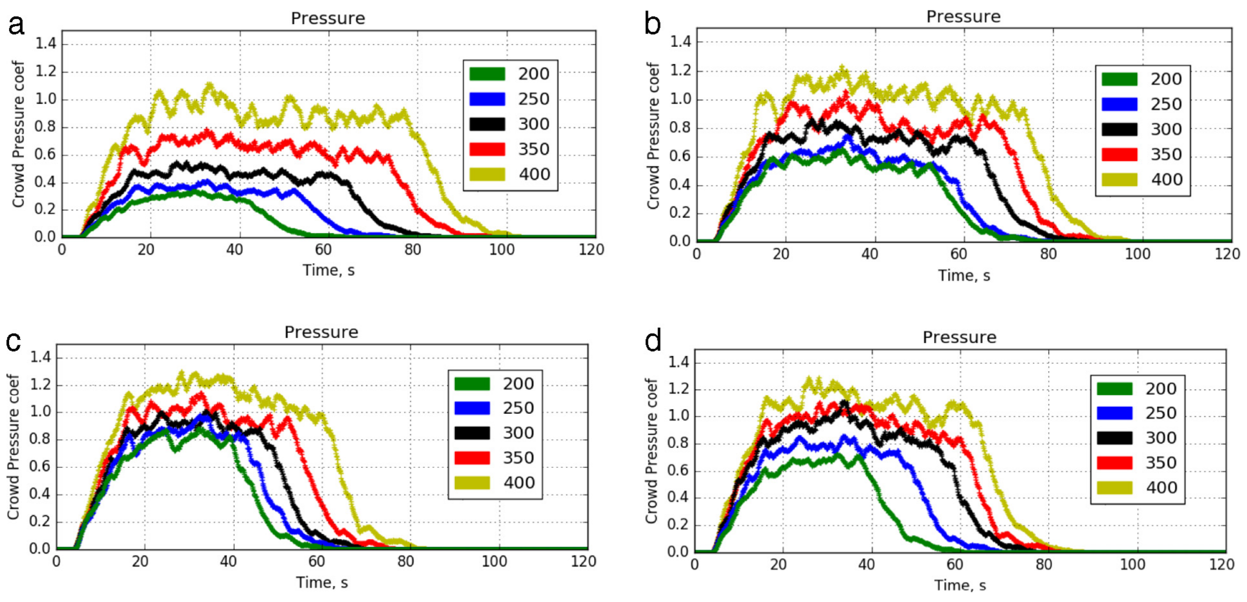


Fig. 11. Ensembles of crowd pressure dynamics for agents evacuating using different available exits: (a) main entrance (b) rear right exit (c) top left exit (d) top right exit.

Fig. 11 depicts the ensembles of the crowd pressure metric values near four different exits from the cinema building. Each of the graphs represents a set of single-run experiments (with a given quantity of agents generated for each of them). The number of agents "attending" a movie (200, 250, 300, 350 and 400 agents) have been explored to achieve these results. As it can be derived from the graphs, increasing quantities of agents produce higher levels of fluctuation in crowd pressure. However, it can be inferred that an upsurge in the number of agents involved in the simulation insignificantly adds to the inter-agent pressure and is equally fair for each of the exits. The entrance, on the other hand, serves as a balanced egress choice as it provides agents with more space to navigate. Top exits hinder the relocations of agents given the spatial constraints in the adjacent area. Moreover, top exits are closer to the central hall and thus provide a faster, yet less "comfortable" alternative to rear right exit, as it is not preceded with the crowd routing fences.

6. Scalability research

In order to explore the scalability of the methods presented, we performed a simple test that includes two series of experiments. The first series depicts scalability by agents (Fig. 12(a)) for the following: (1) model with simple movement system (no social force or RVO models), (2) model with our implementation of social force, and (3) data overheads for the assimilation of an exact amount of agents to common space. The number of agents was altered up to 1 million.

Scalability by number of secondary models was studied as well (Fig. 12(b)). We have ranged the number of agents as in the previous experiment: (1) up to 100 000 agents for one secondary model, and (2) every secondary model added 10 000 agents.

We see a linear relationship for all measured values with the count of agents (Fig. 12(a)). Thus, computation time for one model will increase linearly with the number of agents regardless of the choice of model either with or without social force as well as overhead processing.

However, if we not only vary the number of agents, but also the number of secondary models connected to common space, we no longer see a linear time growth compared to that of only one secondary model operating on the same total amount of agents (Fig. 12(b)).

In this case study, we used parallel implementation of the system and ran it on one PC. Parallel computation is ensured through the execution of different models as independent threads within a single application. In addition, the models are parallel. To achieve this OpenMP and.Net TPL technologies were used.

All series of experiments have been performed on PCs with the following build: quad-core Intel i7-5820k CPU 3.3 GHz, 16 GB Ram, 1 GB/s network.

7. Discussion and conclusion

In this paper we have presented a methodology that can be used for conducting complex experiments involving multiple agent-based models. It is suggested that the behavior of agents can be tuned with the modular structure consisting of flexible sets of compatible units (described in Section 3.1): informational, planning, decision-making mechanisms, navigation and collision avoidance subsystems. To solve the issue of coordinating the operation of models, we also proposed a method of model integration (described in Section 3.2) using common virtual space and abstract common agents. These help us achieve a clear interaction between different models and their entities—agents. In order to initiate the analytical module, we have suggested using crowd pressure calculator as it represents a major factor of injuries in dense crowds. This analytical unit (described in Section 3.3) has been designed specifically to present corresponding estimates in a comprehensible visual form. It uses the environment to collect, register, and chart information about the pressure between agents and obstacles. As output, the module serves the data on relative pressure that allows to identify extreme pressure points.

It has been mentioned throughout the work that although PULSE supports crowd pressure estimation, additional empirical research is yet required to calibrate models to empirical data. More precisely, a methodology for translating actual metrics into model parameter values is required. Additionally, results of the experiments (Section 5) suggest that crowd pressure starts to oscillate along with increasing number of agents involved into the simulation. This observation foreshadows the extension of the

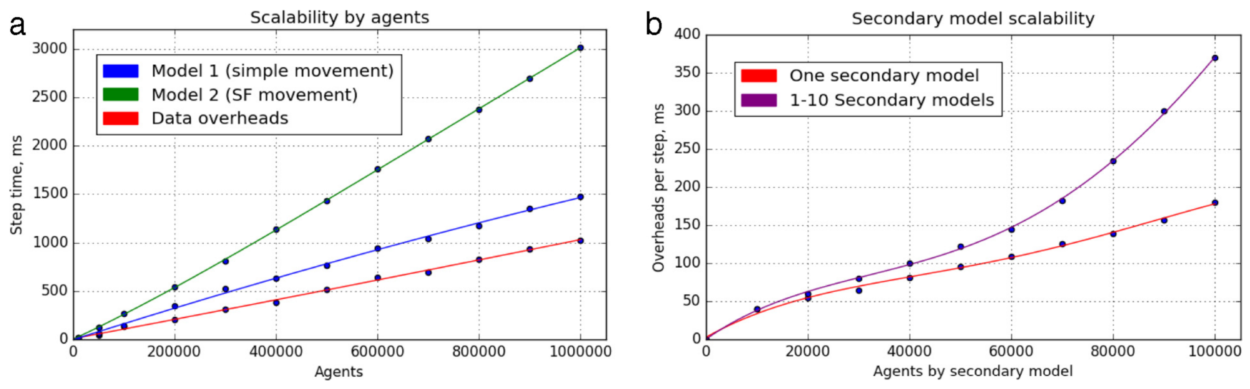


Fig. 12. Scalability test results: (a) scalability by number of agents (b) scalability by number of secondary models.

application of the module for studying pressure waves (by analogy with density waves).

The techniques mentioned above were implemented in the PULSE simulation environment. The PULSE project is an open source solution distributed with MIT license and is openly available via GitHub [1]. It has three areas of application: (1) flexible agent-based model constructor application, (2) middleware model integration support, and (3) simulation analysis and visualization. In order to exhibit its applicability, we have run a set of multi-model evacuation simulations. Two models were used in the experiments: (a) the building-scale model that was implemented to simulate the egress of agents from the cinema, and (b) the city/district-scale model that was used to simulate the daily activity of virtual agents at Vasilyevsky Island area (Saint-Petersburg, Russia). Both models were implemented, integrated, and analyzed through the agent-based library from the PULSE. It is worth noting that the chosen models and corresponding modules related to data analysis and agent behavior have been chosen in accordance with the aim of illustrating the case where model integration is crucial.

Scalability tests show that time step increases linearly when a single model is used. However, an increase in the number of models leads to a change in the form of computation time growth. This casts certain limitations on real-time simulations, as increased numbers of agents and models can significantly slow down the calculations. Moreover, network data transfer shall be studied as well, since overhead has been analyzed here with regard to processing data that has already been received. As such kind of research requires additional methodology and metrics that are outside the scope of this paper. However, we suggest to explore these in future works. Distributed execution was implemented through the UDP protocol. However, as previously mentioned, it has not been researched yet. The distribution of our system is another topic aimed for a separate research, which will be conducted in the future. This research should include different protocols, as well as integration with cloud-based platform, that will be responsible for dynamic allocation of computing resources.

At this point, we have gained plausible results from integrating models for the case derived from the area of emergency crowd management (planning and assessing the evacuation of the visitors of an entertainment facility). In order to provide the presented methodology with more elaborated test, we plan to integrate the proposed scheme into the operational decision-support system in attempt to improve the provision of information, which would assist practice of planners and rescue service officials. To widen the scope of application of the methodology described here, we are looking forward to reproducing other scenarios. These may replicate various interplaying urban processes or other cases from planning and management.

Moreover, we plan to make a few important extensions to PULSE and the related models. One of our current works in-progress is focused on PULSE, its infrastructure and code base. We plan to improve the agents' navigation system and extend it by introducing new levels and scales (in addition to planning, pathfinding and collision avoidance). One of the improvements we are planning to implement in the nearest perspective is the integration of the district/city model (also known as the virtual society model) and a traffic model.

Acknowledgment

We would like to thank Alva Presbitero for her help in editing the manuscript. This paper is financially supported by Ministry of Education and Science of the Russian Federation, Agreement #14.584.21.0015 (11.11.2015).

References

[1] V. Karbovskii, PULSE project, available at: https://github.com/vladkar/pulse-project-open, 2016.
[2] D. Helbing, A. Johansson, H.Z. Al-Abideen, Dynamics of crowd disasters: An empirical study, Phys. Rev. E 75 (4) (2007) 46109.
[3] G. Morvan, Multi-level agent-based modeling-a literature survey, 2012, arXiv preprint arXiv:1205.0561.
[4] J. Gil-Quijano, T. Louail, G. Hutzler, From biological to urban cells: lessons from three multilevel agent-based models, in: Principles and Practice of Multi-Agent Systems, Springer, 2012, pp. 620–635.
[5] N. Gilbert, K. Troitzsch, Simulation for the Social Scientist, McGraw-Hill International, 2005.
[6] D.-A. Vo, A. Drogoul, J.-D. Zucker, An operational meta-model for handling multiple scales in agent-based simulations, in: 2012 IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future, RIVF, 2012, pp. 1–6.
[7] T. Huraux, N. Sabouret, Y. Haradji, A multi-level model for multi-agent based simulation, in: Proc. of the 6th International Conference on Agents and Artificial Intelligence, ICAART, Angers, France, 2014.
[8] L. Jalali, S. Mehrotra, N. Venkatasubramanian, Multisimulations: towards next generation integrated simulation environments, in: Formal Modeling: Actors, Open Systems, Biological Systems, Springer, 2011, pp. 352–367.
[9] B. Camus, C. Bourjot, V. Chevrier, Multi-level modeling as a society of interacting models, in: Proceedings of the Agent-Directed Simulation Symposium, 2013, p. 3.
[10] R. Suleiman, et al. Social Science Microsimulation. Tools for Modeling, Parameter Optimization, and Sensitivity Analysis, 1998.
[11] T. Korhonen, et al. Integration of an agent based evacuation simulation and the state-of-the-art fire simulation, in: Proceedings of the 7th Asia-Oceania Symposium on Fire Science & Technology, 2007, pp. 20–22.
[12] A. Filippoupolitis, et al. Emergency response simulation using wireless sensor networks, in: Proceedings of the 1st International Conference on Ambient Media and Systems, 2008, p. 21.
[13] S. Jain, C.R. McLean, An Integrated Gaming and Simulation Architecture for Incident Management Training. National institute of standards and technology. Technology administrations, US Department of commerce, 2006.
[14] W.K.V. Chan, Y.J. Son, C.M. Macal, Agent-based simulation tutorial - Simulation of emergent behavior and differences between agent-based simulation and discrete-event simulation, in: Proceedings - Winter Simulation Conference, 2010, pp. 135–150.

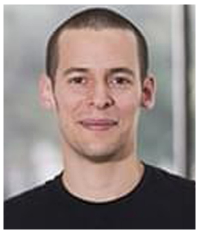
[15] A. Marzinotto, et al. Towards a Unified Behavior Trees Framework for Robot Control., 2014, pp. 5420–5427.
[16] A.J. Champandard, Understanding Behavior Trees, 2007.
[17] S. Patil, J. Van Den Berg, S. Curtis, Directing crowd simulations using navigation fields, IEEE Trans. Vis. Comput. Graphics (2011).
[18] D. Voloshin, D. Rybokonenko, V. Karbovskii, Towards a performance-realism compromise in the development of the pedestrian navigation model, Procedia Comput. Sci. 51 (2015) 2799–2803.
[19] V. Viswanathan, et al. Quantitative comparison between crowd models for evacuation planning and evaluation. 2014, arXiv preprint arXiv:1401.0366.
[20] J. Van den Berg, M. Lin, Reciprocal velocity obstacles for real-time multi-agent navigation, Robot. Autom. (2008).
[21] D. Helbing, P. Molnar, Social force model for pedestrian dynamics, Phys. Rev. E 51 (5) (1995) 4282.
[22] A. Kiselev, V. Karbovskii, S. Kovalchuk, Agent-based modelling using ensemble approach with spatial and temporal composition, Procedia Comput. Sci. (2016).
[23] G. Fiedler, UDP and TCP. Available at: http://gafferongames.com/networking-for-game-programmers/udp-vs-tcp/, 2013.
[24] H. Sawashima, et al. Characteristics of UDP packet loss: Effect of tcp traffic. of INET'97: The Seventh Annual …, 1997.
[25] R.S. Lee, R.L. Hughes, Exploring trampling and crushing in a crowd, J. Transp. Eng. 131 (8) (2005) 575–582.
[26] A. Seyfried, A. Schadschneider, Fundamental diagram and validation of crowd models, in: International Conference on Cellular, 2008.
[27] A. Johansson, et al., From crowd dynamics to crowd safety: a video-based analysis, Adv. Complex Syst. 11 (4) (2008) 497–527.
[28] R. Lee, R. Hughes, Prediction of human crowd pressures, Accid. Anal. Prev. (2006).
[29] E.J. Evans, F. Hayden, Report on tests of static loads on live subjects to determine tolerable forces that can be exerted by crowd control crush barriers, Guildford, UK, 1989.
[30] I.H.G. Hopkins, et al., Crowd pressure monitoring, in: R.A. Smith, J.F. Dickie (Eds.), Engineering for Crowd Safety, Elsevier, Amsterdam, 1993, pp. 389–398.
[31] V.A. Karbovskii, et al., Personal decision support mobile service for extreme situations, Procedia Comput. Sci. 29 (2014) 1646–1655.
[32] D. Voloshin, D. Rybokonenko, V. Karbovskii, Optimization-based calibration for micro-level agent-based simulation of pedestrian behavior in public spaces, Procedia Comput. Sci. 66 (2015) 372–381.
[33] V.V. Krzhizhanovskaya, et al., Distributed simulation of city inundation by coupled surface and subsurface porous flow for urban flood decision support system, Procedia Comput. Sci. 18 (2013) 1046–1056.

Vladislav Karbovskii is a Senior Researcher at the eScience Research Institute (Computational Social Science team) and an Assistant Professor at the High Performance Department at ITMO University, St. Petersburg, Russia. He completed his engineering degree in 2011 and Ph.D. degree in Computer Science in 2015 both at ITMO University. Prior to joining ITMO University, he worked at the industry and developed enterprise high-performance distributed systems. His current research interests include artificial intelligence, agent-based modeling, machine learning and data science, distributed and high-performance computing, software engineering and computational social science.

Daniil Voloshin is a Ph.D. degree candidate at the ITMO University and a research assistant at the international laboratory "Urban Informatics", Saint-Petersburg, Russia. He currently holds B.A. and M.A. degrees in Sociology from Saint-Petersburg State University, Russia. In 2015 he received a grant aimed at supporting young scientists from the Russian Foundation For Basic Research. For the last 4 years he has been involved into work in international collaborative research projects, organized by the ITMO University, Russia and the University of Amsterdam, the Netherlands. His research interests include computational social science, urban studies, collective behavior, crowd studies and agent-based modeling.

Andrey Karsakov is a Ph.D. student at the ITMO University and a research assistant at eScience Research Institute of ITMO University. He currently holds Engineering Diploma in Industrial Electronics from Togliatti State University, Russia. He also has a more than 6 years work experience in business in the fields of design, computer graphics and visualization. His research interests focused on visualization, virtual reality, human–computer interaction and UI/UX design.

Alexey Bezgodov is a Ph.D. Engineering sciences program at ITMO University, Russia. He has also worked in video games industry for two years, mostly involved with real-time rendering. His current research interests include rendering technologies, graphics resource virtualization, networking and artificial intelligence.


Carlos Gershenson is a tenured research professor at the Universidad Nacional Autónoma de México and a leader of the Self-organizing Systems Lab. He is editor-in-Chief of Complexity Digest. He got a B.Eng. in Computer Engineering at the Fundación Arturo Rosenblueth in Mexico City. Later, he obtained his M.Sc. in Evolutionary and Adaptive Systems at the University of Sussex, UK, and a Ph.D. at the Free University of Brussels, Belgium. He was a postdoc at the New England Complex Systems Institute, USA. His current research interests include self-organizing systems, complexity, artificial life, information, evolution, cognition, artificial societies, and philosophy.