Elevator Simulation for Testing Advertisement Scheduling Systems

Daniel Wilfing, Oliver Krauss and Andreas Schuler
University of Applied Sciences Upper Austria
Softwarepark 11, 4232 Hagenberg, Austria
Homepage: http://ehealth.projekte.fh-hagenberg.at/

***Abstract*—An agent-based elevator simulation was implemented to test the validity of an advertisement scheduling system. The elevator simulation imitates the elevators and their advertisement system, thus testing the different schedules and sending the results back to the scheduler. To validate the results, certain data sets of the simulation application are compared with expected values. First results have shown that the created data is in a valid range around the estimations. Finally, methods to further improve the simulation are presented. It was found that agent-based simulations are a good method to test systems, which are too complex or expensive to test in the real environment.**

***Keywords*—agent based simulation, elevator simulation, test case, automatic scheduler**

I. Introduction

Setting schedules for advertisements is a difficult task, as the required work grows exponentially with the problem size [1]. Some advertisers favour continuous advertising, others prefer flighting or pulsing [2]. Certain advertisements should not be displayed after each other, like ads of business rivals. Configuring the schedule manually is doable if only a few schedules have to be created, but becomes more complex and time-consuming when more schedules should be uniquely configured for different display units.

Automatic scheduling systems can take care of this problem, by calculating different schedules given certain constraints [3]. They are several times faster than a manual approach and are capable to incorporate results of previous schedules. They constantly improve their schedule calculation thanks to this. However, automatic systems are still prone to mistakes, like ignoring certain aspects that have not been considered with constraints (e.g. displaying advertisements of business rivals one after the other). Testing the different schedules would be beneficial, but using the real environment is usually expensive or outright impossible, if the necessary devices are not available. Testing distributed a system, or systems which are already in use, proves difficult as well. Using a simulation approach might solve this problem, as it can simulate all kinds of scenarios with little set-up time. Simulations are also able to simulate different behaviours of a model much faster, compared to a real environment [4].

In this work, an application is introduced, that simulates an elevator system. The simulation was tested using the automatic advertisement scheduler. The advertisements defined by the scheduler, are displayed during each elevator ride. The simulation system collects all runs and additional information regarding the run (e.g. start- and end-time of the run, displayed advertisements, etc.). This information is sent back to the scheduler, so it can make adjustments regarding the next schedules. This way, testing of the advertisement system is done without using the real environment. The results of the simulation are evaluated and the validity of this test method presented.

II. Methods

Simulations allow the reenactment of different situations and are usually used for scenarios, which are too complex or time-consuming to perform in the real environment. Core of each simulation is a model, which is a depiction of an event or a system. The simulation then defines the different actions and behaviours for the model. [5]

The two main areas of simulations are continuous and discrete models and simulations. Continuous simulations imitate a continuously changing system and often require real numbers for calculation. Additionally, infinitely many changes can occur in each time interval [6]. For discrete simulations, changes occur at certain instants in the simulations. Changes between those time steps are not calculated and dismissed [7]. The simulation presented in this work implements a discrete approach, as each event happens at certain instants.

The model used in this work describes a single building with multiple floors and persons, as well as an elevator system. During the simulation, the persons move from one floor to another using the elevators. The elevators then transport the persons to their requested floor and start displaying the advertisements. Information regarding this elevator is collected and sent back to the scheduler, like the start- and end-time of each elevator ride and which advertisements were displayed during said rides. This information is then compared with a set of estimations, to verify how genuine the behaviour of the simulation is.

The following sections describe the structure of the model, as well as how the different person and elevator behaviours have been implemented.

A. Structure of the model

The simulation consists of different agents with distinct behaviours, which are interacting with each other. This type of simulation is called agent-based simulation [8]. One of these

agents is represents a building, that contains an arbitrary number of persons and elevators. The building consists of multiple floors, that have a specific type. Possible floor types are *lobby*, *apartment*, *shop*, and *office*. Persons in the simulation require these types, so they can decide which floor they want to visit next. A graphical visualisation of a model can be seen in figure 1.

Different agents are assigned to simulate the behaviour of the elevators. These agents are heavily influenced by the person-agents in the building, which use them to move from one floor to another. Each elevator-agent in the building has attributes that define its speed and capacity, as well as a list of advertisements that the elevator displays to passengers. The advertisements are given by the automatic scheduler and get displayed in a round-robin manner. Round-robin is a form to sequentially process items in a list, starting back from the beginning when the entire list got processed [9].

Person-agents keep the simulation running by autonomously interacting with elevator-agents. These agents are positioned on different floors inside of the building, depending on the simulation setup. During the simulation, they move to different floors using the elevator-agents and generate the necessary elevator rides. A person-agent decides to which floor he wants to move next, but the elevator-agent ultimately decides in which order the different floors will be visited.

The behaviours of each agent have been modeled after state machines, which define the different actions and their order, in which they happen during simulation. This is similar to the approach presented by [10]. The state machines of the different agents influence each other and certain states can only be reached through actions of other agents. The state machines allow a more sophisticated behaviour for the different agents, thus enhancing the agent-based simulation [11].

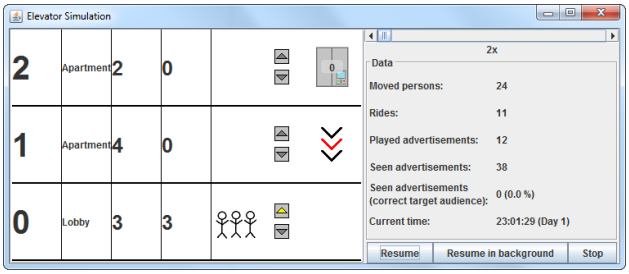

Fig. 1. A graphical representation of the model shows the events during simulation.

B. Person behaviour

The person-agents are constantly interacting with the other agents in the simulation and move from one floor to to another. To make this possible, they need a certain behaviour. With a random behaviour, the simulation would already be functional and generate the required elevator runs. However, this makes it impossible to simulate specific scenarios, like the *In-* and *Out-rush*. These rushs are behaviours that have been observed in real elevator systems and occur at certain times. *In-Rush* happens when many people want to move from to the ground floor to one of the upper floors (e.g. people go to work). *Out-Rush* occurs when people go from the upper floors to the ground-floor (e.g. people go for lunch) [12]. To simulate these scenarios, it is necessary to define at which time a person should go to a specific floor (e.g. at 12:00, person wants to visit the cafeteria). A person should move several times in the simulation, so it requires more than one of those time-based behaviours.

A person-agent in the simulation has to interact correctly with the other objects in the simulation. After it decides to change floor, it should press the correct floor button and wait for the elevator. The person should enter the elevator once it reaches its floor, press the correct button inside the elevator and exit it once the the elevator arrives at the correct destination. All of those actions are implemented as states in the simulation (see figure 2). The state *wait for cabin* and *wait in cabin* are dependant on the states of the elevator-agents. The person watches the advertisements during the state *wait in cabin*. Its also possible that the elevator breaks down. In this case, the person leaves the elevator and tries to enter a different one. Other states (e.g. person left the building) are not relevant for this simulation.

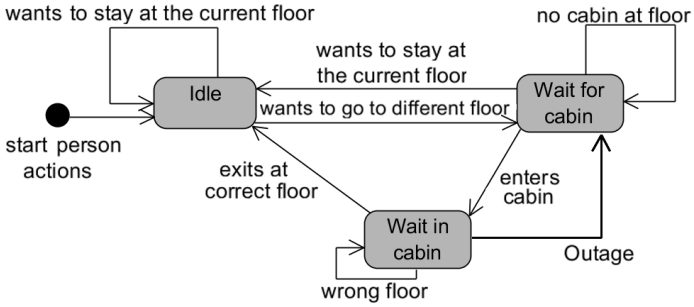

Fig. 2. Each person has one of three possible states, which describe if the person wants to remain on the current floor or not.

C. Elevator behaviour

The elevator-agents transport the person-agents to the different floors. To achieve this, they need a specific behaviour defining how they move frome one floor to another and which actions they have to take. They do not act by themselves, but are controlled by the different floor- and elevator-buttons.

The easiest way would be to randomly move the elevators through the building, but this does not simulate real elevator systems. State-of-the-art elevators move in one direction and visit necessary floors, until there are no more required floors in the given direction. They are then turning around and move in the other direction [13]. Each elevator in the simulation adheres to this behaviour. Additionally, the elevators have to coordinate each other, or every elevator would go to the same floor after a floor button was pressed. The elevator system requires a control unit that defines which elevator should go to which floor to pick passengers up [14].

Besides this, the elevators in the simulation also have to spend time opening and closing their doors, as well as take time to move to a different floor. The elevator can also

send distress calls or break down, depending on the scenario. Similar to the persons-agents, these actions are implemented as states (see figure 3). The state machine of elevators is mainly influenced by the actions of the person-agents, because these agents make requests to move to different floors. They influence every state of elevator-agents, with the exception of the state *outage*. The state *move* is additionally affected by the other elevator-agents in the simulation, as certain floors can be skipped if different elevators are moving to the requested floors already. Each elevator has a unique schedule of advertisements. The commercials will not get displayed while the elevator is waiting. They start once the elevator begins moving to a different floor.

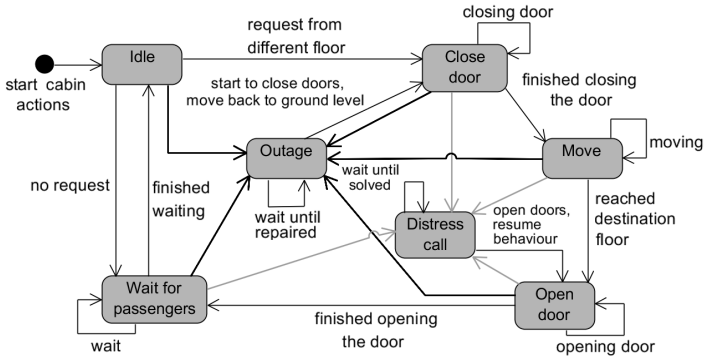

Fig. 3. Elevators are idling until they get a request from a floor. If they have to move to a different floor, they cycle through different states until they have reached their destination.

D. Communication with the scheduler

Each elevator in the simulation requires a schedule. These schedules consist of an arbitrary number of advertisements, that are given in a specific order. Once a schedule has been uploaded to an elevator, it will keep displaying the advertisements, until it is replaced by a new schedule.

The automatic scheduler generates the different schedules on a daily basis. The different elevators take the newest schedule via restful web services and update its advertisements [15]. If the automatic scheduler is not able to generate the schedules in time, the elevators keep displaying the old advertisements. This is to prevent the system from not displaying any advertisements.

Each elevator collects information regarding the advertisement-playbacks. After a certain amount of elevator rides have occurred or a day has passed, the information is sent to the automatic scheduler.

The interval when schedules are created and updated can be shortened significantly, if the system is tested with the simulator. The simulation is able to generate the elevators rides much faster, and therefore tests the different schedules quicker than the real environment. For example, the scheduler creates and sends the schedules at midnight, when using the real system or a slowed down simulation. If the simulation speed is increased by a factor of 4, then the scheduler has to calculate schedules each 6 hours, instead of every 24 hours.

E. Test scenario

Two different test cases were implemented to test the automatic scheduler together with the elevator simulation. A model with 100 different elevators has been simulated. The automatic scheduler calculated the necessary schedules for these elevators using 20 advertisements with different constraints (e.g. specific advertisements should only scheduled on certain weekdays). This scenario has been simulated for one week. Using the time lapse functionality of the simulation, this scenario has been simulated with two different time settings.

In the first test case, the simulation slowed down to require an entire week for the simulation, thus running in real-time [16]. Thanks to this, the behaviour of the automatic scheduler can be inspected in a setting that is as close to the real environment as possible.

The simulation runs without delay in the second test case. The interval, in which the automatic scheduler calculates the advertisement schedules, had to be shortened significantly, otherwise the scheduler would not be able to generate the schedules on time. If the scheduler is too slow, the simulation reuses the old advertisement schedules.

The resulting data of both runs are compared to see if there are any significant differences. Estimations for the different data sets are stated and matched with the results of the test runs, to identify any problems regarding the validity of the simulation. These data sets include the start- and end-time of the generated elevator rides, to check if their data is reasonable and if the elevator-agents where able to correctly simulate their given behaviour. Data regarding the displayed advertisements is also collected, to verify if the advertisements have been presented in the order given by the schedule.

III. Results

The results of the simulations have been collected in an object-relational database. Several SQL queries have been executed to make comparison with different estimators possible, and to verify how rational the data sets are.

Required time for elevator ride

A certain amount of time is necessary until a single elevator ride has occurred. This required time also needs to be simulated correctly, or the time when advertisements are displayed during elevator rides would not be accurate. If this does not get simulated properly, the simulator would create rides that can not happen in the real environment, like a ride that only lasted for one second, but displayed three advertisements lasting 10 seconds each during this time.

Different building types have been simulated, containing a different number of floors, elevators and persons. Possible building types are for example shopping centres, office buildings and apartment complexes. Depending on the settings of the elevator-agents, the time that it takes for one elevator to move from one floor to another should be between 10 and 30 seconds. If a ride takes less time, the elevator is too fast and the simulation becomes unrealistic. The minimum ride times are thus suitable values to consider when reviewing the data

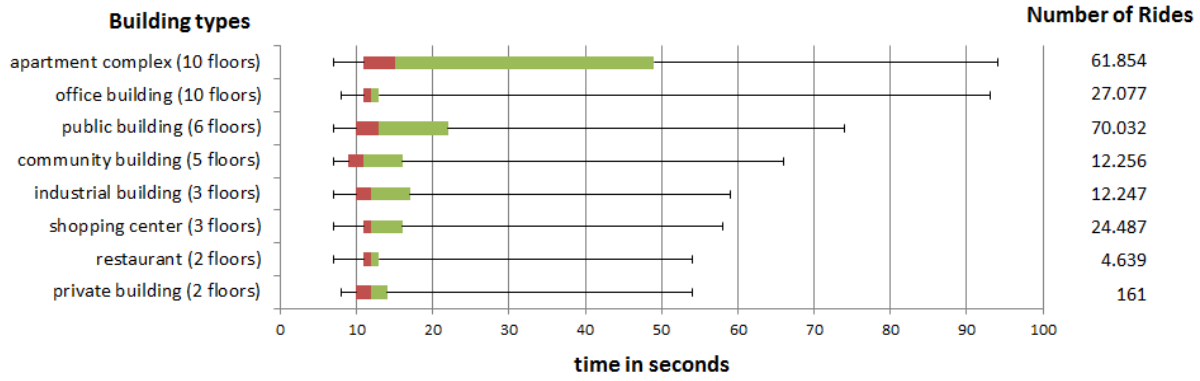

Fig. 4. The necessary times of an elevator movement heavily depend on the number of floors inside the building.

set. The maximum ride times can be considered as well, but they mainly depend on the number of floors in the building. If the building contains more floors, then the elevator rides that can occur take accordingly more time.

The fastest ride over all generated rides was 7 seconds long. This is 3 seconds below the estimation. This ride probably occurred because a person stepped into a very fast elevator and only moved a single floor up or down. All the necessary actions defined by the state machine of the elevator-agent can be simulated during these 7 seconds, but this result is not very realistic. This problem can be solved by decreasing the speed of the elevator-agents.

To evaluate the maximum elevator ride times in respect to how many floors the building has, a box-plot diagram was designed (see figure 4). According to the data, the maximum time of an elevator ride increases with the number of floors in the building. This matches with the previous assumption. Elevator-agents seem to correctly consider the number of floors in the building during the simulation. The maximum ride times seem to be very high, but can be explained with distress calls, which delay the elevator ride significantly.

A. Schedule usage

After receiving the schedules, the simulation application has to properly distribute the different schedules to the elevators. The advertisements have to be displayed in exactly the same order, that was defined in the schedule.

The automatic scheduler prefers simple advertisements with little requirements at the beginning. Over several days, the scheduler generates more specialised schedules for each elevator, using the data sent back from the elevator simulation (e.g. number of displayed advertisements). For the test cases, three simple advertisements are being scheduled together with 17 constrained ads. Constraints define certain criteria that need to be considerer for scheduling. The branch of an advertisement is a possible constraint, or that that the advertisement has to be displayed at a specific time. The only constraint that is considered for each advertisement is, that the same advertisement should not be repeated several times in a row. The first schedule should only consist of three projects, named A, B and C. The scheduler should prefer these three projects, as they don't have any unique constraints defined. A first fitting schedule might be in the order A-B-C-A-B-C-A-B-C-B. This order would be suitable, as no advertisement is repeated twice in a row. The second schedule should already use the commercials with constraints during schedule generation.

After testing this behaviour, a small variation of the estimated schedule was observed for the first schedule (see figure 5). The estimated and generated schedule are similar, but the calculated schedule repeats the advertisement C two times back-to-back. The simulation correctly displays and repeats the given schedule.

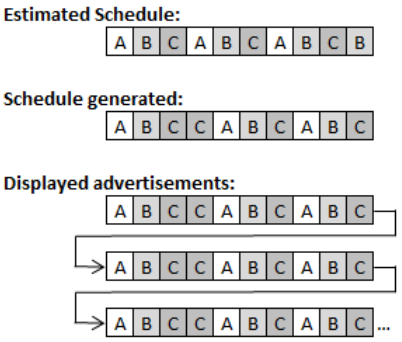

Fig. 5. The elevator simulation correctly incorporates the generated schedule during the simulation.

The last generated schedule for the simulation should differ for the most part compared to the first schedule. It should mostly consist of advertisements with constraints. Making an accurate estimation for this schedule is difficult, because many factors of the simulation are taken into account during the schedule generation, some of which are stochastic.

Figure 6 shows the result of the last run. As required, only constrained advertisements have been scheduled. The elevator in the simulation was also able to display and repeat the schedule correctly. However, there is still room for improvement. The schedule contains three D-advertisements in a row.

It would've been impossible to prevent a repetition with the given number of advertisements, but a schedule like D-D-E-D-D-E-D-F-D-G would've prevented a triple repetition, at the cost of two double repetitions.

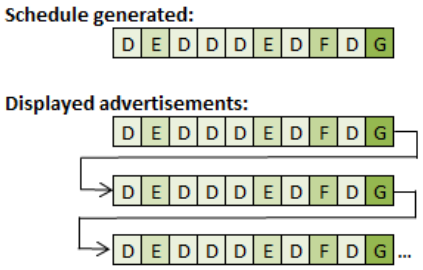

Fig. 6. After collecting data from the simulator, the scheduler starts using the constrained advertisements for scheduling.

All of these behaviours have been successfully observed in both test cases. It didn't matter if the simulation was slowed down to real-time or was simulating as fast as possible. The only difference was, that the interval of the automatic scheduler had to be adjusted, so that the scheduler would calculate the schedules fast enough for the simulation application.

IV. Conclusion

Using an application to simulate certain scenarios, that are otherwise too difficult or expensive to set-up, is a good way to generate test data for a scheduling algorithm and software-systems in general. Data can be generated much faster than it would ever be possible in the real environment. The testing of the scheduling algorithm can be improved, as it allows the creation and calculation of much bigger problems than usual. Simulations also enable to run the simulation at different speeds, allowing to test a certain scenario over any given duration. Because of this its possible to run the scheduler in an environment, that is as close to the real environment as possible. Thanks to this, problems that might arise when a certain problem size is reached can be detected and fixed sooner.

Simulations seem like a fitting way to test without using a real system. By imitating the basic behaviours of the real environment, simulations make it possible to test an application without using the real system. Certain aspects can be monitored and adjusted during the simulation, that can not be inspected as easily in the real system [17]. Complete validity of the simulation has yet to be confirmed by comparing data of the real environment with data of a simulation using a model of the real environment [18].

V. Outlook

As a next step, the results of the simulator are intended to be compared with the data created in the real environment. To do this, a model that is very similar to the buildings, that are currently showing advertisements using manually built schedules, will be simulated. Then the data of the simulation will be compared with the real data. Thanks to this, the differences regarding the behaviour of the simulation and the real system can be detected. Furthermore, the validity of the simulation can be assessed and it can be verified if a simulation application is a suitable substitute for testing a system.

The basic framework of the simulation application should be improved, so that other kinds of models, and not only elevator systems, can be simulated as well. Additional person behaviours might be implemented, so that more types of persons can be simulated, like children or elderly persons [19]. The current system is loosely modelled after a state machine approach, but the state machine template has not been implemented. Further research will be done to asses how effective simulations based on finite state machines are [20].

References

[1] A. S. Jain and S. Meeran, *Deterministic job-shop scheduling: Past, present and future* European Journal of Operational Research, 113, 390-343, 1999.

[2] T. A. Shimp and J. C. Andrews, *Advertising Promotion, and Other Aspects of Integrated Marketing Communications* ISBN-13: 978-0324593600, 2013

[3] M. Ghallab, D. Nau and P. Traverso, *Automated Planning and Acting* `http://projects.laas.fr/planning/` last visited: Apr. 25, 2016

[4] J. A. Sokolowski and C. M. Banks, *Principles of Modeling and Simulation: A Multidisciplinary Approach* ISBN-13: 978-0470289433, 2009

[5] H. Stachowiak, *Allgemeine Modelltheorie* ISBN-13: 978-3211811061, 1973

[6] F. E. Cellier and J. Greifeneder, *Continuous System Modeling* ISBN-13: 978-0387975023, 1991

[7] N. Matloff, *Introduction to Discrete-Event Simulation and the SimPy Language* University of California, Davis, 2008

[8] S. Moss and P. Davidsson, *Multi-Agent Based Simulation* ISBN-13: 978-3-540-44561-6, 2001

[9] A. Silberschatz, P. B. Galvin and G. Gagne, *Operating System Concepts* ISBN-13: 978-0470233993, 2009

[10] A. Shalyto, N. Shamgunov and G. Korneev, *State Machine Design Pattern* Lane Department of Computer Science and Electrical Engineering, West Virginia University, 2008

[11] I. Sakellariou, *Agent Based Modelling and Simulation using State Machines* Department of Applied Informatics, University of Macedonia, 2015

[12] G. F. Newell, *Strategies for Serving Peak Elevator Traffic* Institute of Transportation Studies, University of California, 1998

[13] G. R. Strakosch and R. S. Caporale, *The Vertical Transportation Handbook* ISBN-13: 978-0470404133, 2010

[14] T. K. Khiang, M. Khalid and R. Yusof, *Intelligent Elevator Control by Ordinal Structure Fuzzy Logic Algorithm* Universiti Teknologi Malaysia, 1999

[15] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures* University of California, Irvine, 2000

[16] J. Blanger, P. Venne and J.-N. Paquin, *The What, Where and Why of Real-Time Simulation* Opal-RT Technologies, 2010

[17] M. McGarry, B. Bickell and G. Pelkey, *A Case Study of the Use of Actual Controls in Simulation Trainers* Andritz Automation, 2012

[18] M. S. Maris, *Validation of Simulation Based Models: A Theoretical Outlook* Manipal Institute of Technology, India, 2006

[19] S. Tuomas, J. Sorsa and M.-L. Siikonen *Passenger Behaviour in Elevator Simulation* in Elevator Technology 14, 2004

[20] I. Chiuchisan, A. D. Potorac and A. Graur, *Finite State Machine Design and VHDL Coding Techniques* Universitatea "Stefan cel Mare" din Suceava, Romania, 2010