Impact of Dwell Time on Vertical Transportation Through Discrete Simulation in SIMIO

Marcelo Henriques, António A.C. Vieira, Luís M.S. Dias(✉), Guilherme A.B. Pereira, and José A. Oliveira

University of Minho, Campus Gualtar, 4710-057 Braga, Portugal
marcelo@nhenriques.com,
{antonio.vieira,lsd,gui,zan}@dps.uminho.pt

Abstract. This work has the objective of simulating an elevator system, using SIMIO software. Firstly, two different approaches, and its implementation, will be explained and compared: Vehicle vs. Entity. After selecting the Entity-approach, due to its more flexible processes and the limitations of the Vehicle-approach, it will be used to conduct the simulation experiments. The purpose is to evaluate the impact of dwell time - time in which the elevator remains stopped, allowing for clients to enter and exit - in the performance of the system. That will be achieved analysing the impact on the total time - spent by clients from placing a call until reaching its destination - number of clients inside the system and waiting for the elevator, waiting time, elevator occupation and number of elevator movements. The analysis of the results indicates that, for the properties defined, the best time for the elevator to stay with its doors open is around 10 s.

AQ1

Keywords: Elevator · Lift · Management systems · Intelligent objects · Modelling · SIMIO · 3D simulation · Case study

1 Introduction

The most typical objective of an elevator system is to move people and cargo in a vertical way. In the elevator industry, changing the entire elevator system - or simply the algorithm - has high costs associated and can imply system inoperability for some time. The heart of any elevator system is its elevator management system, which decides what will be the next elevator movement through its algorithm, based on various inputs. A simple algorithm for only one elevator, as in the studied case, can be described as follows (Setchi 2010):

- Move in a certain direction, up or down, stopping at all floors where there are calls or destinations;
- Change its direction when there are no calls or destinations at floors beyond the current floor in the current direction, or when it reaches the last floor, changing from going down to going up when it reaches the bottom floor, or changing from going up to going down when it reaches the upper floor;
- Stop, in case there are no calls or destinations in the system.

A. Paias et al. (Eds.): ICCL 2016, LNCS 9855, pp. 1–16, 2016.
DOI: 10.1007/978-3-319-44896-1_27

One of the main advantages of simulating a system is the possibility to change it in a virtual way and measure the consequences, before physically changing it, with all the investments involved. These measurements allow management to take decisions based on data. Decision making based on simulation data will help management deciding what system to implement, or elevator companies deciding what parameters should be defined. One of these parameters is the dwell time, which is the time that the elevator remains stopped, with its doors open to allow clients to enter or exit it.

This papers describes a simulation model of an elevator system, developed in the discrete event simulation tool SIMIO; Pegden (2007). The basis of this work was a hospital located on the north of Portugal. Two different approaches were addressed and compared. Afterwards, one of the approaches was used to conduct simulation experiments and analyse the obtained data.

The remainder of this paper is organized as follows: First, in Sect. 2, a literature review will be presented, addressing the selected simulation tool for this problem. Afterwards, in Sect. 3, two different approaches for modelling the system in question will be presented. The analysis of the obtained results on one of the approaches will be discussed and lastly, in Sect. 5 the main conclusions will be withdrawn, along with some future work.

2 Literature Review

Most recent models of elevator group management systems (e.g. Destination Dispatch) had, in their genesis, tests and data retrieved from using computing simulation. One simulation tool that outstands in the elevator industry is the software Elevate® (Barney and Al-Sharif 2015), which allows to simulate and analyse elevator traffic, with support for different configurations and applications, e.g. two floor elevators, an elevator system with different speeds and different attending floors ("About Elevate" 2016). This software runs on Windows™ and was developed by the London-based company Peters Research. Another innovation by this company is the software Elevate Live™, which allows checking the status of the elevator management system in real time ("About Elevate Live" 2016).

This software is not the only simulation tool used in the elevator industry, but it is one of the most referred and promoted. But, taking into account the will to share information, the intellectual property protection and the maintenance of market advantage, companies of this industry tend to not reveal which tools are used.

But the need and use of simulation in this field is real (Barney and Al-Sharif 2015; Hakonen and Siikonen 2009; Zhang and Zong 2014), because elevator models can reach high levels of complexity. Taking, for instance, the Shanghai Tower, where hundreds of elevators travel vertically, with certain restrictions and different purposes, the level of complexity associated to this system becomes obvious.

The number of simulation tools is very large. Thus, its comparison is important. However, most scientific works related to this subject "analyse only a small set of tools and usually evaluating several parameters separately avoiding to make a final judgement due to the subjective nature of such task" (Dias et al. 2007).

Hlupic and Paul (1999) compared a set of simulation tools, distinguishing between users of software for educational purpose and users in industry. In his turn, Hlupic (2000) developed "a survey of academic and industrial users on the use of simulation software, which was carried out in order to discover how the users are satisfied with the simulation software they use and how this software could be further improved". Dias et al. (2007) and Pereira et al. (2011) comparing a set of tools based on popularity on the internet, scientific publications, WSC (Winter Simulation Conference), social networks and other sources, claim: "Popularity should never be used alone otherwise new tools, better than existing ones would never get market place, and this is a generic risk, not a simulation particularity" (Dias et al. 2007); however, a positive correlation may exist between popularity and quality, since the best tools have a greater chance of being more popular. According to the authors, the most popular tool is ARENA, Kelton et al. (2009), and the good classification of SIMIO is noteworthy. Based on these results, Vieira et al. (2014) compared both tools taking into consideration several factors. This latter paper is also a good source of information for researcher and practitioners, since it compares SIMIO with the most popular tool (ARENA), giving some basic examples.

SIMIO has two main levels for modelling. The simpler one, called 'Facility', is suitable for practitioners without computer science background, where one can create models in a building-block approach over a physical layout, providing a realistic 3D animation. The second level, called 'Process', enables the creation of detailed behaviour using logical flow charts to specify virtually anything.

Processes, once created, can be used anywhere in the 'Facility' level. Moreover, processes can be "attached" to Entities (objects) to enabling them to react actively and autonomously. This behaviour pushes SIMIO "living" objects to agents. It is controversial to consider SIMIO objects as intelligent, once such term has a connotation to support logical programming and self-learning ability.

Another relevant capability is the support for object class hierarchy, allowing the extension of existing objects rather than creating from scratch.

SIMIO was the chosen tool for this project. It is based on intelligent objects (Sturrock and Pegden 2010; Pegden 2007; Pegden and Sturrock 2011). These "are built by modellers and then may be used in multiple modelling projects. Objects can be stored in libraries and easily shared" (Pegden 2013). Unlike other object-oriented systems, in SIMIO there is no need to write any programming code, since the process of creating a new object is completely graphic (Pegden and Sturrock 2011; Pegden 2007; Sturrock and Pegden 2010). The activity of building an object in SIMIO is identical to the activity of building a model. In fact, there is no difference between an object and a model (Pegden 2007; Pegden and Sturrock 2011). A vehicle, a customer or any other agent of a system are examples of possible objects and, combining several of these, one can represent the components of the system in analysis. Thus, a SIMIO model looks like the real system (Pegden and Sturrock 2011; Pegden 2007). This can be very useful, particularly while presenting the results to someone unfamiliar to simulation.

In SIMIO, the model logic and animation are built in a single step (Pegden and Sturrock 2011; Pegden 2007). This makes the modulation process very intuitive (Pegden and Sturrock 2011). Moreover, the animation can also be useful to reflect the

changing state of the object (Pegden 2007). In addition to the usual 2D animation, SIMIO also supports 3D animation as a natural part of the modelling process (Sturrock and Pegden 2010). To switch between them the user only needs to press a specific key (Sturrock and Pegden 2010). Moreover, SIMIO provides a direct link to Google Warehouse (Pegden and Sturrock 2011).

SIMIO offers two basic modes for executing models: interactive and experimental. In the first it is possible to watch the animated model, which is useful for building and validating the model. In the second, it is possible to define properties of the model that can be changed (Sturrock and Pegden 2010).

3 Comparison of Different Implementation Approaches

To elucidate the need of building the two approaches and its comparison, it is beneficial to explain them individually, their common points and the disadvantages and advantages of each approach. This matter will be addressed in the present chapter.

The use of a SIMIO standard object to transport entities has the advantage of being already developed, and thus the user only needs to place it and edit some properties. However, to model more complex situations can become a hard task. The use of an entity to overcome this situation implies a higher initial effort. Nonetheless, once this has been surpassed, the user will be able to model its own transportation logic. The choice that has to be made, between using the standard Vehicle object, or an Entity, is not an obvious one. In this sense, this chapter will analyse both alternatives.

The model facility has some equal parts between the two approaches implemented, as their objective is common: simulate an elevator. Both facilities are composed by seven floors, named from floor 1 to floor 7. Each floor has its own source, creating entity clients through a random exponential expression. The ground floor has a separate property than the upper floors, allowing different scenarios simulation, like: up-peak, down-peak and mixed movements, common on elevator passenger traffic (Barney and Al-Sharif 2015). Each source is linked to a series of central nodes by a path, guiding clients to the place where they will wait for the elevator to pick them up. Once a client is sent out of the elevator to its destination floor, it will be transferred to a node linked by a path to a final sink, where each client is destroyed.

3.1 Vehicle Approach

The Vehicle approach consists of using the Vehicle object from the standard SIMIO library. Among other properties, the user can specify load capacity, unload time and task selection methods; however, trying to shift the Vehicle from its standard behaviour can be a complex task. This approach has two sides, one based on processes and another based on parameters. Both are used to model client and elevator behaviour.

To start, client behaviour has to be modelled. In this sense, a destination node, representing his or her destination floor, is randomly assigned to each client. This destination node is chosen from a list, which is individual to each node. These options are selected inside the source parameters.

Afterwards, all clients will travel through a Path object that will take them to a TransferNode, where they will wait for the elevator. It should be stressed that the 'Allow Passing' property of these Paths, must be set to false, in order to ensure that no overtaking occurs and that a queue of waiting clients is formed. Moreover, the 'Ride On Transport' property of each Transfer Node must be set to true to obligate clients to wait for the Vehicle and seize it. In this approach, the node where the clients wait for the elevator is the same where the elevator travels, to ensure that the Vehicle can pick the waiting clients.

The major benefit of this approach stems from the automatic transfer of clients from the waiting node to the elevator, since the native Vehicle object was designed specifically for this purpose: transporting people or goods. The transfer steps, responsible for transferring the clients onto the Vehicle, are defined within the Vehicle model, being hidden from the common user. A downside of this aspect is the inability to see and change the processes that allow such automated actions to a more suitable one, according to the needs of the specific system intended to implement.

Contrary to the nodes for clients, which have only one direction: pointing towards the Vehicle, the destination nodes could not be modelled as nodes mutually travelled by clients and the elevator itself, otherwise, the elevator would leave its natural vertical path and enter the final path to the sinks. If the nodes were separated by networks other difficulties would arise, namely transferring the clients out of the elevator, when the automated transfers would not occur, thus needing the same process step and a separate node to transfer the client to as implemented. To overcome this, the central nodes - where clients wait for the elevator - and the out nodes - where clients reach their final path to destination - are physically separated. To make the final transfer from the central nodes to the out nodes, a process was created to each central node, being triggered whenever a client enters that specific central node. Such process is illustrated below in Fig. 1.

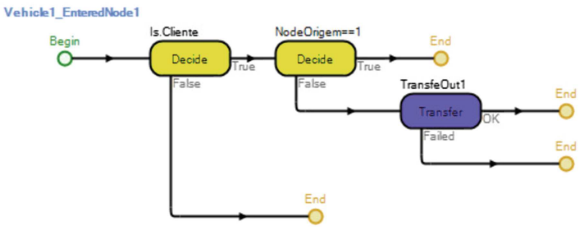

Fig. 1. Process to transfer clients out of the elevator

The first Decide step protects the process from being executed by the Vehicle, as the Vehicle also crosses the central nodes. The following Decide step verifies if the client entered on the current floor. In an affirmative result, the process ends and the client continues waiting for the elevator; if the result is False - meaning that the current node is its destination, since it was transferred out of the vehicle - the Transfer step will transfer the client from the central node to the correspondent out node. Finally, the out node is connected through a path to a sink, responsible for destroying the client.

The Vehicle object, in its default state, displays some difficulties answering elevator calls. When the vehicle is in movement, it answers firstly the destination of those inside its ride, ignoring calls from clients that are waiting on floors that the vehicle passes through. The vehicle only lets clients enter its ride in two occasions: when it stops to leave out a client inside its own ride, or when its ride is empty and starts receiving seize requests from clients waiting, first answering the oldest request, instead of the nearest.

To bypass the aforementioned difficulties and adapt the Vehicle to the current objective, some properties of the Vehicle can be edited, namely:

- Initial Ride Capacity: Specifies the capacity of the elevator;
- Task Selection Strategy: Defines the strategy to select the next task;
- Dwell Time: Defines the time the vehicle will wait, whenever it has to load/unload clients;
- Routing Type: Specifies how the vehicle will decide its next movement, either following a fixed route, or a route based on client demand. The first option sets the Vehicle to a predefined route, e.g. a milk run, whilst the later regulates the Vehicle depending on client requests.

When this last property is set on ‘on demand’, it is possible to manipulate two more properties inside ‘resource logic’: ‘ranking rule’ and ‘dynamic selection rule’. These two properties are responsible for ordering the requests when they are placed, and also allow to dynamically select the next call to answer, respectively.

As such, the selected expression for the property ‘ranking rule’ is the origin of the client, which orders the queue based on the originated floor; and the property ‘dynamic selection rule’ is set to ‘DirectDistanceTo.Object(Candidate.Object)’. ‘Candidate’ refers to the client which is requesting a pick-up, and the main expression returns the distance between the Vehicle and the client. As this later expression is dynamic, it is independent of the previous expression and resulting queue. This decision occurs in the moment when the Vehicle is available to be seized. These expressions are native SIMIO functions.

It can be stated that the elevator modelled as a SIMIO Vehicle object is simple to conceive, relying mostly on native properties and few and simple process steps. Notwithstanding, it still requires prior knowledge on the Vehicle properties being edited, e.g. ‘dwell time’ or ‘resource logic’.

However, even after changing the ‘resource logic’, while testing the simulation model, the aforementioned limitations were still noticed. Furthermore, the dynamic rule to select the next request to answer was not always respected, choosing to answer the request of the client that was waiting the longest, even if the floor difference was between the top and bottom floor, also ignoring all clients in the middle.

Another way to order client requests and Vehicle decision is through priority. That is, a priority level has to be given to each client that performs a request to the Vehicle, e.g. going from high priority to the closest client, and lowest priority to the client that is further away from the elevator. This strategy would require a dynamic calculation of the priority level each time the elevator moves, as the distance from the elevator to the client changes with each movement. This strategy could be implemented through the same properties under ‘resource logic’, facing the same issues as the strategy applied

and mentioned above; or it could be implemented through robust and complex processes, erasing the major advantage of a Vehicle approach: simplicity.

As concluding remarks, a Vehicle approach is simple, but it is very limited and does not fulfil all the needs and requirements of an elevator system, even through new processes it was not possible to change the Vehicle route in the way intended, as it has its own 'logic' behind it. Furthermore, if a change in the algorithm of the elevator system is needed, the change is very limited and unpredictable, as all the Vehicle built-in processes are hidden from the common user. All these limitations would decrease the number of applications which the model could be used to simulate and experiment, and even the importance of itself. All these limitations and factors contributed to the selection of the elevator as an entity, as it would be further demonstrated.

3.2 Entity Approach

Opting for this approach will imply the construction of a SIMIO object, similar to the Vehicle one, from scratch. This approach implies that all actors are modelled in a detailed way. Thus, this section is divided in processes executed by the client and by the elevator.

1. **Client Processes**

Each client, after being created in the source, runs on the entrance path and, upon arrival at the central node of its floor of origin, will execute the process in Fig. 2.

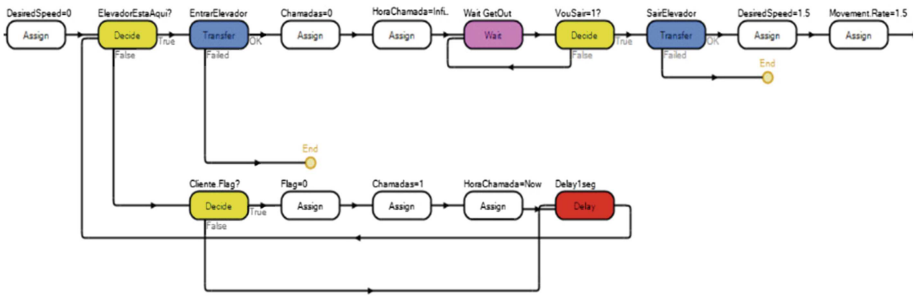

Fig. 2. Process executed by clients

This process is responsible for ensuring that each client waits for the elevator, calls the elevator, enters it when the elevator is on the same floor as the client, and the client gets out of the elevator onto the last path into the sink of the destination floor. Upon executing this process, the client will be stopped, in order to obligate him or her to wait for the elevator to answer the call that will be placed. This is achieved by setting the speed property to zero. Thereafter each client will verify if the elevator is positioned at the same floor he or she is. While waiting for the elevator, the client will change two data structures of the model: one array representing the number of calls on each floor,

and another array that records the time on which calls are made. Both indexes correspond to the floor where the call was placed.

When the elevator is in the same floor as the client, it stops and opens its doors, the client is transferred from the central node to a ride station inside the entity and both aforementioned arrays are reinitialized and updated. While inside the elevator the client is kept in a step of the process, waiting to arrive at its floor destination. It is ensured the client leaves the elevator at the right floor by checking a variable of the entity - that stores the *id* of the destination floor - and comparing it with the index of the floor at which the elevator currently is.

2. Elevator Processes

The approach in question consists of modelling the elevator as an entity, but, apart from giving it the behaviour of a typical elevator - the algorithm - it is necessary to create the object and place it in the right location. To this end, the two first steps of the process represented on Fig. 3 do that.

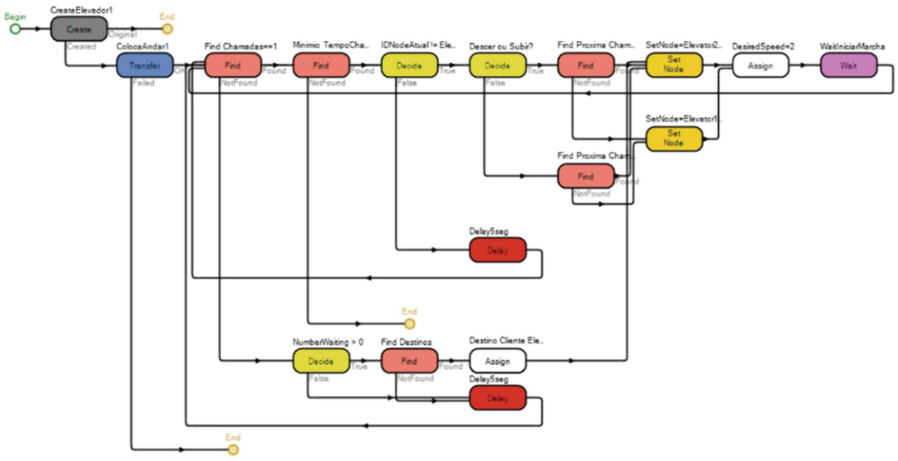

Fig. 3. Main process of the elevator

After those two initial steps, the overall process is an infinite loop responsible for the management system of the elevator, meaning that these steps are responsible for deciding the next elevator movement and will be run for the entire period of the simulation in a closed perpetual loop. Note that the next destination of the elevator will always avoid making direction changes, thus giving priority to floors with calls in the same direction it is traveling. First, the elevator verifies if there are calls registered on the system, through the array mentioned above. If there are no calls registered, the elevator will decide its next destination based on the destination of each client riding it. If there are calls, the elevator then analyses the time in which they were made, thus giving priority to clients that made the calls sooner. After having decided on the next destination, the elevator will be kept on a wait step until its doors are closed and thus it is ready to initiate its trip. To ensure the elevator stops at all floors which have calls

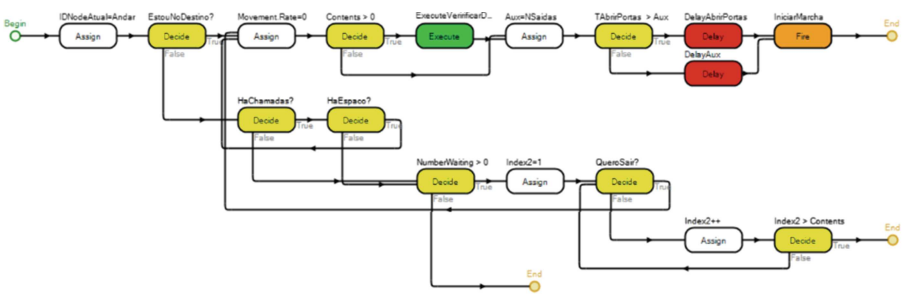

Fig. 4. Process executed by the elevator whenever it arrives to a new floor

registered or a client wanting to exit, the process represented in Fig. 4 is executed, whenever the elevator arrives at a given node that represents a floor.

In this process, the elevator will firstly analyse if it has arrived on the floor which was assigned to it as a destination. If the node in question is not its destination floor, the elevator will then analyse if there are calls on that node - ensuring that it still has capacity to hold additional clients - or if any client inside it wants to exit at that node. If the current node is the elevator destination, has a call placed, or is the destination of a client inside the elevator, the next steps will ensure that the elevator stops, and will model the time that the elevator is kept with its doors open (dwell time), allowing for clients to exit or to enter the elevator. Afterwards, an event will be fired to indicate that the elevator can resume its trip, allowing the process represented in Fig. 3 to continue its loop. In this regard, communication between these two processes is necessary and ensured, since both processes are executed in parallel.

3.3 Final Remarks

The complexity of the model using an Entity as an elevator is evidenced by the size of the developed processes and by their relation, where a process executes another process. Moreover, a process can trigger an event which was holding a token in another process, ensuring the communication among entity clients, entity elevator and processes.

The behaviour of an elevator was modelled with success and its behaviour was taken farther than what was achieved in the approach of the SIMIO Vehicle. In this sense, the approach chosen to conduct simulation experiments was the Entity approach. Figure 5 shows the elevator modelled as an entity in SIMIO. The animation in 3D represents an advantage when interacting with the model and/or showing it to others.

4 Results

Once the model was developed and validated, data was retrieved from it, in order to get relevant information that would lead to conclusions about the developed model. One of the major benefits of using SIMIO is the possibility of conducting simulation

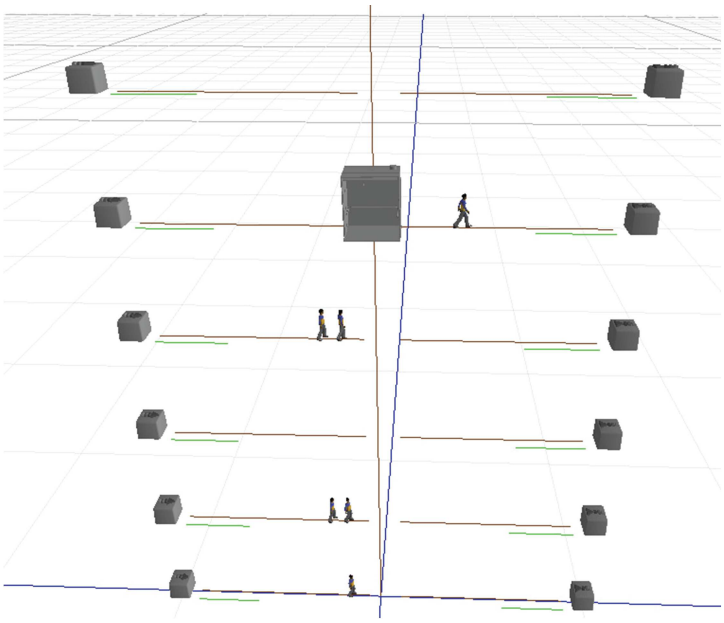

Fig. 5. 3D view of the model during its runtime

experiments on a model. A simulation experiment allows for executing a set of scenarios with different values for the model properties, and the impact of those changes on the model KPIs (Key Performance Indicators). In the present model, dwell time is the main model property in study, being changed from 1 to 20 s. Other properties were implemented and can be analysed in future studies, such as: capacity of the elevator and different arrival rates of clients per floor.

The dwell time is crucial to the total time of a client (waiting time plus travel time) because if it is increased, it increases the probability of clients entering the elevator at a floor, thus diminishing client waiting time on the current floor; but will also increase the waiting time of clients on other floors. If this time is decreased, the probability of clients entering the elevator at each stop decreases and the elevator will move more, thus decreasing the waiting time on other floors. A balance between these two possibilities needs to be found. In order to have a good representation of the impact of this property on all KPIs, the value will vary from 1 to 20 s. To note that a value of dwell time with a good performance on a specific KPI, e.g. average client total time, at up-peak time can have a bad performance on a mixed or down-peak movement of clients, as calls can be placed in a more focused area of the building, e.g. the ground floor, or can be spread across all floors. The main focus was, therefore, to analyse the impact of dwell time in the system performance, namely the following established KPI:

- Average total time in the system, per client: sum of waiting time and travel time for the clients;
- Average elevator occupation (or load): number of clients riding the elevator;

- Elevator movements: number of movements executed by the elevator in the simulation runtime;
- Average number of clients in the system: sum of clients waiting for the elevator with the ones already riding it;
- Average number of clients waiting: number of clients that are waiting for the elevator on all floors;
- Average waiting time per client;

In order to ensure that the results do not contain irrelevant data, as a result of the time needed for the system to achieve a "full-operating status", it is very important to define an accurate warm-up period. In this context, a warm-up period of 3600 s was defined because, on the several tests conducted, it was found that from this time on, the KPI values achieved a more stable status. Furthermore, 10 replications were used to ensure that different random number seeds are used. The simulation time in the experiments was 24 h.

Figure 6 represents the evolution of the average total time in system per client, as a function of dwell time. Dwell time is displayed in seconds, in all graphics, while average total time is in minutes.

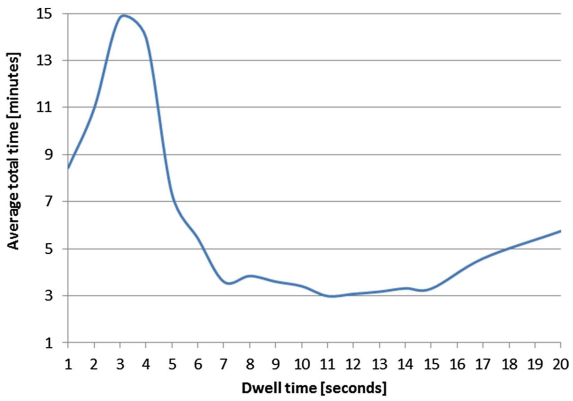

Fig. 6. Total time in system per client as a function of dwell time

For dwell time of 1 to 4 s, an increase of the average total time per client was observed, due to less time that the clients have to enter the elevator. The lower average total time values are seen in the 10 to 15 s band, rounding 3 min between the 11 and 15 s of dwell time. After this value, an ascend curve is seen due to the increase of time in which the elevator is stopped at the same floor, thus not traveling to attend other calls, increasing waiting time on other floors.

Average waiting time is a performance measure which affects the average total time and is responsible for the perception of the system quality on clients using it. Figure 7 presents the obtained results for the evolution of the average waiting time per client as a function of the dwell time.

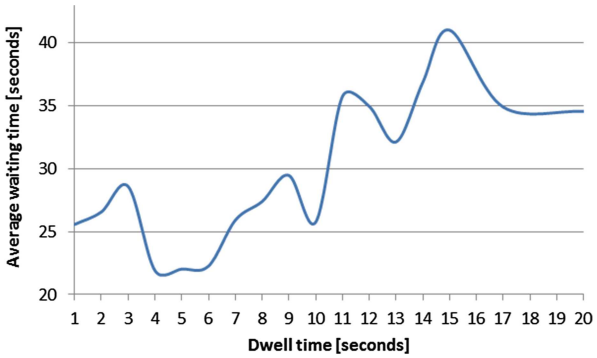

Fig. 7. Waiting time per client as a function of dwell time

The lower values are located on **4 to 6 s** of dwell time, and the maximum is reached in 15 s. The graphic in Fig. 8 shows variation of the average **number** of **waiting clients** on all floors.

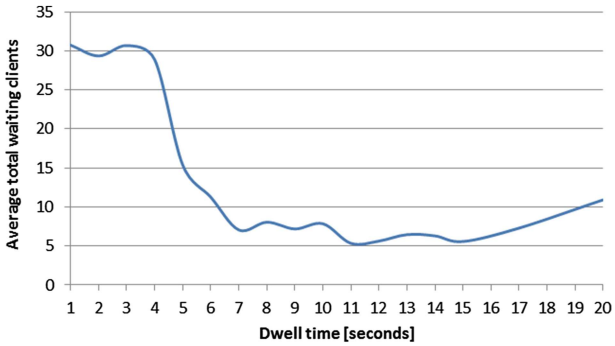

Fig. 8. Number of waiting clients on all floors as a function of dwell time

A dwell time of 1 to 5 s results in the highest values on clients waiting for the elevator, and from 20 s onwards, the curve returns to an ascending state. It is in the 7 to 18 s band that the average number of waiting clients remains below 10. The lowest values of this KPI are reached on the **11 to 16 s** band, where values close to an average of 5 to 6 clients waiting are shown.

Figure 9 shows the evolution of the **total** number of **clients** in the system as a function of the elevator dwell time. As previously stated, the values of this KPI should be the result of the sum of the total number of clients waiting with the load of the elevator. As can be seen, the values indicated by the above graph match the sum of the average number of clients riding the elevator (average occupation) and the average number of clients waiting.

The KPIs analysed until this point focus more on the side of the user of the elevator. However, other perspectives, as the **power consumption** of an elevator, do not

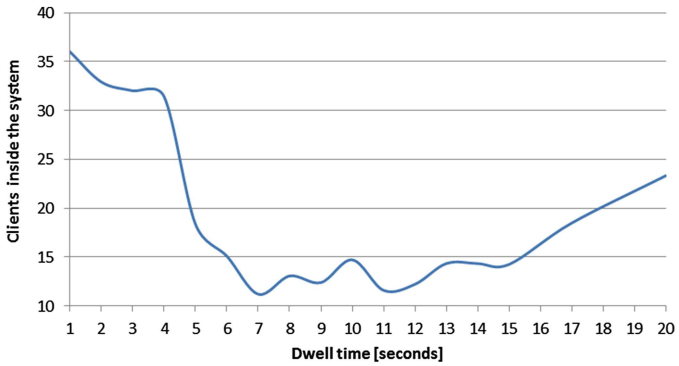

Fig. 9. Number of clients inside the system as a function of dwell time

necessarily react in the same way to a change of the dwell time of an elevator. In this sense, the consumption of the elevator was also analysed. The power consumption of an elevator system is very important, especially considering many of those systems are free of charge. There are two factors influencing this expense: number of movements and elevator occupation. The smaller the movements, the less energy will be consumed. But elevator occupation is not linear, meaning it is related to the difference in weight between the elevator cabin (including the weight carried) and counterweight, which each one is placed on each end of the cables of an elevator system. The weight difference between these two masses is the momentum the elevator engine has to provide, as in every movement one of these masses is going down, and with its gravitational force helps reducing the amount of torque the engine has to provide in order to move the cables and hoist the other mass. The weight of the cabin and the counterweight depend on the system installed, client demand and other design decisions. So, it is not possible to directly relate these two factors without knowing the system itself, but it seems correct to say that the closer it is to 40 % of the cabin maximum load, the less power will be consumed. This claim comes from the general calculation method of counterweights in the elevator industry, essential for reducing the engine effort to hoist the cabin and to maintain traction on all the cables (McCain 2007).

Figure 10 shows the number of elevator movements and its average load as functions of the dwell time. It is difficult to say the exact point on which the system will be more efficient. But it is possible to refer a band on which the system will be more efficient. That band probably lays between 8 to 15 s, due to less movements - which are less than 1/3 of the highest value registered, with a low dwell time - and the load on the elevator reaches up to 30 to 40 % of its maximum load. For this decision, it was considered that the counter-weight of the elevator is calculated for an elevator with 40 % of its load – situation in which it consumes less energy to perform its movements.

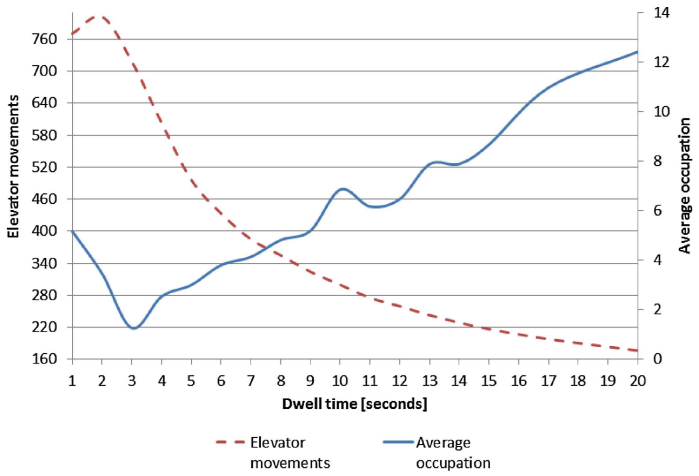

Fig. 10. Number of elevator movements and its average occupation as functions of dwell time

5 Conclusions

An **elevator** system was modelled in **SIMIO** - a recently developed discrete simulation tool. The simulation model was based on a hospital located in the north of Portugal. The tool was chosen due to its similarities to ARENA - the most used simulation tool worldwide - since they were developed by the same authors. Moreover, it fully supports 3D animation, which results in very appealing simulation models, which also contributes to a better understanding of the system in its execution.

Two implementation approaches to model the elevator behaviour were considered. First, the entity was modelled using a SIMIO built-in object, whose purpose is to transport entities from one location to another. This approach enabled a fast basic-modelling of the system, since the standard behaviour is already defined by the transporter. However, it proved to be complex to model different strategies for the elevator. Moreover, different problems with this approach were identified, for instance the elevator would always give priority to customers on board rather than stopping to allow new entrances on the way. Possible workarounds to this problem would require mathematical expressions and as such, modelling different behaviours in the vehicle would be very complex. On the other hand, modelling the elevator as an entity was very challenging, since all the behaviour had to be developed from scratch. Nevertheless, when its modelling was finished, it proved to be more flexible than the first approach, since it could be easily added different strategies to the elevator. In this sense, the model with the elevator modelled as an entity was the one used to conduct simulation experiments. One of the great advantages of using this approach is that it allows different strategies of the elevator to be incorporated. To the study in question, the only strategy modelled was to give priority to the closest floors with clients with higher waiting times. Different strategies, such as the milk run strategy could easily be implemented.

In this analysis phase, the only parameter analysed was the **dwell time**, which is the time that the elevator keeps its doors opened to allow new entrances. In future studies, other properties, such as different arrival rates of clients to the elevator, or the capacity of the elevator, could be analysed. To evaluate the performance of the system, the following Key Performance Indicators (**KPI**) were defined: average total time; average occupation; number of elevator movements; average of waiting clients on all floors and average waiting time.

In the beginning of this study, the authors thought the optimum value for the dwell time was around 5 s. However, the final analysis of the multiple KPI, indicates that the option would be to use a dwell time of around **10** s. It was found that within this time frame of dwell time a balance between all the KPI could be achieved. Different goals of the management system, may lead to the adoptions of other dwell times.

By re-using previously defined SIMIO objects in other models, this elevator model could be used on other **future research**. For instance, in multiple elevators, where ETD (Estimated Time to Destination) or other algorithms could be implemented. Furthermore, the power consumption of the elevator could also be quantified.

Acknowledgments. This work has been supported by FCT – Fundação para a Ciência e Tecnologia in the scope of the project: PEst-OE/EEI/UI0319/2014 (ALGORITMI).

References

About Elevate (n.d.). https://www.peters-research.com/index.php/elevate/elevate-elevate-express. Accessed 23 Jan 2016

About Elevate Live (n.d.). https://www.peters-research.com/index.php/elevate-live/more-about-elevate-live. Accessed 23 Jan 2016

Barney, G., Al-Sharif, L.: Elevator Traffic Handbook: Theory and Practice, 2nd edn. Routledge, New York (2015)

Dias, L., Pereira, G., Rodrigues, G.: A shortlist of the most popular discrete simulation tools. Simul. News Eur. **17**, 33–36 (2007)

Hakonen, H., Siikonen, M.-L.: Elevator Traffic Simulation Procedure. Lift Report (2009)

Hlupic, V., Paul, R.: Guidelines for selection of manufacturing simulation software. IIE Trans. **31**, 21–29 (1999)

Hlupic, V.: Simulation software: an operational research society survey of academic and industrial users. In: Proceedings of the Winter Simulation Conference, vol. 2, pp. 1676–1683 (2000)

Kelton, W.D., Sadowski, R., Zupick, N.: Simulation with Arena, 5th edn. McGraw-Hill Education, New York (2009)

McCain, Z.: Elevators 101, 2nd edn. Elevator World Inc., Masdar (2007)

Pegden, C.D.: SIMIO: a new simulation system based on intelligent objects. In: Simulation Winter Conference, pp. 2293–2300, 9–12 December 2007

Pegden, C.D., Sturrock, D.T.: Introduction to SIMIO. In: Proceedings - Winter Simulation Conference, Phoenix, AZ, pp. 29–38 (2011)

Pegden, C.D.: Intelligent objects: the future of simulation (2013). http://www.simio.com/resources/white-papers/Intelligen-objects/

Pereira, G.B., Dias, L.S., Vik, P., Oliveira, J.: Discrete simulation tools ranking – a commercial software packages comparison based on popularity. In: ISC 2011 – 9th Industrial Simulation Conference, Venice, Italy, pp. 5–11, 6–8 June 2011

Setchi, R.: Knowledge-based and intelligent information and engineering systems. In: Setchi, R., Jordanov, I. (eds.) Knowledge-Based and Intelligent Information and Engineering Systems, vol. 1, pp. 133–134. Springer Science & Business Media, Cardiff (2010)

Sturrock, D.T., Pegden, C.D.: Recent innovations in SIMIO. In: Proceedings - Winter Simulation Conference, Baltimore, MD, pp. 21–31 (2010)

Vieira, A., Dias, L.S., Pereira, G.B., Oliveira, J.A.: Comparison of SIMIO and ARENA simulation tools. In: 12th Annual Industrial Simulation Conference, ISC 2014, EUROSIS, Skovde, Sweden, pp. 5–13, 11–13 June 2014

Vik, P., Dias, L., Pereira, G., Oliveira, J., Abreu, R.: Using SIMIO for the specification of an integrated automated weighing solution in a cement plant. In: Proceedings of the Winter Simulation Conference, Winter Simulation Conference, Baltimore, Maryland (2010) AQ2

Zhang, J., Zong, Q.: Energy-saving-oriented group-elevator dispatching strategy for multi-traffic patterns. Build. Serv. Eng. Res. Technol. **35**(5), 543–568 (2014). http://doi.org/10.1177/0143624414526723