The <morespace> Project: Modelling and Simulation of Room Management and Schedule Planning at University by Combining DEVS and Agent-based Approaches

Shabnam Tauböck [1] *, Felix Breitenecker[1], Dietmar Wiegand[2], Nikolas Popper[3]

[1] Inst. for Analysis and Scientific Computing, Vienna University of Technology, Wiedner Haupstraße 8-10, 1040 Vienna
[2] RED Real Estate Development, Vienna University of Technology, Vienna, Austria; **shabnam.tauboeck@tuwien.ac.at*
[3] dwh simulation services, Neustiftgasse 57-59, 1060 Vienna, Austria

Abstract In this work an approach to integrating agent based modeling into a discrete event simulation system is developed and tested in the course of a project done at the Vienna University of Technology.
Discrete Event Simulation can be efficient and fast but people's behavior and movement is not easy to model. It would be convenient to develop a model, where both systems - discrete event and agent based - coexist. During the last years the feasibility of finding an equivalent discrete event model for any agent based model has already been discussed and formally proven in several publications. This implies that it should be possible to integrate any agent that is part of an agent based system into a discrete event system.
An application of this theory is introduced in this work, showing in detail how the agent based approach was integrated in the discrete event simulation environment Enterprise Dynamics. The <morespace> project supports and controls lecture room management and lecture schedule planning for many curriculas at Vienna University of Technology.

Introduction - The <morespace> Project

The project MoreSpace was launched to develop a software tool to assist the department of Gebäude und Technik during the planning phase of "University2015", a project of the Vienna University of Technology (VUT) to renovate all university buildings and to improve the existing infrastructure and the inherent processes by determining and evaluating the (spatial) resources required. The team responsible for this project used static methods to calculate an assessment of the number of square meters each faculty of the VUT needs for lectures and other student related activities.

This calculation was based on the number of students that took exams and the hours of teaching for each faculty. The resulting numbers did show the need of square meters required to accommodate the number of student during the lectures but did not take into consideration how these square meters are used over time. One thing that complicates things considerably is the fact that lectures at the VUT do not take place one after the other in a strict pattern as it is done at schools or colleges but are set at times favored by the lecturer. This results in a weekly timetable for the student that contains times where several lectures may be very close to each other, even overlapping, and times where no lectures at all are taking place, leaving a big time gap. Over the whole of the VUT one can say that there are certain times during the week that are more or less preferred by many lecturers, resulting in a high demand of rooms during certain time periods, where other times – i.e. Friday afternoons – are not that popular. The fact that the required amount of square meters is available, does not mean it is used in a way that ensures that it will be indeed possible to acquire a room for a lecture at a certain time.

The idea to develop a simulation was born, a tool to reproduce the situation as it currently is concerning the lectures and their demand on room and experiment with the room structure. Opposite to several other approaches used to solve the problem of limited space resources at a university that quite often focus on the timetabling problem [4, 5] this approach focuses mainly on the facility management point of view [6]. The problem of generating a timetable for lectures and courses is not the center of interest for this simulation tool. It uses the existing timetable as a basis to evaluate the current situation according to room utilization and room capacity utilization. It also identifies potential solutions that may result in a room assignment that frees space that seemed to be occupied. Possible modifications that can be generated automatically to the given schedule are time shifts within intervals that can be set by the user and the splitting of a lecture into two parallel events. Further changes to the timetable can only be done with user interactivity.

One major condition to be considered that proved to be a major constraint was to not decrease the quality of teaching. That means that the impact of possible modifications to the time schedule on the students has to be reflected in the simulation result. Instead of trying to generate a timetable to increase the utility of rooms the emphasis of this work lays in the analysis of the space management and the assignment of rooms to lectures is modeled in great detail. Using discrete event simulation in the field of facility management is still a relatively new concept [7].

The first step to developing MoreSpace was to use the data of two of the eight faculties of VUT and test if the main building on Karlsplatz as it was planned would be able to hold all lectures of these two faculties. To be able to experiment with the room structure was quickly advanced by the possibility to use different strategies for the space management. It proved to make a big difference which rules are used in the booking process.

This shows very nicely the analogy to the classical application of DEVS, the simulation of logistic processes in production lines or manufacturing facilities where the order of processing may become crucial for production times and the feasibility of manufacturing certain quotas in time. Here simulation is one of the most evident ways of analyzing and enhancing the existing system.

The classical discrete event simulator Enterprise Dynamics was chosen to develop the dynamic simulation including features like different possibilities of room selection, different management of the resources, variability of classes, class structures and number of students - only to mention a few of the features. This discrete dynamic simulator was combined with - and is guided by - methods and procedures of the real estate management like business process models.

In the course of developing the first studies the working group identified two main problems that are different to tasks which are normally solved by classical discrete event models. On one hand the interfaces to data collection and booking system had to be improved and standardized. Booking features and also the representation of data was not sufficiently optimized neither to the needs of the simulation tool that has to be implemented nor to the needs of the future booking system that has to be simulated. On the other hand the buildings of the Vienna University of Technology – as the university will not move out of the city of Vienna – remain very scattered over several districts of the city. Therefore a simulation of room booking and facility management has to be aware the problem of differently structured buildings within the university and therefore consider travelling times, sometimes even between classrooms within the same building.

1 Basics DEVS and AB

Considering the basic formalism of DEVS [1] a Discrete Event System Specification (DEVS) is a ture $M = (X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta)$, where X is the set of input values, S the set of states, Y the set of output values, δ_{int} the internal transition function, δ_{ext} the external transition function, δ_{con} the confluent transition function, λ the output function, ta the time advance function. Several of this so called atomic models can be put together to form a coupled model.

An agent based model [2] is defined as a tuple $\langle A, E \rangle$ where A is a set of agents with $A = \bigcup a^k$and $1 \leq k \leq N_{agents}$ and E is the Environment. The agent k itself is defined as a function $a^k \colon R_S^k \to \Lambda^k$; an Environment E is defined as tuple $\langle \Sigma, \tau \rangle$ where Σ is the system state, $\tau \colon R_\Lambda \to \Sigma$ is a state transformer function that changes the system state based on $r_i^k \in R_\Lambda$ with:

- $s_j^k$is the j^{th}set of variables that is seen by agent k where $1 \leq k \leq N_{agents}$
- $\alpha_j^k$is the j^{th}action done by agent k in response to a set of state variables s_j^k
- $\Lambda^k = \bigcup \alpha_j^k$ all actions done by agent k
- r_i^k, the i^{th} run of agent a^k, is the i^{th} sequence of interleaved $s_0^k, \alpha_0^k, s_1^k, \alpha_1^k, \ldots$
- $R^k = \bigcup r_i^k$, the set of runs of agent k, where $1 \leq i \leq N_{runs}^k$
- $R_S^k = R^k$ that ends with an s_j^k

During the last years the feasibility of finding an equivalent discrete event model for any agent based model that conforms to the specification given above has been discussed and shown in several publications [3]. This implies that every agent that is part of such an agent based system can be integrated in a discrete event system.

It is obvious that both modeling techniques have their advantages and drawbacks. Discrete Event Simulation is known to be efficient and fast as long as the concept of event driven time steps is able to use its advantage of jumping over time intervals where no changes to the system state occur and only update the system elements of the time points where events are schedules or triggered. People's behavior and movement is hard to model in such a system. Movement i.e. is usually not along a certain foretold line – people do tend to take the shortest route from A to B but the easiest way that seems to be free of obstacles and decisions between different ways are due to individual personal preferences – some people prefer to take the stairs, some rather wait for the elevator, some take on the longer walk to the escalator to save themselves from having to take the stairs. The goal here would be to develop a model, where both systems - discrete event and agent based - coexist.

In this project the discrete event simulation environment Enterprise Dynamics (ED) is used to develop the simulation model, which is designed to develop discrete event simulation systems. The formalism shown by DEVS is very much reflected in the basic make up of ED. Basic elements called atoms would refer to the atomic models described by DEVS. They are all identical in their basic structure, their behavior defined by the transition functions. A simulation model is build by using atoms again, again a concept conform to the coupled models in DEVS. The main idea now is to create an atomic model that behaves like an agent to integrate it in the discrete event simulation.

The main characteristic of an agent is its ability to make its own decisions based on its own state and its environment. To model the human decision making in this work the Utility Theory is used: it assumes that the decision process has two elements: the options and the evaluation function, called utility function that maps each option in the choice set to a numerical value. The function $u: \mathcal{X} \longmapsto \mathbb{R}$ is a utility function if $\mathcal{X}$ is the set of choices. Preferences of the modelled individuals can be: no preferences ($\backsim$), prefer the first over the second option ($\succ$) or the second over the first ($\prec$). If the preferences observed in the individuals modeled correspond to the relations given by $u(\circ)$, $u(\circ)$ is called a valid utility function for the given decision problem.

As application of this theory the MoreSpace project done for the VUT is introduced: The simulation of the booking management for all courses held at the VUT and the student flow to determine the utilization and accessibility of lectures and the allocated space. Dealing with the simulation of the whole Campus in Vienna as well as the student behavior it needs the best of both worlds, DEVS and ABM, to cover all characteristics of this system. Entering and leaving rooms can be best modeled using a queuing system whereas students are best represented by agents to model their individual behavior regarding the attendance of lectures as well as the travelling between lecture halls. The main idea is to use an ED atom to create an agent that interacts with and moves through the discrete event simulation, regarding it as its environment. .

2 The MoreSpace Model

The MoreSpace model actually simulates two different but interlaced things: the booking/assignment of rooms for lectures and the dynamic behavior of students attending these lectures and the resulting utilization of space.

The MoreSpace simulation model uses three different sources for the data basis of a simulation run: the information about courses, all information regarding the lecture halls and their status and information about the number of students attending a course are estimated based on historical data as well as actual trends.

The MoreSpace simulation uses this data for calculating a suggestion for the assignment of rooms according to the selected booking management rules. This leads to successful and not successful booking attempts, both which are part of the simulation result. Other results are i.e. the expected utilization of rooms.

This suggestion is used as a basis for the dynamic simulation of the semester that includes the movement of students through the system. Students are simulated as single agents with their own 'intelligence'; they are able to make decisions i.e. which lecture to attend if two lectures overlap.

During the course of the semester they attend their lectures and move through the university campus, causing the dynamic behavior of the system. This delivers additional results like the capacity utilization of the rooms – the number of students attending the lectures held in a room opposed to the capacity of the room. This gives a greater understanding of the real demand on room compared to the available space. Also a result of the dynamic simulation based on the student - agents is the accessibility of lectures – spatial as well as temporal and the utilization of rooms. Accessibility is of great interest for it takes the size and layout of the VUT Campus into consideration. This is one of the major advantages of MoreSpace: it does not stop at the best possible assignment of rooms but also considers the individual problems that may occur for students to be able to attend a lecture.

The benefit of using a simulation tool in general is that ways of enhancement can be tested without a risk to the real system. The same can be said about MoreSpace: strategies for booking can be experimented with, in scenarios rooms can be closed, added, joined or redecorated with no risk or costs. Results can be achieved very quickly; the comparison of different scenarios can be easily done.

The model is designed to allow three different angles of experimenting:

- The room structure containing the type of rooms and their capacity and location,
- The list of courses held with their time and expected number of students and the required setup of the room
- The booking management used for the assignment of rooms to the single lectures of each course

A main task of creating a simulation of a real system is simplifying the system as it is observed in the real world as much as to reach a model that is only as complex as it is needed to achieve the desired results. In the case of MoreSpace the main focus was on the different rooms available for lectures and the long list of events that had to be managed. Only after working with the input data and analyzing it the realization dawned that one aspect had not been taken into consideration so far: the students. Just as important as it is to find the best equipped room with high enough capacity for every event, is it to ensure that students are able to attend this perfectly planned event. The collision of lectures is a very common problem that forces many a student to decide with lecture he will not attend. This may be due to overlapping times of lectures but even more often it is the time it takes to get from one lecture room to the next. This movement time is often underestimated and causes lectures to interfere with each other although they are not really taking place at the same time. But enabling students to attend their lectures as unopposed as possible is one important factor for keeping the quality of education on the VUT at its best.

So the focus in the model was shifted to also include the students. In a classic DEVS system the students would have been regarded as entities that are routed through the system, in this case through the rooms and the lectures occurring there. In the case of two overlapping lectures the entity would have always followed its designated path: attend the first lecture until it is finished and then proceed to the next. But students are no mindless entities; they have their own priorities and preferences and usually every student is a unique and to some extent unpredictable person. The classical DEVS approach seemed not satisfying and considering a group of people with certain behavior quickly leads to the most common approach for problems of that kind: Agent Based Modelling.

As already described before agent based modeling works with individual ‘agents’ that have a predefined behavior that results from an evaluation of their current situation and a set of rules.

According to this students can be considered as agents, their current situation results from the list of lectures they want to attend and the decision of where they will go is derived from the given set of rules. If certain possibilities are entered for certain decisions the individuality of the single persons simulated can be achieved.

The MoreSpace Model is designed to simulate the behavior of about 20 000 students. The high number of agents to be simulated proved to be too much to remain within reasonable computing time of a simulation run. Therefore the simulation of the exact movements of each agent based on the layout of rooms and corridors was not implemented in ED but in an external simulation model developed in JAVA. The ED model is able to use average travelling times or to interact with the JAVA model by sending the student agents to the external model of corridors and have them re-entering the system as soon as they have reached their destination. The agent based approach is the key for the successful interaction of both models as the agent simply crosses the borders to the other system and returns to the ED model at a later time.

One thing the CA model was able to show was the dependency of the time needed to reach a certain lecture room from the ‘student traffic’. It seems logical that the time will increase if a lot of people move through the same corridors, obstructing each other. The extent of the delay due to a high people density is shown in **Figure 1**. This result could not have been calculated in ED model.

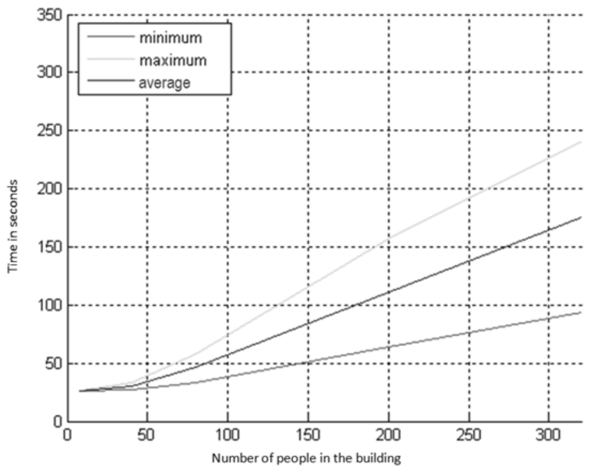

Figure 1. Walking time from room HS1 to room HS2.

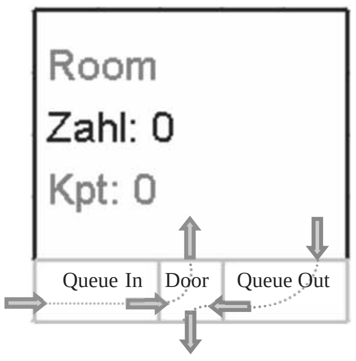

Figure 2. Structure of Room in ED.

To model the act of entering and leaving the room a server – queue combination is used consisting of two queues – one for the people leaving the room, another for those who want to enter – and a multi-server that represents the doorway as shown in **Figure 2**. A multi-server in Enterprise Dynamics is a server with capacity greater than one. A regular server in ED is also able to process several entities at once but only in a so called batch-run. This means all entities are processed simultaneously. The multi-server is able to handle several entities even with deferred entry times. The capacity of the multi-server representing the door corresponds to the size of the doorway. A small door where only one person at a time can pass through will have capacity 1; larger doors an accordingly higher capacity.

Entering and leaving has no fixed priority. Both queues are connected to the Server. A student entering the Queue_In will pass through the Door atom and enter the Room. A student leaving the room will enter the Queue_Out, pass through the Door atom and leave for their next destination.

Each room is represented as a single object within the simulation with its own attributes capacity, category and divisibility as well as its list of events with their date and time.

3 The Student Agent Atom

According to the aforementioned definition an agent goes through a sequence of states s_j^k following certain actions α_j^k; from the DEVS point of view an action α_j^k of an agent is a time consuming activity; therefore it is possible to replace it with an event triggered at the beginning of this activity using δ_{ext} the external transition function; the time consumed by the activity needs to be represented by ta the time advance function. So basically every action α_j^k needs to be replaced by an event $x \in X$ and $\delta_{ext}(x)$ to ensure the correct update of the state variables of the atom itself as well as its environment. The state transformer function τ that updates the environment of the agent needs to be ingrained in the interaction of the agent with the DEVS based system.

The main problem is to coordinate the behavior of the DEVS system with the AB elements. In the case of the project MoreSpace the solution to this problem lays partly in the definition of the problem itself and partly in the simulation package used. Enterprise Dynamics is a discrete event simulation environment that represents the DEVS formalism in its basic makeup. All elements, called atoms, share the same layout and basic functionality. Several atoms can be used to build a more complex atom; several of them can be combined again, reflecting the principle of coupled models.

A given set of events may cause an update of the atoms state, the transition functions can be defined using the ED internal programming language. The time advance function ta is handled using an eventlist that controls the adjustment of time steps.

But additional to everything that forms an atomic or coupled model, ED offers spatial attributes as well as the functionality to move objects through space. And the set X of incoming values to trigger an event includes two elements that assist the ABM as well: the timed event x_t and the message event x_m. Both are not caused by a change of state of a model but in the first case by reaching a certain point in time and in the second by a message from outside of the system. Especially the latter option makes it possible to control elements in the DEVS system from external.

In case of the MoreSpace project this option was used to transfer the agents out of the ED model into a JAVA model for the calculation of travelling times at the movement between two spatial locations.

The agents represent students that attend lectures at a given time; therefore it is possible to use timer events for controlling the activity of the agent atoms. Triggered by a timed event the activity of the agent takes place: they either change their location by moving towards a certain lecture hall, take a decision whether they should attend a lecture and chose one in case of several overlapping, they enter a queue, or leave the ED model for the more exact simulation of peoples movement in a JAVA model.

But it is the agents themselves that control the setting of these events: each atomic agent has its own personal settings and takes its own decisions. The student is considered an atomic model with the behavior of an agent. It is a member of both worlds, DEVS as well as ABM because it still interacts with the DEVS environment as a regular input and output of the DEVS system represented by Queues and Server.

Considering the definition of a run r_j^k an action $\propto_j^k$ takes place in response for every state $s_j^k$the student takes in. $S^k$is the set of all state variables an agent a^k can see. The actions as well as the change to the state variables have to be defined by using the structure of the atomic model.

For the interaction with the discrete event simulation system the agent has to act as an input to the DEVS elements. I.e. in ED an external event of a Queue is triggered by an incoming atom.

The student agent entering the queue atom is treated as the positive input value $x \in X$ and therefore triggers the time advance function and the external translation function, thus the autonomous behavior of the atomic model 'Queue'. The 'Queue' treats the agent atom just like any other input, resulting in the output value $\lambda(s)$ that causes the student atom to be moved into the next atom, the 'Door'. Here again the agent atom is treated as any regular input, generating the output ue $\lambda(s)$ after $ta(x)$ has passed from the 'Door', that causes the agent atom to be positioned in the 'Lecture Room'. Here the control is returned to the agent atom: the lecture room has no further functionality but to count the number of student atoms it contains, it does not generate an output value that influences the agent atom.

The next action of the agent atom is completely independent from the activity of the discrete event system elements: the student will leave again at the time either the lecture ends or it has another activity planned that is of higher interest than its current activity. Leaving is the identical procedure: the student agent enters the queue leaving the room, passes through the door and then moves on to its next activity.

The main characteristic of students is their ability to make their own decisions based on their own state and their environment. To model the human decision making in this work the Utility Theory is used: it assumes that the decision process has two elements: the options and the evaluation function, called utility function that maps each option in the choice set to a numerical value.

In case of the MoreSpace model one decision the student has to take is the case of overlapping lectures. If a lecture x and a lecture y take place at the same time the student has to decide which one to attend.

Basically the utility function for the decision making of the student for overlapping lectures can be derived from the type of lectures the student has to choose between. Of course the individual preferences need to be added as well: the option not to go anywhere can be added to model the possibility that a student might not attend any lecture at all. The decision between lectures may be done stochastically.

But a tendency towards one lecture has to be remembered – the decision the next time may depend on the decisions felled in the past. The decision also depends on the quality of the lecture itself: Are there enough seats? Does the course have a good scriptum? Is it an interesting topic? The depth of the utility function can be easily altered.

4 Student Numbers

A general problem in formulating the model was the lack of distinct information regarding the number of students attending each lecture. Due to the lack of mandatory attendance at most lectures the numbers differ quite strongly from the number of passed exams or even the number of enrolled students.

It is a common occurrence that the numbers of students that attend a lecture tend to drop after the initial weeks of a new semester. This effect is especially distinctive observable in lectures without mandatory attendance.

The real number of students following the course is mostly not matching the number of students really attending each lecture; the best approximation for this number is n_F, the number of students taking the exam for passing the course.

But the number n_{A_t} of students attending a lecture at time t depends on several factors:

- The perspicuity of the lecturer – the quality of the lecture influences the attendance of students considerably. A lecturer that is hard to understand or whose explications are hard to follow will have fewer students than one who presents his lectures in a comprehensible and interesting way.
- Of course the topic itself is always a factor as well
- The quality of the accompanying material: a lecture that offers high quality manuscripts that contain anything the student needs to pass the exam usually lead to a drop in attendance. The strongest factor for attending a lecture after it being mandatory is the need for lecture notes. If they are provided one main factor simply falls away.
- The importance of the lecture: compulsory lectures are usually highest in priority for they must be done. Some courses are precondition for getting a place in a tutorial
- Assessment of student's performance: if during the semester several interim tests are done for the assessment of the students this usually leads to a different behaviour: at the date of the tests most students attend the lecture to take part in the test. Right before and after these dates the number raises and drops respectively.

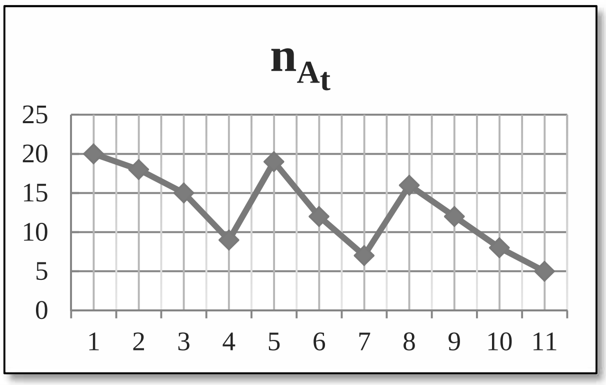

Figure 3. Course with Interim Tests.

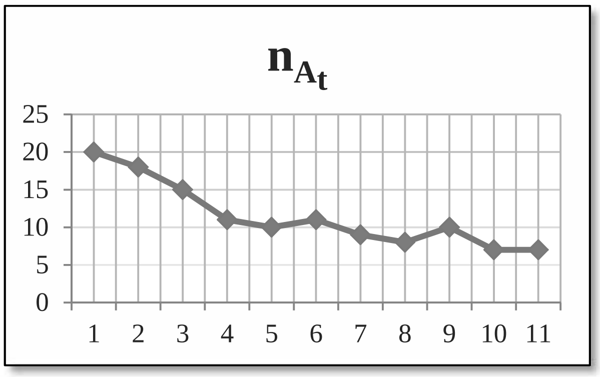

Figure 4. Course without Interim Tests.

Using the agent based approach did allow to use the known data as basis for the number of students. Students have their own behaviour in attending lectures that recreates the attendance behaviour of the real system. Known are:

n_E … Number of enrolled students
n_F … Number of students finishing the course by passing the exam

Not known are

n_{A_t} … Number of attending students at time t
n_{D_t} … Number of drop out students at time t
n_{N_t} … Number of not attending students at time t

We can say for sure:

$$n_E \geq n_{A_t} \geq n_F$$

Considering the current situation on the TU Vienna one may even be sure to say:

$$n_E > n_{A_t} \geq n_F$$
$$n_{A_t} = n_F * F_S$$
$$n_F = n_{A_t} + n_{N_t} - n_{D_t}$$

For courses with a mandatory attendance one might say:

$$n_E > n_{A_t} = n_F + (n_{D_{t_E}} - n_{D_t})$$

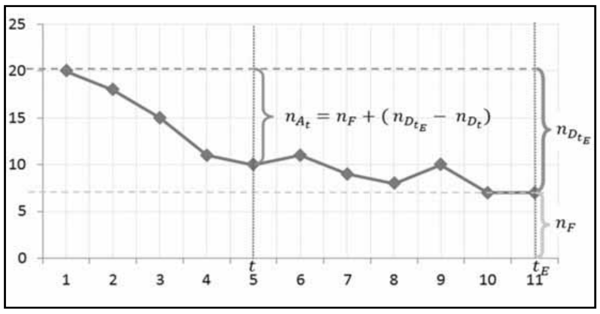

Figure 5. Number of Attending Students.

Using the agent based approach did allow to use the known data as basis for the number of students. Students have their own behavior in attending lectures that recreates the attendance behavior of the real system.

The dynamic simulation involves the student going through the semester and attending their courses. They allocate space in the assigned room and add to the capacity utilization of the room.

The system state changes during the simulation due to events caused by the elements of the simulation model. There are two additional events that influence the system state but are not triggered by the occurrences within the system:

- Assignment of rooms to lectures
- Blocking of rooms during certain time period due to reservation or reconstruction

The booking of rooms for lectures will generally take place before the actual semester begins, but there will always be a demand of space during the semester as well, resulting in additional room assignment as well as an adjustment in the student behavior. Additional lectures or other events will cause student to attend them and therefore make them take new or different decisions than they would have before this change happened.

The unavailability of a room for a certain time period due to reconstruction or other reasons will lead to a reassignment of all lectures booked in this room. This will also affect the students as they have to be informed about the change in location.

The student atom contains the behaviour of the student as well as their list of lectures they have to attend. This list is generated at the time of creation of the student and results from their semester as well as field of study. The generation is based on probability functions to achieve a wide variety of lecture combinations to represent the inhomogeneous groups of students that occur especially in courses in higher semesters.

The behaviour of the students is based on their goals – attending their lectures – as well as on the overall situation. In case of collision of events they wish to attend students have to make a decision.

4.1 Event Matrix

Each student atom owns a matrix of lectures it will attend. The size of the matrix *n* x 4 is given by the number of lectures *n* the student is assigned to. This matrix L_s contains the basic information for the behaviour of the student s during the simulation:

$$L_s = \begin{pmatrix} l_{1,1} & \cdots & l_{1,4} \\ \vdots & \vdots & \vdots \\ l_{i,1} & \cdots & l_{i,4} \end{pmatrix}$$

- The element $l_{i,1}$ contains the begin time of the lecture i.
- The element $l_{i,2}$ contains the end time of the lecture i.
- The element $l_{i,3}$ contains the ID of the course attended at lecture i.
- The element $l_{i,4}$ contains the pointer to the room of the lecture i.

4.2 Attending a Lecture

Here the event for the attending of a lecture is set. The time the student will need to reach the location of the lecture hall has to be taken into consideration. We define:

- T_c current time
- T_i start time of the lecture
- $\hat{T}_i$ end time of the lecture
- $\tilde{T}_i = T_e + t_w$ start time of the lecture considering walking time
- T_e time for the student to start walking
- t_w Time it takes the student to reach the lecture hall from their current position
- t_e time until T_n

If $i \leq n$ we can assume that:

$$\exists\, T_i = L_s(i,1) = l_{i,1}$$
$$\exists\, C_i = L_s(i,3) = l_{i,3}$$
$$\exists\, R_i = L_s(i,4) = l_{i,4}$$

where

$$l_{i,1} \geq T_c$$
$$C_i \neq C_c$$

Hence:

$$t_e = \max(T_e - T_c \;,0)$$

If i > n the students have finished their list of lectures and can leave the simulation.

4.3 Queuing

During this activity the student moves into the queue in front of the lecture hall. The according event for leaving a room is set by evaluating the next lecture the student will visit:

$$T_i = L_s(\mathrm{i},1) = l_{\mathrm{i},1}$$
$$\hat{T}_i = L_s(\mathrm{i},2) = l_{\mathrm{i},2}$$

under the following conditions:

$$T_i \geq T_c$$
$$C_i \neq C_c$$

$$T_{i+1} < \hat{T}_i \Rightarrow Conflict \text{ !!!}$$

According to the given probability $\mathbb{P}$ a student will chose one of the two possibilities for the time T_e when they will leave the current lecture:

$$T_e = \begin{cases} \hat{T}_i \\ \max(T_i - t_w, 0 \end{cases}$$

and

$$t_e = \max(T_e - T_c \ ,0)$$

5 Results of MoreSpace Simulation

5.1 Utilization of Lecture Rooms

This data shows the number of hours each lecture room was booked by the booking procedure. This shows the theoretical utilization, the time the room is booked, but not the time the room is truly used. As past experiences have shown sometimes rooms may be booked for a lecture that does not take place.

5.2 Capacity Utilization of Lecture Rooms

This data shows how many students did attend a lecture in the simulation. The number expected is given; the according number of students is assigned. If the number of attending is lower than that it hints at a problem at the accessibility of the course.

5.3 Not Booked Events

The booking procedure tries to find a lecture room for every lecture planned. If it is not able to assign a room the according lecture is listed in this data. For the comparison of several simulation runs one has to make a distinct decision on which aspect the attention is focused. Depending on this the key data has to be selected.

The following example shown in **Figure 6** illustrates how easily data can be misinterpreted in the comparison of two scenarios:

Scenario 1: 1 lecture from 11.00 to 15.30 for 56 students could not be booked in any lecture room.

Scenario 2: 2 lectures from 10.00 to 11.30 for 23 students and from 15.00 to 16.00 for 41 students could not be booked in any lecture room.

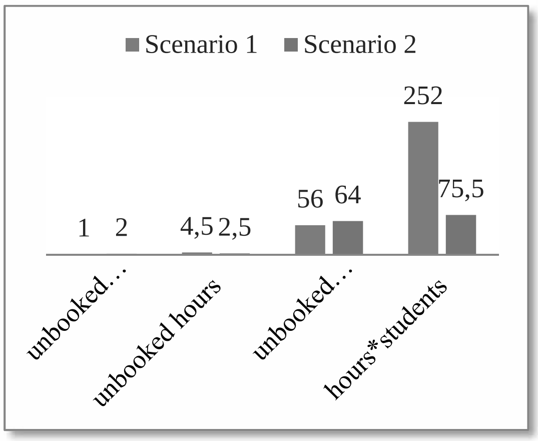

Figure 6. Key Data for Not Successful Booking.

This demonstrates the importance of defining the correct key data; Depending on which value is considered the assessment of the simulation results can be interpreted completely different.

Considering the number of not booked lectures Scenario 1 seems to deliver the better result. The number of not booked hours quickly shows another picture: where in scenario 2 both lectures together result in 2.5 hours that could not find a room, Scenario 1's 1 lecture requires 4.4 hours of time.

The picture again changes if one looks at the number of students that cannot attend a lecture without a room: Scenario 1 is the better one in this regard. But taking the hours of lecture each student misses into account Scenario 1 suddenly looses highly against Scenario 2 again.

5.4 Accessibility of Lectures

The accessibility of lectures can be interpreted in two different meanings:

- Temporal accessibility: this indicates if a lecture overlaps with another lecture.
- The spatial accessibility indicates if a lecture can be reached in time: this considers lectures that take place after each other, even with a time gap between them but the location of the rooms is such, that it is not possible to reach the second lecture on time.

While the first kind can be easily determined by evaluating the given data, the second is much more difficult to estimate: the real time it takes from one lecture hall to another depends on far more than the spatial distance: The density of people moving through the corridors, the waiting time at elevators, the distance to staircases influences the walking time.

This makes the evaluation of the spatial accessibility to one of the simulation results as it is able to deliver far more accurate results than estimation by distance.

6 Conclusion

Using an agent based approach for simulating the students in the <morespace> project proved to be a good approach to cover several demands:

- The possibility to model the exact movement of students during the corridors and across the TU Campus, even covering the eventuality of travelling between buildings that are further apart or emergency evacuation simulation.
- Students have to be regarded as entities with individual preferences and decisions based on their previous behaviour and state.
- The need to hand control over their actions to the students themselves. They are not routed through the system via a course of server and queues but move on their own.

The implementation in ED did result in a hybrid agent atom: it has the basic attributes that describe an agent:

- Autonomy: each agent acts on its own and decides its own behavior
- Social ability: agents are able to communicate with each other
- Reactivity: agents react to their environment and changes therein
- Pro-activeness: Agents do not only react to their environment but act on their own as well

But due to the ED configuration the agent atom still holds the basic functionality that relates to the DEVS concept. This enables the interaction with the ED model at certain points according to the general input/output procedure. This is the case every time a student enters the queue in front of a door leading to a lecture hall: the basic configuration of any atom is also present in the agent, thus making it able to react to the event of entering and exiting another atom. This is used to update the state of the agent

For the specific project MoreSpace this solution was the best combination as the atomic agent belongs to both worlds, making it some kind of double agent that is able to use the resources of DEVS as well as ABM.

The agent based approach provides a very detailed insight into the movements and activities of the single students and therefore allows an assessemesnt of effect of certain scenarios on the quality of teaching as well as on the averall state of the room situation.

Acknowledgements

Development of MoreSpace has been granted by ZIT Technologieagentur der Stadt Wien GmbH, within a FemPower Framework (www.zit.co.at). Implementation of Morespace has been granted by Vienna University of Technolgy. More information at www.morespace.at.

References

[1] B. P. Zeigler. *Theory of Modeling and Simulation*. John Wiley, New York, 1976.

[2] M. Wooldridge,. *An Introduction to Multi Agent Systems*. John Wiley and sons, Chichester, England 2002.

[3] B.S.S. Onggo. “Running Agent-Based Models on a Discrete-Event Simulator”, in *Proceedings of the 24th European Simulation and Modelling Conference*, 25-27 October 2010, Hasselt, Belgium (ESM10). Ostend, Belgium: Eurosis-ETI, 2010.

[4] S. Abdennadher, M. Marte. *University course timetabling using constraint handling rules*. Applied Artificial Intelligence: An International Journal Volume 14 Issue 4, pp 311-325, 2000,

[5] C. Beyrouthy, E. K. Burke , B. McCollum, McMullan, D. Landa-Silva, A. Parkes. *Towards improving the utilisation of university teaching space*. Technical Report NOTTCS-TR-2006-5, School of Computer Science & IT, University of Nottingham, 2006.

[6] D. Wiegand, P. Mebes, V. Pichler. Real Estate und Facility Management– neue Anwendungsbereiche für diskrete ereignisgesteuerte Simulationen. *Procedings zur ASIM 2006, 19. Symposium Simulationstechnik*, pp. 15-20, 2.–14. September 2006, Hannover, Germany.

Submitted: September, 2010
Revised: March, 2011
Accepted: March20, 2011