An Agent-Based Approach to Increasing Space Utilization of Office Buildings

Štefan Emrich

Department of Analysis and Scientific Computing, Vienna University of Technology, Wiedner Hauptstraße 8-10, 1040 Vienna, Austria; *stefan.emrich@datengeschichten.at*

SNE Simulation Notes Europe SNE 25(1), 2015, 9-16
DOI: 10.11128/sne.25.tn.102273
Received: April 10, 2014; Revised December 10, 2014;
Accepted: January 15, 2015;

Abstract. Based on a brief analysis of the status quo of office space utilization (section 1), a hybrid simulation model combining discrete event simulation (DES) and agent-based methons (AB) is developed in section 2. In section 3 it is to analyze some general characteristics of such office systems. Although academic, the results underline the huge potential benefit for an increased space utilization through utilization of mathematical simulation.

Introduction

In private businesses the cost factor is one of the main – if not the major – contributor to decission making. Nevertheless there is a certain blind spot when it comes to space related costs, which is partially induced by the status of owning representative buildings or a spacious office. But aside from this, the awareness for the potential savings (regardless whether of GHG emissions or financial ones) through an efficiency increase in space management is hardly existent, yet. Subsequently current approaches to reduce space related costs focus on buildings' operating costs instead.

1 Potential for improvement

The effect of this focus is illustrated in Fig. 1 and has been described by Zitter et al. Operating costs account for only 20% of the annual building-related costs (1^{st} bar in Fig. 1; left to right). They then provide benchmarks according to which roughly 40% (of the initial 20%) are capable of being influenced —- thus 8% of the total costs (2^{nd} bar). It is further possible to reduce the influenceable costs by 30% (= 2.4% of total costs; 3^{rd} bar). Assuming a realistic reduction of 50% in practice, the total costs can be reduced by a mere 1.2 percent (4^{th} bar)! It is thus obvious that this approach cannot contribute to substantial savings.

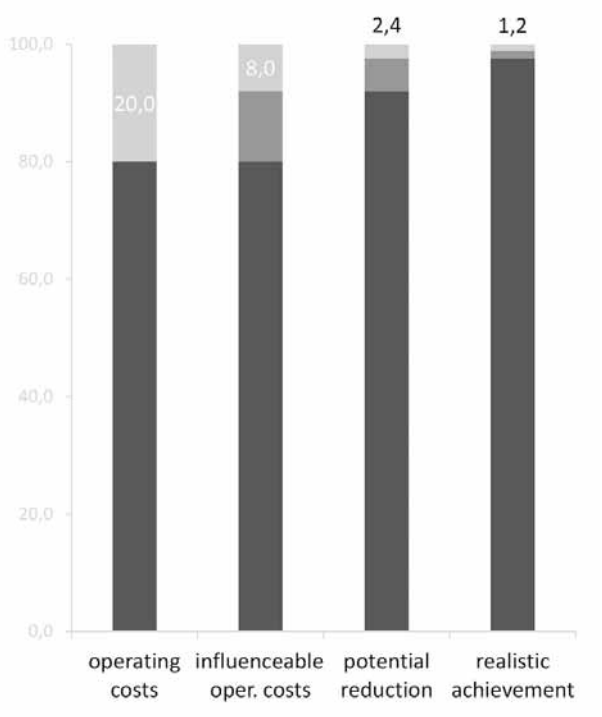

Figure 1: Practically achievable reduction of builing related costs by tweaking of operational costs.

On the other hand buildings are used only for a fraction of their life time only. As explained by Ottomann effective utilization of office space lies around a mere 5%. This includes a working week of five 8 hour work days, holidays and vacation of employees, breaks, sick leaves and social as well as organizational activities.

It is apparent that an efficiency increase in space utilization offers a far bigger potential for savings than reductions of operating costs. A theoretical increase of 5 percentage points (i.e. from 5% to a utilization of 10%) does equal cutting the space required in half – and thus reducing space-related costs by approximately 50%! This is illustrated by following example: A company with 100 offices has a utilization of 5%. Availability of hundred offices per week (7 days á 24 hours) is 16.800 office-hours ($100 \times 7 \times 24$). Utilization of 5% means that a mere 840 office-hours are actually "consumed" by the employees. As the actual need (840 office-hours) is not changed by a more efficient space-management, a raise to 10% utilization efficiency would require an availability of only 8.400 office-hours ($840 = 10\% \Rightarrow 100\% = 8.400$), which calls for (8.400 divided by 7 days á 24h) 50 offices – a 50% reduction of the original 100 offices.

The question that arises is: "How can (office) space be used in a more efficient manner?" - Which is equivalent to that of how much space is truly needed.

1.1 Static Approaches

As stated in Emrich et al., decision makers need to know how much space is truly needed to answer this questions. This of course is hard to answer without adequate information. Approximations can be derived by rule of thumb estimates customary in the particular trade, although they will remain (rough) estimates. Chances are that the need for space will be over- or underestimated. Both outcomes come with significant costs (see Kovacs et al. for financial insight on inefficient utilization of office buildings). Either there will not be enough space for all employees, which not only requires renting additional space, but also disrupts workflows and thus decreases overall productivity. Overestimating required work space, on the other hand, leads to sub-optimal utilization. The situation improves less than it could have.

Trying to improve the results, more detailed calculations could be carried out. Nevertheless these will become extremely complicated and complex when trying to incorporate different behavior of employees. For example will sales representatives have needs different from in-house account managers, who will again have needs, working- and vacation times that differ from those of the IT-staff. Getting exact results under such heterogeneous conditions is challenging, to say the least. In addition, even if it would be possible to obtain exact results for this problem, they would be valid only for this one scenario. A change within the employee structure or a different space management strategy would require starting from scratch, as all calculations and consideration have to be applied to the new scenario. Another flaw of this approach is that it neglects the stochastic nature of the observed system (i.e. employees are not robots that have ever repeating, non-changing routines within their work-cycles).

Another approach is to closely monitor and track the employees' actual work place needs and use the data obtained for statistical analyses (e.g. electronic monitoring of workplace activities, collecting information on employee position, etc.). Nevertheless, this approach has some major drawbacks. First it raises issues regarding privacy. And even if legal it is likely to cause bad blood among employees and/or staff associations. Second if monitoring systems are not installed yet, it is costly to do so. Further it takes a long period of time to acquire sufficient amounts of data. Third data gathered is, by definition, always historic – even in real-time systems. Thus it can only be used to explain and analyze (management) strategies, employee structures and office layouts that have existed and been monitored in the real world (i.e. those from which the data comes from). But the data is only of very limited use when trying to understand the effects of alternative scenarios (e.g. modified employee structure, different working times, changed space management, etc.).

These above two methods (rule of thumb and statistic analysis) are regarded static models as they do depict the system behavior, but without any change over time. This is not to be confused with an (in)ability to "predict" the future state of the system. But the prediction does not change over time, as well as the models itself do not change their states.

2 Model Implementation

To overcome the limitations of static approaches a dynamic model – based on discrete event simulation (DES) and agent-based methods – is being developed. The factors that need to be taken into account when modeling space utilization in an office environment are fairly similar to those of the described university environment. As explained in Emrich et al., these are:

- employee structure
  i.e. which employee types are within the system and how many employees of each type

- employee behavior
  i.e. which working preferences and what kind of behavior do the employees have (e.g. working times, fraction of field work, etc.)
- office environment
  i.e. how many offices/workplaces of what type are available
- The space management in place
  i.e. which rules have to be considered when it comes to assignment of work places, which workplaces are available for whom (employees or employee-types), etc. – these rules strongly depend on the objective of the simulation

2.1 Modeling Approach

In such a setup, as explained in Emrich et al., the individual employee can be regarded as the smallest unit. It is her behavior that defines the simulation result, and subsequently it is necessary to depict the employees in the most accurate way. For this reason "top-down" approaches (such as statistical methods) are only of limited success: they describe the system as a whole – without giving respect to the interactions of the system internal elements. Agent-based (AB) methods, as used in the present case for development of the "More-Space Office Tool", are producing the system's behavior via definition of its smallest units and their respective interactions — the employees and their behavior. AB modeling is treating every instance (i.e. employee) as an independent entity with an individual behavior.

Further, to recreate realistic behavior, the stochastic nature of events has to be incorporated into the simulation model. This is necessary as, for example, employees will not come to work every day at exactly 8:00 A.M. On the contrary they will most often come a bit earlier or later as they have to deal with "unexpected" events, such as traffic jams or delayed public transport. Such events can potentially trigger chain reactions (e.g. missing the first of a series of connections by only a second can lead to a cumulative delay of several hours) and are thus vital for the dynamic nature of the model. Discrete event simulation (DES) is aiming at such problems and is therefore incorporated into the model.

In order to combine the features of AB methods and DES a hybrid model was created using the simulation environment AnyLogic, which is based on the object oriented programming language JAVA and capable of supporting both approaches (AB and DES).

2.2 Employee

Within the model each employee is modeled as individual agent in a class called "Worker". This agent has several parameters and variables and a combined statechart for its health- and work-status (see Fig. 2). The parameters of the object Worker are:

- employeeType
  name of employee type (e.g. customer support or developer)
- employeeColor
  the color of the agent's visual representation
- daysInOffice
  the number of days/week which the agent is working in the office, (e.g. customer support might come into office only once a week and be at customers' locations the remaining days of the week)
- timeInOffice
  the average duration the agent stays in the office, once it comes to the office
- startShiftMin
  the earliest time that the agent will come to work (if it comes to work), i.e. earliest time to start its shift
- startShiftMax
  the latest time that the agent will come to work (if it comes to work), i.e. latest time to start its shift
- fixedWP
  boolean parameter if the agent owns a fixed (exclusive) workplace or not

And its variables are:

- assignedWP
  ID[1] of the workplace currently assigned; if fixedWP is *true* this ID is constant throughout simulation
- sickLeaveDays
  counter of days the agent was on sick leave
- spentVacation
  counter of vacation-days consumed
- sicknessDuration
  used to store the duration of the sickness if the agent turns sick

[1] Remark: pointer to object instance.

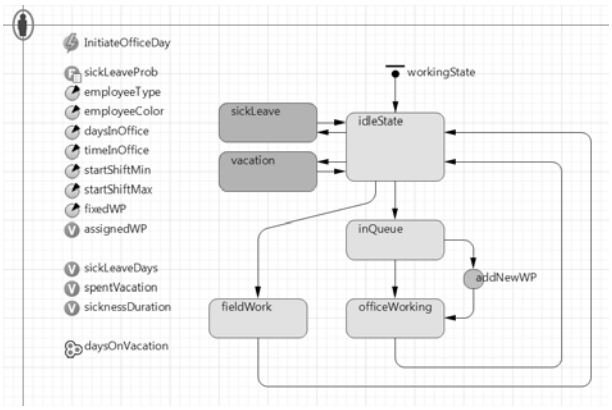

Figure 2: Structure of Worker object in the model, including parameters, variables, statechart and control-elements.

Besides the statechart and the above variables and parameters the Worker-agent is composed of three more control elements that are used to control the agent's behavior. These are *daysOnVacation* (list), *sickLeaveProb* (table function) and *InitiateOfficeDay* (dynamic event) and will be looked at in more detail later.

As mentioned the statechart used combines the working state and the health state of the agent. The reason for the combination is the assumption of their mutually exclusive nature. I.e. that an employee turning sick is not going to go to work. For simplification of the model it is further assumed that an employee is not turning sick during vacation or while at work. The top arrow indicates that the agent enters the statechart into state "idle" (*idleState*). This is the initial state from which every work day is started. From here the agent starts its working day either as an office day (via *inQueue* and *officeWorking*) or working outside the office (*fieldWork*) before it returns to the idle state. In case that the agent turns sick or takes a day off it changes from idle into state *sickLeave* and *vacation* respectively. The last remaining state, *addNewWP*, will be described later.

At initialization of the simulation each agent's *daysOnVacation*-list is filled with 25 days[2] on which the agent is on vacation. These 25 dates are scheduled randomly with following constraints:

- For 50% of all employees a blocked vacation is being scheduled, ...
  - with a length that is uniformly distributed between 10 and 15 days, and
  - which starts randomly (uniformly distributed) within a user-specified "core vacation period" (e.g. summer holidays).
- For 40% of the remaining employees (20% of total) a blocked vacation (of uniformly distributed length between 10 and 15 days) is scheduled at a random time (unif. distrib.) during the year.
- Finally each employees vacation list is filled up (until 25 days are reached) with random (unif. distrib.) days.

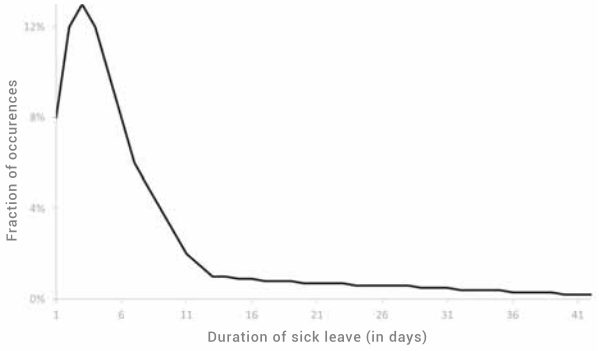

Figure 3: Skewed bell-shaped distribution of sick-leave duration used within the office model: length of sick leave duration (x-axis) as fraction of total occurences (y-axis) of sick leave.

To create a closer to reality behavior of employees a sickness function has been developed. Based upon several data sets from "Statistik Austria" (the Austrian Statistical Central Office) a distribution of the duration of sick leaves has been developed (see Fig. 3). This distribution is naturally only a rough estimate, but sickness duration and frequency strongly depends on the business field and the region/country and can thus not be modeled precise and generally valid at the same time. Instead it is recommended to use real-world data for parametrization and model fitting.

2.3 Employee Structure

With the basic employee being defined in the above described, flexible way, one object class (Worker) can be

[2] By Austrian law employees are entitled to an annual vacation of five weeks. Hence people working five days per week (the majority) receive 25 days of vacation (per year). People working 6 days per week subsequently receive 30 days.

used to represent different employee types. At initialization of a simulation run the user is presented with a GUI (see Fig. 4) that allows to define the numbers of employees per employee type, the core vacation period as well as choosing between flexible and fixed (i.e. individual, not shared) workplaces.

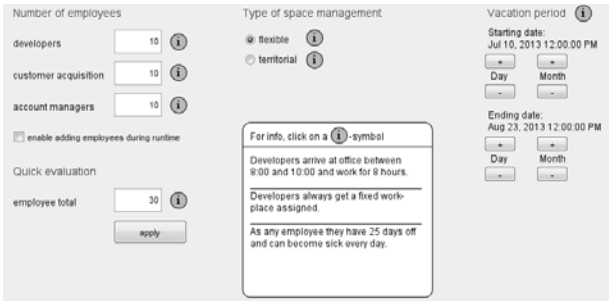

Figure 4: Section of GUI that allows specification of main simulation-parameters.

The model then generates the specified number of employees per employee type (in this case three types are hard-coded) together with their specific characteristics (see parameters, listed before).

An alternative possibility is to control the employee structure using spreadsheet-files, which are loaded by the model at initialization. This approach maximizes flexibility and empowers users without programming skills or access to the source code to precisely control the employee structure. In this case the model processes the spreadsheet-file row by row and creates an employee type (with multiple instances) for each of these. The exemplary spreadsheet depicted in Fig. 5 leads to the creation of three different employee types: "Developers", "Acquisition" and "Forenoon", with respective behavior.

	A	B	C	D	E	F	G	H	I
1	Employee statistics								
2									
3				arrival at office			agent color		
4	type	No. of employees	days in office	earliest	latest	time in office	red	green	blue
5			days/week	hour	hour	hours	0...255	0...255	0...255
6	Developers	30	5	8,00	10,00	7,00	255	165	0
7	Acquisition	30	1	8,00	10,00	2,00	245	20	147
8	Forenoon	40	5	7,00	17,00	2,00	84	84	84
9									

Figure 5: Spreadsheet file controlling model-internal employee structure.

With this approach it is even possible to create an individual class for every single employee and thus incorporate individual behavior (e.g. by adding preferred vacation times and individual, age- or gender-dependent illness probabilities). Besides potential privacy concerns this naturally requires to have the corresponding data in the first place. In relation to the benefit for the simulation result this approach is most likely too costly and thus not reasonable to pursue.

2.4 System behavior

In the current implementation the general goal is to use the model to calculate the number of required workplaces – for a given employee structure – and in this particular case to evaluate the savings potential compared to fixed workplaces[3]. With the employee structure and behavior in place, the next step is to model the general system behavior.

At initiation the model creates all employees as instances of class Worker, which, when coming into office require a workplace. Subsequently the number of required workplaces can be obtained by adding a workplace to the (virtual) building each time one is needed. This approach is supported by the object oriented architecture of the simulation environment. It is possible to create a new instance of a class at runtime. Thus offices and workplaces are implemented as classes (*Office1WP* office with one workplace, *Office2WP* with two workplaces and *Workplace*). When a *Worker* changes its state from idle to *inQueue* a check for free workplaces is performed. If a workplace is available the employee uses it, if not a routine is called which creates an additional instance of *Workplace* and assigns it to the employee. Remark: Since employees with *fixedWP* = *true* do not release their workplace (ID), it also cannot be taken by other employees.

The structure of class *Workplace* is a fairly simple one. Besides information relevant for visualization purposes (e.g. its coordinates) it has following three variables:

- assignedEmployee
  analogous to *Worker*'s *assignedWP* this holds the ID of the employee that is assigned to the workplace; in case of a fixed workplace the ID[4] does not change

- nTimesUsed
  counter for utilization analysis which registers ever use-session

[3] I.e. a workplace model in which every employee has an individual workplace that is not shared

[4] Remark: pointer to object instance.

- nMinUsed
  counter for utilization analysis which registers every minute of usage

As indicated the model has two more object classes: *Office1WP* and *Office2WP*. They represent the frame in which workplaces are set. Depending on the 1WP or 2WP the object houses one or two workplaces. The primary (and sole) reason for their existence is the visualization of the simulation.

During simulation a cyclic event *wakeAgents*, scheduled for 00:01 of each simulated day, triggers the actions of all *Worker*-instances. It schedules their dynamic events *InitiateOfficeDay*, depending on probabilities and other factors. First it checks whether the current day is a workday or not. If it is, it checks how many days per week the respective agent is working "in office" and probabilistically determines whether a day in office or a day of field work is scheduled.

The exact point in time when the (employee-internal) dynamic event *InitiateOfficeDay* is taking place is scheduled with a uniform distribution between the agent's *startShiftMin* and *startShiftMax*. When this point of time is reached, this internal dynamic event performs a state-check on the agent (if healthy and not on vacation) and then (probabilistically) determines whether the agent becomes ill or not — in which case it proceeds to work in office.

3 Simulation and Findings

3.1 Parametrization

Without real-world data to derive an employee structure from and with no benefit of a super-realistic one, a simplified employee structure was used to evaluate the savings potential of a flexible space management compared to a fixed workplace model. Nevertheless expert-knowledge was used to obtain a close-to-real employee structure and behavior.

The structure consists of developers, employees in customer acquisition and account managers, of which only the first are granted a fixed workplace. This is explained by their respective "in office" working times. Developers come to office (if not ill or on vacation) on all workdays, arrive in office between 8:00 and 10:00 (a.m.) and work for 8 hours. Customer acquisition personnel spend four days per week with field work and hence only have one day in office. When they are in office they, as developers, come between 8:00 and 10:00 and work for 8 hours. Account managers are doing mainly field work, but come into office daily, although at irregular and changing times. They arrive between 8:00 and 17:00 and are then in office for two hours.

3.2 Results

For evaluation of the savings effect through flexible workplace utilization a balanced structure with $1/3$ of every employee type was used. Flexible workplace utilization was defined in such a way that all employees (except developers) use any free workplace (except those that are assigned to a developer); developers always use their assigned workplace.

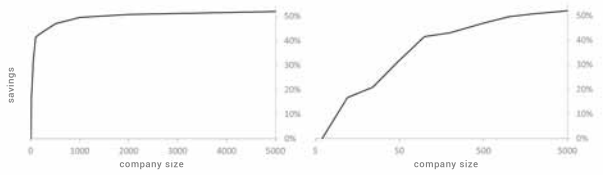

Figure 6: Workplace savings potential (y-axis) as a function of company size (x-axis) – plotted on a linear (left) and a logarithmic scale (right).

The model was used to simulate the workplace requirements of each company size over one year (365 days). To compensate the effect of outliers the Monte Carlo method was applied. For every company size 10 simulation runs (each with a random seed for random number generation) were produced and averaged. The savings potential is then calculated as the difference between the simulation average and the company size[5]. The results are shown in Fig. 6, once on a linear (in figure left) and once on a logarithmic scale (in figure right). It is obvious that small enterprises can draw no and medium-sized ones only limited benefit of a flexible use of workplaces. Large companies on the other hand can cut more than 50% of their workplaces, compared to a fixed (workplace utilization) model!

Arguably averaging of simulation results leads to a lower number of workplaces than required in the "worst case" scenario. But then again flexible use of workplaces always holds a theoretic danger of shortage: in the absolute "worst case" all employees require a workplace at the same time. The question that has to be answered in practice is: how much risk does one want to

[5] As the number of employees equals the number of workplaces required if every employee has their individual workplace.

take? Depending on the answer it is necessary to plan with a sufficiently large buffer. In addition the difference between mean and maximum is very small (see tab. 3.2), which is explained by the (fairly long) runtime of the basic simulation (365 days). The odd employee numbers are explained by the employee structure, which consists of three equally large groups.

Company size	average	maximum	difference
6	6	6	0
12	10	10	0
24	18	19	1
28	32	33	1
99	58	58	0
198	110	113	3
501	260	266	6
1002	501	506	5
2004	981	988	7
5001	2395	2405	10

Table 1: Simulation results for required workplaces: average and maximum (of 10 simulation runs) and difference.

Using the variable *nMinUsed* of the *Workplace*-object it is possible to calculate the effective occupancy — either for every workplace, or for the whole lot. In doing so one must consider that the employees are modeled in such a way that they do not leave their workplace until they finish their workday. I.e. there are no meetings, conferences, lunch-breaks, and the like — which would naturally reduce effective occupancy. Incorporation of such elements would require to consider whether an unoccupied workplace left for such a reason would become available (for use by another employee) or remain reserved although unoccupied by the initial employee. The present implementation has been chosen in order to avoid this problem.

Occupancy has been calculated in two different ways. Once the total time of workplace-usage is divided by the total simulation time (i.e. 24 hours, 7 days a week), the second time it is seen as fraction of the core time (10 hours per business day, i.e. Monday through Friday). With respect to the findings in Fig. 6 a company with 500 employees can already profit significantly of a flexible workplace management. Thus occupancy has been analyzed for this category by simulating a period of one year for 5 times with the flexible workplace utilization as previously described. I.e. developers, who are working full time, have a fixed place, the remaining employees not.

The simulation results (depicted in Table 3.2, labeled “fixed”) show that the total occupancy (labeled “total”) lies at about 19.7% and during core time (labeled “core”) around 66.3%. In a second step the flexibility of the space management has been increased by one notch: developers also use flexible workplaces (results labeled “flexible” in Table 3.2). Even though developers are working full time (i.e. 8 hours per day, 5 days per week) the impact that this change has is dramatic. Only by unblocking the (previously fixed) workplaces blocked during vacation and illness of developers total utilization was increased by one percentage point (relative: 5%) and core utilization by 3.3 percentage points (relative: 5%). Again (compare to tab. 3.2) the deviation of the results is so small that the ranges of the two settings’ results never overlap. The explanation for this is again found in the long simulation period of 365 days, which causes graduation of a lot of random influences.

	total (%)		core (%)	
sim-run	fixed	flexible	fixed	flexible
1	19.88	20.87	66.87	70.19
2	19.51	20.11	65.61	67.66
3	19.63	20.87	66.02	70.21
4	20.03	21.14	67.38	71.10
5	19.29	20.48	64.89	68.87
average	19.70	20.69	66.28	69.61

Table 2: Comparison of workplace occupancy for scenarios “fixed” and “flexible”; simulation results for “total” occupancy and “core” time occupancy (10 hours/business day).

3.3 Conclusion

Calculation of required number of workplaces is a queuing theory problem such as establishing the number of checkout counters in a post office or supermarket and general dimensioning problems in the area of service provision. With the above described hybrid ap-

proach (combining AB methods and DES) it is possible to model workplace utilization at an employee-based level, which has several advantages.

One major advantage is the transparency of the model and hence of the simulation results. It allows the user for whom simulation is carried out to understand and follow the reasoning behind the model, without becoming a simulation-expert herself (e.g. statecharts are easily read and understood). A second is the closeness of the attribute-mapping between implementation and reality, which at the same time is partially responsible for the before mentioned. An employee turning ill is implemented as agent-internal state change from "healthy" to "ill".

Besides the easy interpretation of the implemented model, the object oriented approach also allows for a very flexible and easy modification and adaption. If additional requirements arise they can more often than not be incorporated in a very efficient way (e.g. fine-tuning of agents' behavior, adding of attributes to objects, introduction of additional statistics, etc.).

With the model being of academic nature, the focus of it lies on serving as a proof of concept. These results obtained are thus not carved in stone, as they are a product of employee structure and behavior as well as of the space management in place and of simulation parametrization. All of which was based not on real data but on assumptions, which, although chosen with a claim for authenticity, reflect the simplifications accepted to obtain a slender model.

Against the background of current practice in space management of office buildings the model results point at a huge potential for improvement. Exploitation of it requires – amongst others – raising awareness for the issue, which can be supported by such conceptual models. On the other hand the model already incorporates most of the aspects necessary to conduct analysis of real systems and only calls for appropriate parametrization. In case of potential extensions, the flexible (object oriented) architecture allows for very efficient adaption. Finally, the big question and challenge that remains is whether an institution is willing and capable of incorporating the required changes of business processes, i.e. installing a flexible space management. Without this step the potential for improvement is, as pointed out intially, more than limited.

References

[1] Ottomann H. Wussten Sie, dass Büroflächen nur zu fünf Prozent genutzt werden? Es ist Zeit! *Mensch & Büro*. 1994;2:28–30.

[2] Emrich Š, Wiegand D, Kovacs A. A Mathematical Simulation Tool for Increased Space Utilization Efficiency. In: *Proceedings of EFMC 2013 (Prague, May 2013)*. 2013; .

[3] Kovacs A, Štefan Emrich, Wiegand D. Strategies and Methods to improve Space-Utilisation within Companies. In: *Proceedings of EFMC 2013 (Prague, May 2013)*. 2013; .