Teaching Mathematical Modeling with Computing Technology: Presentation of a Course based on Evacuations

André Greubel
andre.greubel@uni-wuerzburg.de
University of Wuerzburg
Wuerzburg, Germany

Hans-Stefan Siller
hans-stefan.siller@mathematik.uni-wuerzburg.de
University of Wuerzburg
Wuerzburg, Germany

Stefan Ruzika
ruzika@mathematik.uni-kl.de
University of Kaiserslautern
Kaiserslautern, Germany

Lynn Knippertz
knippertz@mathematik.uni-kl.de
University of Kaiserslautern
Kaiserslautern, Germany

ABSTRACT

Mathematical modeling is considered a crucial skills, both in modern life and STEM education. Prior research has identified the relevance of working on *complex and authentic* modelings problems in education. However, up to this point, little of the courses proposed in this area explicitly focus on the role of comprehensive computing technology during mathematical modeling. We bridge this gap by presenting a design and ready-to-use technology for an interdisciplinary course that introduces students to mathematical modeling of complex systems with comprehensive technology. In the course, students are introduced to grid automatons as basic computing model. Furthermore, they can increase their knowledge of mathematical modeling and algorithmic thinking. In this paper, we develop a didactic structure for such a course and present educational technology developed to support this structure. The structure itself consists of three simulation environments and is based on the following problem: "How can we estimate the time it takes to evacuate our school (without an experiment)?". We describe the structure of the course and the simulation environments in more details and outline potential exercises for such a course.

CCS CONCEPTS

• **Computing methodologies** → *Simulation evaluation*; • **Applied computing** → *Interactive learning environments.*

KEYWORDS

STEM Problems, Computing Technology, Digital Simulations

ACM Reference Format:
André Greubel, Hans-Stefan Siller, Stefan Ruzika, and Lynn Knippertz. 2022. Teaching Mathematical Modeling with Computing Technology: Presentation of a Course based on Evacuations. In *Proceedings of the 17th Workshop in Primary and Secondary Computing Education (WiPSCE '22), October 31-November 2, 2022, Morschach, Switzerland*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3556787.3556802

1 INTRODUCTION

Mathematical problem solving and mathematical modeling are crucial skills for modern education [17]. A modern way to foster these skills are "complex and authentic modelling problems": real-life problems used as learning experience for multiple skills that are only little simplified for education (c.f. Section 2.1).

Because of the "necessity of treating authentic, complex modelling problems interdisciplinary" [19], working on these problems frequently requires activities from other STEM fields. Because of this, "[d]esigning authentic learning scenarios is therefore one of the key challenges in education interventions that aim for STEM literacy" [4]. This is especially true for technology, the T in STEM: "Usually, to get a solution, computer programmes (Excel or more sophisticated ones) must be applied" [19].

However, while several approaches for teaching with authentic and complex problems are published (c.f. [19, p. 291f.] for a list), none of the approaches in this list explicitly focus on the use of such "sophisticated" computing technologies during this processes. Furthermore, there is little research into how to introduce such comprehensive technology to students. This is unfortunate, as "there is a need to understand the implications of this technology for all aspects of classroom practice including that of mathematical applications and modelling." [10].

To start bridging this gap, the central goal of this paper is to develop and present a course designed to introduce learners to *core conceptualization of computing*, as well as the *application of comprehensive computing technology* while working on complex and authentic modeling problems. More precisely, we present material[1] for a course designed to introduce the students to:

(1) mathematical modeling with comprehensive computing technology based on an authentic and complex problem.
(2) cellular automatons as one important computing concept frequently used in modeling for complex problems.
(3) certain aspects of algorithmic thinking focused on the analysis and evaluation of algorithms. (c.f. Section 2.5).

Overall, our interdisciplinary course teaches students central aspects of two important topics: mathematical modeling and computing. The course itself is designed for students in higher secondary education (around years 8-10) and based on a complex real-world problem: Estimating the time it takes to evacuate a building.

[1]The development of the material was funded by *Deutsche Telekom Stiftung*.

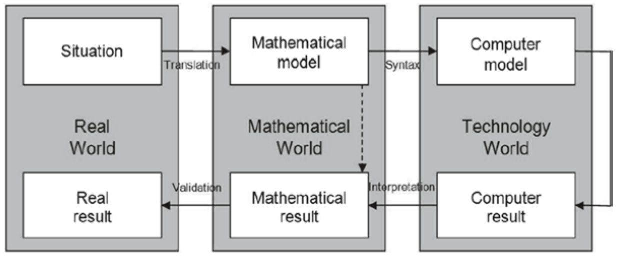


Figure 1: Technology Enhanced Modeling Process [33].

2 EDUCATIONAL BACKGROUND

2.1 Complex and Authentic Problems

Complex and authentic problems are considered to be of high importance for STEM education [19, 23].

In this regard, problems are considered *authentic*, if there is an "alignment of student learning experiences with the world for which they are being prepared" [26]. Such problems "shall articulate the relevance of mathematics in daily life, environment and sciences and impart competencies to apply mathematics in daily life, environment and sciences" [19]. With them, learners can "practice the skills and knowledge that are relevant and real to workplace situations and learn it at the same time" [16]. Furthermore, problems are characterized as *complex*, if they are real-world problems that are "only little simplified" [19], and if they require consistent work over a longer period of time (at least multiple hours). In such problems, learners "have to carry out the whole modelling cycle on their own, only supported by tutors or their teachers, sometimes they even have to carry out the modelling cycle several times, going back and forth quite often in so-called mini-cycles." [19].

Notably, these characteristics bring such problems "into sharp contrast to many other modelling activities usual at school". [19]. Most importantly, they do not focus on only one single skill or subject but are interdisciplinary in nature. As such, such problems are frequently included as parts of project weeks (c.f. [19, 31]). However, it should be noted that "[t]he strong plea of the students for the inclusion of these kinds of examples in usual mathematics lessons support our position that it is appropriate to include these kinds of problems in ordinary mathematics lessons." [19].

2.2 Modeling with Computing Technology

To solve complex and authentic problems, students frequently need to utilize *mathematical modeling*, a central goal of which is the promotion of modeling competencies, i.e., the ability and the volition to work out real-world problems with mathematical means [18, 25].

One established way to describe the inclusion of computing technology in mathematical modeling is the *technology enhanced modeling process*, as described by [33] and depicted in Figure 1.

This process focuses on the explicit translations between the mathematical and computer world, e.g., by implementing an algorithm with a programming language or using a simulation. Note that this explicit translation is not universally accepted as the only way to include computing technology into modeling tasks (c.f. [7, 13]) and different models are available (e.g., [11, 12]). However, it is well suited for our course as we also utilize a direct translation of results.

When using computing technology in mathematical modeling, there should be a *technological surplus value*. This means that the technology should not be included as a mean to itself. Instead, the key benefits of including the computing technology should be apparent to both students and teachers. In our course, this surplus value lies within the automated execution of the simulation algorithm: Given that the manual execution of scenarios with a grid automaton is very time-consuming (even execution of single rooms can easily take more than one hour), it is not only beneficial but necessary to introduce automation to work on authentic problems.

2.3 Cellular Automata as Computing Models

A cellular automaton is one of the oldest computing models, dating back to von Neumann [21]. It consists of cells with neighbours that change their state according to specified rules (the programming of the automaton). A sub-type of a cellular automata is a grid automaton which uses a two-dimensional grid of rectangular cells.

For complex real-world problems, cellular automata are frequently used as computing model. For example, they can be used while modeling climate change [2], urban growth [3], traffic flow [27], kinetic phenomena [22], or the spread of HIV viruses in cells [8].

In computer science education, they are a frequent object of analysis in theoretical computer science [1]. They are are also sometimes used as computing model, e.g., for teaching about parallel computing [35], to assess programming skills [34], or to introduce low-level programming [6]. However, more flexible register-machines are frequently used instead, often in combination with textual or visual high-level programming languages like Python or Scratch [20].

As such, while cellular automata are currently not widespread as computing model in computer science education, they are frequently used as computing model for modeling real-world problems, making them a sensible choice for interdisciplinary courses.

2.4 Evacuations with Grid Automatons

Based on prior research, we use building evacuation as context for our course. This context allows for interesting, real-world modeling problems, complex exercises, and meaningful inclusion of computing technology – while not being too reliant on sophisticated inner-mathematical methods or domain knowledge [14, 15, 30, 31].

To simulate building evacuations, grid automatons with agents are frequently used [24]. In the implementation of such an automaton, each cells can be either *empty*, *full*, *blocked*, or *safe*. While full cells contain (exactly one) agent, empty cells do not. Blocked cells neither do nor can contain an agent. They represent walls in the building. Finally, safe cells remove each agent moving on it from the simulation. They represent the destinations in a scenario.

During each *Simulation Step*, each agent can move to a neighbouring cell – either in four (*neumann neighbourhood*) or in eight directions (*moore neighbourhood*). Whether or how the agents move is described by the *fleeing algorithm*. A simple fleeing algorithm might instruct each agent to move to the cell next on the shortest path of unblocked cells to the nearest safe cell, if this cell is empty.

2.5 Computational and Algorithmic Thinking

Computational thinking is one of the cornerstones of computing education and can be defined as "the thought processes involved

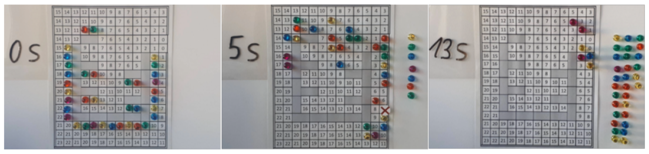


Figure 2: Excerpt of some of the photographed simulation steps (manual execution).

in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.” [36].

One important sub-skill of computational thinking is algorithmic thinking [32], defined as the ability to “to think in terms of sequences and rules as a way of solving problems” [5]. Algorithmic Thinking is based on a firm understanding of the concept of an algorithm and its key properties (like determinism, performance, correctness, and validity). It requires activities like stepping through algorithms step-by-step to work out what they do [5], “assessing that an [algorithm] is fit for purpose” [5], and “the ability to think about all possible special and normal cases of a problem” [9].

In our introductory course, algorithmic thinking activities focus on such evaluating activities and the introduction of key properties of algorithms – rather than the design of new algorithms. As object of analysis, the fleeing algorithm of the grid automaton is used.

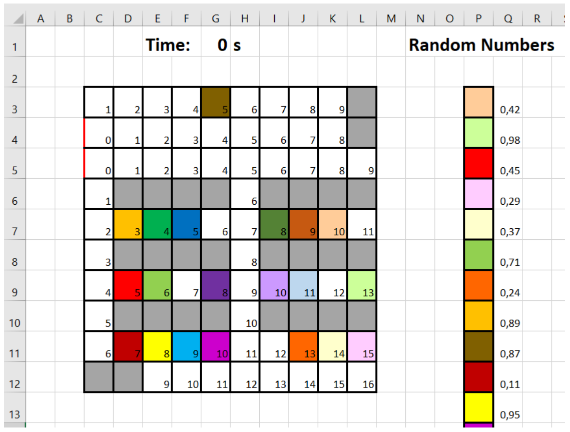
3 PRESENTATION OF THE SIMULATION ENVIRONMENTS

In this section, we introduce high-level idea of our course which is based on a real-world problem: Estimating the time it takes to evacuate a building – like a school – without a real-world experiment. It utilizes one analog and two digital simulation environments developed for this course that consecutively get more comprehensive.

3.1 First Learning Environment: Manual Execution of a Simulation on Paper

The first environment utilizes a playful, hands-on approach. A grid template with objects depicted on the cell structure is made out of paper. Agents are placed on the grid in the form of playing figures. All figures are labeled for a proper distinction between each other.

3.1.1 Execution of the Simulation. To execute a simulation step, agents are moved in a random order within a selected neighborhood such that they shorten their distance to the nearest goal. In doing so, they can only occupy empty cells. If none exists, the agent chooses not to move. If a goal is reached, the agent is removed from the field. After every agent had the chance to move, the simulation step is completed and the next one begins. The simulation stops once the last agent reached a goal. In order to determine the agent that may move, a picture (or label) of every agent can be depicted on a playing card. Shuffling these cards and drawing them one after another enables an intuitive, lightweight use of randomness. After completion of a simulation step, the cellular space with the agents is photographed. Such images, exemplary shown in in Figure 2, can then be played as a stop-motion film sequence.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1					Time:		0 s							Random Numbers				
2																		
3			1	2	3	4	5	6	7	8	9	▨				■	0,42	
4			0	1	2	3	4	5	6	7	8	▨				■	0,98	
5			0	1	2	3	4	5	6	7	8	9				■	0,45	
6			1	▨	▨	▨	▨	6	▨	▨	▨	▨				■	0,29	
7			2	3	4	5	6	7	8	9	10	11				■	0,37	
8			3	▨	▨	▨	▨	8	▨	▨	▨	▨				■	0,71	
9			4	5	6	7	8	9	10	11	12	13				■	0,24	
10			5	▨	▨	▨	▨	10	▨	▨	▨	▨				■	0,89	
11			6	7	8	9	10	11	12	13	14	15				■	0,87	
12			▨	▨	9	10	11	12	13	14	15	16				■	0,11	
13																■	0,95	

Figure 3: Example of a scenario in the second environment (Excel Screenshot).

3.1.2 Didactic Goals of this Environment. This first environment introduces students the concept and basic idea of simulating room evacuations with a grid automaton. Learners get a first, hands-on experience with the domain, approach, and the computing model. The environment clearly pictures the underlying grid automaton and their counterparts of real-world entities (persons, walls, exits) in the simulation. Furthermore, the different time steps (single move and a full movement of all entities) are associated with clear events (drawing a card or finishing the deck of cards). Simple research questions such as "What influence do different numbers of people have on evacuation time?" can already be answered within this learning environment, This visual and enactive approach leads to an easy memorable model that is especially suitable for young learners.

3.1.3 Limitations of this Environment. On the contrary, the movements of the agents require time which limits the number of agents and the size of the rooms to be evacuated. Because of this, the scope of possible exercises is limited: It is not yet possible to study extensive, real-world evacuation scenarios, such as the evacuation of an entire building. Furthermore, the environment involves a material cost and preparation time, as game figures and associated cards have to be purchased, the cellular automaton has to be created on paper and cards, pens and glue dots are needed to mark the figures.

3.2 Second Environment: Semi-Automated Simulation with Excel

The second environment is implemented as an Excel file and introduces automation to the simulation process. The students first need to implement the automaton into this Excel file by setting the background color of cells. This Excel file is seemingly empty: The only content visible at first are step-by-step instructions to create the evacuation scenario: Walls can be filled in with black background, different background colors can be used for agents. Additionally, cells can be marked as safe by filling them with a zero. If at least one cell contains a zero, the sheet automatically updates: Each cell denotes the distance to the nearest safe cell. This information is inserted by a VBA script.

3.2.1 Execution of the Simulation. This script also contains functions to semi-automatically execute the simulation. All functions available are assigned to a key combination. Once this key combination is pressed in the Excel sheet, the corresponding function is called. The most important functions are: 1) assigning a random number to each person, 2) letting the person with the lowest random number move one step, 3) let all persons move one step. These functions can be used to decrease the amount of manual labor and, as such, increase the possible scope of exercises.

Similarly to the first environment, images of the steps can be collected and played as a stop-motion movie, e.g. by using a Screenshot-Tool and PowerPoint.

3.2.2 Didactic Goals of this Environment. The second environment utilizes automation to reduce the overhead of the manual execution method. Hence, it is now possible to shift the focus from learning about the simulation towards working with technology and solving more complex problems with it. For example, it is now possible to build more sophisticated scenarios and analyze aspects of the results that depend on executing the simulation multiple times.

Still, its technological overhead remains very limited. Most importantly, every automation steps corresponds to exactly one step previously executed manually. This highlights that all results are an inherent property of the algorithm - rather than the method of execution.

3.2.3 Limitations of this Environment. As Excel was not specifically designed to support such tasks, it cannot solve some of the problems of the first environment. In particular, some aspects, such as running multiple, similar scenarios, still require a huge manual overhead. Additionally, the scenario can be implemented only in an area specifically designed for it. While the size of this area can be configured by the creator of the Excel file, it cannot be changed afterwards. As such, switching to more and more complex scenarios is unpractical as it requires the preparation of files enabling different scenario sizes. Furthermore, some aspects of the simulation, like the neighbourhood-function, cannot be changed easily in this environment. Both aspects still limit the possible scope of the scenarios and thus also the possible problems to be investigated.

3.2.4 Access to additional material. All Excel files, alongside a detailed description, are available at https://dbtools.mathematik.uni-kl.de/evakuierung/index.html. The material is freely accessible. Files for different building sizes are available.

3.3 Third Environment: Automated Simulation with a comprehensive Web Application

The third environment is implemented as comprehensive, customizable web applications. This significantly increases both the complexity of the environment, as well as the possible scope of exercises.

The WebApp provides a scenario editor that can be used to build scenarios by inserting different kind of tiles (e.g., containing persons, walls, or safe zones) into the scenario. Furthermore, scenarios can also be stored and loaded as a base-64 text, enabling easy transmission of scenarios between students and learning places, e.g., via e-mail or chat messenger. Multiple example scenarios (that are used in the course) are also directly available from within the environment via the press of a button.

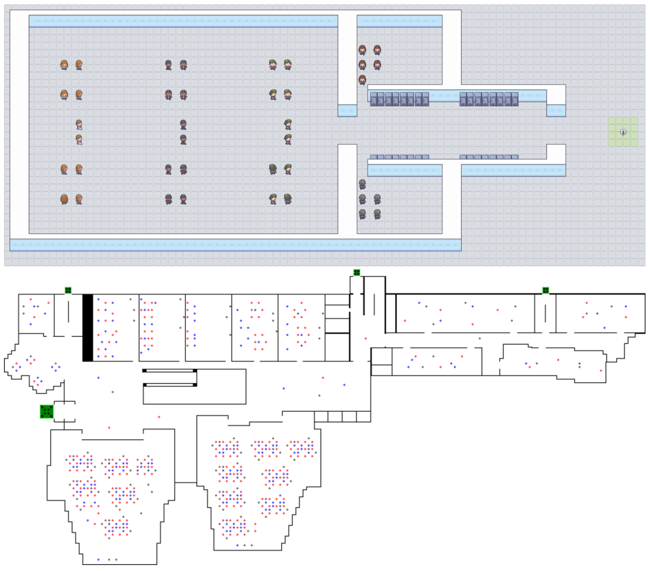


Figure 4: Two scenarios with different sizes and aesthetics in the third environment (web app screenshots).

3.3.1 Execution of the Simulation. Before the execution, the fleeing algorithm used for the execution, the neighbourhood function, and the graphical design can be customized. After starting the execution of a scenario with a click on a button, the scenario is executed automatically. Then, central statistical information about the simulation (necessary steps, total movements, and used configuration) are shown. Additionally, it is still possible to visualize the execution of the simulation step-by-step or movement-by-movement by using buttons to navigate the current state of the simulation (e.g., one movement ahead, or one step ahead). It is also possible to highlight the path to a safe cell a person took by hovering over that person.

3.3.2 Didactic Goals of this Environment. The third environment removes the last overhead that remained in the second environment, offers a vastly extended functionality, and has many quality-of-life features for smooth working. It enables the rapid execution of different scenarios and configurations: The execution of the large university building on the bottom of Figure 4 (consisting of 180×86 tiles and 600 persons with a total of over 20.000 individual movements) takes around $10s$ ($\pm$ 10s) on typical school laptops. This reduction in manual labor enables sophisticated exercises, as well as a meaningful analysis and comparison of the mathematical model and the algorithms used in the simulations.

3.3.3 Limitations of this Environment. However, while powerful, the main focus of this environment is to appeal to non-specialized learners interested in comprehensive mathematical modeling.

First, it is not optimized to achieve extremely accurate results. Instead, it provides a reasonable result and performance only for up to middle-sized buildings, like schools with around 100 classrooms. For scientifically valid results, or bigger buildings and building complexes, professional software is necessary.

Second, the standard choices of some parameters are suitable for common, but not all, scenarios. In some edge cases, they need to be understood and customized, requiring a deeper understanding of the environment in order to achieve good results.

Third, it is not (yet) possible to fully customize the environment. All options have to be selected from a list and no custom fleeing algorithms can be implemented (yet). It is also not possible to simulate individual behavior of agents like accidents or stubbornness.

3.3.4 Additional Material. The simulation itself, alongside a detailed description of the environment (including user handbooks, and technical details like the available algorithms and their most important properties) is freely available at https://evadid.it/eva2.

4 PHASES FOR THE IMPLEMENTATION OF A COURSE

In this section, we describe multiple phases that can be used to guide working during the course. The phases build on each other and either introduce skills necessary for the following phases, or work towards one of the course goals (c.f. Section 1). Thus, all phases form a structure that accounts for all necessary learning requirements and achieves all goals of the course. For each phase, we will outline the central goal, why it was chosen, and the activities done by the students to achieve this goal.

4.1 Phase 1: Introducing Preliminaries

The central goal of the first phase is the introduction of the grid automaton as computing concept and the modeling process (second goal). As such, this phase introduces the preliminaries for the following phases. This explicit phase is reasonable as neither the modeling process, the grid automaton, nor the execution of the simulation are straight-forward to understand. Thus, they should be taught explicitly to enable a low floor by reducing cognitive load (c.f. Plass et al. [28], Resnick et al. [29]).

Student activity at this point focuses on the comprehension of the model, rather than the quality of solutions. Thus, exercises should motivate students and require the proposed model and the modeling process to be solved. A suitable leading question can be stated as "How long does it take to evacuate the classroom we are sitting in?" This question has two main benefits. First, it is unlikely that students can solve this problem without either a model and a formalized modeling process, or an experiment. Second, the question incorporates the surroundings of the students and, as such, is likely to be meaningful and motivating to them.

4.2 Phase 2: Introducing Automation

Once students understand the model and modeling process, more complex problems can be stated. For example, it is possible to extend the question not only to the room, but the building the teaching takes place. This naturally leads to the problem of scalability, as the manual execution is very time consuming. The second phase tackles this problem by demonstrating how computing technology can be included sensibly into mathematical modeling. As such, this phase primarily focuses on the first central goal of the course.

To do so, one should first introduce the core activity of the technology enhanced modeling process: the translation between mathematical and computer world. Switching to the second environment resolves the scalability problem without introducing additional overhead, as model and algorithm are already known.

While demonstrating this translation, it is important to note that the results are dependent only on the mathematical model, rather than the method of execution. This can be verified by comparing the execution of both environments step-by-step. Understanding this aspect is relevant to recognize that modeling with and without computing technology are not fundamentally different things – but rather alternatives that have to be balanced depending on the scope and resources of the project.

4.3 Phase 3: Optimizing the Modeling Quality

This third phase increases the knowledge of mathematical modeling in complex problems (first goal). At this point, the quality of the answer (rather than the process) becomes the main focus of the course. More precisely, students should be made aware about what properties of the modeling process are relevant for a good solution. For example, results can be presented as number, as range, or as distribution and be interpreted as exact result, good estimate, rough approximation, or order of magnitude.

Exercises can be phrased as "what-if"-scenarios that encourage execution and comparison of multiple, similar scenarios. Thus, questions might include "What if we want to evacuate the whole building?", "What if the random number generator chose different people to move?", "what if there were double as many people in a certain room?", "What if the doors were smaller?", ...

4.4 Phase 4: Optimizing the Model Quality

In this phase, the focus remains on the quality of the answers to exercises (first goal). However, in this phase, a different sub-question is central: What properties of the model and algorithms are relevant for a good solution? Specially crafted exercises can be used to highlight edge cases or unrealistic behavior of the basic model.

For example, movement takes unrealistically long in diagonal hallways, if one uses the Neumann neighbourhood. Furthermore, it would be more realistic to let agents switch to another safe destination if their way to the nearest one is congested. Notably, this behavior is only shown in certain algorithms (e.g., `multiple goals` but not `closest goal`).

For such tasks, it is beneficial to switch to the third environment to account for these results and improve the quality of the modeling results by using different configurations.

4.5 Phase 5: The need for Algorithm Evaluation

This phase should demonstrate the necessity to evaluate algorithms in more details (third goal). This necessity can be shown by demonstrating the impact of the configuration changes on the simulation result. For example, allowing for diagonal movement lowers the simulation steps in the left scenario depicted in Figure 4 by 50%. Using a different fleeing algorithm raises them by 50%.

Exercises can include the deliberate construction of edge-case scenarios that lead to results that deviate from reality as far as possible. Furthermore, students can be asked to identify properties the scenario must fulfill for suitable results of the algorithms. Additionally, it is possible to compare the quality of results with real-world evacuations – e.g., with data collected from regular training evacuations from the school.

4.6 Optional Phase 6: Evaluating and Improving

Optionally, one can then continue to explicitly show not only why an evaluation is necessary, but also how such an evaluation would be performed. To do so, one needs to introduce additional techniques, such as statistical or in-depth analysis of algorithms. Furthermore, students can be asked to design novel algorithms that perform well for the concrete buildings they are working with. Notably, depending on the underlying computer science and mathematics curricula, these techniques might not be known to the students at that point. However, such exercises demonstrate that this context enables a high ceiling (c.f. [29]) for such a course both for teaching mathematical modeling and computational thinking.

5 CONCLUSION

In this paper, we presented a course to introduce modeling with computing technology to students. It is based on a motivating real-world problem (building evacuation) that has been verified to be interesting in prior research. In the course, students work with three different simulation environments – two of which utilize freely-available and ready-to-use digital simulations. These environments are utilized in different phases of the course that introduce the desired knowledge step-by-step, frequently in an enactive way.

While working with these environment and the problems presented in the course, students get first-hand experience with grid automatons as an example of an important computing model for simulating scenarios. They get first experiences with algorithms and their key properties and learn about the inclusion of computing technology in problem-solving. Furthermore, they can improve their modeling competences and learn about both the need and basic approaches to verifying algorithms during modeling.

The next important step for this project is an empirical verification of the course quality. This includes design-based research (into the quality of material like handbooks for the environments), qualitative analysis (into the working behavior of students), and quantitative verification (of learning outcomes). However, the design and ready-to-use technology presented in this practice paper can already be used by practitioners in classes.

Overall, we hope that this papers motivates practitioners to use our design and technology in modeling classes and lays the groundwork for empirical research into the role of computing technology in problem solving with authentic and complex problems.

REFERENCES

[1] Michal Armoni, Susan Rodger, Moshe Vardi, and Rakesh Verma. 2006. automata theory: its relevance to computer science students and course contents. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education.*

[2] PD Carey. 1996. DISPERSE: a cellular automaton for predicting the distribution of species in a changed climate. *Global Ecology and Biogeography Letters* (1996).

[3] Yimin Chen, Xia Li, Xiaoping Liu, Hu Huang, and Shifa Ma. 2019. Simulating urban growth boundaries using a patch-based cellular automaton with economic and ecological constraints. *International Journal of Geographical Information Science* 33, 1 (2019), 55–80.

[4] Lucian Ciolan and Laura Elena Ciolan. 2014. Two perspectives, same reality? How authentic is learning for students and for their teachers. *Procedia-Social and Behavioral Sciences* 142 (2014), 24–28.

[5] Andrew Csizmadia, Paul Curzon, Mark Dorling, Simon Humphreys, Thomas Ng, Cynthia Selby, and John Woollard. 2015. Computational thinking-A guide for teachers. (2015).

[6] Gabriele Di Stefano and Alfredo Navarra. 2012. Scintillae: How to Approach Computing Systems by Means of Cellular Automata. In *Cellular Automata.*

[7] Helen M Doerr, Jonas B Ärlebäck, and Morten Misfeldt. 2017. Representations of modelling in mathematics education. In *Mathematical modelling and applications.*

[8] Rita Maria Zorzenon Dos Santos and Sérgio Coutinho. 2001. Dynamics of HIV infection: A cellular automata approach. *Physical review letters* 87, 16 (2001).

[9] Gerald Futschek. 2006. Algorithmic thinking: the key for understanding computer science. In *International conference on informatics in secondary schools-evolution and perspectives.* Springer, 159–168.

[10] Vince Geiger. 2011. Factors affecting teachers' adoption of innovative practices with technology and mathematical modelling. *Trends in teaching and learning of mathematical modelling* (2011), 305–314.

[11] Gilbert Greefrath, Corinna Hertleif, and Hans-Stefan Siller. 2018. Mathematical modelling with digital tools—a quantitative study on mathematising with dynamic geometry software. *ZDM* 50, 1 (2018), 233–244.

[12] Gilbert Greefrath, Hans-Stefan Siller, and Jens Weitendorf. 2011. Modelling considering the influence of technology. *Trends in teaching and learning of mathematical modelling* (2011), 315–329.

[13] André Greubel and Hans-Stefan Siller. 2022. Learning about black-boxes: A mathematical-technological model. In *Twelfth Congress of the European Society for Research in Mathematics Education (CERME12).*

[14] Andre Greubel, Hans-Stefan Siller, and Martin Hennecke. 2020. Teaching Simulation Literacy with Evacuations. In *European Conference on Technology Enhanced Learning.* Springer, 200–214.

[15] Andre Greubel, Hans-Stefan Siller, and Martin Hennecke. 2021. EvaWeb: A Web App for Simulating the Evacuation of Buildings with a Grid Automaton. *European Conference on Technology Enhanced Learning* (2021).

[16] Lam Bick Har. 2013. Authentic learning. *The Active Classroom The Hong Kong Institute of Education* (2013).

[17] Hyewon Jang. 2016. Identifying 21st century STEM competencies using workplace data. *Journal of science education and technology* 25, 2 (2016), 284–301.

[18] Gabriele Kaiser. 2014. *Mathematical Modelling and Applications in Education.* Springer Netherlands, Dordrecht, 396–404.

[19] Gabriele Kaiser, Martin Bracke, Simone Göttlich, and Christine Kaland. 2013. *Authentic Complex Modelling Problems in Mathematics Education.* Springer International Publishing, Cham, 287–297. https://doi.org/10.1007/978-3-319-02270-3_29

[20] Kanika, Shampa Chakraverty, and Pinaki Chakraborty. 2020. Tools and techniques for teaching computer programming: A review. *Journal of Educational Technology Systems* 49, 2 (2020), 170–198.

[21] Jarkko Kari. 2005. Theory of cellular automata: A survey. *Theoretical computer science* 334, 1-3 (2005), 3–33.

[22] Lemont B Kier, Paul G Seybold, and Chao-Kun Cheng. 2005. *Modeling chemical systems using cellular automata.* Springer Science & Business Media.

[23] Richard Lesh and Guershon Harel. 2003. Problem solving, modeling, and local conceptual development. *Mathematical thinking and learning* (2003).

[24] Yang Li, Maoyin Chen, Zhan Dou, Xiaoping Zheng, Yuan Cheng, and Ahmed Mebarki. 2019. A review of cellular automata models for crowd evacuation. *Physica A: Statistical Mechanics and its Applications* 526 (2019), 120752.

[25] Katja Maaß. 2006. What are modelling competencies? *ZDM* 38, 2 (2006), 113–142.

[26] Anthony D McKenzie, Christopher K Morgan, Kerry W Cochrane, Geoff K Watson, and David W Roberts. 2002. Authentic learning. In *Proceedings of the 25th HERDSA Annual Conference.* Citeseer, 426–433.

[27] Kai Nagel and Michael Schreckenberg. 1992. A cellular automaton model for freeway traffic. *Journal de physique I* 2, 12 (1992), 2221–2229.

[28] Jan L Plass, Roxana Moreno, and Roland Brünken. 2010. *Cognitive load theory.* Cambridge university press.

[29] Mitchel Resnick, Brad Myers, Kumiyo Nakakoji, Ben Shneiderman, Randy Pausch, Ted Selker, and Mike Eisenberg. 2005. Design principles for tools to support creative thinking. (2005).

[30] Stefan Ruzika, Lynn Knippertz, and Eva Rexigel. 2019. *Simulation von Evakuierungen auf Grundlage zellulärer Automaten.* Technical Report. Uni. of Kaiserslautern.

[31] Stefan Ruzika, Hans-Stefan Siller, and Martin Bracke. 2017. Evakuierungsszenarien in Modellierungswochen. In *Neue Materialien für einen realitätsbezogenen Mathematikunterricht 3.* Springer, 181–190.

[32] Cynthia Selby and John Woollard. 2013. Computational thinking: the developing definition. (2013).

[33] Hans-Stefan Siller and Gilbert Greefrath. 2010. Mathematical modelling in class regarding to technology. In *Proceedings of the sixth congress of the European Society for Research in Mathematics Education.* 2136–2145.

[34] Thomas Staubitz, Ralf Teusner, Christoph Meinel, and Nishanth Prakash. 2016. Cellular Automata as basis for programming exercises in a MOOC on Test Driven Development. In *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE).* https://doi.org/10.1109/TALE.2016.7851824

[35] Antonio J. Tomeu Hardasmal and Alberto G. Salguero. 2020. Teaching Parallelism With Gamification in Cellular Automaton Environments. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje* 15, 1 (2020), 34–42.

[36] Jeanette Wing. 2011. Research notebook: Computational thinking—What and why. *The link magazine* 6 (2011), 20–23.