A Model Factory in Augmented Reality as an Eye-Catcher at Exhibitions and Fairs

Torsten Munkelt[1*], Steven Behne[1], Markus Wacker[1], Sven Völker[2]

[1]Dresden University of Applied Sciences, Friedrich-List-Platz 1, 01069 Dresden, Germany;
Torsten.Munkelt@htw-dresden.de

[2]Ulm University of Applied Sciences, Prittwitzstraße 10, 89075 Ulm, Germany

SNE 27(1), 2017, 1 - 8, DOI: 10.11128/sne.27.tn.10361
Received: February 5, 2017 (Selected ASIM STS 2016 Postconf. Publ.), Accepted: February 20, 2017
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Our universities required an eye-catcher at exhibitions, conferences, open houses etc. We decided in favor of a model factory in augmented reality demonstrating a queueing system, material flow, and aspects of pro-duction planning and control. When visitors look at an empty table 'through' their personal mobile devices, they can watch a factory producing on the table. The visitors can alter control variables and watch the factory change its behavior accordingly. We explain the underlying simulation model and its configuration, the composition and functionality of the mobile app for augmented reality, and the communication via database. Furthermore, we present a solution for the automatic generation of the model factory's layout, and we explicate how to eliminate the blurring of the model factory in augmented reality. Finally, we share experiences and user feedback from first exhibitions of the model factory and outline our plans for its future de-velopment.

Introduction

At industrial trade fairs, recruitment events and similar occasions, exhibitors compete for visitors' attention. Most exhibitors try to contact as many visitors as possible because the overall number of contacts correlates with the number of leads. To initiate contact, an eye-catcher is needed which attracts visitors to the booth. Posters, on-screen presentations and even video clips do not serve this purpose well, since these types of media are omnipresent at fairs. The ideal eye-catcher should be physical, moving, and sending optical and acoustic signals (e.g. it should blink and beep), thus, appealing to multiple senses. According to experience, the eye-catcher often does not relate to the products or services of the exhibitor and attracts visitors nonetheless.

The Faculty of Computer Science/Mathematics of Dresden University of Applied Sciences inspired this work. The Faculty required an eye-catcher at trade fairs, exhibitions, conferences, open houses etc. The ultimate goal is to acquire new students for our universities, faculties, and courses of study.

The original idea was to build a transportable model factory. Since many universities already possess model factories illustrating industrial engineering, the new model factory was to focus on queueing systems and material flow instead. Hence, our eye-catcher illustrates some principles of production planning and control which relates the eye-catcher to our services and consequently makes it even more appealing to visitors.

Furthermore, it is required that the model factory

- can be influenced by setting structural and behavioral parameters, e.g. the number of machines, the routings, and the control strategy,
- produces perpetually, e.g. does not run empty,
- blinks and beeps occasionally,
- is easy to transport and to set up, and
- fits into the trunk of a station wagon.

When developing a concept for the model factory, the focus shifted from a physical model to a model factory in augmented reality (AR) due to several reasons: State-of-the-art technology of mobile devices and augmented reality frameworks allow the creation of an AR model factory with reasonable effort. Such a model factory is configurable according to the specific exhibition, and visitors can manipulate it directly without the risk of physical damage. Furthermore, there is currently no comparable AR model factory known, which gives us a 'unique selling point'.

A realization of an AR model factory must fulfill the following basic requirements:

- Visitors can see the factory (machines and production orders in process) through their personal mobile devices (smartphones or tablet computers) but not with bare eyes.
- Mobile devices project the factory on the spatially restricted top of a conventional table.
- Production orders are visualized during their entire lifecycle, e.g. while they are moving from station to station, waiting in queues, and being processed at the machines.
- Visitors may set values of control parameters online on their mobile devices and observe the resulting effect on key performance indicators like adherence to delivery dates.

1 Types and Applications of Model Factories

Regarding their technical basis, there are different types of model factories: virtual factories, downscaled physical factories, and life-size physical factories. Model factories serve different purposes, e.g. evaluation, demonstration, and education.

Virtual factories are digital simulation models that do not contain any physical components. They are widely used for evaluating alternative production concepts (e.g. [1]) as well as for educational case studies (e.g. [2]), since they can represent arbitrary complex production systems and are easy to modify.

Downscaled physical model factories consist of small physical models of factories' resources. The resource models are either simple blocks or functional resources that handle simplified 'products' like small bricks. Components for creating this type of model factories are commercially available (e.g. [3]). Downscaled physical model factories typically serve educational purposes.

Life-size model factories are equipped with machines also used in real factories. Although they are usually restricted to a few workstations, these model factories require considerable space and are quite expensive. Their purpose is to demonstrate state-of-the-art technology (e.g. [4]), professional training (e.g. [5]), and research (e.g. [6]). Some of the life-size model factories use AR applications to provide manufacturing information to workers.

An investigation concerning AR based model factories did not come up with any existing solution similar to the one described in this paper.

2 Overview of the System Architecture of the AR Model Factory

Figure 1 shows an overview of the system architecture of the AR model factory.

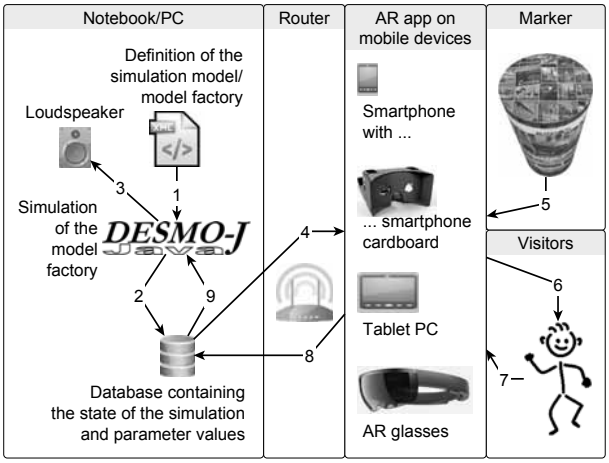

Figure 1: Overview of the system architecture.

The system operates as follows: The simulation environment, DESMO-J, loads an XML configuration file (1), which contains all relevant parameters of the factory, e.g. machines, routings, and the product mix. DESMO-J automatically configures an executable simulation model based on these parameters. Then the simulation starts. Initially, the simulation writes master data to the database. As the simulation progresses, DESMO-J writes the current state of the model factory to the database and updates it at every change in state (2).

Acoustic signals mark certain events during the simulation (3). A WLAN router connects the mobile device to the notebook/PC. The mobile device periodically polls the current state of the model factory from the database (4). The visitor points the camera of the mobile device to the marker standing on top of the table. Now, the app on the mobile device receives the camera's picture (5) and recognizes the marker's position in relation to the position of the camera. The app overlays the picture the camera is receiving with the visualization of the model factory according to the position of the marker in relation to the position of the camera.

If the visitor looks at the table 'through' the mobile device, she or he will see the model factory in its current state on top of the table (6). Figure 2 shows the visitor's view of the AR model factory.

The app updates the view in short intervals resulting in an animation of the production process. The visitor may change control parameters of the model factory via the mobile device (7). The app writes the new parameter values into the database (8), the simulation system reads the new parameter values (9) and adopts its behavior accordingly. Then, the process starts anew, and the visitor watches the changed behavior of the AR model factory on the mobile device. Several visitors may view the model factory at the same time using their personal mobile devices.

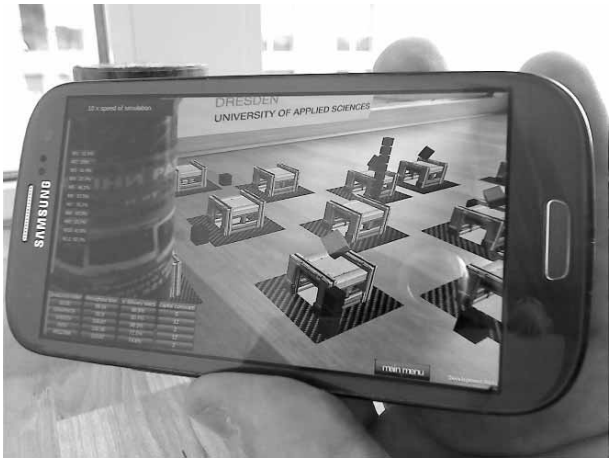

Figure 2: Visitor's view of the AR model factory.

The following three sections describe the components of the system and their functionality in detail.

3 Event-based Simulation of the Model Factory

The model factory is a job shop system (see [7]): An infinite stream of production orders enters the factory. Each production order requires a sequence of operations to be processed. Each operation is performed by a specific machine and requires a certain processing time. Each production order belongs to a certain product type assigned to a specific routing (i.e., a specific sequence in which the production order seizes the machines). A machine can process only one production order at a time, and one production order can only seize one machine at a time. While a machine is processing an order, other incoming orders enqueue in front of the machine.

Every machine has its own queue with unlimited capacity. After an operation is completed, the production order moves on to the next machine. In our model, only the transportation distance influences the required transportation time. Transportation resources are not part of the model. For demonstrating the model factory, we applied a production system consisting of eleven machines and producing five types of products (see [8] und [9]). Figure 3 pictures the production system.

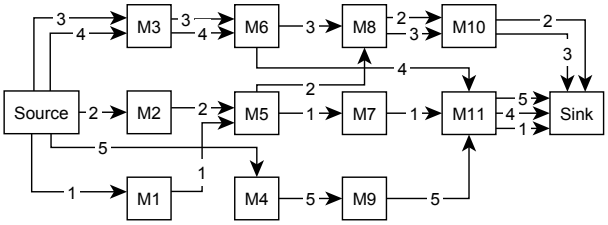

Figure 3: Production system containing eleven machines and the routings of five different products.

Typically, production planning and control pursues four objectives, namely [10]

- adherence to delivery dates,
- short throughput times,
- high utilization of the machines, and
- low level of work in process.

Four key performance indicators (KPI) operationalize these objectives. The KPIs are calculated and updated during the simulation.

A number of parameters is necessary to describe the model factory:

- number of machines,
- distribution of interarrival times,
- product types, their probability, and their specific routings (sequences of operations processed on certain machines and requiring certain processing times),
- velocity of transport, and
- control strategy applied to select orders from waiting queues.

An administrator writes the values of these parameters to an XML file. This configuration file serves as input for generating an event-based simulation model.

A variety of simulation systems is able to simulate job shop systems. We chose DESMO-J (see [11] and [12]), version 2.5.1c, developed by the Department of Computer Science of Hamburg University. DESMO-J is an open-source object-oriented simulation framework written in JAVA™.

JAVA™ is also used to realize the simulation models itself. DESMO-J supports process-oriented and event-oriented modeling [11]. We chose the latter because of its global view, its simplicity, and much lower computational costs.

While data-driven model generation is a well-known technique in simulation [13], there is one aspect that is worth mentioning: For sake of simplicity, the configuration file does not contain information about the physical location of the machines on the shop floor or a geometric description of the network of transportation routes. Hence, the factory layout has to be created automatically: An algorithm for layered drawing of directed acyclic graphs places the virtual machines relative to each other on the factory floor. The machines are the nodes of the graph, the transportation routes between consecutive machines are the directed edges of the graph. The algorithm aims at minimizing the number of crossings while keeping the factory layout as small as possible (see [14]). In order to achieve both, the algorithm sorts the machines topologically and assigns them to virtual layers (see [15]). Afterwards, the algorithm heuristically exchanges the machines at each layer to minimize the number of crossings of transportation routes (see [16]). All transportation routes are either parallel or perpendicular to each other and they may have perpendicular bends.

The model generator writes the coordinates of the machines and the transportation routes to the database (see Section 5). This way, the AR app can access the geometric information regarding the factory layout.

Once the model generation is completed, the simulation starts and runs until manually terminated. Visitors can not only view the operating model factory but also change some parameters at runtime:

One parameter is the utilization of the production indicated by the utilization of the bottleneck machine. The system limits the utilization to 85 % because the factory and the queues should not overflow with production orders. When a visitor enters a new utilization, the system calculates an appropriate interarrival time of the production orders applying the probabilities of the product types and the routings. Thus, the system generates production orders more or less often, and after a certain time the utilization converges to the required value. [17] gives an example of the above calculations.

The visitor may also change the product mix. She or he can choose one of three options:

- the initial distribution read from the configuration,
- a uniform distribution (all product types appear with the same probability), or
- a distribution where the next product type is twice as probable as the current one, i.e., if there are product types $i = 1, \dots, n$, the next arriving order will require product i with probability $p_i = 2^{i-1}/(2^n - 1)$.

Furthermore, the visitor can choose from several queueing strategies [18]:

- first come, first served,
- shortest processing time first,
- shortest slack time first, and
- highest value first.

All queues will be re-sorted when the global queueing strategy changes. The slack time and the values of the production orders are calculated and updated during the course of the simulation.

When a visitor changes a parameter value, the model factory needs time until it reaches a steady state again. The method ‘crossing of the means’ (see [19]) indicates when this is the case. Until then, visitors cannot change parameter values further. This way, the visitor can experience the effects of her or his change on the KPIs representing the four objectives of production planning and control.

4 The Composition and the Functionality of the AR Mobile App

Two frameworks support the visualization of the AR model factory, namely Unity and Vuforia. Unity is a runtime and development environment for 3D visualization and interaction (see [20] and [21]). It is a product of Unity Technologies, Inc. We applied free Unity Personal, Version 5.2.2. Vuforia is a software development kit for mobile devices that enables the creation of AR applications (see [22] and [23]). Vuforia was developed by the Qualcomm Developer Network and is now product of the Parametric Technology Corporation, Inc. We applied the non-commercial version Vuforia 5.5.9.

The 3D visualization of the model factory in Unity is straightforward: The machines are ready-made visual objects. The production orders are simple cubes differently colored in order to distinguish their routings/products. If necessary, they are stacked in front of the machines in order to resemble waiting queues.

If a stack becomes too high, its order cubes are scaled down proportionally. If a machine is processing a production order, a spinning cube above the machine indicates that the machine is working.

The transportation routes are drawn as parallel or perpendicular lines with perpendicular bends if necessary. The transportation routes are colored according to the color of the routings/products. The data for the visualization stems from the simulation (see Section 3). Unity calculates the movement of the production orders on the transportation routes, which results in a visualization of a smooth flow of the orders along the routes. Figure 4 shows the AR model factory containing machines, waiting queues, production orders, transportation routes etc.

Unity also provides control widgets. We applied widgets to change the utilization of the model factory, to change the product mix, and to change the queueing strategies (see Section 3). The widgets do not appear inside the 3D visualization of the factory but at the 2D graphical user interface (GUI) of the mobile app. Furthermore, the GUI displays the current values of the objective variables. This way, the visitor can observe how the change of the parameter values effect the objectives. Figure 4 shows the widgets and the display of the values of the four KPIs mentioned in Section 3.

Figure 4: Screenshot of the AR model factory; values of the objective variables at the bottom left corner.

All visitors watching the model factory via their mobile devices see the same factory at the same state. After one of them changes parameter values, all controls on all mobile devices become inactive until the model factory reaches a steady state again which usually takes less than two minutes. This way, the visitors can observe the undistorted effect of the change of the parameter values.

When we developed the AR app, we decided on marker-based augmented reality. The app knows the marker beforehand, and thus, the app identifies the marker more easily. In marker-less AR settings, the app has to identify certain objects, like faces, without knowing every object beforehand, which is difficult and therefore computationally expensive [24].

We decided on a cylindrical marker pasted up with parts of picture postcards (one of them from Dresden, Germany). Figure 1 shows the physical marker in the top right corner. Next, we defined a corresponding virtual marker at the website of Vuforia [25]. Figure 5 shows the marker's definition.

Figure 5: Definition of the marker: top view and overview over the complete surface.

The virtual marker resembles the physical marker. Its surface shows exactly the same pictures at exactly the same place. We downloaded a package from the Vuforia website, which contains the logical/virtual marker and a virtual AR camera. We imported this package in Unity and placed the logical/virtual marker inside the model factory, which already exists as rudimentary 3D model in Unity. We placed the logical/virtual marker at the entry of the model factory to represent the source of the production orders. Figure 6 shows the logical/virtual marker in Unity.

In order to establish an AR scenario, we need to insert a 3D visualization of a virtual world into a real-time video stream of a real-world scene. The camera of the mobile device provides the necessary video stream. In our case, the scene consists of a table with the physical marker standing on top of it. The AR app receives the video stream, identifies the marker, calculates its position and orientation, and provides an origin of coordinates as reference.

With the virtual camera, the Vuforia-part of the app then projects the virtual Unity model on the video stream, i.e. on the table (and the marker). The app finally delivers the augmented video stream to the visitor on the screen of her or his mobile device in real-time. The visitor gets the impression that she or he is watching the model factory on top of the table through the mobile device.

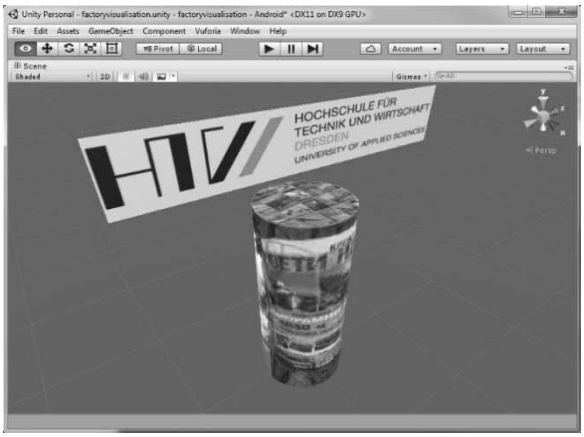

Figure 6: Representation of the virtual marker in the Unity framework.

One major problem occurred when Vuforia overlaid the real-time picture with the 3D visualization: The visualization alternated abruptly and noticeably between two positions in consecutive resulting pictures. The reason for the undesirable blurring is a picture downscaling effect: For faster real-time pattern recognition of the marker in the real-world picture, Vuforia scales the input picture down to a size of 640×480 pixels.

The remaining pixel array serves the detection of the edges of the marker. If the mobile device and its camera move only slightly, their position will repeatedly alter marginally in relation to the physical tracker and the detected edges of the marker in the picture will jump between adjacent pixels. Scaled up again, the distance between the marker's edges detected in one picture differs considerably from the edges detected in the next picture. Thus, Vuforia overlays the 3D visualization on the real-world pictures at different positions in consecutive pictures. Consequently, the 3D visualization jumps noticeably from one resulting picture to the next. We solved the problem by using the moving averages of the last five detected positions and orientations of the marker in the Vuforia software module when overlaying the real-world picture with the 3D visualization. This proved to be sufficient to avoid jumps of the 3D model in consecutive resulting pictures.

After compiling the app in Unity, it can be exported as Android Package (APK) for the Android operating system or as XCode for iOS.

5 The Database for the Communication between Simulation and Mobile App

The database is stored on the notebook and managed by MySQL. Both the simulator and the app are connected to the database via JDBC.

The database serves as interface for the bidirectional communication between the simulation and the AR app. In brief, the simulation writes master data and events to the database while the mobile app reads the master data and the events from the database and applies them to the visualization. On the reverse channel, the mobile app writes control data to the database while the simulation reads the data from the database and adopts the simulation accordingly.

Figure 7 shows a simplified data model in crow's foot notation (see [26]). Initially, the simulation model generator writes master data to the database: machines and their locations, transportation routes with their starts, stops, and bends. As a simulation run proceeds, the relevant events of the order lifecycle are written to the database: entering the factory, enqueuing in a waiting queue, seizing a machine, releasing a machine, and leaving the factory. Moreover, the simulation writes KPIs to the database. The simulation adds timestamps to all records. Database triggers aggregate, simplify, and prepare all data relevant for the visualization and distribute the data to small tables containing only very few records. Reading these tables is very fast.

The mobile app reads the master data at the beginning of the simulation and creates the 3D visualization of the factory layout. Then, the mobile app periodically polls events from the database. Every period comprises a few frames. According to the events, the mobile app updates the 3D visualization of the production orders. As mentioned in Section 4, the only data not provided by the database are the locations of the production orders on the transportation routes during transport. The mobile app continuously calculates these locations in relation to the length of the routes and the given velocity.

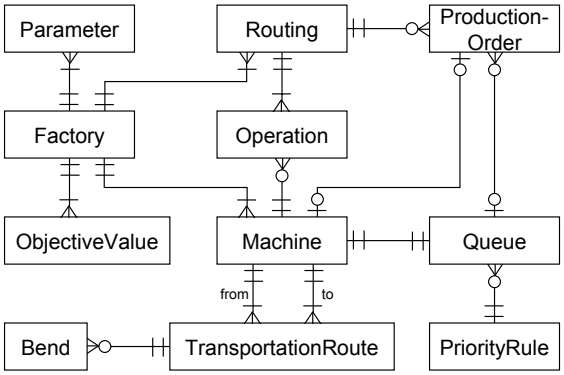

Figure 7: Simplified data model.

When a visitor changes the value of a control parameter, the mobile app writes the change and a timestamp to the database. All connected apps read the new parameter values from the database, update the parameter values in their graphical user interface, and disable the controls until the simulation reaches a steady state again. Thus, no visitor can change parameter values until then. If another visitor changes the parameter values in the meantime, the database will discard the changes because of the later timestamp. In parallel, the simulation also reads the new parameter values and adopts its behavior accordingly. Thereupon, the simulation writes newly occurring events to the database and updates the KPIs. The mobile app reads the updated KPIs from the database and displays them to the visitor. When the simulation reaches a steady state again, the simulation writes the information to the database and the mobile app reads the information and enables the controls.

6 Watching and Operating the Model Factory through a Personal Mobile Device

When exhibiting the AR model factory, we provide mobile devices, so visitors can watch the factory without their personal mobile devices. However, there may not be enough mobile devices for all visitors and some visitors might want to watch the factory through their personal devices.

In order to install the AR app on his or her own device, the visitor has to scan a QR code printed on a poster provided at the booth. The QR code encodes an URL referring to a website located on the local notebook/PC at the booth. Using the URL, visitors can download the installation files of the mobile app for different versions of different operating systems.

Once the app has been installed, it can be used immediately to view and control the AR model factory.

7 Recent and Future Use of the AR Model Factory

We already exhibited the AR model factory twice at the career-start fair (KarriereStart) in Dresden, several times during open houses at Dresden University of Applied Sciences, and during Long Night of Sciences (Lange Nacht der Wissenschaften) in Dresden. Visitors were very interested in the model factory. More than 150 of them watched the model factory 'through' our prepared tablet computers on the table; more than 40 of them installed the mobile app on their own mobile devices. All visitors changed at least the utilization of the factory and watched the waiting queues lengthen or shorten. The feedback was positive without exception. The most frequent question asked by visitors was if we could also visualize and control their real-life factories in AR in real-time.

The exhibition scheduled next is the upcoming Long Night of Sciences in Dresden. When introducing Dresden University of Applied Sciences at high schools, we will also exhibit the AR model factory. We also plan to present the model factory during poster sessions at scientific conferences.

8 Summary and Outlook

We described the development of an AR model factory. The overall goal was to build an eye-catcher we can apply at trade fairs, exhibitions, conferences, open houses etc. We achieved the goal in full by building the AR model factory. Evidently, the model factory stimulates the visitors to approach the booth and to gather information of the courses of study offered at our faculties/universities. Moreover, the model factory has a strong relation to a specific domain of business informatics, namely production planning and control: We can use the model factory to demonstrate some principles of production planning and control. The eye-catcher is not physical, but some of its parts are moving, and it sends optical and acoustic signals when certain events occur in the factory. In essence, the visitor can watch the model factory 'through' her or his smartphone. The model factory seems to stand on top of a physically empty table and it is visibly processing production orders of different types.

Visitors may manipulate the behavior of the model factory by changing the utilization, the product mix, and the queueing strategy. The changes result in different values of the KPIs (like adherence to delivery dates or work in process), which the visitor can also monitor.

At a next stage of extension, the visitor could interact with the mobile app via speech and gestures but speech and gesture recognition might slow down the visualization. If the simulation and the database run on a remote server, the notebook/PC will become obsolete at the booth and we will need the marker only. The internet connection should be broad and stable for this scenario, which is often not the case at trade fairs. The visualization could display the number of waiting orders above the queues. Furthermore, the development of the KPIs could be visualized as time lines in diagrams. A printout of the factory layout and physical building blocks representing the machines could serve as a physical model of the factory. This way, the model factory would interlace reality and AR even more and it would more closely resemble a physical model factory.

References

[1] März L, Krug W, Rose O, Weigert G. *Simulation und Optimierung in Produktion und Logistik*. Berlin: Springer; 2011. 220 p.

[2] Bickel S, Yildiz M, Blessing B, Kriz WC, Oslowski G, Eiselen T. The e-Beer game: an online simulation game for the training of supply chain management. In: Mayer I, Mastik H. editors: *Organizing and Learning Through Gaming and Simulation. Procedings of ISAGA 2007*; 2007 Jul; Nijmegen. Delft: Eburon. 39-46.

[3] http://www.fischertechnik.de/en/Home.aspx

[4] Mayer F, Pantförder D, Diedrich C, Vogel-Heuser B. *Deutschlandweiter I4.0-Demonstrator*. Lehrstuhl für Automatisierung und Informationssysteme, Technische Universität München, 2013.

[5] http://www.festo-didactic.com

[6] http://www.arena2036.de

[7] Blazewicz J, Domschke W, Pesch E. The Job Shop Scheduling Problem: Conventional and New Solution Techniques. *European Journal of Operational Research.* 1996; 93: 1-33.

[8] Schulz R. *Parallele und Verteilte Simulation bei der Steuerung komplexer Produktionssysteme*. Technische Universität Ilmenau; 2002.

[9] Bube O. *Architektur und Implementierung von Prozeßidentifikatoren*. Technische Hochschule Ilmenau; 1993.

[10] Buzacott JA, Corsten H, Gössinger R, Schneider, HM. *Production Planning and Control: Basics and Concepts.* München: Oldenbourg; 2012. 246 p.

[11] Page B, Kreutzer W. *The Java Simulation Handbook: Simulating Discrete Event Systems with UML and Java*. Aachen: Shaker; 2005. 516 p.

[12] Page B, Lechler T, Claasen S. *Objektorientierte Simulation in Java mit dem Framework DESMO-J*. Books on Demand; 2000. 197 p.

[13] Bergmann S. *Automatische Generierung adaptiver Modelle zur Simulation von Produktionssystemen*. Technische Universität Ilmenau; 2013.

[14] Battista GD, Eades P. *Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, NJ: Prentice-Hall, 1998. 397 p.

[15] Coffman EG, Graham RL. *Optimal Scheduling for Two-Processor Systems. Acta Informatica*. 1972; 1: 200–213.

[16] Barth W, Jünger M, Mutzel P. *Simple and Efficient Bilayer Cross Counting. Journal of Graph Algorithms and Applications*. 2004; 8(2): 179-194.

[17] Munkelt T. *Potenzial Bayes'scher Netze zur Unterstützung der Produktionsplanung und -steuerung*. Technische Universität Ilmenau; 2009.

[18] Shtub A, Karni R. *ERP: The Dynamics of Supply Chain and Process Management*, New York: Springer; 2010. 281 p.

[19] Mahajan PS, Ingalls RG. Evaluation of Methods Used to Detect Warm-up Period in Steady State Simulation. *In: Ingalls, R. G., editors: Proceedings of the Winter Simulation Conference;* 2004 Dec; Washington, DC. 663-671.

[20] Goldstein W. *Unity Game Development Essentials.* Birmingham: Packt; 2009. 316 p.

[21] Chittesh J. *Das Unity-Buch: 2D- und 3D-Spiele entwickeln mit Unity 5.* Heidelberg: dpunkt; 2015. 512 p.

[22] Kwik F, Bahana R. *Using Augmented Reality to Enhance Aetherpet, a Prototype of a Social Game.* Procedia Computer Science. 2015; 59: 282-290.

[23] Imbert N, Vignat F, Kaewrat C, Boonbrahm P. *Adding Physical Properties to 3D Models in Augmented Reality for Realistic Interactions Experiments.* International Conference on Virtual and Augmented Reality in Education; 2013 Nov; Puerto de la Cruz. 364-369.

[24] Chi HL, Kang SH J, Wang X. *Research trends and opportunities of augmented reality applications in architecture, engineering, and construction.* Automation in Construction. 2013; 33: 116-122.

[25] https://developer.vuforia.com/target-manager

[26] Rob P, Coronel C. *Database Systems: Design, Implementation & Management*. 8th ed. London: Cengage Learning EMEA; 2008. 704 p.