IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 25, No. 5, October, 2020, pp. 699-707
Journal homepage: http://iieta.org/journals/isi

Design and Application of a Virtual Simulation Teaching System Based on Cloud Service

Xiaogai Shen

Department of Management, Shijiazhuang University of Applied Technology, Shijiazhuang 050081, China

Corresponding Author Email: shenxgzjl@163.com

https://doi.org/10.18280/isi.250518

Received: 7 June 2020
Accepted: 15 September 2020

Keywords:
virtual simulation (VS), cloud service (CS), VS teaching system, simulation system design

ABSTRACT

To satisfy the demand of virtual simulation (VS) teaching system for storage capacity and computing power, it is of great significance to probe deep into the VS system oriented to cloud service (CS). This paper designs a CS-based VS teaching system, drawing on the relevant results on distributed VS system, the formalization of VS system architecture, and the semantic organizability of VS system. Firstly, the functional requirements of the CS-based VS teaching system were analyzed, and decomposed into three layers, namely, goal, function type, and function name. Then, a dynamic model of CS-based VS teaching system was constructed, whose simulation accuracy and precision were verified through experiments. The experimental results demonstrate the excellent simulation performance of the proposed system. The research findings provide reference for applying CS system design in other fields.

1. INTRODUCTION

Virtual simulation (VS) aims to analyze the system architecture and functions of the object, providing a reference for the decision-making of applications based on that object. The VS process can be broken down into four stages: object modeling, model structuring and programming, model running and verification, and comparative analysis of experimental results [1-5].

The VS has been widely used in the field of teaching. Teachers and researchers rely on the VS to explore actual complex nonlinear systems and use them to teach students. If the target system has complex functions, it will be complicated and compute-intensive to simulate the system [6-8]. The massive data have challenged various aspects of VS teaching system applications, ranging from program development, architecture design, to the accuracy and combination of simulation elements.

As a new computing service model, cloud service (CS) has become a research hotspot [9-12]. CS and VS are two research fields with different purposes and contents [13-15]. But it is CS that provides the basic platform and application environment for the VS based on logical framework. Currently, there is no unified standard for CS platforms of VS teaching. Many key details need to be solved before applying CS to VS teaching [16, 17].

The existing studies on CS-based VS mainly focus on distributed VS system, the formalization of VS system architecture, and the semantic organizability of VS system [18, 19]. Most of the VS systems on the market are developed under the high-level architecture (HLA). After extending the HLA, Schaefer, Schaefer et al. [20] designed simulation systems for application scenarios like high-energy chemical data analysis, tide warning, and ultraviolet lighting analysis. Bajpai et al. [21] introduced the Cosim-Grid, a simulation grid project of China National Grid (CNGrid) for industrial production scenarios (e.g. product supply, product output, and manufacturing cycle), and emphasized that the project can complete such services as resource allocation to varied production and marketing modes, and management of supply chain members. Loukriz et al. [22] designed the Aurora system to overcome the performance defect of distributed simulation system in CS environment; through the system, the workers that undertake lots of computing tasks are controlled through the master, and all cached VS data and processes are integrated into one data packet to be exchanged between different simulation elements.

On the formalization of VS system architecture, Maeda et al. [23] developed a universal hierarchical modular VS framework called discrete event VS (DEVS) for the numerous discrete events in the real world, and successfully described the time-varying effects and associations between the modules within the object. Ghosh et al. [24] put forward a parallel DEVS modeling paradigm oriented to the logic process paradigm, which makes up the insufficiency of DEVS in depicting the execution capability of parallel events.

On the semantic organizability of VS system, Ferraresi et al. [25] proposed a hierarchical extended simulation model based on high-level concepts for combined discrete events, and demonstrated that the proposed model has high accuracy and strong ability to handle complex problems.

To satisfy the demand of VS teaching system for storage capacity and computing power, it is of great significance to probe deep into the VS system oriented to CS, and rationalize the description and data processing of the object. Thus, this paper designs a VS teaching system based on CS. Firstly, the functional requirements of the CS-based VS teaching system were analyzed, and decomposed into three layers, namely, goal, function type, and function name. Then, a dynamic model of the CS-based VS teaching system was established, plus the sequence diagram of the modeling process. After that, the simulation accuracy and precision of the proposed system were verified through experiments. The experimental results show that the proposed system is fast in response, and

powerful in resource management and operation execution.

2. DEMAND ANALYSIS

The CS-based VS teaching system mainly targets program developers and users. Figure 1 shows the use cases of VS teaching system functions. Five kinds of functions can be summarized for VS teaching system:

(1) The CS-based VS teaching system needs to support four basic functions: interfacing module interactions; managing developer and user configurations; environment design and program running; data collection and processing.

(2) Four levels of interfaces should be provided for the interactive services between modules: control interface, connection interface, data interface, and basic interface.

(3) The configurations of developers and users should be managed in four aspects: account and password; program uploading and modification; platform parameters and strategies; type and number of tasks.

(4) The cloud simulation service engine and its functions should be expanded and packaged through environment design and program running.

(5) The data on VS teaching system should be collected, processed, and presented accurately.

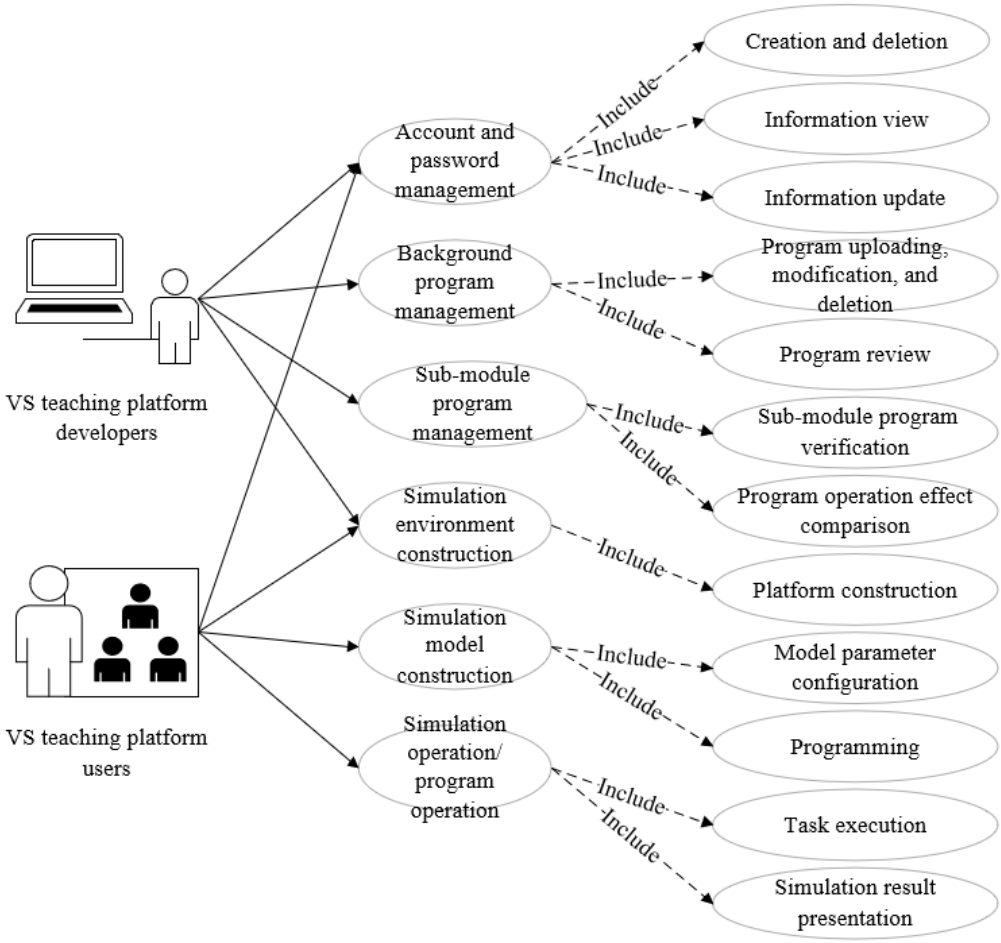

Figure 1. The use cases of VS teaching system functions

The specific functions of CS-based VS teaching system are as follows:

Layer 1 (goal)

D={D_1, D_2, D_3, D_4}={interfacing module interactions, managing developer and user configurations, environment design and program running, data collection and processing}

Layer 2 (type)

D_1={D_{11}, D_{12}, D_{13}, D_{14}}={control interface, connection interface, data interface, basic interface};

D_2={D_{21}, D_{22}, D_{23}, D_{24}, D_{25}, D_{26}, D_{27}, D_{28}}={user creation and deletion, user information view, user information update, user simulation operation, managing program upload and configuration modification, managing platform parameter configuration, managing computer resource configuration, managing type and number of tasks}

D_3={D_{31}, D_{32}}={environment design, program running}

D_4={D_{41}, D_{42}, D_{43}}={data collection, data processing, data presentation}

Layer 3 (name)

D_{11}={D_{111}, D_{112}, D_{113}, D_{114}, D_{115}, D_{116}}={platform, task, program configuration interface, user control command management interface, program operation control interface, simulation result output and query interface}

D_{12}={D_{121}, D_{122}, D_{123}}={simulation element combination interface, task scheduling and configuration interface, simulation module control interface}

D_{13}={D_{131}, D_{132}, D_{133}, D_{134}, D_{135}}={main task interface, simulation data library interface, task creation interface, task execution interface, simulation element and module scheduling management interface}

D_{14}={D_{141}, D_{142}}={simulation module output data interface, platform output data interface}

D_{21}={D_{211}, D_{212}}={user creation, user deletion}

D_{22}={D_{221}, D_{222}, D_{223}, D_{224}}={viewing user basic information, viewing user program information, viewing user model information, viewing user program running state}

D_{23}={D_{231}}={updating user basic information}

D_{24}={D_{241}, D_{242}}={running user simulation, stopping user simulation}

D_{25}={D_{251}, D_{252}, D_{253}, D_{254}}={program uploading, program modification, program deletion, program query}

D_{26}={D_{261}, D_{262}}={simulation database parameter configuration, computer parameter configuration}

D_{27}={D_{271}, D_{272}, D_{273}, D_{274}, D_{275}}={processor resource allocation, memory resource allocation, bandwidth resource allocation, storage resource allocation, task scheduling priority}

D_{28}={D_{281}, D_{282}, D_{282}}={type of task type, execution time of task, number of tasks}

D_{31}={D_{311}, D_{312}, D_{313}, D_{314}, D_{315}}={construction of platform, determination of simulation principle, program preparation, comparison of experimental results, program validation}

D_{32}={D_{321}, D_{322}, D_{323}, D_{324}}={starting task, pausing task, resuming task, ending task}

D_{41}={D_{411}, D_{412}, D_{413}}={CS data, energy consumption data, task execution time data}

D_{42}={D_{421}, D_{422}}={data analysis algorithm, data processing algorithm}

D_{43}={D_{431}, D_{432}}={graphic presentation of simulation result, tabular presentation of simulation result}

3. DYNAMIC MODELING

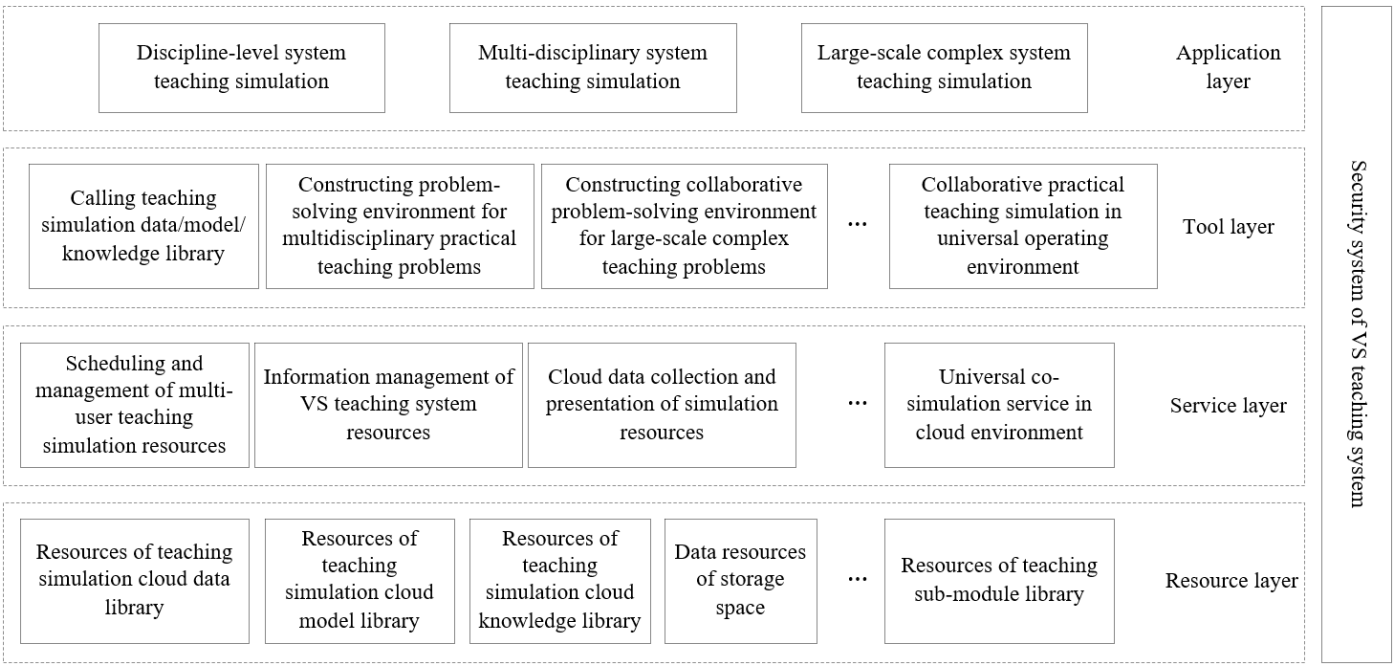

Figure 2. The hierarchical structure of CS-based VS teaching system

According to the requirements of tasks, the CS-based VS teaching system should be able to remove the coupling between sub-modules, simulation tools, sub-nodes of program running, and connection nodes of simulation elements, such as to ensure the efficient and reliable operation of the program. Figure 2 illustrates the hierarchical structure of CS-based VS teaching system. It can be seen that the key techniques in the dynamic modeling of the CS-based system are virtualizing and standardizing the target simulation elements of CS.

The basic operation process of the dynamic model is as follows: After different users upload the program, the CS center will parse the requirements of the program for computer resources, simulation element combination, and sub-module scheduling, select and call relevant simulation elements and sub-modules from the simulation libraries as per the parse results, and combine these elements and sub-modules to construct the main task. Figure 3 gives the simplified architecture of the CS-based VS teaching system.

It is impossible to dynamically model the CS-based VS teaching system, without virtual packaging the simulation elements and modules on Docker. Before building the dynamic model, the mapping between the simulation elements and modules, which are related to the task, in Docker containers must be determined, such that the program can start in the shortest time and operate at the highest efficiency (Figure 4).

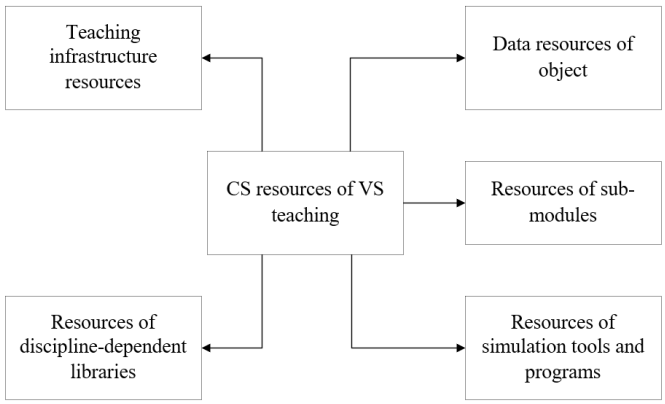

Figure 3. The simplified architecture of the CS-based VS teaching system

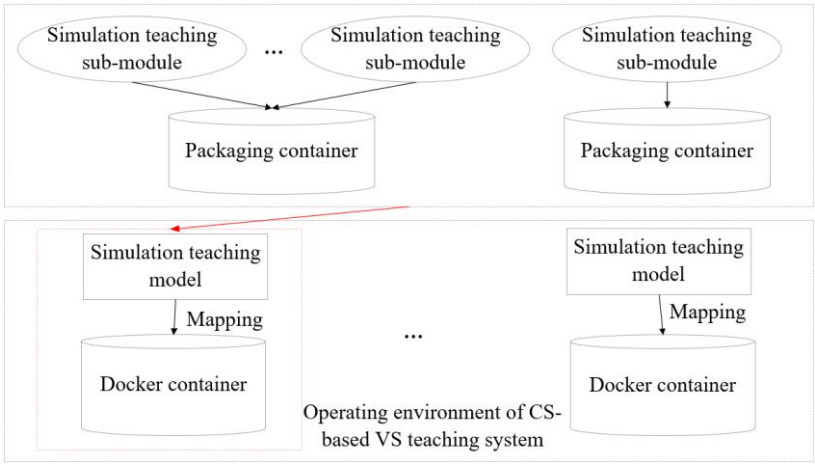

Figure 4. The mapping of dynamic model to Docker containers

The startup time and operating efficiency of the program are mainly determined by the computing power and memory allocated to program running sub-nodes and connection nodes of simulation elements, and also affected by the bandwidth and storage space allocated to each sub-module during the reception of simulation data. Therefore, the mapping of dynamic model to Docker containers was established based on five factors: processor resource allocation D_{271}, memory resource allocation D_{272}, bandwidth resource allocation D_{273}, storage resource allocation D_{274}, and task scheduling priority D_{275}. The computing power required for the dynamic model can be quantified by:

$$C_{DM\text{-}C} = \omega_1 C_{cpu} + \omega_2 C_{ram} \tag{1}$$

where, C_{cpu} is processor resource allocation; C_{ram} is memory resource allocation; ω_1 and ω_2 are the weight coefficients of C_{cpu} and C_{ram}, respectively. The networking performance required for the dynamic model can be quantified by:

$$C_{DM\text{-}N} = \omega_3 C_{band} + \omega_4 C_{SS} \tag{2}$$

where, C_{band} is bandwidth resource allocation; C_{SS} is storage resource allocation; ω_3 and ω_4 are the weight coefficients of C_{band} and C_{SS}, respectively. Thus, the dynamic model of the CS-based VS teaching system can be expressed as:

$$DM = \{M_{Dyn}, N_{Dyn}, RA_{Dyn}, NP_{Dyn}\} \tag{3}$$

where, $M_{Dyn}=\{model_1, model_2, \ldots, model_n\}$ is the set of n sub-modules in the dynamic model; $N_{Dyn}=\{N_{modeli,modelj} | model_i, model_j \in N_{Dyn}\}$ is the set of connection nodes between the sub-modules; $RA_{Dyn}=\{RA_{modeli} | model_i \in N_{Dyn}\}$ is the set of computer resource configuration performance (CRCP) in the dynamic model; $NP_{Dyn}=\{P_{modeli,modelj} | model_i, model_j \in N_{Dyn}\}$ is the set of networking performance between sub-modules. Each Docker container can be described as:

$$DOCKER = \{D_{Doc}, N_{Doc}, RA_{Doc}, NP_{Doc}\} \tag{4}$$

where, $D_{Doc}=\{Doc_1, Doc_2, \ldots, Doc_m\}$ is the set of m Docker containers; $N_{Doc}=\{N_{Doci,Docj} | Doc_i, Doc_j \in N_{Doc}\}$ is the set of connection nodes between the containers; $RA_{Doc}=\{RA_{Doci} | Doc_i \in N_{Doc}\}$ is the set of CRCP provided by the containers; $NP_{Doc}=\{P_{Doci,Docj} | Doc_i, Doc_j \in N_{Doc}\}$ is the set of networking performance required between the containers. Then, the mapping $map=h(model)$ of sub-modules in the dynamic model to the containers can be defined as:

$$MAP = \{D_{map}, I_R, C_R, U_R\} \tag{5}$$

where, $D_{map}=\{model_{map1}, model_{map2}, \ldots, model_{mapm}\}$ is the set of containers after the deployment of sub-modules, $D_{map}=D_{Doc}$; $N_{map}=\{N_{model\text{-}mapi,model\text{-}mapj} | model_{mapi}, model_{mapj} \in N_{map}\}$ is the set of connection nodes between the containers after the deployment of sub-modules; $N_{map}=N_{Doc}$;$RA_{map}=\{RA_{model\text{-}mapi} | model_{mapi} \in N_{map}\}$ is the set of CRCP within container Doc_i as required by sub-modules; $NP_{map}=\{P_{model\text{-}mapi,model\text{-}mapj} | model_{mapi}, model_{mapj} \in N_{map}\}$ is the set of networking performance required for the sub-module deployed between Doc_i and Doc_j ($\forall N_{model\text{-}mapi,model\text{-}mapj} \in N_{map}$, $h(model_{mapi})=Doc_i$).

In general, the total CRCP required by the dynamic model must be smaller than the CRCP of Docker containers, and the total networking performance required by the dynamic model must be smaller than the networking performance of Docker containers. Figure 5 presents the sequence diagram of the dynamic modeling. The number of sub-modules to be deployed in the containers can be derived from the optimal mapping.

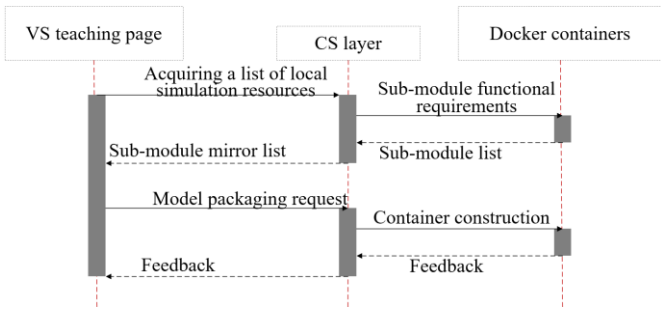

Figure 5. The sequence diagram of the dynamic model

4. ACCURACY VERIFICATION

The simulation accuracy of the VS teaching system could be affected by various factors. This paper evaluates the simulation accuracy of the system from eight dimensions: object, attributes of active and passive states, associations

between these attributes, model resolution, model error, simulation sensitivity, response time, and simulation precision. Figure 6 shows the evaluation model of the system simulation accuracy.

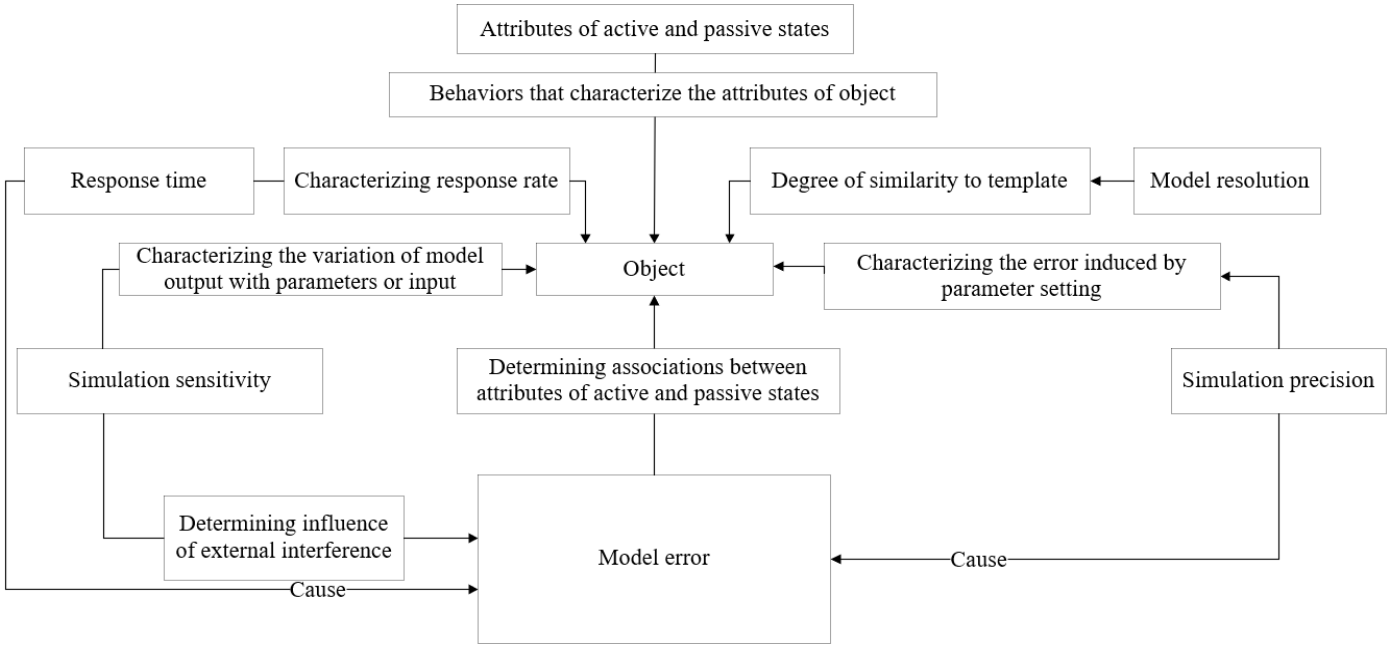

Figure 6. The accuracy evaluation model

The simulation accuracy of the dynamic model was evaluated in the following steps, according to the requirements on the CS-based VS teaching system and the eight evaluation dimensions.

(1) Verification of system requirements

The verification of system requirements aims to judge whether the CS-based VS teaching system can satisfy the user needs for simulation, and the developer needs for development, maintenance, and update, from the following four aspects: interfacing module interactions; managing developer and user configurations; environment design and program running; data collection and processing.

Let DEM_U be the set of user needs for simulation, and DEM_S be the set of functions realized by the system. In addition, the object and the simulation model are denoted as O^U and O^S, respectively; the attributes of the object and those of the simulation model are denoted as A^U and A^S, respectively; the associations between the attributes of the object and those of the simulation model are denoted as R^U and R^S, respectively. Then, DEM_U and DEM_S can be expressed as:

$$\begin{cases} DEM_U = \{O^U, A^U, R^U\} \\ DEM_S = \{O^S, A^S, R^S\} \end{cases} \tag{5}$$

where, ΔO^U, ΔA^U, and ΔR^U are the deviations between the object and the simulation model. If the deviations are acceptable to the developer or user, the functions of the VS teaching system can meet the user needs for simulation.

(2) Verification of sub-module accuracy

The verification of sub-module accuracy aims to judge whether the sub-modules can satisfy the user demand for simulation accuracy from the eight dimensions. Here, the sub-module and the evaluation model for sub-module are denoted as O^M and O^E, respectively; the attributes of the sub-module and those of the evaluation model for sub-module are denoted as A^M and A^E, respectively; the associations between the attributes of the sub-module and those of the evaluation model for sub-module are denoted as R^M and R^E, respectively. Then, the set MS_M of sub-modules and the set MS_E of evaluation models for sub-module can be respectively expressed as:

$$\begin{cases} MS_M = \{O^M, A^M, R^M, \Delta E_A^M, S_A^M, P_A^M\} \\ MS_E = \{O^E, A^E, R^E, \Delta E_A^E, S_A^E, P_A^E\} \end{cases} \tag{7}$$

where, $\Delta E^M{}_A$ and $\Delta E^E{}_A$ are the set of errors of the active and passive state attributes of the sub-module accepted by the developer/user, respectively; $S^M{}_A$ and $S^E{}_A$ are the set of simulation sensitivities of the active and passive state attributes of the sub-module accepted by the developer/user, respectively; $P^M{}_A$ and $P^E{}_A$ are the set of simulation precisions of the sub-module accepted by the developer/user, respectively. Each sub-module of the VS teaching system needs to satisfy the following constraints:

$$\begin{cases} O_L^M \le O^E \le O_H^M \\ A_L^M \le A^E \le A_H^M \\ R_L^M \le R^E \le R_H^M \\ \Delta E_{AL}^M \le \Delta E_A^E \le \Delta E_H^M \\ S_{AL}^M \le S_A^E \le S_H^M \\ P_{AL}^M \le P_A^E \le P_H^M \end{cases} \tag{8}$$

where, $O_L{}^M$ and $O_H{}^M$ are the minimum and maximum of O^M, respectively; $A_L{}^M$ and $A_H{}^M$ are the minimum and maximum of A^M, respectively; $R_L{}^M$ and $R_H{}^M$ are the minimum and maximum of R^M, respectively; $\Delta E_{AL}{}^M$ and $\Delta E_{AH}{}^M$ are the minimum and maximum of $\Delta E_A{}^M$, respectively; $S_{AL}{}^M$ and $S_{AH}{}^M$ are the minimum and maximum of $S_A{}^M$, respectively; $P_{AL}{}^M$ and $P_{AH}{}^M$ are the minimum and maximum of $P_A{}^M$, respectively. If a sub-module of the VS teaching system satisfies the above formula,

then the sub-module meets the required simulation accuracy.

(3) Verification of simulation model accuracy

The verification of simulation model accuracy aims to judge whether the simulation model, which is made up of the sub-modules, can satisfy the user demand for simulation accuracy from the eight dimensions.

Here, the simulation model and its evaluation model are denoted as O^F and O^{FE}, respectively; the attributes of the simulation model and those of the evaluation model are denoted as A^F and A^{FE}, respectively; the associations between the attributes of the simulation model and those of the evaluation model are denoted as R^F and R^{FE}, respectively. Then, the set MS_F of simulation models and the set MS_{FE} of evaluation models can be respectively expressed as:

$$\begin{cases} MS_F = \left\{O^F, A^F, R^F, \Delta E_A^F, S_A^F, P_A^F\right\} \\ MS_{FE} = \left\{O^{FE}, A^{FE}, R^{FE}, \Delta E_A^{FE}, S_A^{FE}, P_A^{FE}\right\} \end{cases} \tag{9}$$

As can be seen from formula (9), if the deviations between the simulation model and its evaluation model are acceptable to the system developer or user, then the simulation model, which is made up of the sub-modules, passes the accuracy verification.

(4) System acceptance

Through the quantitative verifications above, the evaluators of simulation accuracy will make the final decision on the acceptance of the VS teaching system.

5. PRECISION ANALYSIS

Precision is an important indicator of the reliability, credibility, and simulation accuracy of the VS teaching system. For the CS-based VS teaching system, the output of each sub-module is partially taken as the input of another VS teaching system. Considering the information transmission error between the sub-modules, it is necessary to analyze the simulation sensitivity of multiple sets of parameters in each sub-module, and evaluate how much the parameters that induce large simulation error affects the simulation model. Figure 7 provides the block diagram of the simulation accuracy analysis model for the VS teaching system.

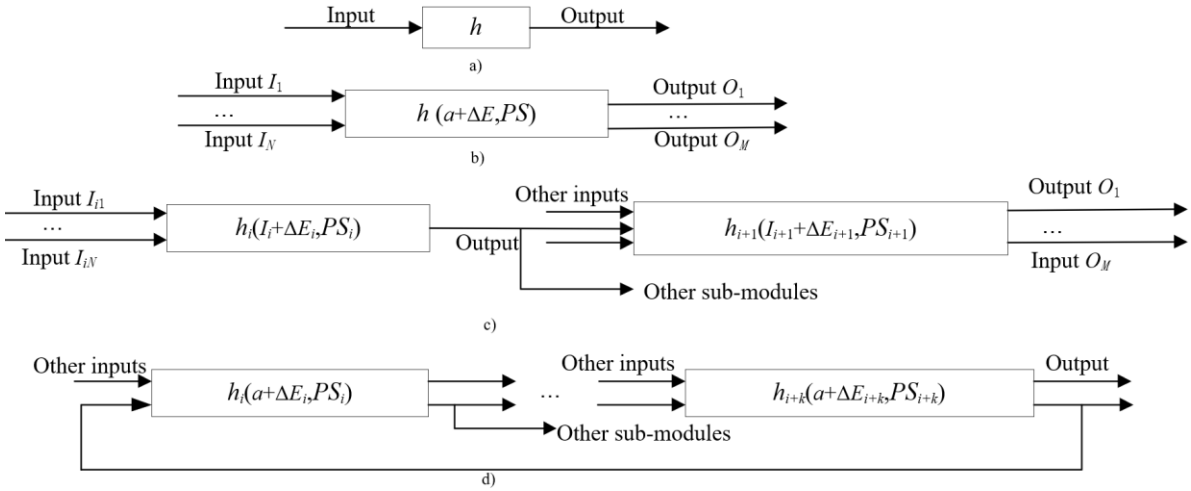

Figure 7. The block diagram of the simulation accuracy analysis model

As shown in Figure 7(a), it is assumed that an input a of the object is simulated by model $model_i$, whose error is ΔE_i. Then, the simulation sensitivity of $model_i$ can be expressed as:

$$S_i = \left|h_i\left(a+\Delta E_i\right) - \bar{h}_i\left(a\right)\right| / \bar{h}_i\left(a\right) \tag{10}$$

where, $h_i(a)$ is the input-output relationship of the i-th sub-module; $h_i(a)$ is the input-output relationship of the corresponding object. If the parameter settings of the sub-module have deviation, then $model_i$ will output different results under the same input a. The object often has multiple input-output relationships, and maintains nonlinear uncertain mappings with each sub-module. Suppose I_N and O_M are the input and output vectors of the object, respectively, and PS is the set of sub-module parameters with simulation deviation.

As shown in Figure 7(b), it is assumed that I_N contains x precise inputs. Then, the simulation sensitivity of the simulation model composed of a single sub-module can be expressed as:

$$S = \left|h\left(a+\Delta E\right) - \bar{h}\left(a\right)\right| / \bar{h}\left(a\right) \tag{11}$$

where, $h(a)$ is the input-output relationship of the simulation model; $\bar{h}(a)$ is the input-output relationship of the corresponding object.

As shown in Figure 7(c), if the simulation model is composed of multiple sub-modules in series, the simulation sensitivity of the i-th sub-module needs to be computed by formula (11). Then, the output error $h_i(a+\Delta E_i)$- $\bar{h}_i(a)$ of the i-th sub-module will be inputted as the error ΔE_{i+1} of the i+1-th sub-module.

As shown in Figure (d), if the simulation model is composed of multiple sub-modules in series, and if the errors are fed back via the coupling between these sub-modules, it is assumed that the multiple sub-modules have no simulation deviation in parameter setting and all adopt the single-input single-output mode. Then, the input and output of sub-module $model_i$ at time t can be expressed as I^t_i and $O^t_i=h_i(I^t_i)$, respectively; the output of sub-module $model_{i+1}$ at time t can be expressed as $O^t_{i+1}=h_{i+1}[h_i(I^t_i)]$, which is the input of sub-module $model_{i+2}$ at time t. This derivation process continues until the output of the i+k-th sub-module $model_{i+k}$ is taken as the input of $model_i$ at time t+1. Then, a new derivation process will begin. The simulation state of each sub-module can be obtained by:

$$\begin{cases} I_i^t = O_{i+k}^{t+1}, O_i^t = h_i\left(I_i^t\right), \ \Delta E_i = h_i\left(I_i^t + \Delta E_{i+k}\right) - \overline{h}_i\left(I_i^t\right) \\ I_{i+1}^t = O_i^t, O_{i+1}^t = h_{i+1}\left(I_{i+1}^t\right), \ \Delta E_{i+1} = h_{i+1}\left(I_{i+1}^t + \Delta E_i\right) - \overline{h}_{i+1}\left(I_{i+1}^t\right) \\ \cdots \\ I_{i+k}^t = O_{i+k-1}^k, O_{i+k}^t = h_{i+k}\left(I_{i+k}^t\right), \ \Delta E_{i+k} = h_{i+k}\left(I_{i+k}^t + \Delta E_{i+k-1}\right) - \overline{h}_{i+k}\left(I_{i+k}^t\right) \end{cases} \tag{12}$$

From formulas (11)-(13), the simulation sensitivity of each sub-module and that of the overall model composed of these sub-modules can be obtained.

6. EXPERIMENTS AND RESULT ANALYSIS

The CS-based VS teaching system must be able to store the important simulation data in the teaching process in a continuous manner. When the VS teaching system crashes due to software or hardware problems, the data of the continuous storage unit will be read to restore the previous system state.

In this paper, a simulation system is built upon the MySQL cloud database. The simulation data were extracted, stored, and managed through the object-relational mapping (ORM) under the Hypertext Transfer Protocol (HTTP) framework for the rapid development of Go applications. The key information is about developer and user, simulation elements, sub-modules, and mirror database. In addition, the CS-based VS platform was also constructed. Some data are listed in Tables 1-3.

As shown in Table 1, the cloud database for our VS teaching system was designed from the perspectives of ID, name, address, service structure, version, and domain. To ensure the performance of cloud mirror service, an independent storage space was allocated for the mirror data, and a mirror address was allocated by the Docker container. Furthermore, the authors also defined the types of sub-modules required for cloud simulation task, as well as the total number of parallel execution tasks that characterizes the simulation capability of the simulation system.

The proposed CS-based VS teaching system was compared with VMware, kernel-based Virtual Machine (KVM), QEMU, common language runtime (CLR) of .Net, and Docker. To ensure the accuracy of the comparison, the simulation model of three sub-modules was tested five times under each environment. The mean value of the five tests was taken as the final result under the corresponding environment.

Table 1. The cloud database of the VS teaching system

Type	Format	Size	Null	Description
ID	Integer	15	/	Database function
Name	Character	20	/	Name database
Address	Character	26	/	Database IP address
Service interface	Integer	20	/	Connection ports between database and other service modules
Version	Integer	5	/	Database version
Field	Character	5	/	Database type

Note: ID and IP are short for identity and Internet protocol, respectively.

Table 2. The cloud mirror data

Type	Format	Size	Invalid	Description
ID	Integer	15	/	Mirror function
Name	Character	20	/	Mirror name
Address	Character	26	/	Mirror address assigned by Docker container
Size	Integer	15	/	Mirror data capacity
Date	Long date format	5	/	Mirror creation time
Type	Integer	5	/	Mirror type
Class	Integer	5	/	Mirror tool, federal class
Storage space	Character	15	/	Mirror data storage space

Table 3. The data on cloud simulation task

Type	Format	Size	Null	Description
ID	Integer	15	/	Task function
Name	Character	20	/	Task name
Number of tasks	Integer	10	/	Number of tasks
Creation time	Long date format	5	/	Task creation time
Execution time	Long date format	20	/	Task execution time
Type	Integer	5	/	Task type
Class	Integer	5	/	Task module class

Figure 8 compares the startup time under different frameworks. Obviously, the startup time of the simulation program varied significantly between environments. The startup time under KVM and QEMU was 12-14 times that under Docker. The proposed system consumed a relatively short time to start up, only slightly longer than the startup time under Docker. But our system is simple and efficient, as it maps the dynamic model into Docker containers. The high simplicity and efficiency offset the slightly longer startup time.

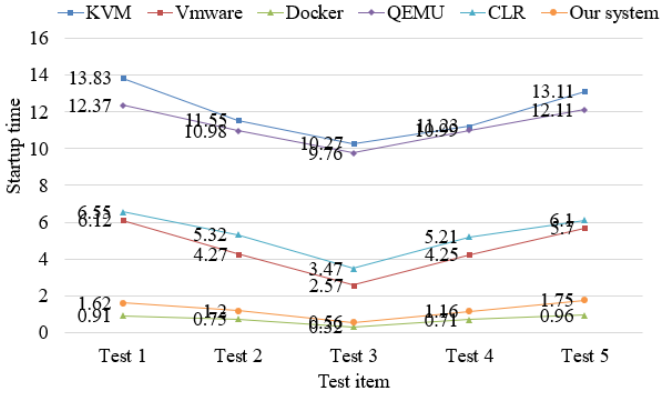

Figure 8. The comparison of startup time

In addition, a simple first-order automatic control system was simulated under the above environments in the presence of disturbance. Figures 9 and 10 compares the CPU utilization and memory utilization under different environments. It can be seen that the system did not consume much CPU or memory resources under the proposed CS-based VS teaching platform, while facilitating the management and execution of simulation resources and sub-modules.

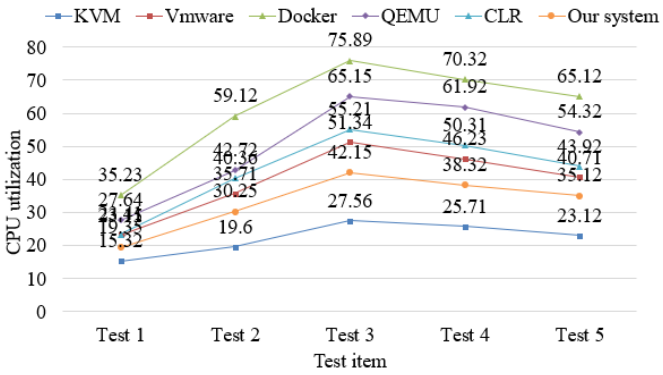

Figure 9. The comparison of CPU utilization
Note: CPU is short for central processing unit.

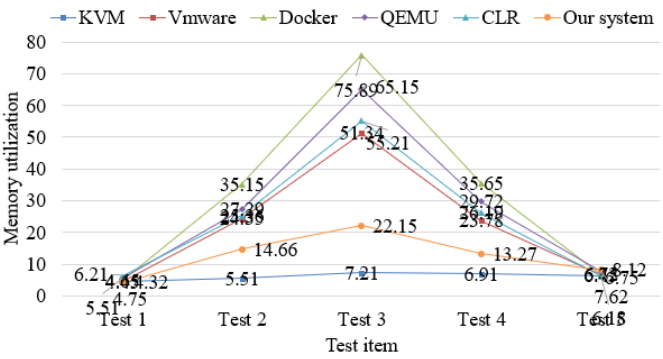

Figure 10. The comparison of memory utilization

7. CONCLUSIONS

This paper designs an innovative CS-based VS teaching system. Firstly, the functional requirements on the CS-based VS teaching system were analyzed, and decomposed into such three layers as goal, function type, and function name. On this basis, a dynamic model was established for the CS-based VS teaching system. The simulation accuracy and precision of the model were verified in multiple dimensions. To verify the performance of the proposed system, a simulation system was built upon the MySQL cloud database, the simulation data were extracted, stored, and managed through ORM under the HTTP framework for the rapid development of Go applications, and the CS-based VS teaching platform was constructed. The experimental results confirm that the proposed system not only boasts good simulation performance, but also enjoys advantages in simplicity and efficiency, owing to the mapping of the dynamic model to Docker containers. The startup time, CPU utilization, and memory utilization of our system were all controlled at low levels. The research results provide a reference for the application of CS-based system design in other fields.

REFERENCES

[1] Kim, J.W., Choi, J.W., Lee, I.W. (2016). Design and simulation of an optimal microgrid system. 2016 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, pp. 1061-1064. https://doi.org/10.1109/ICTC.2016.7763368

[2] Zeraatkar, N., Farahani, M.H., Rahmim, A., Sarkar, S., Ay, M.R. (2016). Design and assessment of a novel SPECT system for desktop open-gantry imaging of small animals: A simulation study. Medical Physics, 43(5): 2581-2597. https://doi.org/10.1118/1.4947127

[3] Baghli, F.Z., El Bakkali, L. (2016). Design and simulation of robot manipulator position control system based on adaptive fuzzy PID controller. Robotics and Mechatronics, 243-250. https://doi.org/10.1007/978-3-319-22368-1_24

[4] Alavinia, R., Zhu, Z., Zhang, S. (2016). Design and simulation of a meteorological data monitoring system based on a wireless sensor. International Journal of Online and Biomedical Engineering (iJOE), 12(5): 27-32. http://dx.doi.org/10.3991/ijoe.v12i05.5733

[5] Mahmoud, A.E., Mahmoud, O.E., Fatouh, M. (2016). Development of design optimized simulation tool for water desalination system. Desalination, 398: 157-164. https://doi.org/10.1016/j.desal.2016.07.022

[6] Singh, A., Ramesh, M.V., Divya, P. (2016). Design and simulation of elephant intrusion detection system. 2016 6th International Workshop on Computer Science and Engineering, WCSE, Tokyo, Japan, pp. 749-753.

[7] El-Nemr, M.K., Omara, A.M. (2016). Design and simulation of a battery powered electric vehicle implementing two different propulsion system configurations. 2016 Eighteenth International Middle East Power Systems Conference (MEPCON), Cairo, pp. 877-882. https://doi.org/10.1109/MEPCON.2016.7836999

[8] Ijagbemi, C.O., Oladapo, B.I., Campbell, H.M., Ijagbemi, C.O. (2016). Design and simulation of fatigue analysis for a vehicle suspension system (VSS) and its effect on global warming. Procedia Engineering, 159: 124-132. https://doi.org/10.1016/j.proeng.2016.08.135

[9] Ekinci, S., Demirören, A. (2016). Modeling, simulation, and optimal design of power system stabilizers using ABC algorithm. Turkish Journal of Electrical Engineering & Computer Sciences, 24(3): 1532-1546.

[10] Jeong, S., Lee, M.S., Cho, K., Jang, Y.J. (2016). System

model and simulation for optimal parameter design of dynamic wireless charging EVs. 2016 IEEE Transportation Electrification Conference and Expo, Asia-Pacific (ITEC Asia-Pacific), Busan, pp. 082-089. https://doi.org/10.1109/ITEC-AP.2016.7512927

[11] Hong, J.H., Ryu, K.R. (2017). Simulation-based multimodal optimization of decoy system design using an archived noise-tolerant genetic algorithm. Engineering Applications of Artificial Intelligence, 65: 230-239. https://doi.org/10.1016/j.engappai.2017.07.026

[12] Modaresi, A., Safikhani, A., Noohi, A.M.S., Hamidnezhad, N., Maki, S.M. (2017). Gating system design and simulation of gray iron casting to eliminate oxide layers caused by turbulence. International Journal of Metalcasting, 11(2): 328-339. https://doi.org/10.1007/s40962-016-0061-3

[13] Liu, W., Liu, S. (2017). Design of medical track logistics transmission and simulation system based on internet of things. Acta Technica, 62(1): 207-220.

[14] Cheshmehbeigi, H.M., Khanmohamadian, A. (2017). Design and simulation of a moving-magnet-type linear synchronous motor for electromagnetic launch system. International Journal of Engineering, 30(3): 351-356.

[15] Avital, E.J., Salvatore, E., Munjiza, A., Suponitsky, V., Plant, D., Laberge, M. (2017). Flow design and simulation of a gas compression system for hydrogen fusion energy production. Fluid Dynamics Research, 49(4): 045504. https://doi.org/10.1088/1873-7005/aa73ba

[16] Aguilar-Gonzalez, A., Nolazco-Flores, J.A., Vargas-Rosales, C., Bustos, R. (2017). Characterisation, design and simulation of an efficient peer-to-peer content distribution system for enterprise networks. Peer-to-Peer Networking and Applications, 10(1): 122-137. https://doi.org/10.1007/s12083-015-0412-5

[17] Hashemipour, M., Stuban, S., Dever, J. (2018). A disaster multiagent coordination simulation system to evaluate the design of a first-response team. Systems Engineering, 21(4): 322-344. https://doi.org/10.1002/sys.21437

[18] Savastru, D., Miclos, S., Savastru, R., Lancranjan, I. (2018). Simulation and design of a LPGFS system for detection of Escherichia coli bacteria infestation in milk. Journal of Optoelectronics and Advanced Materials, 610-617.

[19] Nasri, F., Alqurashi, F., Nciri, R., Ali, C. (2018). Design and simulation of a novel solar air-conditioning system coupled with solar chimney. Sustainable Cities and Society, 40: 667-676. https://doi.org/10.1016/j.scs.2018.04.012

[20] Schaefer, R., Wesuls, J.H., Köckritz, O., Korte, H., Windeck, K.J. (2018). A mobile manoeuvring simulation system for design, verification and validation of marine automation systems. IFAC-PapersOnLine, 51(29): 195-200. https://doi.org/10.1016/j.ifacol.2018.09.492

[21] Bajpai, A., Ghasemi, M., Song, X. (2019). Design and simulation of a lab-scale down-hole drilling system. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 233(8): 2591-2598. https://doi.org/10.1177/0954406218797978

[22] Loukriz, A., Messalti, S., Harrag, A. (2019). Design, simulation, and hardware implementation of novel optimum operating point tracker of PV system using adaptive step size. The International Journal of Advanced Manufacturing Technology, 101(5-8): 1671-1680. https://doi.org/10.1007/s00170-018-2977-7

[23] Maeda, M., Furutaka, K., Kureta, M., Ohzu, A., Komeda, M., Toh, Y. (2019). Simulation study on the design of nondestructive measurement system using fast neutron direct interrogation method to nuclear materials in fuel debris. Journal of Nuclear Science and Technology, 56(7): 617-628. https://doi.org/10.1080/00223131.2019.1611500

[24] Ghosh, A., Jain, A., Sarkar, S.K. (2019). Design and simulation of nanoelectronic data transfer system with an emphasis on reliability and stability analysis. Analog Integrated Circuits and Signal Processing, 101(1): 13-21. https://doi.org/10.1007/s10470-018-01384-9

[25] Ferraresi, C., De Benedictis, C., Maffiodo, D., Franco, W., Messere, A., Pertusio, R., Roatta, S. (2019). Design and simulation of a novel pneumotronic system aimed to the investigation of vascular phenomena induced by limb compression. Journal of Bionic Engineering, 16(3): 550-562. https://doi.org/10.1007/s42235-019-0045-0