Software Environment for Learning Continuous System Simulation

Marijana Despotović-Zrakić[1], Dušan Barać[1], Zorica Bogdanović[1], Branislav Jovanić[2], Božidar Radenković[1]

[1]Faculty of Organizational Sciences, University of Belgrade, Jove Ilića 154, 11000 Belgrade, Serbia, maja@fon.bg.ac.rs, barac.dusan@fon.bg.ac.rs, zoricab@fon.bg.ac.rs, boza@fon.bg.ac.rs

[2]Institute of Physics, University of Belgrade, Pregrevica 118, 11000 Belgrade, Serbia, brana@ipb.ac.rs

Abstract: In this paper, we describe a new graphical environment for learning continuous simulation. This open source software solution enables engineering students to learn and understand various real system models in the CSMP (Continuous Simulation Modelling Programme) language easily, and at the same time learn about the implementation aspects of simulation languages. The software provides efficient tools for creating, storing, and executing continuous system simulation models that are described with differential equations. The graphical block-diagram interface allows students to drag-and-drop predefined modelling blocks, connect them together and create different models of continuous systems. Further, we present an example of using the developed application in solving a typical continuous simulation problem. Finally, we present the results of the evaluation performed on a testing sample of 160 students of undergraduate course Simulation and simulation languages at the Faculty of Organizational Sciences, University of Belgrade.

Keywords: Continuous system simulation; teaching simulation and simulation languages; CSMP software

1 Introduction

In this study, the authors set out to develop a software tool that would effectively support teaching and learning continuous simulation and simulation languages. The focus is on creating an open source tool that would provide an environment for learning continuous systems modelling and simulation as effectively as using commercial software such as Matlab/Simulink, but, unlike commercial tools, provide a testbed for learning simulation languages and the concepts of their implementation.

Simulation is used to describe the behaviour of a dynamic system by means of a model and using this validated model in a series of experiments designed to provide an insight into the future behaviour of the system under specific conditions [1]. The area of computer simulation has been successfully applied to the study and modelling of processes, applications, and real-world objects. Simulation tools provide an environment for the analysis of various features of the system: feasibility, behaviour, stability, performance, etc. [2], [3], [4]. In addition to its many practical applications in industry, commerce, research, and other areas, computer simulation can also be used as a powerful teaching aid [5], [3]. Computer simulation gives the freedom and flexibility to adjust various parameters of the system before design [6]. In some cases, simulation models can substitute real systems that are too expensive or unsafe to install in teaching laboratories [1]. In addition, using simulation is sometimes a requirement, e.g. when designing a manufacturing prototype [7].

All physical systems exist in the time-space continuum. The type of model one selects depends on the degree of aggregation of individual phenomena. One may choose either a continuous or a discrete approach to model development. Continuous models are useful when the behaviour of the system depends more on the aggregate flow of events than upon the occurrence of individual events [5]. A discrete system is one in which the state variables change only at a discrete set of points in time, while in a continuous system the state variables change continuously over time.

1.1 Continuous Simulation e-Learning Course

Simulations play a vital role in engineering education, especially in laboratory exercises [8]. Computer simulation enables engineering students to perform the analysis of various types of systems, investigate relations and interactions among a system's components, project design, and implementation of a system, verify different solutions and environments, test capabilities and technical characteristics [9]. The knowledge gained from simulation courses help students understand, analyze and design systems, develop software components, test software solutions, and solve different real life problems [10], [11].

There are lot of specialized languages for continuous simulation, such as: CSMP (Continous Simulation Modelling Programme) [12], [13], ESL (European Simulation Language) [14], ACSL (Advanced Continuous Simulation Language) [15], CSSL4 (Continuous System Simulation Language 4) [16], Simulink [17], [18], Matlab [19], Modelica [20] and others that have been developed to simplify modelling, and to minimize problems related to programming continuous systems [21]. In the past years, there has been much research dealing with developing tools, languages, and methods for continuous simulation [6], [15], [22], [23], [24], [25]. Using Matlab/SIMULINK for designing simulation models is one of the

most frequent scenarios [6]. Matlab is one of the most powerful simulation platforms as it provides a plethora of features [26]. However, a majority of simulation tools have limitations such as high costs, platform dependence, maintenance difficulties, and limited reusability.

Computer simulation has been studied for many years at the Faculty of Organizational Sciences, University of Belgrade, in the scope of the Simulation and simulation languages course. The course is organized at the fourth year of undergraduate studies in the area of information systems and Internet technologies. Before attending this course, students are obliged to take several exams in the field of programming and are familiar with programming concepts and several programming languages (Java, C). The main goal of the course is to introduce the basic concepts and applications of computer simulation and simulation languages.

The course is realized using the combination of traditional classroom-based teaching and e-learning technologies, i.e. blended learning [27]. As a support to the teaching process, we use a learning management system Moodle [28]. The course lasts for three months. The course syllabus is divided into three main blocks: continuous simulation, discrete event simulation and 3D simulation. All the topics are covered by practical exercises using an appropriate simulation language: CSMP/FON (Faculty of Organizational Sciences - in the Serbian language "*Fakultet Organizacionih Nauka*"), GPSS (General Purpose Simulation System), X3D (XML-based file format for representing 3D objects), respectively. Students are obliged to implement their own models and solutions to different problems in the simulation of complex, real systems. In the part of the course that deals with continuous simulation, students model systems from the area of: motion dynamics, electrical engineering, mechanics, fluids, social phenomena.

Solving real cases by means of specialized software helps students to be highly motivated and contributes to overcoming possible deficiencies in their mathematical background. Many of the simulation programmes based on mathematical models have to be used by experts. These programmes cannot be used for providing basic knowledge or gaining experience to engineers or users who are not experts in this subject. On the other hand, an important part of the course curriculum is related to design and implementation of the simulation languages.

Therefore, the main goal of this research was to develop an efficient software tool for creating, storing, and executing continuous system simulation models. This software tool will provide students with a cheap environment for practice, particularly suited for students in the areas of information technologies and software engineering. The idea was to use all the existing features of CSMP and enrich it with interactive graphic environment. The main requirements in the design and implementation of the software environment for learning continuous system simulation were the following:

Create a testbed for learning and realization of continuous simulation languages – the main idea behind the CSMP/FON is to provide students with an environment for both using the simulation language features and learning how to implement key continuous simulation concepts within a simulation language. These main simulation concepts should include emulation of parallelism, solving problems related to algebraic loops, blocks, implicit functions, etc. [4]. Using the CSMP/FON students are enabled to learn more about the structure of the CSMP language and how the CSMP implements all the above-mentioned concepts. As an open source and free solution the CSMP/FON can be extended and modified according to any kind of request. For instance, each student can implement an integration method of their own choosing and perform simulation using the implemented method.

Matlab/Simulink, as the most widely used solution on the market, provides numerous features, but the source code is not open. For its users, blocks are provided as "black boxes", so users cannot change or adapt anything. Accordingly, users can use advanced features related to simulation, but they cannot learn anything about how the simulation language was realized. Therefore, Matlab/Simulink is not entirely adequate for a course like this.

Support for scientific research – scientific research within our laboratory often requires creating complex models that include both continuous and discrete event simulation. In order to combine different simulation models effectively, we need simulation environments that are interoperable, opened for modifications, and easily adjustable for specific problems.

Saving costs - One of the main reasons for implementing our own version of the CSMP was related to the cost of licenses for commercial CSMP and Simulink implementations. Licenses would have to be provided for more than 100 computers at the Faculty of Organizational Sciences. Compared to Matlab/SIMULINK, the CSMP/FON has almost the same features. However, we made the CSMP/FON free and open source; therefore, the application of this software in teaching and learning is cheap and every student can easily learn about the simulation of complex systems on a personal computer.

Simple and rich user interface - In the CSMP/FON we implemented all the newest concepts that are used in the modern software design: simplicity, clarity, responsiveness, efficiency, richness of colours and images, etc. The user interface of the CSMP/FON can be adapted and changed according to the users' needs. In addition, the architecture of the application is designed in order to enable an easy integration of CSMP/FON functionalities in the web-based simulation solutions. Although Matlab provides quite clear interface, it is much less user-friendly, its design is old-fashioned and does not follow any of the abovementioned concepts.

Suitable and easy to use for educational purpose - while Matlab/Simulink provides students with various features and possibilities, a research showed [26] that it is not suitable for beginners in the area of simulation. Using

Matlab/Simulink requires good comprehension of numerical analysis, linear algebra and many mathematical functions, which makes this solution quite complex and inappropriate for our students. The CSMP/FON abstracts mathematical issues and enables students to learn basic simulation concepts without the need to have advanced knowledge in the area of mathematics. Considering suitability of other solutions for learning simulation, for instance, in [29] the authors introduce a comprehensive object-oriented, distributed, and extendable research solution for business simulation, named DSOL. Numerous features and simulation concepts are provided. Further, this software focuses on linking simulations to business information systems, such as ERP systems and databases. The main constraint of this solution is that it requires advanced knowledge of the Java programming language. Further, GUI features are not rich enough for learning basic simulation concepts. Accordingly, the DSOL is not entirely suitable for teaching and learning simulation languages.

Minimize required hardware resources and improve speed of execution – Matlab/Simulink provides a wide range of advanced features and services, but consequently requires powerful hardware infrastructure. Further, our experience with simulation in Matlab/Simulink showed that it could be time consuming. The CSMP/FON is a lightweight software solution that minimizes hardware consumption and optimizes the time of simulation execution.

2 CSMP/FON Application

The CSMP simulation language can be classified into the group of block-oriented languages for solving systems of differential equations. Each block is specified by a set of inputs and parameters and a graphic symbol. The graphic display of elements in the general form is presented in Fig. 1. Each available element specifies the relation of three input variables e_1, e_2, e_3 and three parameters p_1, p_2, p_3. The output is a scalar whose value depends on the concrete relation f for that element $e_0=f(e_1, e_2, e_3, p_1, p_2, p_3)$.

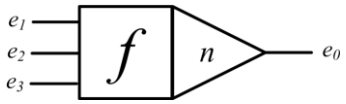

Figure 1
Graphic display of elements in the general form

The CSMP was developed by IBM in the early 1960s and it represented a real analogue simulator [15]. The language development of the CSMP II and later the CSMP III was followed by hardware development in IBM. These programmes could not be used interactively nor simultaneously from different platforms. These

shortages were eliminated in an interactive implementation of simulation language CSMP/FON. The CSMP/FON implements the simulation concepts introduced by IBM's CSMP, but all the libraries and classes in our solution were implemented from the scratch. GUI elements are completely new and adapted for teaching and learning simulation. In this implementation of the CSMP/FON a user-friendly graphical user interface was developed, and the efficiency of the programme improved. The CSMP/FON is an open source solution available to the students and can be downloaded from the web site http://www.elab.rs/laboratorija-za-simulaciju/. The CSMP/FON simulation tool allows users to develop complete, interactive simulations in three steps: describing the mathematical model, building the user interface using off-the-shelf graphical elements and connecting certain properties of these elements to the variables of the model. Operating principle of the CSMP/FON is shown in Fig. 2.

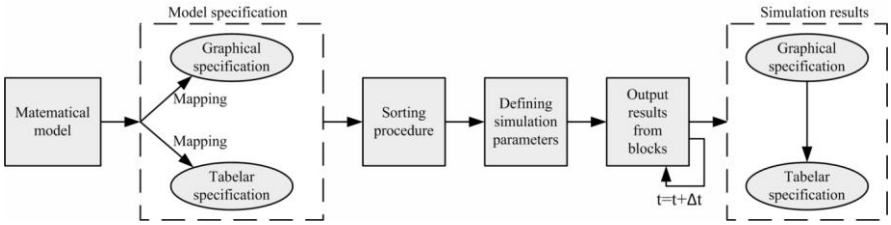

Figure 2
How CSMP/FON works

The programme's engine is based on an algorithm for sorting ordinal numeral of blocks. It gives an order needed to calculate the output values from every block and after each integration interval the output values for every block are known. We use the Runge Kutta IV method for integration.

The CSMP/FON provides an integrated and user-friendly environment for creating, testing, and analyzing continuous system models. It enables students to configure simulation models, execute simulation, and analyze results. Particular benefits from using the CSMP/FON include an improved performance of teaching activities, integration of activities in teaching and learning processes, learning simulation on a variety of real models, learning about the implementation of a continuous simulation language. An interactive environment with resident editor, processor and result analyzer, a fast and easy model debugging, different views and analyses of simulation results are some of the most important features of the application. Users are allowed to interact with a simulation, monitor values of variables, change parameter values, and rerun the simulation, redefine output requirements, etc.

2.1 CSMP/FON Architecture

The architecture and key components of the CSMP/FON application are presented in Fig. 3. The components are explained in the following text.

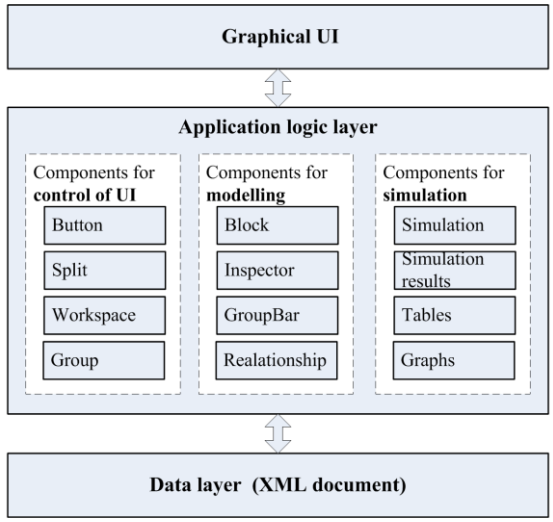

Figure 3
Conceptual model of CSMP/FON software solution

The architecture of the CSMP/FON solution is based on the MVC pattern (Model View Controller) [30]. Each component in the application logic layer in Figure 3 is implemented through appropriate MVC classes.

2.1.1 User Interface

A software application for learning simulation, in addition to the model, should have an interface for displaying the simulation in a proper way, and a facility to change the parameters while the model is running to provide dynamic control. The assigning of values to parameters, the specification of initial conditions, and control during the execution need to be achieved in ways that directly correspond to the concepts or procedures that are to be learned [4]. The user interface shown later in Fig. 5 is one of the key advantages of the CSMP/FON in comparison with similar solutions. In essence, the user interface of the application is a collection of graphical windows whose components are active, dynamic, and clickable. Students can change any active element in the graphical interface, the recalculation and dynamic presentation are immediate. In this way students instantly perceive how their modifications affect the model.

The graphical interface contains three basic menus: File, Edit, Help; and a toolbox with eight buttons for model manipulation. On the left side of the main screen there is a list of available blocks. Blocks are categorized in groups: Mathematical elements, Trigonometry, Generators, Limiters and Other. The user creates a model by dragging and dropping blocks from the list to the workspace, and then connects blocks using options from the toolbox. After the block is dropped, a form for input parameters of the specific block is shown. Each block has a predefined number of parameters (min 1, max 3). When the model is created, the user chooses the option for starting the simulation from the toolbox, and sets simulation parameters: the length of the simulation, the integration interval, and the interval for printing the results. The simulation results are shown in the form of tables and graphs.

2.1.2 Application Logic

The source code of the application is grouped into logical parts: components for control of user interface, components for modelling, and components for simulation (Fig. 3).

Components for control of user interface implement functionalities of the standard Windows forms buttons with some handy visual add-ons, functionalities for visual split of controls and space saving. There is a container for any type of a control that enables an easy hiding and showing of controls. A visual representation of controls can be changed during the run time. "Workspace" is a component that represents the canvas on which the model is drawn. This component handles most of the work related to visual representation and editing of the model. We use workspace as a canvas to draw CSMP blocks and relationships.

Components for modelling implement functionalities for creating and saving block diagrams. "Block" is the main component of this programme and it handles the graphical representation of any element of the CSMP language. This component is abstract and extensible, therefore programming new blocks with new properties and parameters is as simple as changing the elements of an XML file. The XML file Manifest.xml contains the properties of each block in the model and keeps the information about the visual grouping of blocks. Using this approach we achieved extensibility of the application. For each CSMP block in the model an object (named TCSMPBlock) is instanced. This object holds the properties and parameters of the block, information about all of the input and output connections, as well as the values calculated during the simulation. We can move blocks across the workspace to achieve a better representation of the model. When moving blocks, the position of all input and output connections is preserved. "Connections" is a component for managing the graphical display of connections between blocks. To achieve a "breaking" of the line, we introduced helper objects called "holders". We pin the holder on a line and create a polyline. These holders are also used to create junctions (a connection between a

relationship and a block). Junctions are used when there already is an output link from a block and we want to create another one. The CSMPInspector is a component for displaying and editing properties of the selected CSMP block.

Part of the application that handles the simulation resides in a component titled "**Simulation**". This part of the programme prepares the model for simulation, checks the model for consistency, and represents the "brain" responsible for the simulation itself. The program is implemented in Delphi on Win32 platform.

2.1.3 Data Layer

Models created using the CSMP/FON are stored in XML files. XML is widely used and it is de facto a standard in modern software architectures. The structure of this XML file (with the extension .CSMP) is shown in Fig. 4:

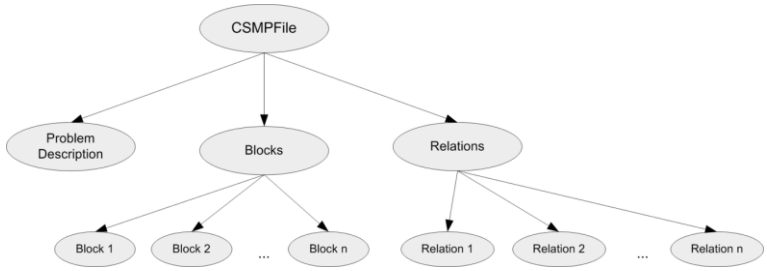

Figure 4
Structure of the CSMP file

Each CSMP model is described using the <block> XML tag. Each block is described through the following tags: <block_id>, <name>, <parameters>, <inputs>. Using the XML technology for storing models enables platform independence, reading and editing the models in any software tool that supports XML, providing compatibility of models developed using old or future versions of the application. Further, XML notation enables an integration of simulation models in web-based solutions.

3 Using CSMP/FON in Simulation of a Spacecraft Landing

3.1 Problem Description

In this problem, we discuss a simplified model of vertical landing of a spacecraft on the surface of the Moon. The total mass of the spacecraft is labelled as M and

consists of two elements: the fixed mass of spacecraft (m_s) and the variable mass of fuel in the spacecraft (m_f). The acceleration of the spacecraft can be represented with the well-known formula shown in Equation (1).

$$M \cdot a = -\frac{g\left(m_s + m_f\right)}{\left(1 + s/r\right)^2} + g_p * g_f \tag{1}$$

where: g – gravitational acceleration on the Moon 1.62ms^{-2}; r – Moon radius (1 738000 m); m_s – fixed mass of the spacecraft (15000 kg); m_f – variable mass of fuel in the spacecraft (initial value 1000 kg); s – distance of the spacecraft from the initial position; g_p – fuel consumption (kg s^{-1}); g_f – factor for conversion of fuel consumption in units of force (4 000 Ns kg^{-1}).

The change in the mass of fuel in the spacecraft is equal to the negative value of variable g_p that depends on time t and can be calculated using the formula (2).

$$g_p = 2 + 1{,}5t \tag{2}$$

If the mass of the fuel drops to zero, fuel consumption must also be equal to zero. The spacecraft stops when it lands on the Moon surface. In the beginning of the simulation, the distance from the Moon surface is 5000 m.

3.2 Solution

The total mass of the spacecraft M is the sum of fixed mass of the spacecraft m_s and variable mass of the fuel in the spacecraft m_f. Since the acceleration of the spacecraft can be presented as the second derivative of the distance of the spacecraft, we can transform (1) into (3). Equation (3) is suitable for creating a block diagram.

$$\ddot{s} = -\frac{g}{\left(1 + s/r\right)^2} + \frac{g_p * g_f}{m_s + m_f} \tag{3}$$

Using (3) we created a block diagram in CSMP/FON, shown in Fig. 5.

After creating the block diagram, we set up simulation parameters (integration interval and time of simulation) and perform the simulation. The simulation results are shown in graphical format (Fig. 6).

Fig. 6 shows how the distance of the spacecraft from the land changes in time. The initial position of the spacecraft is set to 4000m above the ground. It takes about 38 time units to land. After 38 time units, the simulation is over, because the quit block stops the simulation when the spacecraft touches the ground.

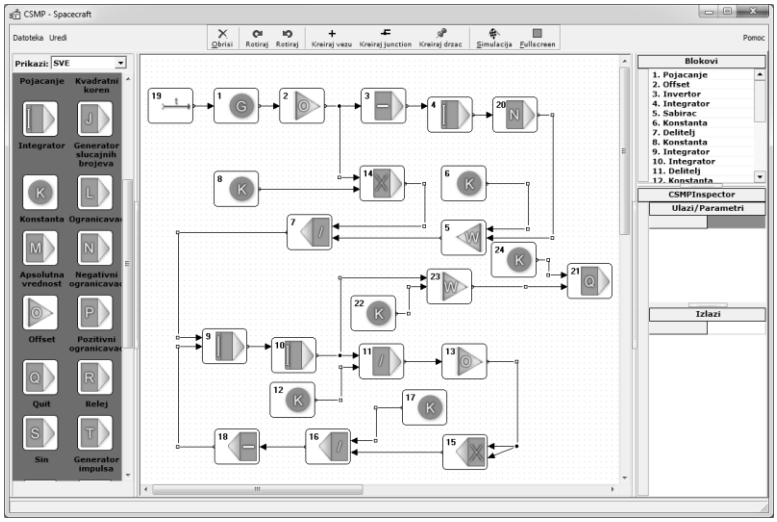

Figure 5
CSMP/FON block diagram

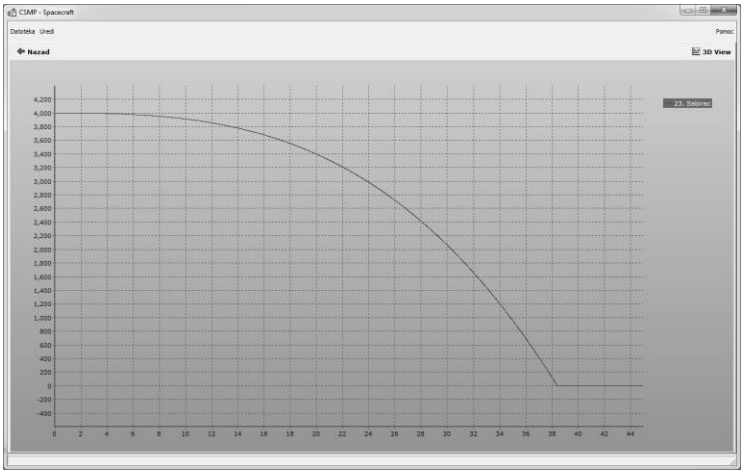

Figure 6
Simulation results showing the distance of the spacecraft

4 Evaluation

This study aims to investigate if the CSMP/FON application can be effectively used to test students' knowledge in the area of continuous simulation and

simulation languages. The experiment was conducted on a sample of 160 undergraduate students of the Faculty of Organizational Sciences, University of Belgrade. Students who attended the Simulation and simulation languages course were randomly divided into experimental (80 students) and control groups (80 students). All students, both experimental and control groups, had a block of lectures presented in the traditional way. Then they attended lab classes where they used the CSMP/FON application to simulate continuous systems. In order to measure the research results, we used the knowledge test that students take at the end of semester. In the knowledge test, students solve problems from the area of continuous systems modelling and simulation. A typical task in the test includes the following requirements: 1. Mathematical modelling of the continuous system described through a verbal model; 2. Creating a block diagram; 3. Creating a table of configuration; 4. Editing a simulation model and adjusting it to specific requirements. The test we applied in the experiment was a standard test used for testing students for more than five years. The students in the experimental group took the final test using the CSMP/FON applications, while the control group students took the final test in the standard form, i.e. on paper. A descriptive comparative statistics of results achieved on the knowledge test is presented in Table1.

Table 1

Descriptive comparative statistics of results achieved on knowledge test

	N	**Mean**	**Std. Deviation**
Experimental group	80	9.26	1.088
Control group	80	8.82	1.271

The distribution of grades in the Experimental and Control groups is shown in Table 2. Grades range from 6 (equivalent to E in ECTS grading scale) to 10 (equivalent to A in ECTS grading scale). We can see that the number of students who achieved high grades is higher in the experimental group. The results of further statistical analysis show that a larger number of students from experimental group that achieved high marks in comparison with the number of students from the control group is statistically significant, $F(1,159)=5.473$ ($p<0.05$).

Table 2

Distribution of grades in Experimental and Control groups

	10	**9**	**8**	**7**	**6**
Experimental group	49	13	9	8	1
Control group	34	18	12	12	4

The described quantitative analysis shows that usage of CSMP/FON within the exam contributed to students' results. This may be explained by the fact that students in the experimental group had a feedback during the exam, but on the other hand, this confirms that this software needs to be included as an integral part

of the course. Also, a qualitative analysis is required in order to make better conclusions.

Therefore, after the test, students in the experimental group were asked to fill in a questionnaire and evaluate the CSMP/FON application. Table 3 shows the students' responses.

Table 3
Questionnaire about CSMP application quality results
(Strongly agree – 5; Agree – 4; Neutral – 3; Do not agree – 2; Strongly disagree - 1)

Question	5	4	3	2	1	Mean score	Std. dev
Using CSMP is simple and clear	70	27.5	1.25	1.25	0	4.66	0.57
User interface for developing simulation model is well designed	45	41.25	10	2.5	1.25	4.26	0.84
User interface for simulation results representation is well designed	35	45	13.75	5	1.25	4.08	0.90
The application contains appropriate information for creating a simulation model	55	37.5	7.5	0	0	4.48	0.64
Information and dialogues during testing and executing simulation models are appropriate	38.75	46.25	12.5	1.25	0	4.24	0.72
Simulation executing speed is satisfactory	70	25	2.50	1.25	1.25	4.61	0.72
I think that user interface suits me	48.75	40	7.5	1.25	1.25	4.35	0.79
The application has enhanced my interest in the area studied within the course	13.75	28.75	33.75	13.75	8.75	3.25	1.14
I have understood the course contents better after using the application	42.50	41.25	11.25	3.75	0	4.24	0.80
While using the application, I have discussed with my colleagues how to create a proper simulation model	28.75	31.25	12.50	10	16.25	3.47	1.43

Mean scores are mainly over 4, so it can be concluded that students are generally satisfied with the CSMP application. However, lower mean values indicate that the application needs to be improved in several aspects: a) the user interface for simulation results should be enhanced. This can be done by implementing better methods for zooming across results, a function for searching results, and a better design. b) The application does not enhance students' interest in the area of continuous simulation. In order to improve this aspect, we should implement some features that support game-based learning and edutainment concepts. c) The application does not encourage collaboration. In further improvements, new features for collaboration need to be implemented.

One of the most important questions was related to the role of the CSMP/FON in understanding the course contents. Since the mean score for this question was

4.24, we can conclude that the usage of this application contributes to students' understanding of continuous systems simulation and simulation languages. This can be explained by the openness and ease of use of the CSMP/FON.

One of the drawbacks of the evaluation presented in this paper is related to the lack of comparison between the results that students achieve with the CSMP/FON and the results achieved with Matlab/Simulink. Additional research is needed to compare how these two software tools affect students' knowledge. On the other hand, the impact of Matlab/Simulink and CSMP/FON on students' knowledge in simulation languages is incomparable, since the former cannot be applied for this purpose.

Conclusion

The main contribution of this work is the development of a new, efficient, and open source environment for learning continuous simulation. The CSMP/FON is a free simulation tool that improves students' understanding of the theoretical concepts of computer simulation and simulation languages. The proposed solution keeps all the existing characteristics of the CSMP language and adds new graphical user interface. The CSMP/FON supports creating interoperable simulation models. This flexible and extensible solution enables developers to implement new simulation algorithms or integration methods easily. The results of the experiment realized in order to evaluate the developed application show that the CSMP/FON application can be used effectively to learn the simulation and simulation languages concepts. This is also in agreement with the fact that the CSMP/FON has already been adopted by other faculties that have a computer simulation in their curriculum, i.e. Faculty of Transport and Traffic Engineering, University of Belgrade.

Finally, we acknowledge some limitations of this study. The CSMP/FON is a desktop application not fully integrated in the e-learning process. Therefore, future researches are directed towards the development of a web CSMP/FON application with the same features and integration of the CSMP/FON into a learning management system. A full coordination of all processes in e-learning will lead to an improved quality of the e-learning system. In addition, we plan to create a library of the most important models and sub models from the area of continuous simulation that can be simply integrated as model's components. Simulation models created in the CSMP/FON should be fully interoperable with analogue models created in other simulation languages.

Acknowledgement

This work was supported by the Ministry of Education, Science, and Technological Development of the Republic of Serbia, grant number 174031.

References

[1] J. Banks: Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, John Wiley & Sons, New Jersey, 1998

[2] J. Banks, J. S. Carson, B. L. Nelson, D. M. Nicol: Discrete-Event System Simulation, 5th Ed. Prentice Hall, New Jersey, 2010

[3] J. Calvo, M. Boada, V. Díaz, E. Olmeda: SIMPERF: SIMULINK-based educational software for vehicle's performance estimation, Computer Applications in Engineering Education, 17(2), pp. 139-147, 2009

[4] R. Granlund, E. Berglund, H. Eriksson: Designing web-based simulation for learning, Future Generation Computer Systems, 17 (2), pp. 171-185, 2000

[5] C. François, K. Ernesto: Continuous System Simulation, Springer-Verlag, New York, 2006

[6] M. Sadiku, M. Tofighi: A tutorial on simulation of queueing models, International Journal of Electrical Engineering Education, 36 (2), pp.102-120, 1999

[7] F. Kentli, H. Çalik: Matlab-Simulink Modelling of 6/4 SRM with Static Data Produced Using Finite Element Method, Acta Polytechnica Hungarica, 8(6), pp. 23-42, 2011

[8] C. Blake, E. Scanlon: Reconsidering simulations in science education at a distance: features of effective use, Journal of Computer Assisted Learning, 23 (6), pp. 491-502, 2007

[9] M. Despotović-Zrakić, D. Barać, Z. Bogdanović, B. Jovanić, B. Radenković: Integration of web based environment for learning discrete simulation in e-learning system, Simulation Modelling Practice and Theory, 27, pp. 17-30, 2012

[10] A. Rodić, G. Mester: The Modeling and Simulation of an Autonomous Quad-Rotor Microcopter in a Virtual Outdoor Scenario, Acta Polytechnica Hungarica, 8(4), pp. 107-122, 2011

[11] M. Badida, R. Králiková, E. Lumnitzer: Modeling and the Use of Simulation Methods for the Design of Lighting Systems, Acta Polytechnica Hungarica, 8(2), pp. 91-102, 2011

[12] F.H. Speckhart, W. L. Green: A guide to using CSMP-the Continuous system modeling program: a program for simulating physical systems, Prentice Hall, New Jersey, 1976

[13] R. D. Brennan, M. Y. Silberberg: Two Continuous System Modeling Programs, IBM Sysems Journal, 6(4), pp.242-266, 1967

[14] R. Nilsen, W. Karplus: Continuous-System Simulation Languages: A State-of-the-Art Survey, Mathematics and Computers in Simulation, 16 (1), pp.17-25, 1974

[15] E. L. Edward, Mitchell: Using a Continuous Simulation Language (ACSL) to Model and Control a Hybrid Simulation, Mathematics and Computers in Simulation, 19 (2), pp.133-140, 1977

[16] J. Strauss, et al: The SCI Continuous System Simulation Language (CSSL), Simulation, 9, pp. 281-303, 1967

[17] A. Demiroren, H. L. Zeynelgil: Modelling and Simulation of Synchronous Machine Transient Analysis using SIMULINK, International Journal of Electrical Engineering Education, 39 (4), pp. 337-346, 2002

[18] The Mathworks Inc: Using SIMULINK, Dynamic System Simulation for MATL AB, 5h Ed., Mathworks Inc, USA, 1997

[19] Z. Ghassemlooy, R. Saatchi: Software Simulation Techniques for Teaching Communication Systems, International Journal o Electrical Engineering Education, 36 (4), pp. 287-297, 1999

[20] S. E. Mattsson, H. Elmqvist, M. Otter: Physical System Modeling with Modelica, Control Engineering Practice, 6, pp. 501-510, 1998

[21] D. Matko, R. Karba, B. Zupancic: Simulation and Modelling of Continuous Systems - A Case Study Approach, Automatica, 30 (11), pp. 1808-1810, 1994

[22] G. Lipovszki, P. Aradi: Simulating Complex Systems and Processes in LabVIEW, Journal of Mathematical Sciences, University of Tokyo, 132 (5), pp. 629-636, 2006

[23] H. Klee: Some Novel Applications of a Continuous System Simulation Language, Computers and Industrial Engineering, 11 (1-4), pp. 385-389, 1986

[24] M. Alfonseca, J. Lara, E. Pulido: An Object-oriented Continuous Simulation Language and Its Use for Training Purposes, Proceedings of 5th International Workshop on Simulation for European Space Programs, Noordwijk, Netherlands, pp. 49-54, 1998

[25] R. Thomas, A. Close, P. McAndrew: Tailoring Simulations for Teaching and Learning: the Potential of the MultiVerse Environment, International Journal o Electrical Engineering Education, 37 (1), pp.1-12, 2000

[26] A. Canizares, Z. Faur: Advantages and Disadvantages of Using Various Computer Tools in Electrical Engineering Courses, IEEE Transactions on Education, 40 (3), pp. 1-7, 1997

[27] M. Despotovic, A. Markovic, Z. Bogdanovic, D. Barac and S. Krco, Providing Adaptivity in Moodle LMS Courses, Educatoional Technology and Society, 15 (1), pp. 326-338, 2012

[28] M. Trenas, J. Ramos, E. Gutierrez, S. Romero, F. Corbera: Use of a New Moodle Module for Improving the Teaching of a Basic Course on Computer Architecture', IEEE Transactions on Education, 54 (2), pp. 222-228, 2011

[29] P. Jacobs, N. Lang, A. Verbraeck: D-SOL; a Distributed Java-based Discrete Event Simulation Architecture, Simulation Conference, 2005 Proceedings of the Winter, Orlando, FL, USA 1, pp. 793-800, 2002

[30] P. Sauter, G. Vögler, G. Specht, T. Flor: A Model–View–Controller Extension for Pervasive Multi-Client User Interfaces, Personal and Ubiquitous Computing, 9 (2), pp 100-107, 2005