Int j simul model 20 (2021) 4, 661-671
Original scientific paper

CODEVS: AN EXTENSION OF DEVS FOR INTEGRATION OF SIMULATION AND MACHINE LEARNING

Kim, B. S.[*]; Kim, T. G.[**] & Choi, S. H.[***,#]

[*] School of Software Convergence, Myongji University, 03674, Seoul, Republic of Korea

[**] School of Electrical Engineering, KAIST, 34141, Daejeon, Republic of Korea

[***] Department of IT Convergence and Application Engineering, Pukyong National University, 48513, Pusan, Republic of Korea

E-Mail: kimbs@mju.ac.kr, tkim@kaist.ac.kr, shchoi@pknu.ac.kr ([#] Corresponding author)

Abstract

When we model a system to analyse it, there are two main methods we can use. First, there are knowledge-based simulation modelling methods using system operations, such as discrete event system specification (DEVS). Conversely, there are data-driven modelling methods using data analysis without explicit system knowledge, such as machine learning. These two models can be used appropriately in situations where it is difficult to model sufficiently with one method, and through this, the advantages of each method can be maximised. In other words, for this, a method is required to specify one system by using two methods at the same time. Therefore, in this paper, we introduce an extension of DEVS formalism, called Cooperative DEVS (CoDEVS), which enables representation of both a simulation model and a machine learning model. It consists of a simulation model, data model, and interface models that convert events between the simulation and data models. We also introduce a modified simulation algorithm that can interpret the new formalism and simulate a distributed file system to show the validity of the proposed work.

(Received in August 2021, accepted in October 2021. This paper was with the authors 1 week for 1 revision.)

Key Words: DEVS Formalism, Cooperative DEVS (CoDEVS), Machine Learning, Data Modelling, Simulation Modelling

1. INTRODUCTION

A complex system can be analysed as many components with many relationships among them, where the behaviour of each component depends on the behaviour of others [1]. Such a system is intrinsically difficult to model due to numerosity, interactions, and hierarchical organisation. Nevertheless, analysis of complex systems is continuously being studied as various internet and communication technologies develop [2]. The results are very important because they help to predict the future behaviour of the system and maximize its performance [3]. In general, two modelling methods can be used to analyse these systems. First, simulation modelling is a knowledge-based approach based on the laws of physical operation of the target system. It is a modelling method that clearly shows the causal relationship between the input and the corresponding output. Discrete Event Systems Specification (DEVS) formalism is a set theoretical specification of discrete event systems that has been widely used for modelling many science and engineering applications [4]. It is hierarchical, modular, and object-oriented, so it is suitable for modelling dynamic systems [5].

The next is a data-driven machine learning method that is modelled through data analysis without explicit knowledge of the system [6]. It can indicate a correlation between data sets, and a model can be built through a learning process that reduces errors between actual data and predicted results [7]. This data-driven machine learning model can be called a data model as opposed to a simulation model [8, 9]. These two methods involve a series of processes to analyse a complex system by making a model of the system.

These two modelling methods have different characteristics in various aspects. The first difference is regarding causality and correlation. One of the limitations of a data model is the

absence of causality representation. Such a model can only express a correlation between two variables, not causality [10]. On the other hand, a simulation model can accurately represent causality. Another difference is regarding predictions under changed conditions. The condition for valid prediction of a data model is to have the same structural operational conditions after learning. However, if the structure, operation, and parameters change after learning, future predictions are invalid. Although data models have these limitations, simulation models are not always better. Because it is not always possible to obtain complete knowledge of the system for meaningful modelling, it may be difficult to make a simulation model complete in all situations. In addition, simulation models show differences in various characteristics, such as analysis level, input type, model fidelity, and decision time [11].

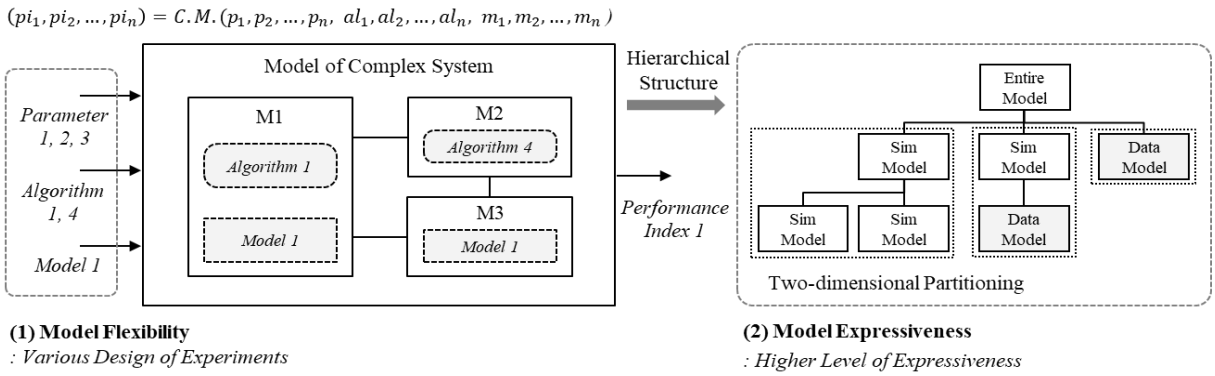


Figure 1: Modelling of a complex system from the perspective of model flexibility and expressiveness.

Furthermore, even if the same system is modelled, the model may vary depending on the analysis objective. For model flexibility, it is necessary to have various inputs as attributes. In other words, depending on the purpose, performance analysis is required according to changes in not only parameters but also algorithms and models, as shown on the left side of Fig. 1. Then, it is important to decide which of the two modelling methods described above to use to model the sub-models of the complex system. For example, a sub-model for parameter change can be expressed as both a data model and a simulation model. However, only the simulation model can be used to experiment by changing the algorithm and model. Because the algorithm and model are part of the simulation model, it is possible to experiment by changing them, but it is difficult to change the data model after it is learned. In other words, it is difficult to model the entire system with only one modelling method in these specific situations because the more complex the model is, the more likely it is to encounter these situations. It is essential to use each model simultaneously and appropriately for improved modelling. It is also necessary to reduce the complexity by identifying and separating the entire system in a modular and hierarchical manner for reliable modelling, as shown on the right side of Fig. 1. Through this two-dimensional partitioning of the entire model, the expressiveness of the model can be increased.

To solve the above-mentioned challenges, an approach to determining which model to apply when system modelling (that is, a method to classify the model) is necessary. Additionally, modelling semantics of how the classified models interact and integrate are needed. In recent years, some studies have been conducted on co-modelling of a simulation model and machine learning in various domains. First, there were studies on the co-modelling methodology of a simulation model and a data model [11, 12]. They specifically compared the characteristics of each model and provided an overall modelling methodology for this purpose [12]. However, although they provided the modelling process, they did not provide integrated modelling semantics that can express it in detail.

Next, some studies modelled network-centric warfare by combining neural networks (data model) with DEVS (simulation model) [13, 14]. Through this, they could reduce the

communication overhead of interoperation by abstracting the simulation model into a neural network model. They extended the DEVS formalism, but it can only be applied to specific domains and simulations. That is, general modelling formalism and interfaces are required. In addition, there have been studies on domain-specific collaboration [15]. In these cases, collaboration is derived in a specific manner depending on the domain characteristics. The researchers utilized simulations and machine learning to model the whole system [16-19]. However, they did not model the entire system separately based on a clear co-modelling methodology.

As we can see from the above, no studies have explicitly presented the aforementioned problems. In other words, a modelling formalism that can specify both methods at once and an algorithm that can interpret it have not been studied. Therefore, we propose an extension of DEVS formalism called Cooperative DEVS (CoDEVS) that enables representation of both machine learning-based data models and simulation models. We also introduce a modified simulation algorithm that can interpret the newly proposed formalism. This paper is organised as follows. Section 2 describes the background and preliminaries. Then, Section 3 provides the proposed CoDEVS formalism. Section 4 discusses the case study of applying the Hadoop Distributed File System (HDFS). Finally, Section 6 concludes the study.

2. PRELIMINARIES

Prior to describing our proposed approach, this section briefly deals with preliminaries about modelling methods. As we said earlier, complex systems cannot be fully modelled through a single modelling approach. For this reason, a cooperative modelling process between simulation and data models was studied based on the general modelling process [12]. It consists of requirements analysis, conceptual modelling, detailed modelling, model integration, verification and validation, experimental design, and results analysis [20].

One of the major differences between the general and the co-modelling process is the conceptual modelling step. In conceptual modelling, we can define the models clearly as functional units and then classify two types of models according to the proposed criteria. In other words, it is necessary to select, according to clear criteria, how to model each derived sub-model. For example, how much system knowledge or actual data is available for modelling can be an important criterion. If the data covers the operating range of the system, then a data model could be a better alternative to a simulation model involving idealistic assumptions and constraints [21]. Model fidelity and efficiency requirements can also be important criteria for classifying sub-models. In this case, the model execution speed can be a measure of model efficiency. When a fast execution speed is required, but relatively high fidelity is not required, it is generally advantageous to use a data model. On the other hand, when high fidelity is required regardless of speed, it can be advantageous to use a simulation model.

After going through the model classification process, the classified conceptual models can be modelled in detail and integrated. Although complex systems can be modelled and analysed through a series of processes, in addition to the modelling process, an explicit formalism for co-modelling is required. Therefore, we suggest a new cooperative modelling formalism, which allows us to develop an enhanced model for complex systems by taking advantage of two modelling methods.

3. CODEVS FORMALISM

In this paper, we propose CoDEVS formalism that extends DEVS formalism. DEVS formalism is hierarchical, modular, and object-oriented, and it largely consists of an atomic DEVS model

representing the system behaviour and a coupled DEVS model representing the structure of the system [4, 5].

Meanwhile, CoDEVS formalism, which includes both simulation and data models, enables hierarchical and modular modelling for performance analysis of complex systems. The proposed formalism consists of a coupled model (CM) and cooperative unit model (CUM). A CM provides the method of assembly of several atomic and/or coupled models to build complex systems hierarchically. A CM is basically identical to the DEVS formalism coupled model, except that it can have a CM and CUM as internal components. Formally, a CM is defined as follows:

$$CM = \langle X, Y, \{M\}, EIC, EOC, IC, SELECT \rangle, \tag{1}$$

where:

X: a set of input events;
Y: a set of output events;
$\{M\}$: a set of all component models; M consists of a CM or CUM;
$EIC \subseteq CM.X \times \cup M.X$, an external input coupling relation;
$EOC \subseteq \cup M.Y \times CM.Y$, an external output coupling relation;
$IC \subseteq M.X \times \cup M.X$, an internal coupling relation;
$SELECT: 2^M - \emptyset \rightarrow M$, a tie-breaking function.

A CUM is a unit model, which is a functional minimum unit. A CUM consists of an atomic simulation model (SM), a data model (DM), and an interface model (IM) that convert data between the simulation and data model, as shown in Fig. 2. A CUM cannot have any child models. There is no limit in the hierarchy except for a CUM. Finally, a CUM is defined as follows:

$$CUM = \langle X, Y, \{M\}, EIC, EOC \rangle, \tag{2}$$

where:

X: a set of input events;
Y: a set of output events;
M: a set of all component models; M consists of an SM, DM, or IM;
$EIC \subseteq CM.X \times \cup M.X$, an external input coupling relation;
$EOC \subseteq \cup M.Y \times CM.Y$, an external output coupling relation.

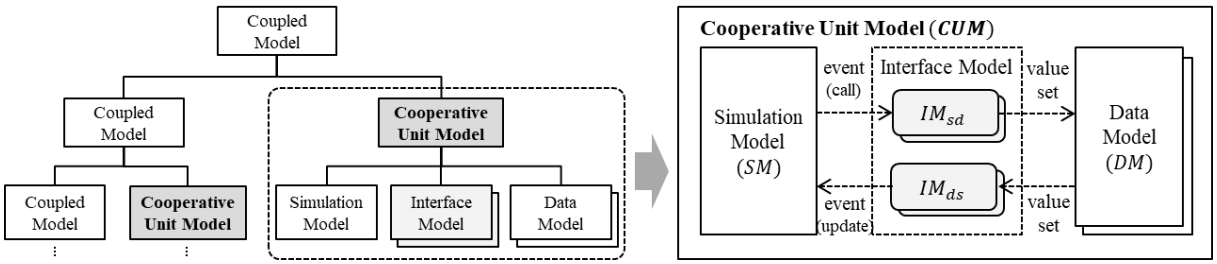


Figure 2: Overall structure of CoDEVS model.

An SM is the basic model and contains the specifications for the dynamics of the model. It is basically identical to the atomic model of DEVS formalism. Formally, a 7-tuple specifies an SM as follows. Additionally, a DM is defined as follows:

$$SM = \langle X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta \rangle, \tag{3}$$

where:

X: a set of input events;
Y: a set of output events;
$\delta_{ext} : S \times X \rightarrow S$, an external transition function;
$\delta_{int} : S \rightarrow S$, an internal transition function;

$\lambda : S \rightarrow S$, an output function;
$ta : S \rightarrow R_{0,\infty}{}^{+}$(non-negative real number), a time advance function.

$$DM = \langle X, Y, f_{dm} \rangle, \tag{4}$$

where:
X: a set of input events;
Y: a set of output events;
$f_{dm}: Y = f_{dm}(X)$, a function of the DM.

Functions in DM can be obtained through data modelling, such as artificial neural networks. The next element is the IM. An IM has two types: IM_{sd} and IM_{ds}. IMs are used to convert from events of an SM to real values of a DM (IM_{sd}), and vice versa (IM_{ds}). Formally, IMs are defined as follows:

$$IM_{sd} = \langle X_s,\ Y_d, dm,\ i_{sd} \rangle, \tag{5}$$

where:
$X_s \subseteq E_{(s,\,t)}$, a set of events (state and function pairs) of SM;
Y_d, a set of real numbers of dm;
dm, the name of the target DM;
$i_{sd}: E \rightarrow R$, an interface for converting from SM to DM,
E is a set of events, and R is a set of real numbers.

$$IM_{ds} = \langle X_d,\ Y_s, dm,\ i_{ds} \rangle, \tag{6}$$

where:
X_d, a set of real numbers of dm;
$Y_s \subseteq E_{(s,\,t)}$, a set of events (state and function pairs) of SM;
dm, the name of the source DM;
$i_{sd}: R \rightarrow E$, an interface for converting from DM to SM; S is a state,
R is a set of real numbers, and E is a set of events.

Fig. 3 a is an example of constructing CUMs using the formalism suggested above. They can be distinguished by the standalone simulation model, standalone data model, and cooperative model. Additionally, they can be modelled independently or combined through coupling relationships. These CUMs can be gathered to complete the entire complex system model. As explained earlier, a simulation model is an abstraction of the system using physical or operational laws. A data model, on the other hand, has three roles, as shown in Fig. 3 b: an output prediction model that changes y to predicted y', a transmission model that delays transmitting with predicted execution time, and a generator model that periodically generates a predicted value.

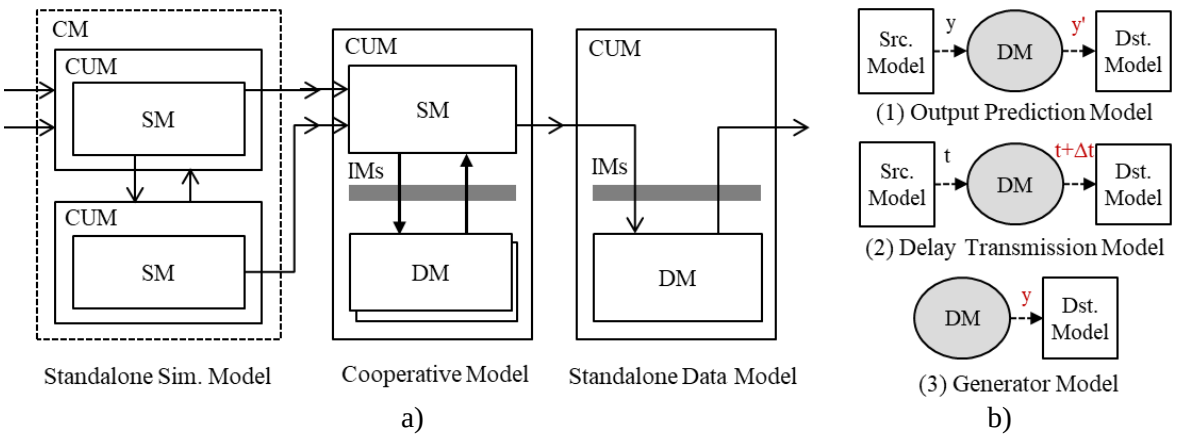


Figure 3: a) forms of model integration, b) roles of data model.

These models have not only different modelling situations and roles but also different algorithms for interpreting their semantics. Therefore, an extended simulation algorithm/tool that can understand the semantics of the proposed CoDEVS for both simulation and data model is necessary. Various tools have been developed to provide a way to implement the existing DEVS formalism using several programming languages [22]. For example, various tools have implemented the DEVS theory, including DEVSim++ [23], CD++ [24], DEVSJAVA [25], and others. DEVSim++, which was developed in C++, has been widely used to implement various DEVS models [26]. It can be used to run a general DEVS simulation model. However, to analyse the simulation model and data model at the same time, DEVSim++ must be interoperated with other tools for data models, or an interface and algorithm for executing the data model must be added to DEVSim++.

In this paper, we modified DEVSim++ instead of directly interoperating with other tools. We briefly introduce it in this section using Fig. 4. The original DEVSim++ was designed without considering a data model. Therefore, when we build a co-model like the one shown on the left of Fig. 4, the data model CUM2 cannot be handled directly inside DEVSim++. Only simulation models CUM1 and CUM3 are directly coupled. That is, to solve this problem, we modified the simulation engine to handle CUM2 with an event going from CUM1.out to CUM3.in [27]. First, CUM2 receives the event outgoing from CUM1.out of the source model; it then performs an operation for calculating a trained data model [28]. At this time, the data model has already been learned through other tools or libraries and then embedded inside the DEVSim++ modelling environment. After that, when an event is updated through the calculation result, CUM2 sends the updated event back to CUM3.in of the target model. Through this modified DEVSim++, two models can be integrated in only one engine.

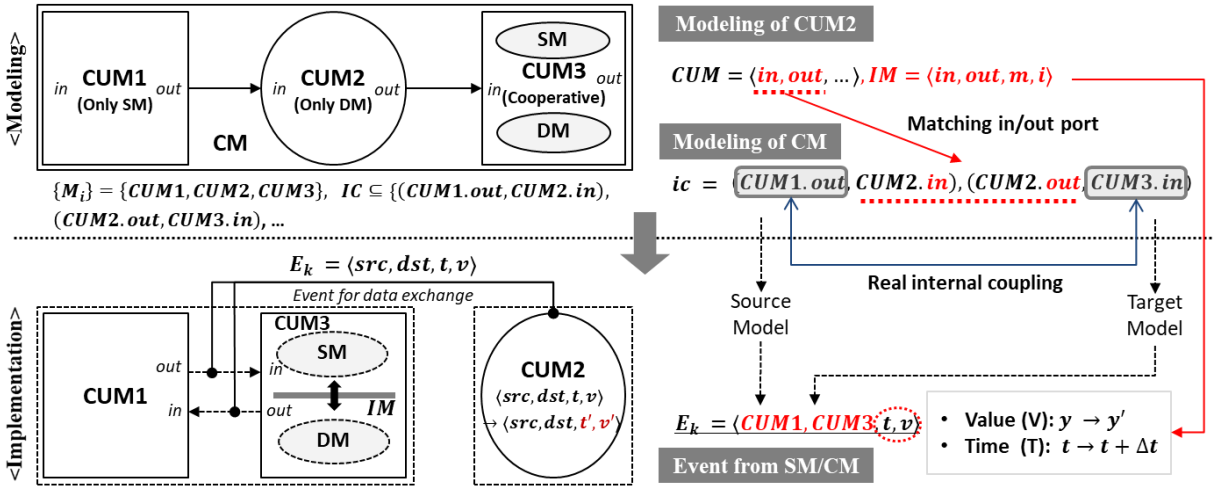
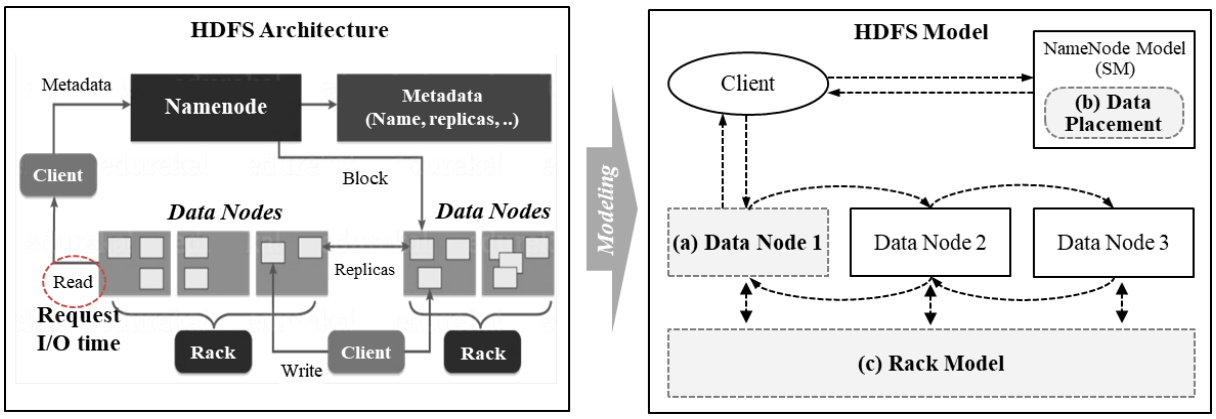


Figure 4: Modified implementation method for CoDEVS.

4. CASE STUDY

In this section, we apply the proposed method to the HDFS to demonstrate the effectiveness of the proposed work. Hadoop is an open-source framework used for distributed storage and processing of big data [29]. HDFS is a distributed file system of Hadoop that stores data reliably using commodity hardware [30]. It has a single name node and a cluster of data nodes. Each data node serves up blocks of data over the network using a block protocol specific to HDFS [29]. It stores the files in blocks of 64 MB, called chunks. It is robust against hardware failure by replicating and storing one block in several blocks.



a) Chunk size of data node (parameter)
b) Data placement algorithm (algorithm)
c) Rack configuration model (model)

Figure 5: HDFS architecture and its abstracted model.

The objective of this case study is to predict the behaviour of a distributed file system based on HDFS in a large cluster [31]. The left side of Fig. 5 shows the overall architecture of HDFS, and the right side shows a simplified model that abstracts it. This HDFS model can be modelled with each modelling approach according to the characteristics of the simulation/data model mentioned previously.

Fig. 6 shows the detailed modelling of the agent model (HCAgentCM), which is one of the data node internal models. HCAgentCM is a CM composed of various internal CUMs. Among them, the HCReaderCUM is a unit model related to file system reading. This corresponds to the cooperative model described above (Fig. 3 a), and because it is difficult to express only with the simulation model, it has a disk data model (DiskDM) related to the reading time inside. At this time, the data model is trained (modelled) using a neural network model based on actual data. We trained the neural network using the pre-processed HDFS data set from the small-sized real HDFS cluster. Then, we constructed the integrated model for the agent by embedding the disk data model in the reader model.

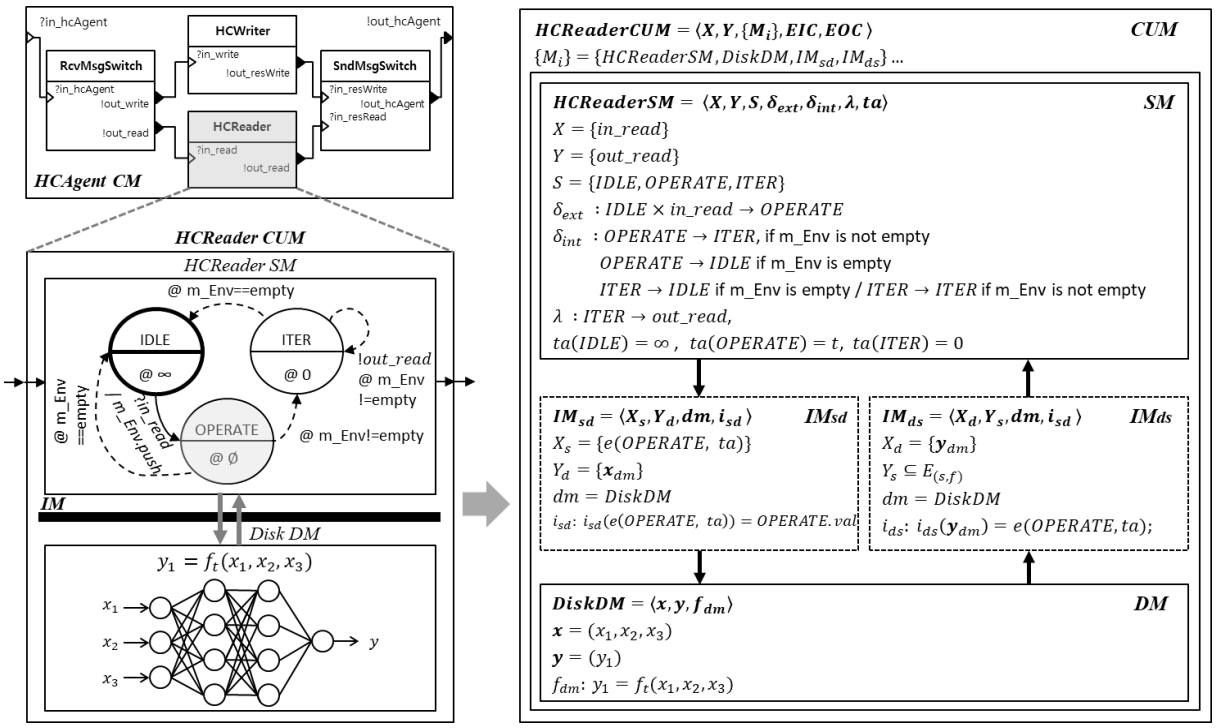


Figure 6: Detailed modelling result using CoDEVS.

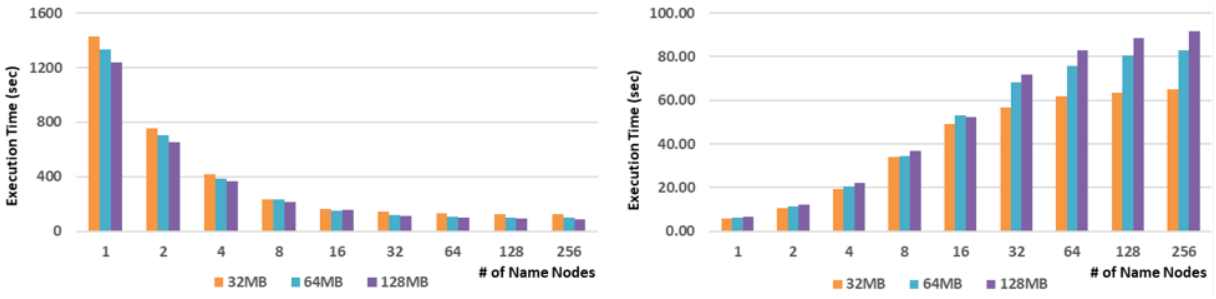
The right side of Fig. 6 shows the results of modelling these models using the proposed CoDEVS formalism. HCReaderSM is a sub-model of HCReaderCUM, a simulation model that simulates the reading behaviours. Additionally, IM_{sd} and IM_{ds} provide an interface between HCReaderSM and DiskDM. The specifications in Fig. 6 show the inputs/outputs, operations, and relationships of each model. The completed HDFS model can be executed using the modified DEVS simulation engine described in Section 3.

Data modelling and simulation modelling generally differ in purpose and features. One of them is related to model flexibility. As discussed earlier, a simulation model can use algorithms, object models, and so on, as well as parameters as inputs. This makes it easy to perform experiments according to the changes of system algorithms or models. However, a data model can reflect only parameter changes. It is difficult to use algorithms or object models as data model inputs. To consider them in a data model, we need to collect new data and perform the data modelling process again. Additionally, it is difficult to analyse the system behaviour, such as failure analysis and topology analysis, with a data model. However, the proposed model makes this possible with high extensibility. Because the model has the advantages of simulation modelling, it can use various types of inputs. In other words, in addition to the numerical parameters, it is possible to simulate the HDFS by changing the algorithms and object models.

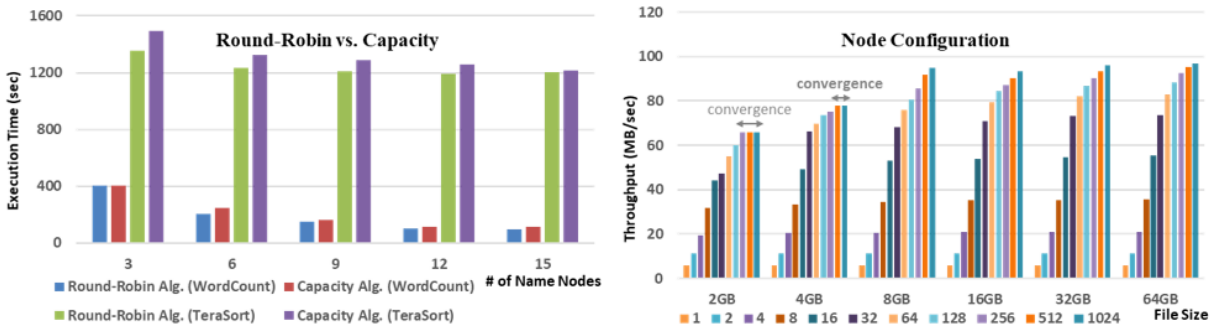
In this case study, we perform the simulations by changing parameters, algorithms, and models. We use the chunk size for the parameter change and the data placement algorithm for the algorithm change. The rack model is used for the model change. This can show that the proposed model is more scalable than a simulation model. Table I shows the experimental design for the model flexibility experiment.

Table I: Experimental design.

Type of attribute	Name	Value
a) Parameter	Chunk size	16, 32, 64 MB
b) Algorithm	Data placement algorithm	Round-Robin algorithm vs. capacity algorithm
c) Model	Rack model	Rack configuration: # of nodes (adding data nodes)



a) Parameter – Chunk size



b) Algorithm – Data placement

c) Model – Rack model

Figure 7: Experimental results.

Figs. 7 a and 7 b show the results of the parameter change experiment and the algorithm change experiment, respectively. Fig. 7 c shows the results of the model change experiment. These experiments show that the proposed model can test and analyse new algorithms and models based on a reliable model. We can see that the proposed model has advantages in model extensibility over a standalone model.

5. CONCLUSION

With the advent of the big data era, research on machine learning has been exploding across diverse research fields. It can be used to analyse and predict complex systems by building data models through training. Conversely, in addition to the data-driven machine learning models, dynamic systems can be analysed through system knowledge-based simulation models. Because these two approaches each have their pros and cons, it is difficult to fully express a complex system with only one approach. To this end, this paper proposes an extended formalism for the integrated modelling of the two modelling methods. It is a CoDEVS formalism that is an extension of the existing DEVS formalism. It consists of a simulation model, data model, and interface models that convert data between the simulation model and data model. Then, the models can be executed using the modified simulation environment. To demonstrate the validity of the proposed work, in this paper, we also applied it to develop a model of HDFS. From this study, we can see how to complement the defects of both methods and to model with one unified formalism. In future works, we will continue our research from the perspective of a model development platform that integrates a simulation engine, interoperation interface, database, and analysis tools into the proposed co-modelling method. Then, we will also apply it to the analysis, prediction, diagnosis, design, and optimization of various other research fields.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea (NRF) funded by the Korea Government (Ministry of Science and ICT) under Grant 2021R1G1A100355911.

REFERENCES

[1] Simon, H. A. (1991). The architecture of complexity, Klir, G. J. (Ed.), *Facets of Systems Science*, Springer, Boston, 457-476, doi:10.1007/978-1-4899-0718-9_31

[2] Vieira, A. A. C.; Dias, L. M. S.; Santos, M. Y.; Pereira, G. A. B.; Oliveira, J. A. (2018). Setting an Industry 4.0 research and development agenda for simulation – a literature review, *International Journal of Simulation Modelling*, Vol. 17, No. 3, 377-390, doi:10.2507/IJSIMM17(3)429

[3] Villalpando, L. E. B.; April, A.; Abran, A. (2014). Performance analysis model for big data applications in cloud computing, *Journal of Cloud Computing*, Vol. 3, No. 1, Paper 19, 20 pages, doi:10.1186/s13677-014-0019-z

[4] Zeigler, B. P.; Praehofer, H.; Kim, T. G. (2000). *Theory of Modelling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*, 2nd edition, Academic Press, San Diego

[5] Kim, T. G. (1995). DEVS formalism: Reusable model specification in an object-oriented framework, *International Journal in Computer Simulation*, Vol. 5, No. 4, 397-416, doi:10.5555/222274.222281

[6] Zhao, Y.; Zhang, H. (2021). Application of machine learning and rule scheduling in a job-shop production control system, *International Journal of Simulation Modelling*, Vol. 20, No. 2, 410-421, doi:10.2507/IJSIMM20-2-CO10

[7] Bock, H. G.; Carraro, T.; Jäger, W.; Körkel, S.; Rannacher, R.; Schlöder, J. P. (2013). *Model Based Parameter Estimation: Theory and Applications*, Springer, Berlin, doi:10.1007/978-3-642-30367-8

[8] Li, J.; Pan, S.; Huang, L.; Zhu, X. (2019). A machine learning based method for customer behavior prediction, *Technical Gazette*, Vol. 26, No. 6, 1670-1676, doi:10.17559/TV-20190603165825
[9] Sönmez, Y.; Kutlu, H.; Avci, E. (2019). A novel approach in analyzing traffic flow by extreme learning machine method, *Technical Gazette*, Vol. 26, No. 1, 107-113, doi:10.17559/TV-20171128220125
[10] Aldrich, J. (1995). Correlations genuine and spurious in Pearson and Yule, *Statistical Science*, Vol. 10, No. 4, 364-376
[11] Kim, B. S.; Kang, B. G.; Choi, S. H.; Kim, T. G. (2017). Data modeling versus simulation modeling in the big data era: case study of a greenhouse control system, *Simulation*, Vol. 93, No. 7, 579-594, doi:10.1177/0037549717692866
[12] Kim, B. S.; Kim, T. G. (2019). Cooperation of simulation and data model for performance analysis of complex systems, *International Journal of Simulation Modelling*, Vol. 18, No. 4, 608-619, doi:10.2507/IJSIMM18(4)491
[13] Kang, B. G.; Choi, S. H.; Kwon, S. J.; Lee, J. H.; Kim, T. G. (2018). Simulation-based optimization on the system-of-systems model via model transformation and genetic algorithm: a case study of network-centric warfare, *Complexity*, Vol. 2018, Paper 4521672, 15 pages, doi:10.1155/2018/4521672
[14] Kang, B. G.; Seo, K.-M.; Kim, T. G. (2018). Communication analysis of network-centric warfare via transformation of system of systems model into integrated system model using neural network, *Complexity*, Vol. 2018, Paper 6201356, 16 pages, doi:10.1155/2018/6201356
[15] Kim, B. S.; Kim, T. G. (2020). Modeling and simulation using artificial neural network-embedded cellular automata, *IEEE Access*, Vol. 8, 24056-24061, doi:10.1109/ACCESS.2020.2970547
[16] Elbattah, M.; Molloy, O. (2016). Coupling simulation with machine learning: A hybrid approach for elderly discharge planning, *Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 47-56, doi:10.1145/2901378.2901381
[17] Li, X.; Yeh, A. G.-O. (2001). Calibration of cellular automata by using neural networks for the simulation of complex urban systems, *Environment and Planning A: Economy and Space*, Vol. 33, No. 8, 1445-1462, doi:10.1068/a33210
[18] Almeida, C. D.; Gleriani, J. M.; Castejon, E. F.; Soares-Filho, B. S. (2008). Using neural networks and cellular automata for modelling intra-urban land-use dynamics, *International Journal of Geographical Information Science*, Vol. 22, No. 9, 943-963, doi:10.1080/13658810701731168
[19] Saric, T.; Simunovic, G.; Simunovic, K. (2013). Use of neural networks in prediction and simulation of steel surface roughness, *International Journal of Simulation Modelling*, Vol. 12, No. 4, 225-236, doi:10.2507/IJSIMM12(4)2.241
[20] Sung, C.; Kim, T. G. (2012). Collaborative modeling process for development of domain-specific discrete event simulation systems, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 42, No. 4, 532-546, doi:10.1109/TSMCC.2011.2135850
[21] Janakiraman, V. M. (2013). *Machine Learning for Identification and Optimal Control of Advanced Automotive Engines*, PhD Thesis, University of Michigan, Ann Arbor
[22] Kim, T. G.; Sung, C. H.; Hong, S.-Y.; Hong, J. H.; Choi, C. B.; Kim, J. H.; Seo, K. M.; Bae, J. W. (2010). DEVSim++ toolset for defense modeling and simulation and interoperation, *The Journal of Defense Modeling and Simulation*, Vol. 8, No. 3, 129-142, doi:10.1177/1548512910389203
[23] Kim, T. G.; Park, S. B. (1992). The DEVS formalism: hierarchical modular systems specification in C++, *Proceedings of the 1992 European Simulation Multiconference*, 152-156
[24] Wainer, G. (2002). CD++: a toolkit to develop DEVS models, *Software: Practice and Experience*, Vol. 32, No. 13, 1261-1306, doi:10.1002/spe.482
[25] Sarjoughian, H. S.; Zeigler, B. P. (1998). DEVSJAVA: basis for a DEVS-based collaborative M&S environment, *Proceedings of the 1998 International Conference on Web-Based Modeling and Simulation*, 29-36
[26] Zeigler, B. P.; Moon, Y.; Kim, D.; Kim, J. G. (1996). DEVS-C++: a high performance modelling and simulation environment, *Proceedings of HICSS-29: 29th Hawaii International Conference on System Sciences*, 350-359, doi:10.1109/HICSS.1996.495481
[27] Kwon, S. J.; Kang, B.; Choi, C.; Kim, T. G. (2017). Adaptive discrete event simulation systems to embrace changes of requirements using event control models, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 50, No. 3, 1147-1160, doi:10.1109/TSMC.2017.2747604

[28] Kwon, S. J.; Kim, T. G. (2012). Design and implementation of event-based DEVS execution environment for faster execution of iterative simulation, *Proceedings of the 2021 Symposium on Theory of Modeling and Simulation*, Paper 14, 8 pages, doi:10.5555/2346616.2346630

[29] The Apache Software Foundation. Apache Hadoop, from *http://hadoop.apache.org*, accessed on 23-07-2020

[30] Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. (2010). The Hadoop distributed file system, *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies*, 10 pages, doi:10.1109/MSST.2010.5496972

[31] Aguilera-Mendoza, L.; Llorente-Quesada, M. T. (2013). Modeling and simulation of Hadoop distributed file system in a cluster of workstations, Cuzzocrea, A.; Maabout, S. (Eds.), *Model and Data Engineering*, Springer, Berlin, doi:10.1007/978-3-642-41366-7_1