SILESIAN UNIVERSITY OF TECHNOLOGY PUBLISHING HOUSE

SCIENTIFIC PAPERS OF SILESIAN UNIVERSITY OF TECHNOLOGY 2023
ORGANIZATION AND MANAGEMENT SERIES NO. 171

# REAL-TIME DATA PROCESSING IN SIMULATION MODELS WITH 3D VISUALIZATION

Piotr JANKE[1*], Tomasz OWCZAREK[2]

[1] Silesian University of Technology, Faculty of Organization and Management; piotr.janke@polsl.pl, ORCID: 0000-0001-8065-9013
[2] Silesian University of Technology, Faculty of Organization and Management; tomasz.owczarek@polsl.pl, ORCID: 0000-0003-2532-4127
* Correspondence author

**Purpose:** The aim of this paper is to demonstrate how to integrate the FlexSim simulation environment with an R engine to enable use of the predictive algorithm results for real-time simulation run management.
**Design/methodology/approach:** Using a simple but easily generalizable simulation model representing a broad set of real-world solutions used in logistics and manufacturing, we outline how to combine FlexSim and the R language. In particular, we present the necessary settings of the objects that make up the simulation model, including event-initiating triggers, as well as the structure of a program written in R, together with the necessary instructions.
**Findings:** The case study showed that synchronous, i.e. real-time, communication between the simulation model and the R computing environment is possible and allows the simulation process to be controlled.
**Research limitations/implications**: The main limitation of the presented approach is the synchronization of file read and write operations, and this is an area for further investigation.
**Practical implications:** The solution proposed in this paper allows for the development and testing of simulations created in FlexSim using the results of optimization models or predictive algorithms developed using the R language, in real time – i.e. in the course of the simulation process. This approach can be used to simulate any process that is modeled using a discrete-event simulator, particularly in fields such as logistics or manufacturing.
**Originality/value:** The study can be of value to anyone involved in modeling and simulating real-world processes using FlexSim, for whom so far the problem has been to incorporate the output of algorithms obtained with other computing environments into the simulation run.

**Keywords:** FlexSim, discrete event simulation, prediction, R language.

**Category of the paper:** technical paper, case study.

# 1. Introduction

In times of ever growing necessity for the optimization of logistics and manufacturing processes, the use of simulation tools is becoming increasingly prominent. The search for process bottlenecks, workload analyses, or the capacity of corridors and pathways represent just a few of the concerns related to efficient management in the field of logistics. The selected computer simulation tools can provide support in these areas by providing new opportunities for testing organizational shifts and seeking optimal solutions without generating excessive costs. Simulation models can reduce costs by running experiments in a virtual environment without the need for expensive reorganization. The optimization modules enable the selection of appropriate model parameters for minimizing or maximizing the target function. Such optimization is possible by querying the solution space for a wide range of parameters, taking into account the constraints defined in the simulation model. For instance, in the field of logistics, it is possible to find optimal settings for minimizing the operator's path in the picking process for fast-moving goods (Pawlewski, 2018). Determining the surface area index for highway freight terminal facilities using the simulation tool in the FlexSim application was presented in the following studies (Chen, Hu, Xu, 2013). Whereas the studies of (Dallasega, Rojas, Rauch, Matt, 2017) used a flow chart process to compare traditional centralized manufacturing planning and control systems with modern 4.0 solutions involving decentralization and real-time planning and control. Another simulation model that shows the performance of autonomous vehicles with respect to the placement of charging points is presented in the studies by (Chen, Chen, Teng, 2021). Similarly, a case study in the paper by (Li et al., 2020) on AGVs with resource allocation in the field of manufacturing logistics was analyzed using a simulation model developed in the FlexSim environment. Furthermore, Yu & Nielsen (2020) emphasizes the use of computer simulation tools in the field of Industry 4.0 and the challenges associated with the implementation of this technology in simulations of factory operations and, in particular, the importance of using artificial intelligence – machine learning – in solutions of this sort.

The primary purpose of this study is to demonstrate a method of integrating a logistics and manufacturing process simulation environment with an R-based engine, enabling the implementation of algorithms for real-time prediction of selected parameters of a running simulation model.

As a discrete event simulator with 3D visualization, the authors of this dissertation chose the FlexSim environment. For the R environment, the engine 4.2.2 from the CRAN package repository was selected.

## 2. Simulation model in the FlexSim environment

The developed simulation model consists of five fixed resources and a simple flow logic. For the execution of the environment connection mechanism, a system that is unrelated to any real-world processes has been prepared. Nevertheless, this model can serve as a reference for more complex models, as the set-up and logic presented herein are often used for processes in the field of logistics or manufacturing. A core element of the model is the decision point that allows the selection of the target (destination) of the flow items, as is customary in real-life processes involving, e.g. storage. The consequence of classifying the freight units is the selection of a particular rack in the warehouse. Such solution is presented in the study by (Zhu et al., 2014). Likewise, a similar situation often occurs on production lines (Zomparelli, Petrillo, Salvo, Petrillo, 2018) and (Aljorephani Elmaraghy, 2016).

For a list of objects used in the model, see Figure 1.

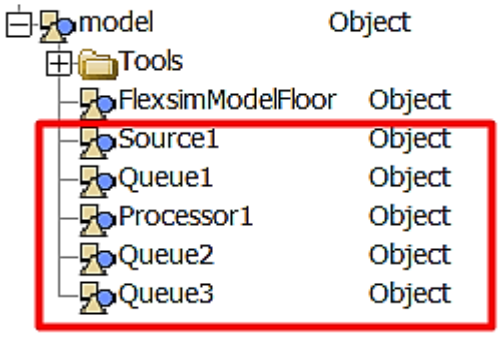
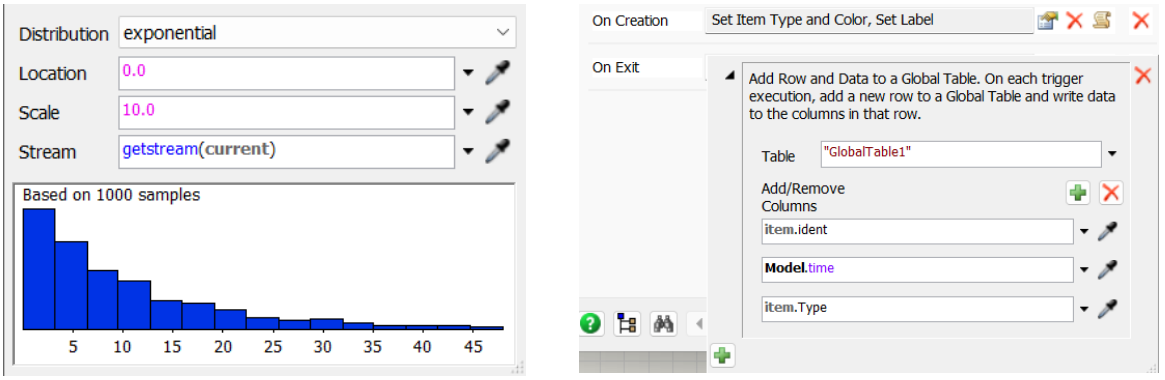



**Figure 1.** Highlighted list of objects used in the model.

Source: Own study.

The model uses three buffer fields from Queue1 to Queue3 with default settings. Furthermore, a single machine "Processor1" and a single data source were used for the "Inter-arrival time" setting and the default "exponential" distribution with default parameters, as shown in Figure 2.

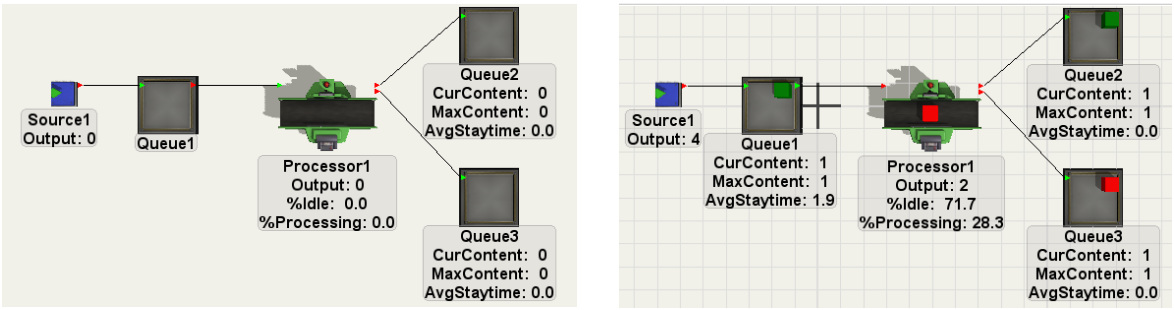



a) b)

**Figure 2.** The distribution parameters for the source of the model a) and the trigger settings for the source b).

Source: Own study.

Two triggers were placed on the source. The first one allows the creation of 2 different values for the default label "Type" randomized according to the "duniform" distribution. Moreover, each newly created flow item contains identifier information with a number based on the order in which the object was created, as well as information on the current time status in the model. The information is stored in a global table using a second trigger for the "On Exit" status of the source. The system is connected using the "Connect Object" directional connections. The connections are shown in Figure 3, with the finished simulation model.




a) b)

**Figure 3.** Simulation model: a) stop, b) start

Source: Own study

The object flow in the simulation model follows the figure from left to right. The predefined global table stores information about the identifier, type and time, and is saved to a csv file via the trigger at "Queue1". The data structure of the first global table can be seen in Table 1.

**Table 1.**

*Structure of the simulation model output for a few selected flow items*

| *Id* | *time* | *decision* |
|---|---|---|
| *0* | *36.20* | *1* |
| *1* | *75.08* | *2* |
| *2* | *79.42* | *1* |
| *3* | *90.73* | *2* |

Source: Own study

The table shows the data for the first 4 flow items. The decision column is a column that stores information about the type of object previously generated using the trigger. The output file generated in this manner is the input for the program running on the R engine.

The final and essential element of the model is the trigger that enables the import of the file processed by the R engine. This trigger is activated at the end of the machine process. Its task is to load the file in real time and execute a SQL query according to the object identifier. This will facilitate a decision on the choice of machine output port for the flow item towards queue 2 or 3. The input data structure of the simulation model is shown in Table 2.

**Table 2.**
*Structure of the simulation model input for a few selected flow items*

| *id* | *pred* |
|---|---|
| *0* | *2* |
| *1* | *1* |
| *2* | *2* |
| *3* | *1* |

Source: Own study

The identifier column remains the same. The *pred* column contains the new data determined by the R program that are linked to the flow element with the correct identifier. The structure of the trigger, based on which the decision related to the selection of the output port of the machine is made, is shown in Figure 4.

```
Object item = param(1);
Object current = ownerobject(c);
/***popup:GlobalTableLookupNew*/
/***tag:Description*//**Using Global Lookup Table ( tab )*/
Variant tableID = /**\nTable: *//***tag:TableName*//**/"tab"/**/;

Table table;
switch (tableID.type) {
    case VAR_TYPE_NODE: table = tableID; break;
    case VAR_TYPE_STRING: table = Table(tableID.as(string)); break;
    default:
        table = reftable(tableID.as(int));
    break;
}

Variant row = /**\nRow: *//***tag:row*//**/item.ident/**/;
Variant col = /**\nColumn: *//***tag:col*//**/2/**/;

double result = Table.query("SELECT pred FROM tab WHERE id == $1",row)[1][1];
return result;
```

**Figure 4.** Logic of the developed SendtoPort machine trigger in the simulation model
Source: Own study

The SQL query executed via the trigger returns a value from the second column of the input table for the row with the corresponding identifier.

The general structure of the proposed solution can be found in Figure 5.

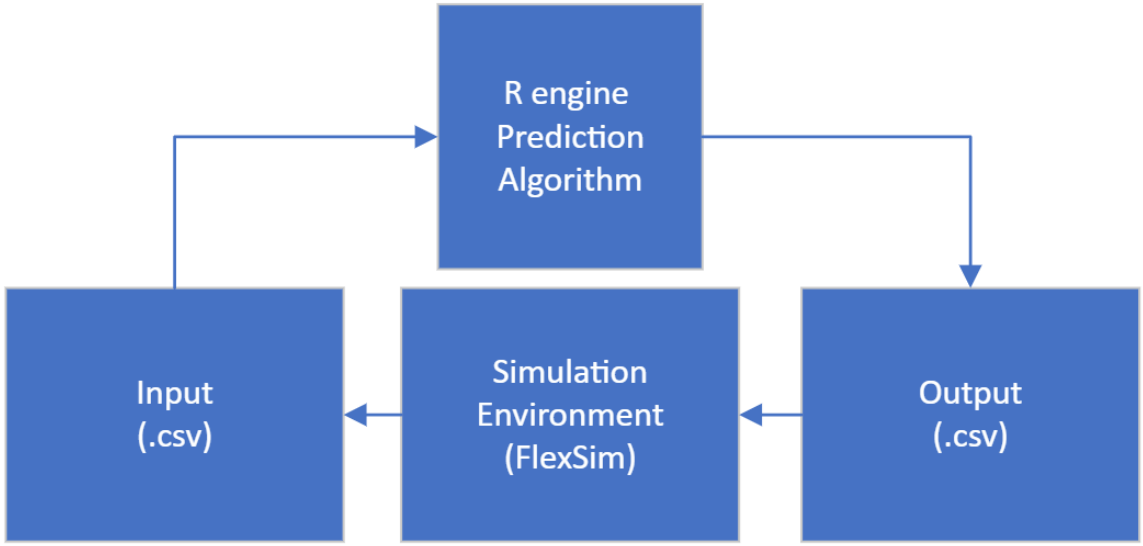



**Figure 5.** A model for the integration of the FlexSim environment and the R engine.
Source: Own study

## 3. The R-based program

A function in R that allows interaction with FlexSim and makes predictions based on simulation data must contain the following necessary elements:

1. Loading of object data generated during simulation.
2. Making prediction.
3. Saving the prediction to a csv file (each record saved should contain – in addition to the prediction value – the identifier of the relevant object).

Also, an optional, albeit useful solution is to display a current job status message (e.g. the number of objects for which a prediction has been made).

As the program is intended to run in real time, it is necessary to use an infinite loop that can be stopped manually. This kind of a solution requires exception handling mechanism so as to avoid aborting the loop in the event of any error (e.g. a file read error in a single iteration). The code for the loop used in the program is given in Figure 6.

```
while(TRUE){
    tryCatch({

        # 1 - read input file
        dat <- read.csv2(paste0(path, inFile))

        # 2 - make predictions, create dataframe
        df <- data.frame(id = dat$id,
                         pred = predict_sim(dat))

        # 3 - save dataframe with predictions
        write.csv2(df, paste0(path, outFile),
                   row.names = FALSE, quote = FALSE)

        # 4 - display message
        print(paste(Sys.time(),
                    "saving predictions for", nrow(df), "records")
              )

    }, error = function(err){
        # on error
        print(paste(Sys.time(), "error: ", err))

    }, finally = {
        # wait some time
        Sys.sleep(interv)

    }
    )
}
```

**Figure 6.** Code for the loop used in the program.

Source: Own study

An infinite loop is provided by the use of the *while* statement. This is immediately followed by the exception handling function (*tryCatch*). At every iteration, it attempts to execute a code consisting of four instructions: loading the data, making a prediction, saving the data and displaying a message about the number of records for which a prediction has been made. If any error occurs at this stage, the program proceeds to the *error* section, which displays an error message in this case. Whether or not an error occurs, the loop ends with the execution of a piece of code in the *finally* section – the use of the *Sys.sleep* function enables the loop to wait a certain amount of time before starting the next iteration. This is done to reduce the frequency of loading and writing files to disk.

The primary purpose of the integration of the FlexSim environment and the R language described in this study is to enable the use of machine learning algorithms. Therefore, the function making the prediction, which is referred to as *predict_sim* in the code presented in Figure 6, plays a key role in the program. This can be a function that returns predictions based on any machine learning algorithm (the example presented here applies specifically to a classification problem, but this does not prevent similar solutions from being used for a regression problem). The only condition is that the value returned by the function is a vector with predicted values for the individual flow items.

Additionally, the developed program uses global variables responsible for the path to the folder holding the simulation model and the file names. Furthermore, the loop described earlier is placed inside a function that resets the data file generated by the simulation model at the very start. Details of the program code are shown in Figure 7.

```
# path to project folder with FlexSim model
path <- "C:/Users/user_name/Documents/FlexSim 2023 Projects/"

# name of the input file (generated by FlexSim)
inFile <- "in.csv"

# name of the output file (read by FlexSim)
outFile <- "out.csv"


operate <- function(interv = 0.1){

    # reset input file
    write.csv2(data.frame(id = integer(),
                          time = double(),
                          decision = integer()),
               paste0(path, inFile), row.names = FALSE, quote = FALSE)

    while(TRUE){

        # code of loop

    }
}

# running the function
operate()
```

**Figure 7.** Detailed program code (without the loop code).

Source: Own study

## 4. Results

The solution presented here uses a simple prediction function that, based on information about the object type (contained in the *decision* column of the global table exported by the simulation model), makes a decision to redirect the object in question to the appropriate buffer field. For a type 1 object, this will be the Queue3 buffer field (the second machine output, so in this case the function will return a value of 2), while for a type 2 object it will be the first machine output (the function will return a value of 1, the object will be sent to the queue labeled Queue2). The code of the prediction function used is provided in Figure 8.

```
predict_sim <- function(dat){

    # for this test predictions are just switched decision values
    # but any ML prediction function can be incorporated here
    pred <- ifelse(dat$decision == 1, 2, 1)

    # vector of predictions is returned
    return(pred)
}
```

**Figure 8.** Code of the prediction function used.

Source: Own study

The first step involved running an R script with the function described in the previous section. Assuming the correct file paths are given and the function generates an empty input file at the start, the results are messages displayed at the frequency specified by the *interv* argument, which gives the time and number of processed records. As long as the simulation model is not launched, this number will be zero (Figure 9).

```
> operate()
[1] "2023-02-24 15:36:20 saving predictions for 0 records"
[1] "2023-02-24 15:36:20 saving predictions for 0 records"
[1] "2023-02-24 15:36:20 saving predictions for 0 records"
[1] "2023-02-24 15:36:20 saving predictions for 0 records"
[1] "2023-02-24 15:36:21 saving predictions for 0 records"
[1] "2023-02-24 15:36:21 saving predictions for 0 records"
[1] "2023-02-24 15:36:21 saving predictions for 0 records"
[1] "2023-02-24 15:36:21 saving predictions for 0 records"
[1] "2023-02-24 15:36:21 saving predictions for 0 records"
[1] "2023-02-24 15:36:21 saving predictions for 0 records"
[1] "2023-02-24 15:36:21 saving predictions for 0 records"
[1] "2023-02-24 15:36:21 saving predictions for 0 records"
[1] "2023-02-24 15:36:21 saving predictions for 0 records"
[1] "2023-02-24 15:36:22 saving predictions for 0 records"
```

**Figure 9.** Result of the program before starting the simulation.

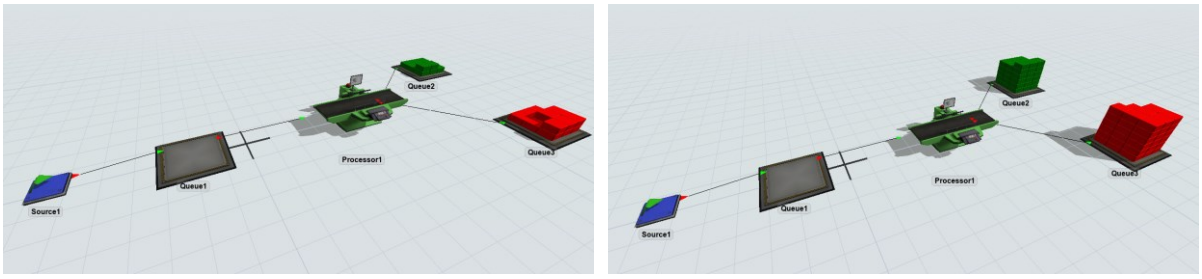
Source: Own study

After running the simulation model described in Section 2, the program will display the number of objects (records) for which a prediction has been made in successive iterations (Figure 10).

```
[1] "2023-02-24 15:42:20 saving predictions for 0 records"
[1] "2023-02-24 15:42:20 saving predictions for 1 records"
[1] "2023-02-24 15:42:20 saving predictions for 1 records"
[1] "2023-02-24 15:42:20 saving predictions for 1 records"
[1] "2023-02-24 15:42:20 saving predictions for 1 records"
[1] "2023-02-24 15:42:20 saving predictions for 1 records"
[1] "2023-02-24 15:42:20 saving predictions for 2 records"
[1] "2023-02-24 15:42:21 saving predictions for 2 records"
[1] "2023-02-24 15:42:21 saving predictions for 2 records"
[1] "2023-02-24 15:42:21 saving predictions for 2 records"
[1] "2023-02-24 15:42:21 saving predictions for 2 records"
[1] "2023-02-24 15:42:21 saving predictions for 4 records"
[1] "2023-02-24 15:42:21 saving predictions for 4 records"
[1] "2023-02-24 15:42:21 saving predictions for 4 records"
[1] "2023-02-24 15:42:21 saving predictions for 4 records"
[1] "2023-02-24 15:42:21 saving predictions for 4 records"
[1] "2023-02-24 15:42:21 saving predictions for 4 records"
[1] "2023-02-24 15:42:22 saving predictions for 4 records"
[1] "2023-02-24 15:42:22 saving predictions for 4 records"
[1] "2023-02-24 15:42:22 saving predictions for 4 records"
[1] "2023-02-24 15:42:22 saving predictions for 4 records"
[1] "2023-02-24 15:42:22 saving predictions for 5 records"
[1] "2023-02-24 15:42:22 saving predictions for 5 records"
[1] "2023-02-24 15:42:22 saving predictions for 5 records"
[1] "2023-02-24 15:42:22 saving predictions for 5 records"
[1] "2023-02-24 15:42:22 saving predictions for 7 records"
```

**Figure 10.** Initial execution of the program upon starting the simulation.
Source: Own study

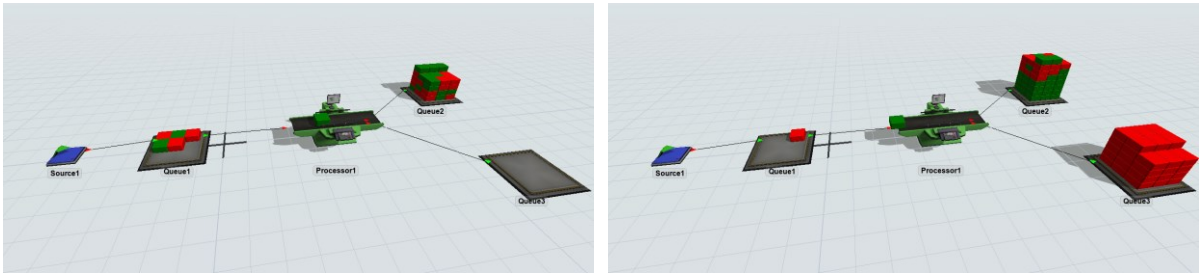
At the same time, the simulation model, or the Processor1 object to be more precise, will decide whether to send a given object to the first (Queue2) or second (Queue3) output, based on the predictions returned by the program. Figure 11 shows the simulation results after 25 and 100 processed objects, respectively, with the program running simultaneously. It shows that all green objects (type = 2) were passed to the first output, while all red objects (type = 1) were passed to the second output.




a) b)

**Figure 11.** Simulation results with the program running: a) after 25 objects, b) after 100 objects.
Source: Own study

Launching the simulation without a running program results in all objects being passed to output 1, regardless of their type (Figure 12a). Similarly, if the program is stopped during the simulation, all remaining objects generated during the simulation will be passed to output 1 (Figure 12b).




a) b)

**Figure 12.** Simulation results: a) without the program running, b) after the program was stopped.
Source: Own study

## 5. Summary

The example of an integration of a discrete event simulator enabling 3D visualization of real-world logistical processes with an R-based engine allowing the use of effects of machine learning, presented in this study, is part of a hybrid approach in modeling and simulation (von Rueden et al., 2020). It is a response to the increasing need to use computer simulations combined with modern optimization and prediction methods in the context of process automation and the concept of Industry 4.0 (Yu & Nielsen, 2020).

However, the solutions presented in the literature are mostly based on asynchronous communication between the simulation and optimization environments, where the results of prediction or optimization using historical data are either incorporated into the scheduling process before the simulation starts or are used to optimize processes post-simulation (Azab et al., 2021; Atalan et al., 2022). The integration proposal presented in this study is based on synchronous communication, where the simulation process continuously uses the output of the machine learning model and makes decisions based on it. A similar solution, using the FlexSim program and Python, was presented in (Leon et al. 2022), whereby the method of communication between environments was based on a client-server architecture. The solution proposed here involves communication by means of shared files, which in extreme cases ( with an inappropriately selected simulation speed with respect to the loop delay in the program) can cause synchronization problems. This may offer an area for further research. However, the solution itself is relatively simple to implement and the results can be easily validated.

## References

1. Aljorephani, S. K., & Elmaraghy, H. A. (2016). Impact of Product Platform and Market Demand on Manufacturing System Performance and Production Cost. Procedia CIRP, 52, 74–79. https://doi.org/10.1016/j.procir.2016.07.068
2. Atalan, A., Şahin, H., & Atalan, Y. A. (2022). Integration of Machine Learning Algorithms and Discrete-Event Simulation for the Cost of Healthcare Resources. Healthcare, 10(10), 1920. https://doi.org/10.3390/healthcare10101920
3. Azab E, Nafea M, Shihata LA, Mashaly M. (2021). A Machine-Learning-Assisted Simulation Approach for Incorporating Predictive Maintenance in Dynamic Flow-Shop Scheduling. Applied Sciences, 11(24), p. 11725. https://doi.org/10.3390/app112411725
4. Chen, J. C., Chen, T. L., & Teng, Y. C. (2021). Meta-model based simulation optimization for automated guided vehicle system under different charging mechanisms. Simulation Modelling Practice and Theory, 106(May 2020), 102208. https://doi.org/10.1016/j.simpat.2020.102208
5. Chen, L.-H., Hu, D.-W., & Xu, T. (2013). Highway Freight Terminal Facilities Allocation based on Flexsim. Procedia - Social and Behavioral Sciences, 96(Cictp), 368–381. https://doi.org/10.1016/j.sbspro.2013.08.044
6. Dallasega, P., Rojas, R. A., Rauch, E., & Matt, D. T. (2017). Simulation Based Validation of Supply Chain Effects through ICT enabled Real-time-capability in ETO Production Planning. Procedia Manufacturing, 11(June), 846–853. https://doi.org/10.1016/j.promfg.2017.07.187
7. Leon, J. F., Marone, P., Peyman, M., Li, Y., Calvet, L., Dehghanimohammadabadi, M., & Juan, A. A. (2022). A Tutorial on Combining Flexsim with Python for Developing Discrete-Event Simheuristics. In 2022 Winter Simulation Conference, 1386-1400. IEEE. https://doi.org/10.1109/WSC57314.2022.10015309
8. Li, G., Yang, S., Xu, Z., Wang, J., Ren, Z., & Li, G. (2020). Resource allocation methodology based on object-oriented discrete event simulation: A production logistics system case study. CIRP Journal of Manufacturing Science and Technology, (2019). https://doi.org/10.1016/j.cirpj.2020.07.001
9. Näsänen, J., & Vanharanta, O. (2016). Program group's discursive construction of context: A means to legitimize buck-passing. International Journal of Project Management, 34(8), 1672–1686. https://doi.org/10.1016/j.ijproman.2016.09.008
10. von Rueden, L., Mayer, S., Sifa, R., Bauckhage, C., & Garcke, J. (2020). Combining machine learning and simulation to a hybrid modelling approach: Current and future directions. In Advances in Intelligent Data Analysis XVIII: 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27–29, 2020, 548-560. https://doi.org/10.1007/978-3-030-44584-3_43
11. Pawlewski, P. (2018). Using PFEP for Simulation Modeling of Production Systems. Procedia Manufacturing, 17, 811–818. https://doi.org/10.1016/j.promfg.2018.10.132

12. Yu, F., & Nielsen, C. P. (2020). A data-driven approach for decision-Making support of factory simulation solutions. Procedia CIRP, 93, 971–976. https://doi.org/10.1016/j.procir.2020.04.129
13. Zhu, X., Zhang, R., Chu, F., He, Z., & Li, J. (2014). A flexsim-based optimization for the operation process of cold-chain logistics distribution centre. Journal of Applied Research and Technology, 12(2), 270–278. https://doi.org/10.1016/S1665-6423(14)72343-0
14. Zomparelli, F., Petrillo, L., Salvo, B. Di, & Petrillo, A. (2018). Re-engineering and Relocation of manufacturing process through a simulative and multicriteria decision model. IFAC-PapersOnLine, 51(11), 1649–1654. https://doi.org/10.1016/j.ifacol.2018.08.220