Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

DEVELOPMENT OF DES APPLICATION FOR FACTORY MATERIAL FLOW SIMULATION WITH SIMPY

So-Hyun Nam
Seung-Heon Oh
Hee-Chang Yoon
Young-In Cho[1]
Ki-Young Cho[2]
Dong-Hoon Kwak
Jong Hun Woo

Department of Naval Architecture and Ocean Engineering
Seoul National University
1th Gwanak-ro, Gwanak-gu, Seoul 08826, South Korea

ABSTRACT

Since most studies on logistics simulation have used commercial software, there has always been a limit in terms of customization and performance. In this research, a discrete-event simulation (DES) model, a program that is capable of evaluating logistics for shipyard layout changes, was developed. The DES model was substituted yard layouts with network models in order to model the complex factory layouts and road configurations of shipyards. An application that is capable of not only performing high-speed calculations on the frequency of road use, travel distance of transporters, travel distance of blocks, and workloads in stockyards and factories through simulation was developed, and given the ability to analyze productivity changes according to various layout configurations, production plans, product configurations and resource conditions.

1 INTRODUCTION

The biggest consideration when it comes to improving the factory layout is logistics. Unlike general machinery industries, in which the process flow is standardized once the product is decided, in shipyards, logistics indicators change due to factors such as ship composition, scheduling, and routing. Because of this, a methodology to evaluate layouts based on these various factors is required.

Research on logistics analysis using Discrete Event Simulation (DES) as representative layout verification method is being conducted. In Halim et al. (2020), the performance of arrangement at an automobile assembly plant was evaluated using ARENA, a commercial DES software. In Zúñiga et al. (2020), optimization for various manufacturing processes such as electrical cabinets and water pumps was conducted using several commercial DES software. In Hamzas et al. (2017), a layout verification simulation for motorcycle assembly plants using Witness, another commercial DES software, was developed. Several studies on simulations for verification and evaluation of semiconductor plant layouts have also been conducted. In Ndiaye et al. (2016), the commercial software AutoMod was used to design a semiconductor plant AVG (automated guided vehicle) transport system. Kim et al. (2012) conducted a study on the improvement of semiconductor parts production lines using simulation and AHP/DEA.

Active research using DES is also being conducted in the shipbuilding sector. The following are some examples of studies that have been conducted in the field of shipbuilding production management. In Song et al. (2009), a methodology for the application of DES simulation, and to identify tasks that can be expected

to improve productivity using DES simulation was presented for production management in small and medium-sized shipyards. In Woo and Song (2014), a methodology to define core performance indicators using business administration BSCs (balance score cards) and performance pyramids for DES simulation-based shipyard production management was proposed. Also, the applicability of KPI (key performance indicator) identification methods and DES simulation for the solution of business problems in shipyard block assembly plant systems was researched. Woo and Oh (2018) and Lee et al. (2020) conducted a study on assembly plant planning optimization (Woo & Oh, 2018) and mid-term planning productivity improvement (Lee et al., 2020) by linking DES simulation through actual shipyard production planning systems and interfaces.

Also, the following studies on the design and improvement of shipyard layouts have been conducted. Shin et al. (2009) conducted a framework research on the design of shipyard layouts. In this research, a framework that incorporates methodologies that consider the characteristics of shipbuilding processes based on the traditional systematic layout planning methods was proposed. Specially, the role of DES simulations when it comes to the quantitative evaluation of layout alternatives was defined. Also, in Song et al. (2010), applications for the framework designed by Shin et al. (2009) were developed to demonstrate the feasibility of shipyard layout design based on theories and systems. There are also cases in which logistics depending on shipyard layout changes have been analyzed using DES simulations. For example, in Woo et al. (2010), a comparative analysis on logistics efficiency following the establishment of new factories using DES simulation to analyze investment effects from the perspective of logistics before large capital yard investment (for construction of new shipbuilding factories) was conducted. Also, Jeong et al. (2018) developed a DES kernel to overcome existing commercial DES software limitations (customizing difficulties, slow simulation speed, etc.), and conducted a study for equipping the DES model with GIS, route search, and spatial arrangement algorithms for shipyard logistics simulations.

The existing studies show the similarities of design and improvement activities for DES-based production systems. Specially, using DES for general manufacturing process and layout design is becoming essential in terms of cost reduction and productivity improvement. However, it has not yet reached the stage of full-scale commercialization in the shipbuilding field due to some problems. The first problem is that most commercial DES software are designed to be suitable for general manufacturing industry types, and lack of a customizing environment that is able to reflect factors such as complex product structures like those of ships, and the constantly changing schedules of shipyards. Another problem is the simulation speed; medium- and long-term simulations that consider a variety of factors such as product information and schedules are required for shipyard layout simulations, but existing research using commercial software has not been able to solve speed problems. The afore-mentioned research (Jeong et al., 2018) developed its own DES software with a flexible customizing environment specialized for shipyards, but could not solve simulation speed issues due to performance problems of the self-developed DES kernel.

This research was conducted to increase customizing flexibility and securing fast simulation speed. Since most existing DES research was implemented using Windows, which is considered to have limitations when it comes to processing quickly large amounts of data such as that of shipyard simulations. For the two goals, the simulator is developed by open-source DES software. Dagkakis and Heavey (2016) compared and evaluated some open-source DES software by some dimensions – language, license, the latest release, etc. According to those features, the algorithm for evaluating material flow in production yard is used SimPy which is python-based DES kernel as the package has merits on reliability, cost-effectiveness, interoperability with machine learning resources, memory efficient computation for Python. Since the release in 2002, SimPy have gained reliability by many user's feedback and updates. Also, as SimPy is based on MIT license, it has perfect cost effectiveness. Consequently, cost effectiveness is the key consideration because this study aims at the field that, if possible, seeks to improve productivity without cost. Many machine learning packages can run in Python environment. Therefore, SimPy has the powerful interoperability with machine learning packages and the interoperability can facilitates the future-works, such as the integration with reinforcement learning.

2 METHODOLOGY

2.1 Simulation Framework

In this research, the simulation framework for DES program development is defined as shown in Figure 11. The simulation framework is composed by an adapter module that is responsible for data preprocessing, a modeler module that connects data and simulation, a simulation module that performs simulations, and an analyzer and reporter module that calculates and outputs results according to the purpose of simulation. In other words, it consists of an adapter and a modeler for pre-processing, an analyzer and a reporter for post-processing, and a DES Kernel for performing simulations.

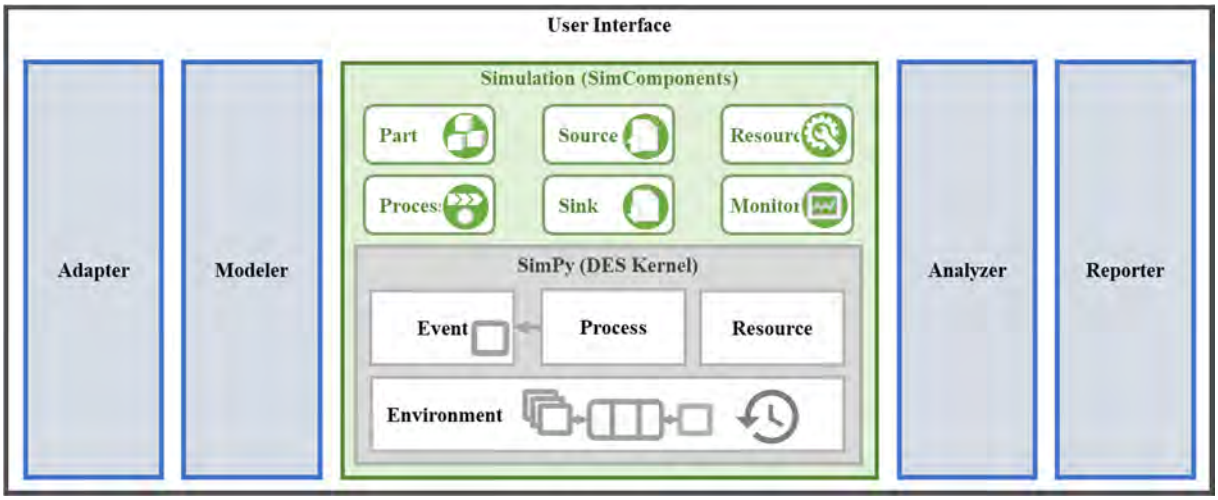

Figure 1: Simulation framework.

The adapter module receives external information (such as product, process, and schedule information) and converts it into the data required for simulation modeling. The modeler module uses pre-processed data to model components of the simulation model (part, source, resource, and process) as objects, and also creates objects called sink and monitor to analyze simulation results.

In the simulation module, the modeled objects are used for simulation execution. This module is a kernel for DES execution that uses SimPy, which is a Python-based DES package. The simulator for this research is developed from 'SimComponents' which Grotto Networking(https://www.grotto-networking.com/DiscreteEventPython.html#Intro) made for the transferring object between processes as SimPy offers only basic components for simulation, such as queue, event, resource, etc. Six classes in the modified 'SimComponents' in Figure 11 are key objects for the actual production system turning into a DES model.

2.2 Pre-Processing

Next, the pre-processing algorithms corresponding to the adapter module will be described. This is a step that is necessary for DES simulation, and in which schedule and product data is combined and converted into data for process-oriented DES. In this research, pre-processing is conducted in two steps. First, activity data that is suitable for the simulation purpose is selected. And second, the assembly relationship of bill-of-material (BOM) data is connected to the processed activity data.

2.2.1 Activity Pre-processing

Activity data is processed in accordance with the simulation purpose. The activity on a block cannot have any overlapping with other activities on the same block. Figure 2 shows the entire cases of the relation between activities for the same block and the output of pre-processing of each case.

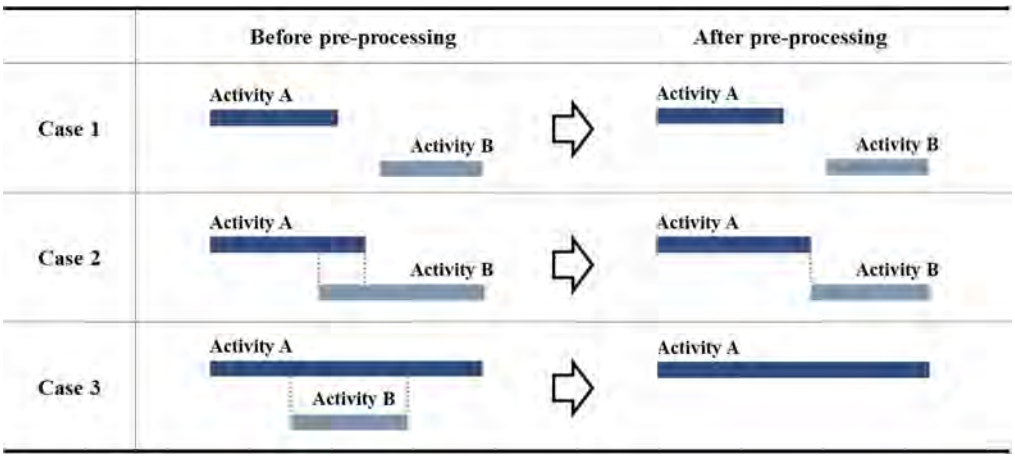

Figure 2: Cases of activity pre-processing.

2.2.2 BOM Pre-processing

Bill of material (BOM) data includes the relationships of blocks which will be assembled and characteristics of each block. For example, using Figure 33, the BOM data has the information that the Part A through Part C will be combined and turn into Part D. Also the data contain parts' height, weight etc. That numbers will used in the algorithms for selecting resources. From a simulation point of view, when a product part that was being processed according to an activity is brought together with a product part of a superior level, child parts are gathered together in the corresponding process to create a new parent part through BOM data in which the relationship between the current product part (child part), and the superior product part (parent part) are specified. This logic is reflected in simulation data during pre-processing, and is also reflected in the simulation model's logic.

Various parent-child relationships are processed through several steps as shown in Figure 33. In this research, these relationships are generalized and reflected in the pre-processing data and model logic for implementation of the assembly process.

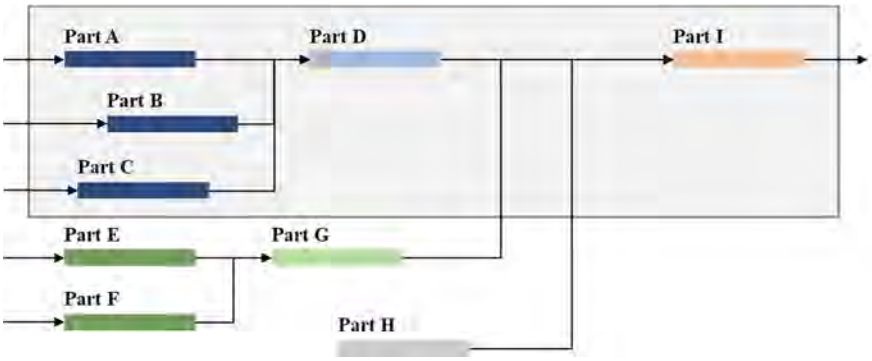

Figure 3: BOM data structure.

2.2.3 GIS Pre-processing

Road implementation in commercial DES-based shipyard simulations has its limitations. Since existing commercial DES programs are specialized for modeling environments with fork lift or AGV (automatic ground vehicle) movement, they have limitations when it comes to reflecting general road shapes such as in shipyards.

In this research, to compensate for these shortcomings, distance matrices between each shipyard layout location were created in advance and loaded from the GIS into the DES program. Shipyard layouts were modeled as shown in Figure 44 using Esri ArcGIS. Factories and roads were organized into layers representing geographic data collections in ArcGIS. Roads where block logistics flows were modeled as omnidirectional graph data structures with the form of edge layers that connect the corresponding point and node layers modeling major points. Here, nodes represent points where the characteristics of roads or factory entrances/exits change. The weighted value of edges represents the distance between each node. Once modeling is over, factory area data and distance data between entrances and exits is exported in csv format.

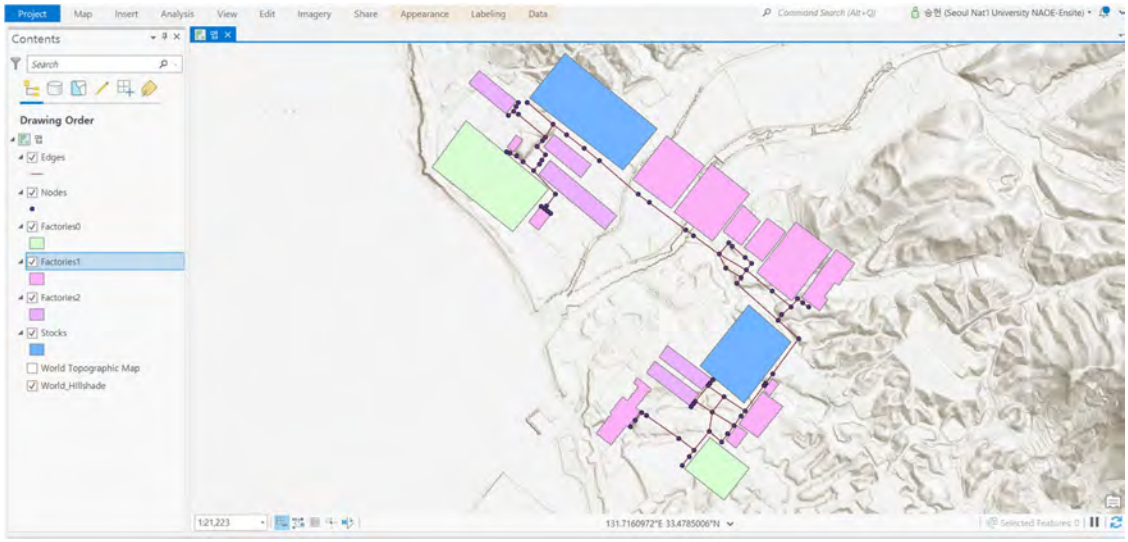

Figure 4: Layout representation by Esri ArcGIS.

As shown in Table 11, distance data between nodes exported from ArcGIS creates symmetric tables containing information on the shortest distance between each node using Dijkstra algorithms based on Python environment NetworkX packages. Factory area data is used as capacity required for block tasks, and symmetric tables are used to calculate the travel distance and travel time of blocks. Applying distance matrices to the simulation can reduce significantly the time required for route search during simulation.

Table 1: Example of shortest distance table between nodes.

	Factory A	**Factory B**	**Stockyard A**	**Factory C**	**Stockyard B**	**…**
Factory A	0	69	120	653	631	…
Factory B	69	0	51	584	561	…
Stockyard A	120	51	0	533	510	…
Factory C	653	584	533	0	22	…
…	…	…	…	…	…	0

2.3 Simulation Module

2.3.1 SimPy

In this research, SimPy (https://simpy.readthedocs.io/en/latest/index.html), which is a Python-based Open Source DES Kernel, is used. SimPy is a Python package that provides DES modeling environments and functions, and can be used with Python's various data processing packages. SimPy.Environment manages time, and stores and progresses events. SimPy.Event is in charge of progressing time during simulation, calling resources provided by SimPy, and generator modeling. Finally, SimPy.Resource, in order to model the resources required for each process, is composed by a Resource that models resources with the same characteristics, a Store that stores discrete resources, and a Container that stores continuous resources.

2.3.2 Simulation Module

Since there are still some short comings with the goal of this research's simulation when only using SimPy, SimComponents was developed for this research by borrowing from Grotto Networking's SimComponents the idea of process connection using SimPy. Resource's Store. Part class, which is shown in Figure 11, is one of the components of SimComponents, refers to objects on which work is being performed during a process. Part class represents the product in the simulation and includes various property (size, weight, etc.) information of the product. Source and sink classes are virtual processes that perform part class creation and destruction. Parts created in the source class are moved to the first process of the corresponding part. Parts whose planned work has been finished are moved to sink class and deleted. Process class is an object where works related to parts are made. Process receives parts from preceding processes (or from Source, in case of the preceding process being the initial process) and moves them to the following process (or to Sink, in case of the following process being the last process) after working on them during a set period of time according to the schedule. Resource class is an object that models facilities, areas, workforce, and transporters required for process progress. It uses resources classes provided by SimPy, or, briefly, creates and expresses internal variables. Finally, Monitor class record event logs that occur during simulation. The analyzer module analyzes results based on these records.

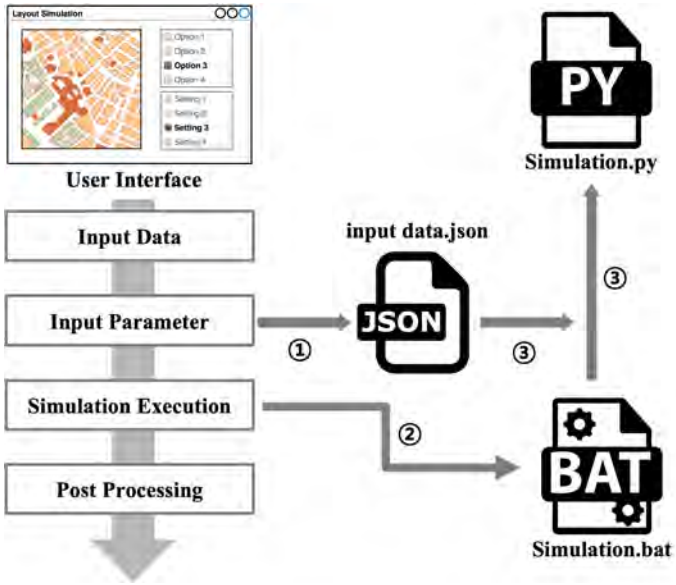

Figure 5: The way of connection with user interface and python files.

2.4 User Interface

In this research, for simulation data input, adjustment of parameters, and verification of results, a method of connection between a Windows User Interface and Python simulation algorithms was devised. Although there are many cases in which user interfaces are developed in C languages (C, C++, C#, etc.) by 3rd party software providers that provide rich UI object libraries, in this research, the DES program was developed with Python. Thus, connection with programs developed in different environments is necessary.

Figure 55 shows the method of connection between user interface and the DES program. Both the user interface and the DES program are developed independently. The user interface is equipped with functions for data input and verification, and on-screen display of post-processing indicators. The procedure of connection between user interface and the DES program is as follows. First, the user inputs data required for the simulation. The input data(data path, parameters, etc.) is saved by json format(①). When the user click execution button, the program makes the execution file as batch format (②). Right after creating run file(②), simulator has run with input data file(①) and algorithm file(③). After this, the DES program executed in the batch file proceeds with data pre-processing, simulation, and post-processing. The user interface reads post-processed files that have been stored in the path specified for each project by the DES program and displays the results visually on the screen. The advantages of this method are that, since simulation algorithms and program environments operate independently, both the user interface and the DES program can be debugged separately when a problem occurs. Also, the features of Python programs can be shared in a Windows environment.

3 DES FRAMEWORK FOR LAYOUT SIMULATION

The target system of simulation is the logistics of a shipbuilding company. This research sets the minimum unit of this simulation as the block which should be moved by transporter. The grand block which is after small blocks are put together at assembly process should pass through pre-outfitting process for working inside the block, painting process, and pre-erecting process which the last process before combined blocks to ship. If the time between the end of prior activity and the start of post activity is more than 3 days, the block should get to the stockyard.

Even though this research is based on the DES Framework (Figure 11), some SimComponents parts corresponding to simulation were partially modified and applied to layout evaluation simulation (Figure 11). First, the class structure was adjusted based on the movement of hull blocks, which are objects of logistics movement. Since the creation and destruction of blocks is closely related to the block assembly process, source class and process classes were unified as process classes. Also, the reporter module was replaced by a user interface.

3.1 Adapter and Modeler Module

In the adapter module, data entered by the user is converted into a format that can be processed in the simulation. In addition to Section 2.2's pre-processing, activities that are irrelevant to logistics were also removed. After this, the modeler module sorts data processed in the adapter module and connects it with the necessary information in each class of the simulation module. Table 22 shows input information linked to the main classes.

3.2 Simulation Module

SimComponents, which corresponds to simulation implementation and logic, is composed by part class (task objects), process class (part class creation and tasking), resource class (movement and assignment of resources required for task), sink class (part class deletion), and monitor class (recording of simulation events).

Table 2: Input data for each class.

Class		Input data
Part class		Part Information (name, weight, …)
		Initial location
		Working data
Source class		Part classes
		Distance information
		Resource information
Resource class	Transporter	Transporter Information (name, capacity, working yard)
	Factory, Stockyard	Name
		Area
Monitor class		Path of Event Tracing File

3.2.1 Part Class

Part class models hull blocks. Class objects themselves were implemented to move according with the movement of actual blocks. In part class, block characteristics (name, weight, area, and data) are defined as property values. Part classes work while moving between workshop and stockyard, and are deleted in sink objects after assembled into a block of a higher rank.

3.2.2 Process Class

Process class manages the tasking and movement of block objects, and is in charge of all decision-making from block creation to workplace selection, movement to workshop and stockyard, and deletion after they have been assembled into a block of a higher rank. The first decision-making that occurs in Process class is whether to create a block or not. There are two types of block creations, as shown in Figure 66. In the three types, it is assumed that Part C, which is parent block, is created after Part A and Part B, which are child blocks, are assembled.

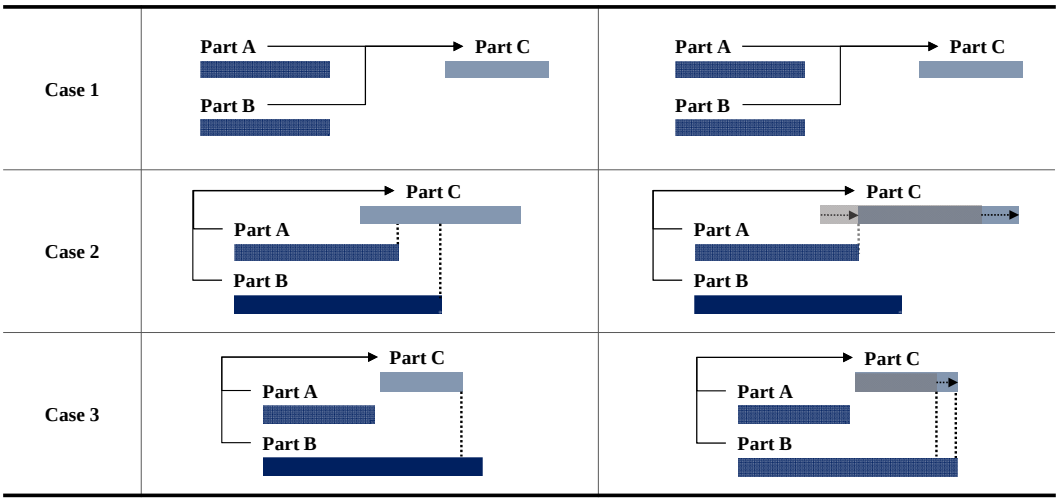

Figure 6: Constraints for assembly.

In this research, the restrictions of cases 2 and 3 were solved through the Store objects of SimPy.Resource. Store releases objects that are stored in it if they are called from the outside. If there are no objects to be released inside Store, the outside waits until an object is put inside Store, and, once there

is an object to release, the requested task is performed without delay. Likewise, in this simulation, by setting the condition of item input inside Store for child blocks to be completed, parent block creation and first task end signals are detected and executed without iterative loop search and time delay.

3.2.3 Resource Class

The resource class is divided into a class that models each resource, and a class that manages resources (management class). Resources required for the production process are transporters, which are in charge of movements, and factories and stockyards, whose area is occupied for block tasks. Each resource determines whether it is preempted or available, and uses SimPy. Resource´s Resource and Container objects for allocation without time delay. Classes that manage resources are composed by methods that select transporters, factories, and stockyards by type. Once a resource is requested by a process class, said resource is released to the requested block taking into consideration the resource´s status, capacity, and distance. Just like in process class block creation, resource characteristics provided by SimPy are used for resource search and allocation. In case of queue due to lack of resources, the time when the resources will be available is searched without execution of iterative loops, and the resources are used without time delay.

Transporter search is also similar to workshop/stockyard search. First, transporters with a maximum loadable block capacity that is greater than the weight of the block to be moved are selected among currently idle transporters. Then, the transporter that is closest to the block to be moved is selected among those found in the first step. After this, the selected transporter is called. In case the search range is expanded to include even transporters that have already been assigned to other blocks, time delay between resource request and assign occurs. This task too was designed to search whether transporters are idle or not without time delay by reflecting SimPy.Resource characteristics. This transporter search method finishes its job when the selected transporter arrives at the block's current location, and returns the name of the selected transporter to Process class.

3.2.4 Monitor Class

The Monitor class logs events that occur during simulation. It logs block creation and deletion, allocated resources, and task progress events. This event log is used by the analyzer module to analyze simulation result indicators. Table 33 shows the list of events.

3.3 Analyzer Module

In the analyzer module, road usage, travel distance by block, transporter travel distance, and process loads are calculated based on event logs such as those shown in Table 3. In this research, the user interface was developed using DevExpress (Version 21.1.6), which is a C#-based Visual Studio Components Library. According to the user interface functions, users can input and edit data, input parameters, and check results.

4 RESULTS AND DISCUSSION

In terms of the user environment, Figure 7 exemplarily shows a screen through which the input data can be checked, and through which users can edit and store data inside the program. Data edited by users is reflected in real time in actual data files and stored again. In a further screen, parameters to be used in simulations can be set. In this screen, lag time, which is the basis for simulation target period and stockyard movement, can be input. Users input all information and perform simulation by pressing the 'Run' button. Here, data paths and parameters input by user are stored as json files and delivered to batch files that perform simulations. Once the simulation starts, its progress is displayed in the form of a console.

Table 3: Events of DES.

Class	Event	Description
Process	Block Created	Block starts at Process class
	Work Start	Block starts working at Process class with preempted area
	Work Finish	Block finishes working at Process class
	Process to Process/Stockyard	Block moves from Process to another Process/Stockyard
	Stockyard to Process	Block moves from Stockyard to next Process
	Child to Parent	Child block moves over to its parent block for assembly
Resource	Area Request	Block requests area of factory or stockyard to Resource class at Process class
	Area Preempted	Block gets resource's area
	Area Release	Block release taken area to Resource class
	Transporter Requested	Transporter is requested after finding appropriate transporter
	Transporter Assigned	An appropriate transporter is assigned requesting block as the transporter gets available
	Transporter Loading Start	Transporter put in requesting block's location factory/stockyard and transporter takes the block
	Transporter Loading Completed	Transporter and Block arrive at destination factory/stockyard
Sink	Block Completed	Block deleted in Simulation at Sink class

Figure 8 exemplarily shows the road usage result screen, in which road usage is expressed in GIS by colors depending on the frequency of use. Figure 9 shows a screen in which the daily usage of resources (transporters, factories, and stockyards) is shown by a graph. The upper graph shows the daily usage of transporters, in which users can check results by selecting the operating zone and capacity. The lower graph shows the daily rate of operation of factories and stockyards. Figure 10 shows a histogram of the total travel distance of blocks. BOM information of blocks is shown in the table to the left, and to the right, a histogram of the travel distance of blocks is shown.

This study is to develop a layout simulation program using SimPy, and the developed layout simulation program shows superior performance compared to the commercial DES program (Plant Simulation of Siemens) used in shipyards. It is known that it takes about 20 to 30 minutes to accurately check the simulation speed of Plant Simulation for a 6-month production schedule. The execution speed of the layout simulation program developed in this study took about 2 minutes under similar conditions. The exact execution time is 113 seconds for half year under Intel® Core i7-1065G7 CPU, RAM 16GB. This result was certificated by official certification authority. This performance improvement is expected to make it possible to perform various experiments on the process variables affecting the layout by overcoming the disadvantages of the existing slow speed.

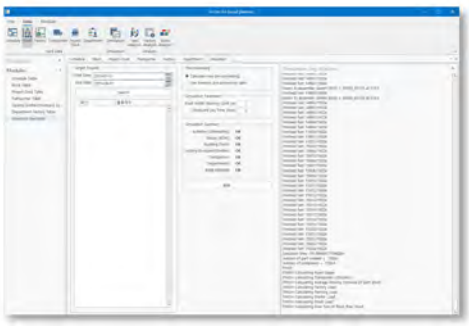

Figure 7: Input parameter for simulation.

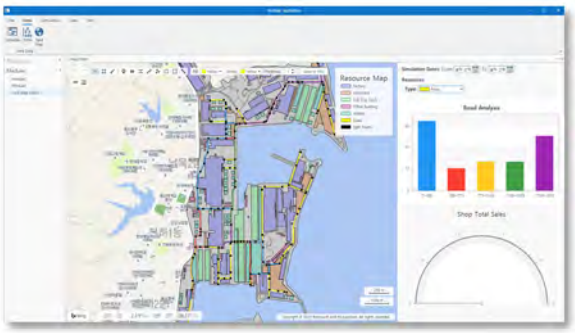

Figure 8: Result of road used frequency.

Figure 9: Result of workload of resource.

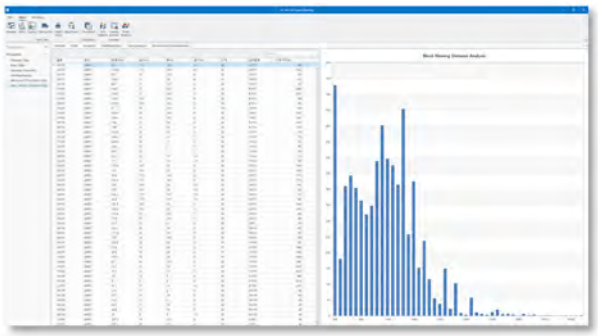

Figure 10: Result of moving distance of blocks.

5 CONCLUSION

In this research, a DES program based on packages of SimPy, which is a Python-based open source, was developed for production system analysis. A logistics simulation program through which shipyard layouts can be evaluated was developed based on the proposed DES Framework and applied to actual production yards of the shipbuilding industry. In this program, whether blocks have been assembled or not can be determined according to the characteristics of ship production, in which many blocks are brought together to form a single ship. The developed DES program can overcome customizing difficulties and low speed problems, which are the disadvantages of existing commercial DES programs.

REFERENCES

Dagkakis, G., & Heavey, C. 2016. A Review of Open Source Discrete Event Simulation Software for Operations Research. *Journal of Simulation* 10(3): 193-206.

Halim, N. N. A., Shariff, S. S. R., & Zahari, S. M. 2020. "Modelling an Automobile Assembly Layout Plant Using Probabilistic Functions and Discrete Event Simulation". *International Conference on Industrial Engineering and Operations Management*, August 10th -14th, Detroit, Michigan, USA

Hamzas, M., Bareduan, S., Zakaria, M., Tan, W., & Zairi, S. 2017. "Validation of X1 motorcycle model in Industrial Plant Layout by using WITNESSTM Simulation Software". *American Institute of Physics Conference Proceedings (1885)*, 020182-1 - 020182-10, https://aip.scitation.org/doi/10.1063/1.5002376.

Jeong, Y. K., Lee, P., & Woo, J. H. 2018. "Shipyard Block Logistics Simulation using Process-Centric Discrete Event Simulation Method". *Journal of Ship Production and Design* 34(02): 168-179.

Kim, D. S., Park, C. S., & Moon, D. H. 2012. "Determination of New Layout in a Semiconductor Packaging Substrate Line using Simulation and AHP/DEA". *Industrial Engineering interfaces* 25(2): 264-275.

Lee, Y. G., Ju, S., & Woo, J. H. 2020. "Simulation-Based Planning System for Shipbuilding". *International Journal of Computer Integrated Manufacturing* 33(6): 626-641.

Ndiaye, M. A., Dauzère-Pérès, S., Yugma, C., Rullière, L., & Lamiable, G. 2016. "Automated Transportation of Auxiliary Resources in a Semiconductor Manufactoruing Facility". In *Proceedings of 2016 Winter Simulation Conference*, edited by Theresa M.K. Roeder, Peter I. Frazier, Robert Szechtman, Enlu Zhou, Todd Huschka, and Stephen E. Chick, 2587-2597. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Shin, J. G., Song, Y. J., Lee, D. K., & Woo, J. H. 2009. "A Concept and Framework for a Shipyard Layout Design based on Simulation". *Journal of Ship Production* 25(3): 126-135.

Song, Y. J., Lee, D. K., Woo, J. H., & Shin, J. G. 2010. "System Development and Applications of a Shipyard Layout Design Framework". *Journal of Ship Production and Design* 26(2): 144-154.

Song, Y. J., Woo, J. H., & Shin, J. G. 2009. "Research on a Simulation-Based Ship Production Support System for Middle-Sized Shipbuilding Companies". *International Journal of Naval Architecture and Ocean Engineering* 1(2): 70-77.

Woo, J. H., & Oh, D. 2018. "Development of Simulation Framework for Shipbuilding". *International Journal of Computer Integrated Manufacturing* 31(2): 210-227.

Woo, J. H., & Song, Y. J. 2014. "Systematisation of Ship Production Management and Case Study for Ship Block Assembly Factory". *International Journal of Computer Integrated Manufacturing* 27(4): 333-347.

Woo, J. H., Song, Y. J., Kang, Y. W., & Shin, J. G. 2010. "Development of the Decision-Making System for the Ship Block Logistics Based on the Simulation". *Journal of Ship Production and Design* 26(04): 290-300.

Zúñiga, E. R., Moris, M. U., Syberfeldt, A., Fathi, M., & Rubio-Romero, J. C. 2020. "A Simulation-Based Optimization Methodology for Facility Layout Design in Manufacturing". *IEEE Access 8*: 163818-163828.

AUTHOR BIOGRAPHIES

SO-HYUN NAM is a graduate student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. Her research interest is in DES simulation, Reinforcement learning and queuing. Her email address is sohyon525@snu.ac.kr.

SEUNG-HEON OH is a graduate school student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He has a Bachelor of Science in Korea Naval Academy and is a officer (lieutenant commander) in R.O.K. Navy. His research interests include modeling & simulation and reinforcement learning. His email address is suenghuny@snu.ac.kr.

HEE-CHANG YOON is a graduate school student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He has a Bachelor of Science in Naval Architecture and Ocean Engineering. His research interests include DES simulation, queuing theory, optimization, and machine learning. His email address is gmlckd12@snu.ac.kr

YOUNG-IN CHO is a graduate school student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He has a Bachelor of Science in Naval Architecture and Ocean Engineering. His research interests include DES (Discrete Event Simulation) and reinforcement learning. His email address is whduddlsi@snu.ac.kr.

KI-YOUNG CHO is a graduate school student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He has a Bachelor of Science in Naval Architecture and Ocean Engineering.. His research interest is in optimization and machine learning. His email address is kiyoung8@snu.ac.kr.

DONG-HOON KWAK is a graduate school student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He has a Bachelor of Science in Naval Architecture and Ocean Engineering. His research interests include production engineering, queuing theory, optimization, and machine learning. His email address is s2arta2s@snu.ac.kr.

JONG HUN WOO is an associate professor in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He holds a PhD in Naval Architectur and Ocean Engineering from Seoul National University. His research interest is in DES simulation, APS(Advanced Planning and Scheduling), machine learning and queuing, and he has relevant research experiences in the application areas of shipbuilding. His email address is j.woo@snu.ac.kr.