Simulation Modelling Practice and Theory 71 (2017) 45–60

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Intelligent simulation: Integration of SIMIO and MATLAB to deploy decision support systems to simulation environment

Mohammad Dehghanimohammadabadi[a,*], Thomas K. Keyser[b]

[a] Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA 02115, USA
[b] Department of Industrial Engineering and Engineering Management, Western New England University, Springfield, MA, USA

ARTICLE INFO

Article history:
Received 14 February 2016
Revised 23 August 2016
Accepted 25 August 2016

Keywords:
Intelligent simulation
Simulation-Optimization (SO)
Iterative Optimization-based Simulation (IOS)
SIMIO
MATLAB

ABSTRACT

Discrete-event simulation is a decision support tool which enables practitioners to model and analyze their own system behavior. Although simulation packages are capable of mimicking most tasks in a real-world system, there are some decision-making activities, which are beyond the reach of simulation packages. The Application Programmers Interface (API) of SIMIO provides a wide range of opportunities for researchers to develop their own logic and apply it during the simulation run. This paper illustrates how to deploy MATLAB, as a computational tool coupled with SIMIO, as a simulation package by using a new user-defined step instance named *"CallMATLAB"*. A manufacturing system case study is introduced where the CallMATLAB step instance is used to create an Iterative Optimization-based Simulation (IOS) model. This model is created to evaluate the performance of different optimizers. The benefits of this hybridization for other industries, including healthcare systems, supply chain management systems, and project management problems are discussed.

1. Introduction

Simulation modeling is increasingly being used to address a variety of issues in several disciplines including but not limited to: healthcare, manufacturing, service industry, supply chain systems, and risk analysis [1–4]. To adopt improvements in organizations, sufficient justification is required to convince managers [5]; therefore, simulation could be utilized as a pivotal decision-making and risk analysis tool to determine and justify the attractive configurations of the system [4]. Simulation software packages provide an abstraction of the real world with the most common functional needs for the modeling. However, there are some activities in real-world systems which simulation software packages are not able to address properly. These activities are primarily human decision-making analysis or highly computational support tools, which are developed to improve the functionality of business organization processes. With the aim of long-term strategic planning, it is crucial to bring these decision-making activities into a simulation model. In order to clearly explain the challenge and benefits of the presented model in this paper, an illustrative example of a manufacturing system is presented, which will be revisited throughout the paper.

Consider a three-stage flow shop system with parallel non-identical machines at each stage exhibited in Fig. 1. Different product types (jobs) with different operating times in each machine are processed based on their priority. The setup type for each product is sequence-dependent, which indicates the setup time between two different products on the same machine

* Corresponding author.
E-mail address: m.dehghani@neu.edu (M. Dehghanimohammadabadi).

http://dx.doi.org/10.1016/j.simpat.2016.08.007
1569-190X/

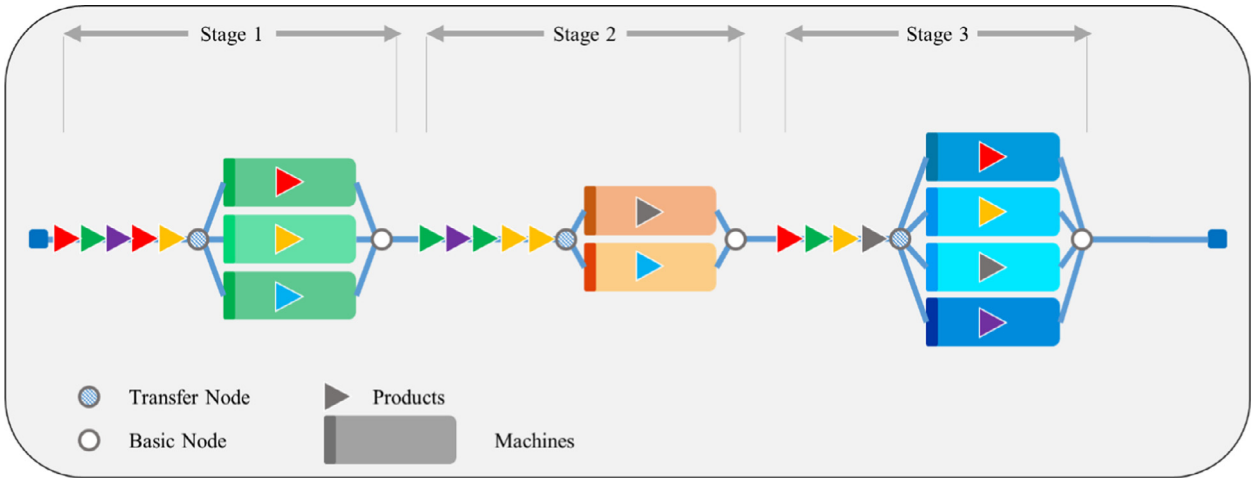

Fig. 1. A three-stage flow shop manufacturing system with parallel non-identical machines at each stage.

might have different values depending on both the product just completed and the product to be processed [6]. Transfer nodes dispatch the products among the available machines based on the selection rule that is defined by the user. Deploying traditional dispatching rules such as: FIFO (first in-first out) or SPT (shortest processing time) in most of the simulation packages is trivial; however, the challenge arises when one is intended to apply the same optimization algorithm that is being used in a real-life system to prioritize the products. Therefore, an optimization module needs to be integrated with the simulation model to handle products' scheduling whenever needed. But, the fact is, due to the notable limitations of the simulation packages and languages, deployment of optimization modules within the simulation run leads to difficulties. These limitations and drawbacks are described as follows.

First, to efficiently implement most of the decision-making type of activities, a custom programming is required, which, due to the absence of simulation commands, is not practical. Also, simulation languages are not general purpose programming languages [7,8] and certainly, the amount of time and effort needed to learn and become experienced with a general purpose simulation language is not trivial [9]. Moreover, these languages are not well suited to be problem-specific [10]. As a result, to advance flexibility and efficiency of the simulation model, it is desirable to integrate simulation with a computational tool instead of contributing the simulation engine to the heavy computational tasks. In this integration, an interactive software package is called up to externally perform human decision-making tasks, such as optimization, when needed. This perfect mechanism extends the functionality of the simulation and makes it potentially more intelligent and accurate, because it provides a realistic perspective of a decision-making environment.

As is discussed later in Section 2, there is no evidence in the literature for adding a generic block-set module to a simulation engine which could solve heavy mathematical computations in a simulation environment. Although this integration is promising, embedding this module into the current simulation software packages is quite challenging. More importantly, the embedded module has to be deployed with minimum effort and least amount of development time, just like applying traditional dispatching rules. Fortunately, with the advance of current simulation packages, this advancement is attainable.

In this study, a generic hybridization of a problem-solving module and a simulation software is developed by integrating SIMIO and MATLAB. SIMIO is a simulation software package designed around basic object-oriented principles [11]. One of the advantages of SIMIO, among many, is its *Application Programmers Interface (API)* capability, which allows users to customize or extend their desired model properly. The API capability of SIMIO is leveraged to exploit MATLAB, as a computational manager to SIMIO, as a simulation manager. The purpose of this integration is to allow users to easily implement their own logic in simulation. This integration is essential because it adds a new dimension to the simulation modeling approaches. Using this advancement, the simulation model could invoke MATLAB to perform an intelligent processing algorithm within the simulated time. This addition can support different problem solving functions including optimization, multi-criteria decision-making (MCDM) analysis, fuzzy logic, neural networks, mathematical calculations, and more. Therefore, this paper endeavors to attain the following goals:

- leverage the API capability of SIMIO in order to create a new user-defined step instance which incorporates MATLAB as an integral part of the simulation manager,
- present an Iterative Optimization-based Simulation (IOS) model case study to illustrate the pitfalls and promises of the new advancement,
- demonstrate the benefits of this hybridization for a few other industries including healthcare, supply chain management systems, and project management problems to provide some insights for researchers, practitioners, and policymakers to consider how to deploy their own logic within a simulation model.

The rest of this paper is organized as follows: the literature review of intelligent simulation frameworks is provided in Section 2. The details of designing, developing and implementing an intelligent simulation model are provided in Section 3. One of the applications of the IOS framework is comparing the performance of different optimizers, which is discussed in Section 4. In this section, three optimizers including: Genetic Algorithm (GA), Simulated Annealing (SA) and Particle Swarm Optimization (PSO) are applied in a simulation model of a manufacturing system, and their performance is evaluated. The benefits and future advancements of the current integration for a few other industries are discussed in Section 5.

2. Literature review

Discrete Event Simulation (DES) has been proven as a powerful tool to accurately model complex systems for analysis and planning [12–14]. Although available simulation software packages could undertake a wide spectrum of realistic activities, they struggle to properly incorporate all of the operational policies in a model. Therefore, programmers have tried to integrate multiple modules or applications to simulation software in order to add their desired logic to the simulation model. In this section, a variety of intelligent simulation models are reviewed, and relevant studies are addressed. Then, the novelty of this study is lined out.

To enable a modeler to abstract a domain-specific language to a level that permits its interaction with other system models, Paredis and Johnson [15] explored a method which is based on graph transformations. This model translates structural descriptions of a system into corresponding simulation models ready to be executed in a simulation tool. Shirazi et al. [16] presented an intelligent co-simulator for real-time production control of a complex flexible manufacturing system (CFMS) with machines and tools flexibility. Their developed system is coupled with the Centralized Simulation Controller (CSC), a real-time simulator for enforcing dynamic strategies of shop floor control. The intelligent co-simulator attempts to find the values of the input parameters of a simulation model such that all of the objectives are satisfied.

Integration of simulation and optimization is a promising avenue of research, which increases the simulation functionality. A variety of approaches exist in the literature for combining simulation and optimization, but only a few of them discussed imbedding an optimization module into a simulation engine. The study by Subramanian and his colleagues [17] is one of the few, in which an optimization module is used to solve a stochastic optimization problem in a Monte-Carlo simulation platform. Mejtsky [18] developed a simulation-optimization model, in which a branching approach mirrors the decision tree of an optimization problem to evaluate different simulation runs during their execution. This model is elaborated with a pruning phase, which controls the number of concurrent simulation models and returns “friendly” models to the objective function. Sivakumar [19] presented a DES model for an “on-line near-real-time” dynamic multi-objective scheduling system. Their model was capable of deploying rules and policies within the simulation run.

One of the approaches to include uncertainty to the simulation model under real-world conditions is using fuzzy logic. Embedding fuzzy logic into DES enables modelers to adequately capture imprecise, subjective, and linguistically expressed knowledge in the simulation inputs [20]. To facilitate the decision making regarding the improvement of a project quality management (PQM) system, Corona-Suárez et al. [21] presented a Fuzzy DES (FDES) model. This model incorporates the subjectivity and uncertainty implicit to assess the effect of PQM factors on the performance of construction operations. Hosseini et al. [22] integrated a fuzzy expert system with DES to properly configure the resource levels of an emergency unit. Sadeghi et al. [20] developed an FDES framework, in which a new approach is utilized to calculate event times to increase the accuracy of simulation time estimation. Ali and Seifoddini [23] addressed dynamics of a manufacturing system using an intelligent knowledge system in order to have a better representation of real-life models. In this study, fuzzy rule-based modules are included into the simulation-modeling environment to transform the human expertise into IF-THEN rules.

Another way to create an intelligent simulation model is using MCDM methods. In the study conducted by Rabelo et al. [31], Analytic Hierarchical Process (AHP) analysis is used at the back-end of the simulation results in order to reach a better decision. Another application of the MCDM combined with DES is discussed by Eskandari et al. [32] to efficiently investigate the patient flow of the Emergency Department (ED) using AHP and TOPSIS.

In some studies, neural networks are incorporated with simulation to significantly reduce the computational burden involved by the simulation engine, and to provide reliable estimation [29]. The studies conducted by [27–32] are some examples of such attempts to integrate neural networks and simulation.

MATLAB, as a comprehensive mathematical application with computational capabilities, is a fruitful candidate for intelligent simulation development. Liang and Yao [24] presented a hybrid analytic and simulation modeling approach for manufacturing systems using Arena, MATLAB’s Simulink, and Stateflow. This integration is used to model a large, hierarchical, complex manufacturing system consisting of one or more lines. In their study, this hierarchical model is divided into three levels: workshop level, machining state level, and cutting. These levels are built in the Arena, MATLAB Stateflow and MATLAB Simulink environments respectively in order to form a hybrid modeling approach. Integration of the Simulink debugger into the .NET platform is introduced in a study by Gao et al. [25] to enhance interactivity for designing simulation applications. In their paper, a case study for a basic hydraulic pump system simulation is presented as a demonstration. Mušič and Matko [26] developed a blockset of a continuous simulation tool that facilitates the design of the discrete part of a simulation model, and integrated that into the Simulink environment. This addition can support control design in process control and manufacturing.

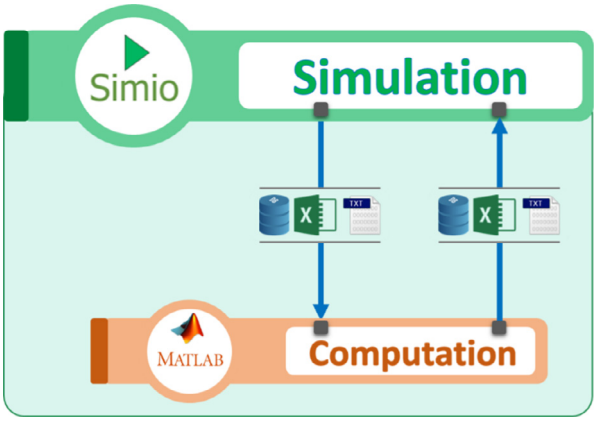

Fig. 2. Integrating SIMIO and MATLAB: The generalized framework.

All of the above mentioned studies tried to add a new dimension to the simulation modeling either by integrating optimization, fuzzy logic, MCDM or etc. In some cases, a specific combination of a simulation software and numerical/computational software tools is demonstrated. However, they are all customized and restricted for predefined user needs and none of them address a generic framework which could be used for several simulation model developments. Moreover, switching from one computational approach to another needs intensive effort and time. For instance, substituting an optimization module with an MCDM approach is not practical in the existing intelligent models. More importantly, in most of these studies, the simulation engine and the computational software/module are operating separately. This is an obstacle to building a real-life model, where decisions are changing continually, and therefore, the intelligent module needs to be called multiple times within the execution of the simulation model.

This paper presents an addition to the simulation environment, SIMIO, to enable modelers to apply the above mentioned capabilities for their modeling in a simplified and practical way. The presented addition to the simulation package could embrace controls, decision-making models, optimization, calculation etc., similar to the real-world situation. The framework structure and a case study are presented as follows.

3. Framework structure

In this section, a generalized conceptual architecture of the model is described, and its components are introduced. Subsequently, the components' related operational processes are identified. Finally, the implementation of the proposed advancement on the previously provided example of the manufacturing system is explained.

3.1. Intelligent simulation framework: a generalized structure

Simulation packages can capture most of the real-world systems' behaviors without demanding a particular implementation. However, when a sophisticated human decision-making needs to be deployed, the limitations of the simulation packages become an obstacle. Therefore, integration of a simulation and a highly computational support tool is expected to make the simulation model more efficient and robust. In this study, as shown in Fig. 2, a generalized hybridization of a simulation software and a problem-solving module is developed by integrating SIMIO and MATLAB. The main stakeholder of the model is simulation, and whenever a need for a sophisticated calculation arises, the computation software is called to handle it. It is noted that, during this transaction, simulation run is paused, and delayed until the solution is provided by the computational software.

A constant data connection is required to accomplish this integration. Both SIMIO and MATLAB support common data formats including text file (*.txt), Excel file (*.xlsx) and even databases like MySQL. Depending on the simulation model, users may prefer to use any other data transferring format over another. In the case of long simulation runs with huge amounts of data, using a database is recommended since it is secure, reliable, fast enough to handle the big data [33] and can easily integrate with both packages.

3.2. Components of the framework

Two software applications used in this study, SIMIO and MATLAB, are explored in this section. The capabilities of both of packages are explained, and the reasons that these are selected are provided.

3.2.1. MATLAB: the computational manager

MATLAB® is a computational software package with a flexible environment for technical computing, which integrates mathematical computing, visualization, and numerical analysis [34]. It has a powerful computer programming language,

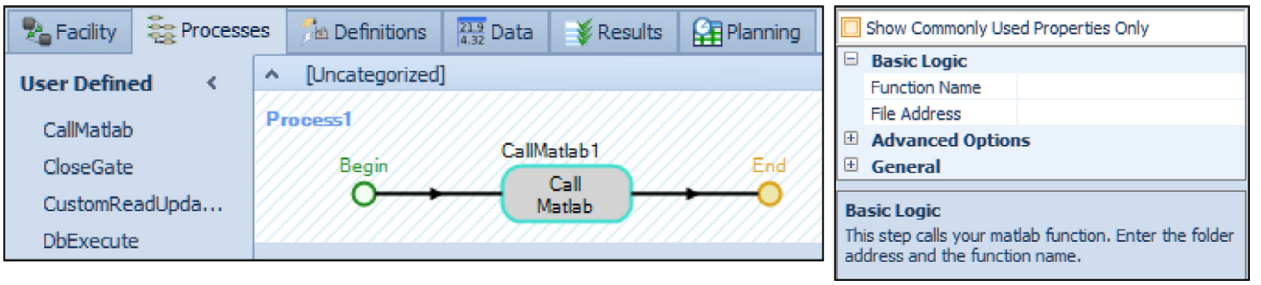

Fig. 3. Using CallMATLAB step instance with 'Function Name' and 'Folder Address' properties to develop a SIMIO process.

which is faster than more conventional programming languages [35]. Due to its simple and intuitive interface, advanced tools and competencies to solve the problems, MATLAB is one of the most widely used mathematical computing environments in technical computing [36]. The problems with mathematical calculations like, optimization, MCDM, fuzzy sets, neural networks, etc., can be conveniently carried out in MATLAB. Moreover, because of its Component Object Model (COM) technology [37], incorporating MATLAB with other software applications is straightforward. Using COM, developers and end users can select application specific components produced by different vendors and integrate them into a complete application solution. Hence, all of the above-mentioned reasons led us to select MATLAB as the *"Computational Manager"* in order to develop the intelligent simulation model.

3.2.2. SIMIO: the simulation manager

SIMIO is a discrete event simulation software package designed around basic object-oriented principles [11]. It has a user-friendly environment and incorporates concurrent 2D and 3D views in order to assist developers in selecting displays that are better suited for the model development. A key innovative feature of SIMIO is its *"processes"* which enable its programmers to modify an object's behavior. As a part of the object execution in the simulation model, arbitrary processes of the model can be executed to change or expand the underlying object's behavior. Creating a process for each object is similar to creating a stylized flowchart, except that the components of the flowchart are logical *"steps"* such as: Assign, Decide, Seize, Transfer, Delay, Release, etc. [38]. One of the advantages of SIMIO, among many, is its *"Application Programmers Interface" (API)* capability, which is the main reason that we decided to select SIMIO as the *"Simulation Manager"*. This capability allows users to be extremely productive by providing limitless enhancement possibilities to customize or extend their desired model [39]. The extension could be adding new steps, elements and rules, importing and exporting data, enhancing experimentation with external algorithms, or interfacing from external programs [39]. Therefore, we leveraged the API capability to develop a user-defined step instance labeled *"CallMATLAB"*. This step instance is designed to exploit MATLAB, as a computational tool within the simulation run in SIMIO. The details about this integration are provided in Section 3.3.

3.3. SIMIO and MATLAB integration

In this section, the developed CallMATLAB user-defined step is briefly presented. Since the development process is technical, we refer interested readers to Appendix A, where a complete user-defined step development guideline, plus C# coding are provided. We hope the provided guidelines would be beneficial for interested readers to get familiar with the API capability of SIMIO and even customize or extend the proposed user-defined step instance.

After installing SIMIO, the Visual Studio template has to be installed. This setup adds a template for custom *"step"* and custom *"element"* in Visual Studio under the Visual C# tab. By default, SIMIO provides a C# class template called *"UserStep.cs"* which enables programmers to develop their desired steps by coding their own logic. This template is customized and coded to create the desired user-defined step, CallMATLAB, which requires two properties. As it is shown in Fig. 3, the first property of this step is *"Folder Address"*, which refers to the directory that the MATLAB file is located in. The second designed property is *"Function Name"* which denotes the name of the MATLAB file that source codes are saved in.

Once this step is added to the SIMIO environment, it could be found at the *"User Defined"* panel within the *"Processes"* window. Note that the coded DLL file and the instruction to add the developed *CallMATLAB* step instance to the SIMIO environment are available at the SIMIO user forum website at this address: http://www.SIMIO.com/forums/viewtopic.php?f=33&t=2185.

Similar to the existing step instances in SIMIO like: Assign, Seize, Delay, Release, Decide, etc., the user could easily drag-and-drop CallMATLAB while building processes. Once the token passes through the process, the simulation model is momentarily paused until the MATLAB function executes and provides the solution. The following section illustrates how this step could be utilized when building a simulation model.

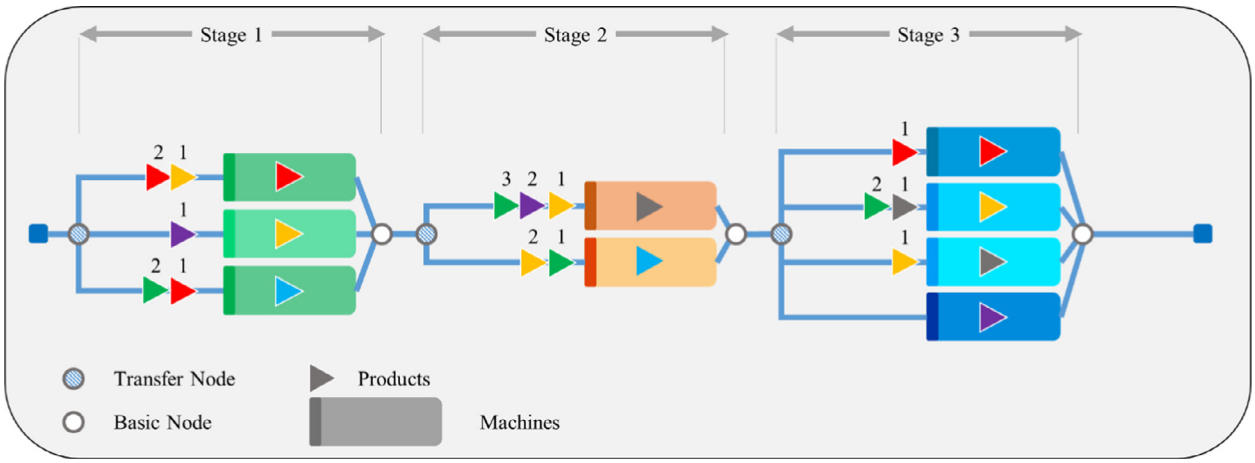

Fig. 4. Dispatching products among the machines according to the scheduling provided by the optimization algorithm within the simulation run.

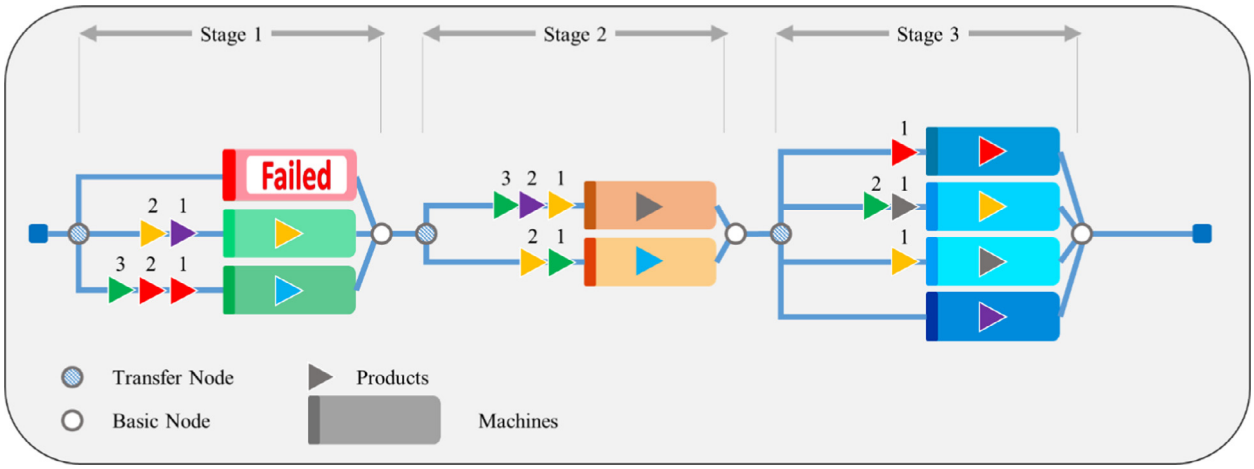

Fig. 5. Re-distributing the products among the available machines in Stage 1, due to the machine failure.

3.4. Implementing the CallMATLAB step: the manufacturing system example

Contemporary simulation optimization techniques are becoming increasingly popular among researchers. Using the developed step instance, CallMATLAB, simplifies developing an IOS model. In the IOS, an optimization module is embedded inside a simulation manager to optimize the system's performance [14,40]. Using the CallMATLAB step instance simplifies developing an IOS model. This section shows how the newly developed step instance could be applied to implement an intelligent simulation model for the previously mentioned example.

In fact, in this manufacturing example, instead of using heuristic dispatching rules, an optimization algorithm is applied for the products' distribution and prioritization among the machines. In order to accurately mimic this optimization task in a simulation model, the CallMATLAB step can be used during model development. As shown in Fig. 4, the optimization algorithm solves a scheduling problem and assigns the prioritized products to the machines. This assignment optimizes goals of the model according to the most recent status of the system. Therefore, any combination of products and machines could represent, and on machines with more than one job on the queue, the job with the lowest priority number is processed.

In most real-world environments, scheduling is an ongoing reactive process, and is continually subject to change due to the inevitable presence of a variety of deviations [41]. These deviations are called trigger events, which could be predictable i.e., scheduling on the beginning of each shift, or unpredictable, i.e., re-scheduling due to machines' failure [42]. The unpredictable changes are monitored either by limit-charts, thresholds or events [43].

To illustrate such events in this example, consider a situation when one of the machines fails, and its assigned products have to be taken over by the other available machines. In other words, in this case, a mathematical problem needs to be solved considering the current status of the system, to provide the new scheduling of products. Then, the simulation is able to use the new scheduling to re-distribute the products among the available machines with new priorities. As illustrated in Fig. 5, one of the machines at the Stage 1 has failed. Hence, the simulation models is paused at this point, and a new

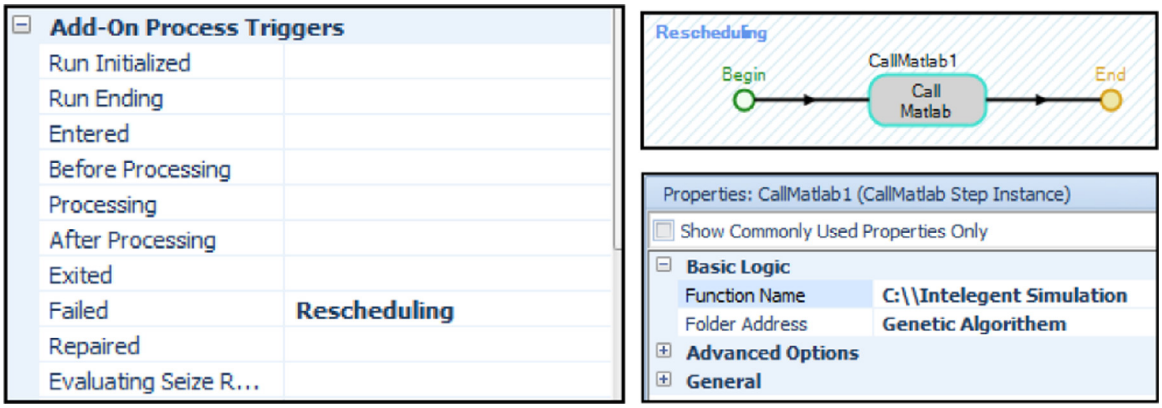

Fig. 6. Adding an *"Add-on Process Trigger"* for using CallMATLAB step instance.

mathematical model is re-parameterized considering the new status of the Stage 1. The optimization problem is solved, and the new scheduling is provided to the simulation. Consequently, simulation continues its run by applying the new configuration.

Adding such a capability to SIMIO is straightforward using CallMATLAB. To implement this, an *"Add-on Process Trigger"* is required to be added to the *"Failed"* status of the machine (server). As shown in Fig. 6, we labeled this process *"Rescheduling"* which includes a CallMATLAB step with the MATLAB function required information. Therefore, once the Machine fails, a token passes through this step and triggers the optimization algorithm in the MATLAB file.

Since the aim of the simulation is to mirror the real-world activities, users can apply this step instance whenever they routinely use optimization in a real-life situation. Depending on how often optimization is used in the system, the provided step would be called up during the simulation execution, to trigger the optimizer in MATLAB.

4. Case study: evaluating the performance of different optimizers

Since different optimizers have different performances for different systems, researchers can leverage the IOS by plugging their preference of optimizers to compare their performance before implementing them in a real-world system. Integrating MATLAB to SIMIO makes the simulation environment more dynamic. Regardless of the optimization's type, one could utilize MATLAB either to run the user customized optimization algorithm e.g. metaheuristics, or call commercial optimizers e.g. CPLEX, GMAS or LINGO.

In this case study, the *CallMATLAB* step is used to apply three metaheuristic algorithms including: Genetic Algorithm (GA), Simulated Annealing (SA) and Particle Swarm Optimization (PSO) in an IOS framework of a manufacturing system.

In this section the conceptual simulation model is defined, and the optimization algorithms are introduced. Then, the experimental results are analyzed and a few insights regarding the algorithms' performance are provided.

4.1. Conceptual simulation model

The problem modeled in this work is a scheduling problem with unrelated parallel machines. In classical scheduling problems, there are N number of jobs which can be processed by any of M available machines without preemption. Jobs have different processing times on different machines with Sequence-Dependent Setup Times (SDST). The objective is to solve a scheduling problem in terms of combined maximum completion time of the jobs, and overall tardiness and earliness cost, which generally is defined [44] as: $R \mid S_{ijm} \mid C_{\max} + \sum^{(} T_i. + E_i)$. The characteristics of the scheduling problem for investigation are presented in the following subsections. Several assumptions are made in this case, which are stated as follows:

- All machines are ready to be scheduled in time zero;
- Preemption of operations of each job is not allowed;
- The setup times are sequence-dependent (setup time varies from one job to another job on each machine);
- Different jobs have different processing times on each machine;
- Each machine can process only one job at a time;
- Each job has a distinct due date;
- All machines operate independently;
- There is unlimited buffer capacity in front of each machine.

To maintain shop floor load, the arrival mode of jobs to the system is *"On Event"*. In other words, once the total number of jobs waiting in queues drops below a certain threshold, a new batch of jobs enters the system. The simulation stopping condition is based on run length and is set for 2 days.

Machines	1			2				3				4	
Jobs	6	9	12	2	8	1	13	10	7	5	11	4	3
Priority	1	2	D	1	2	3	D	1	2	3	D	1	2

D: Delimiters

Fig. 7. An example of a permutation of 10 jobs and 4 machines.

Three trigger events are considered to develop this IOS model including: (1) new jobs arrivals to the system, (2) machines' failure, and (3) machines' repair. Once a new job or a new batch of jobs enters the system, a new scheduling is required to distribute the jobs among the available machines. Upon machines' failure or repair, the number of total available machines in the system changes, and therefore, unprocessed jobs have to be distributed among available machines. The occurrence of any of these major changes in the system pauses the simulation manager (SIMIO), and triggers the optimization manager (MATLAB) to get a fresh schedule according to the current status of the system. The objective function of this problem is to minimize weighted completion time of the jobs, and overall tardiness/earliness cost at the same time. Eq. (1) indicates the objective function which is as follows:

$$\min Z = w_1 C_{\max} + w_2 \sum_{i=1}^{N} (\alpha_i \rho_i + \beta_i e_i) \tag{1}$$

where w_1 and w_2 are weights of the completion time, and the overall tardiness/earliness cost of the jobs, $C_{\max}$ represents total completion time of jobs, ρ_i and e_i are tardiness, and earliness of the jobs, and the corresponding penalty costs of each are represented by α_i and β_i respectively. Parameter N denotes the total of number of jobs in the system.

4.2. Optimizers

As previously mentioned, three metaheuristic algorithms are utilized in the IOS framework to compare the effects of each of these optimizers on three measures of the simulation model. These algorithms are introduced as follows:

- **Genetic Algorithm (GA):** The first optimizer is GA, which mimics biological evolution. This algorithm employs random choices to have a highly exploitative search, keeping a balance between exploration of the feasible domain and exploitation of good solutions [45].
- **Simulated Annealing (SA):** SA is developed by Kirkpatrick, Gelatt, and Vecchi [46], and its name is inspired by annealing from metallurgy and has shown successful applications in a wide range of combinatorial optimization problems. This fact has motivated researchers to use SA in many SO problems [43,47]. This algorithm starts with an initial feasible solution, then uses a cooling schedule to move from one solution to another for identifying the optimal solution [47].
- **Particle Swarm Optimization (PSO):** This algorithm mimics the social behavior of the flying birds and their methods of information exchange. In this algorithm, the local best solution (local search) and global best solution (global search) are combined to improve the search efficiency [48].

Interested readers are referred to [49] for more details of these algorithms.

For a system with N number of total jobs and M number of available machines, a permutation size of $(N+M\ --\ 1)$ is needed to allocate machines. For instance, for a manufacturing system with 10 jobs and 4 machines, a permutation of 13 is needed. As shown in Fig. 7, a permutation of jobs is generated and the numbers greater than N in each permutation are considered as *"delimiters."* Delimiters separate the sequence of the jobs on one machine from the next. For instance, in this case, delimiters are 12, 13 and 11 respectfully. Therefore, jobs 6 and 9 are assigned to Machine 1, and Job 6 is prior to Job 9. The same rule applies to the other jobs allocation. Each of the above mentioned algorithms use their own logic to generate a number of permutations. These permutations are evaluated and the optimal/near optimal solution is found.

It is worthy to note that the aim of this case study is not showing superiority of one algorithm over another, but giving researchers and programmers insights about the potential capability of this addition to the simulation engine. To perform this case study, the conceptual model is validated and an experimental analysis is carried out. The model validation procedure, experimental setting and its results are provided in the following sections.

4.3. Verification and validation of the simulation model

To validate and verify the developed framework and simulation conceptual model, a multi-level verification exercise is performed to ensure correctness and credibility of the model which are listed as follows:

- Numerical assessment: Multiple numerical experiments are performed to ensure the simulation results match the expected values. For instance, the results provided by each of the optimization iterations are compared with the solutions provided by a manual optimizer, to verify the model structure.

Table 1
Experimental setting.

Optimizer types	Responses of interest
• Genetic Algorithm (GA)	• Execution time
• Simulated Annealing (SA)	• Average tardiness\earliness cost
• Particle Swarm Optimization (PSO)	• Max completion time of the jobs

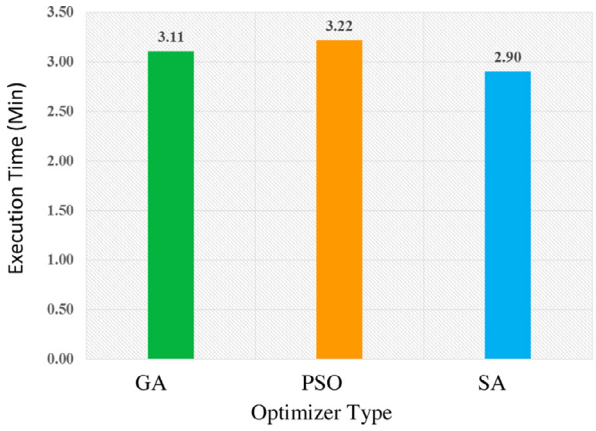

Fig. 8. Execution time of model for different optimizers.

- Debugging algorithms: The functionality of the developed user-defined step instance, CallMATLAB, is verified prior to inclusion into the simulation by debugged source codes and executing several pilot tests.
- Checking the process logic of the simulation model: A variety of different processes are created in order to develop the simulation model. All of this processes' logic are verified and evaluated under multiple simulation environments.
- Robustness of the simulation model: Trace capability and visually realistic animation of the model are used to validate the behavior of the developed system. These two techniques are utilized to ensure the validity of model's behavior.

4.4. Experimental setting and results

The considered factor in this experiment is optimizer type. The responses of interest are execution time of the model and two system performance measures, including: average tardiness/earliness cost of the job and max completion time of the jobs (Table 1).

Each scenario is replicated 10 times using the developed IOS model. Same Number of Function Evaluations (NFE) is used as a stopping criterion for each of the optimization algorithms. NFE indicates the finite number of times that each algorithm evaluates the objective function before resulting the optimal/good solution. In this study, NFE is set equal to 2000 for all of the algorithms. Analysis of variance (ANOVA) is used to analyze the experiment results. The initial results indicate that there is not a significant difference between the optimizers in terms of execution time, because p-value $= 0.337$ is greater than $\alpha = 0.05$. Although these optimizers are using different algorithms, this result is reasonable because, the same NFE is set as a stopping criterion for all of optimizers. As shown in Fig. 8, on average each replication takes about 3 min to run.

The first system performance measure that is studied is the average tardiness/earliness cost of the jobs. Statistical analysis indicates SA has better performance compared with GA and PSO. The ANOVA result shows p-value $= 0.004$, which is less than $\alpha = 0.05$. The graphical comparison of this test is shown in Fig. 9, which illustrates that 95% confidence interval of SA for tardiness/earliness cost is lower than the other two methods.

The max completion time of the jobs (C_{max}) is another response of interest in this study. Although the PSO average is lower than the other two algorithms, the statistical analysis shows there is not a significant difference between these optimizers in terms of C_{max}. The statistical confidence interval plot is provided in Fig. 10.

As a result, in this specific case study, SA showed a better performance than GA and PSO. Although these three optimizers performed equally in terms of execution time and C_{max} of the system, but SA dominated GA and PSO in terms

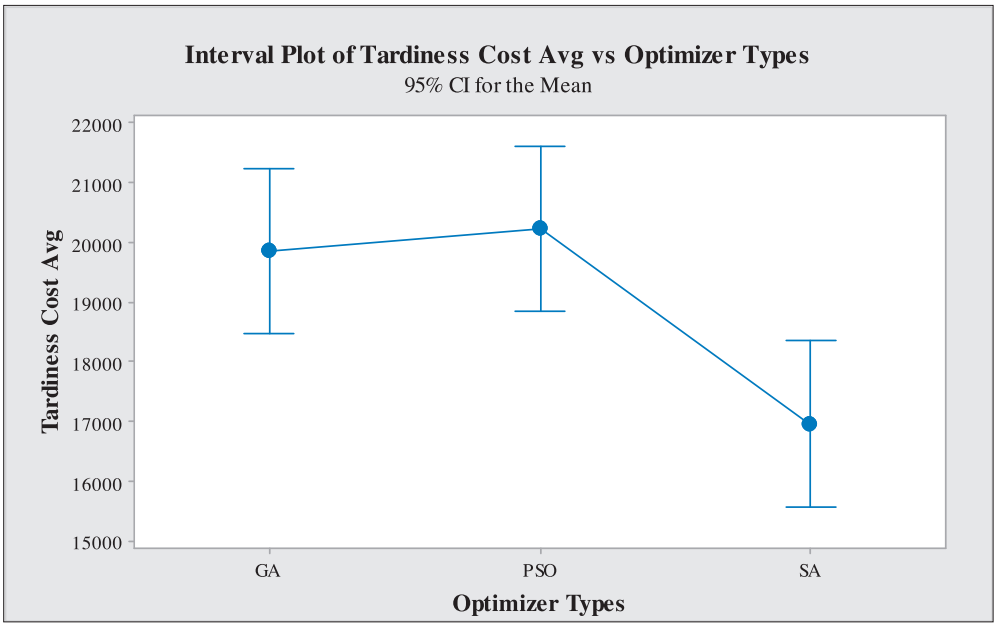

Fig. 9. Average tardiness/earliness cost of the jobs for different optimizers.

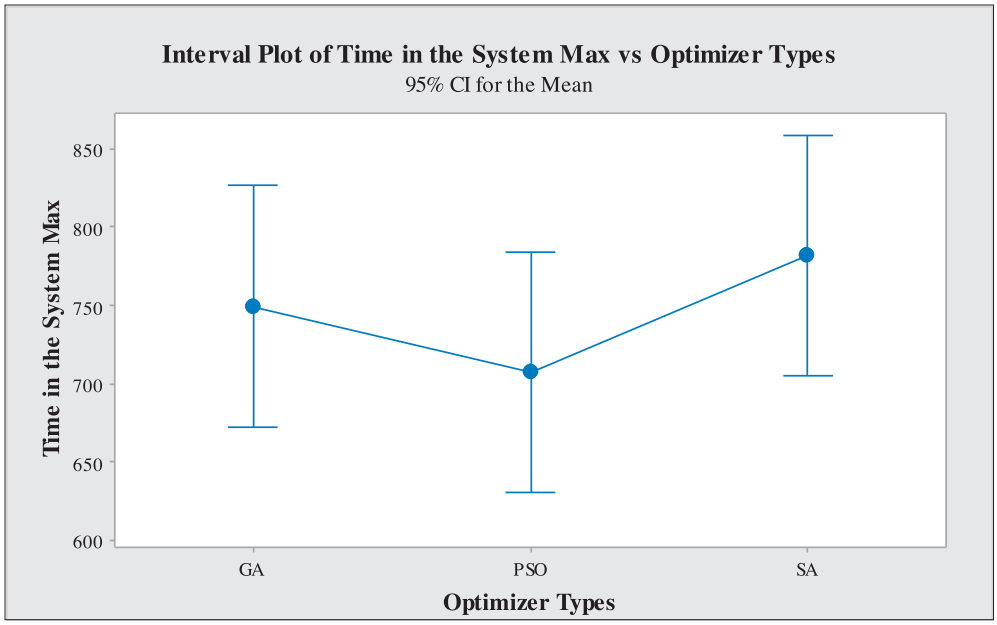

Fig. 10. Max completion time of the jobs for different optimizers.

of tardiness/earliness cost of the jobs. Therefore, SA could be a suitable candidate to be used for scheduling in a similar real-world manufacturing system.

5. Application in other systems

As previously mentioned, the developed CallMATLAB step instance is applicable to every discipline where simulation applies. In this section, the potential applications of this advancement are presented to provide interested readers with a starting point for their own application and research.

5.1. Application in healthcare systems

Scheduling the patients, nurses or physicians is a crucial task in hospitals and ERs. The developed CallMATLAB step could be used to develop an IOS framework to model daily scheduling operations in a hospital where unplanned events exist, such as having two ill physicians. The following are more examples of these types of trigger events: an announcement of a bus accident that overflows an ER, a delay in the arrival of critical equipment, a reallocation of staff in a particular shift because they have been informed that a key staff member has taken time off [50], etc. In general, these events require immediate compensating actions that may be assessed with a simulation model.

5.2. Application in supply chain management (SCM) systems

A simulation model could embrace diverse SCM models with their non-linear and complex relationships. Customers' demand and suppliers' capacities are constantly changing both in terms of variety and price range. Moreover, supply chain relationships between suppliers, manufacturers, distributor stores and retailers evolve over time. One could apply IOS models for a supply chain system, while any of those above mentioned changes in the system could be considered as a trigger point. For instance, if customers' demand changes, supplier capacity varies, new suppliers are selected, the number of available transports fluctuates or even prices change, the simulation could pause itself and optimize the system according to the new SCM condition. Applying the proposed IOS approach provides companies an opportunity to design their own supply chain system, not only by optimizing internal operations, but also by examining and improving the entire supply chain performance over the long-term.

5.3. Application in project management

Another potential application is to the field of project management. Each project contains a set of related activities with precedence constraints (start-to-start, start-to-finish, finish-to-finish, and finish-to-start), and shared pool of limited resources. In reality, time, cost and required resources for each task and even task success have a stochastic nature. Therefore, implementing an IOS approach provides an optimal resource constrained project selection and task-scheduling problem in the face of uncertainty. Subramanian et al. [17,51] is a good example of how this framework is applicable in a project management problem. In their study, an IOS model starts with the initial tasks, and at the end of each task(s) a simulation model generates stochastic duration, number of required resources etc., and optimization solves a mathematical model to find the next task to be performed. This cycle continues until a predetermined end of the planning horizon is reached.

6. Conclusion

It is difficult for simulation software packages to model a large number of human decision-making methods and computational aspects of a real-world system in an efficient way. This paper proposed a new SIMIO step instance which integrates a simulation software to a computational agent in order to perform high computational operations like optimization. Several applications presented to illustrate the potential applicability of the proposed CallMATLAB step, such as IOS modeling. This step instance is not limited to perform optimization and could be utilized to execute any type of extensive computational calculation. Note that the data transfer between MATLAB and simulation could be done either via a database or other types of data file, such as excel or text file.

We believe this advancement adds a new dimension to the simulation modeling approaches. This would enable experts to enjoy the modeling simulation while implementing their own logic and decision-making tools within the simulation run by taking advantage of powerful computational resources, such as MATLAB. The proposed CallMATLAB step instance in SIMIO, could be used to develop an IOS framework which helps programmers to apply their own algorithm and optimization techniques to evaluate long-term performance.

Acknowledgment

We are grateful to the anonymous reviewers for their insightful feedback and valuable suggestions, which have improved the overall quality of the manuscript.

Appendices

A. Guideline to create a user-defined step instance for SIMIO

After installing SIMIO, the Visual Studio template has to be installed. As is shown in Fig. 11, this setup adds a template for custom *step* and custom *element* in Visual Studio under Visual C# tab. Platform *.NET Framework 4* is required to run and develop the *User-defined Step and Element*. Note that, the coded DLL file and the instruction to add the developed CallMATLAB step instance are available at SIMIO user forum website at this address: http://www.SIMIO.com/forums/viewtopic.php?f=33&t=2185.

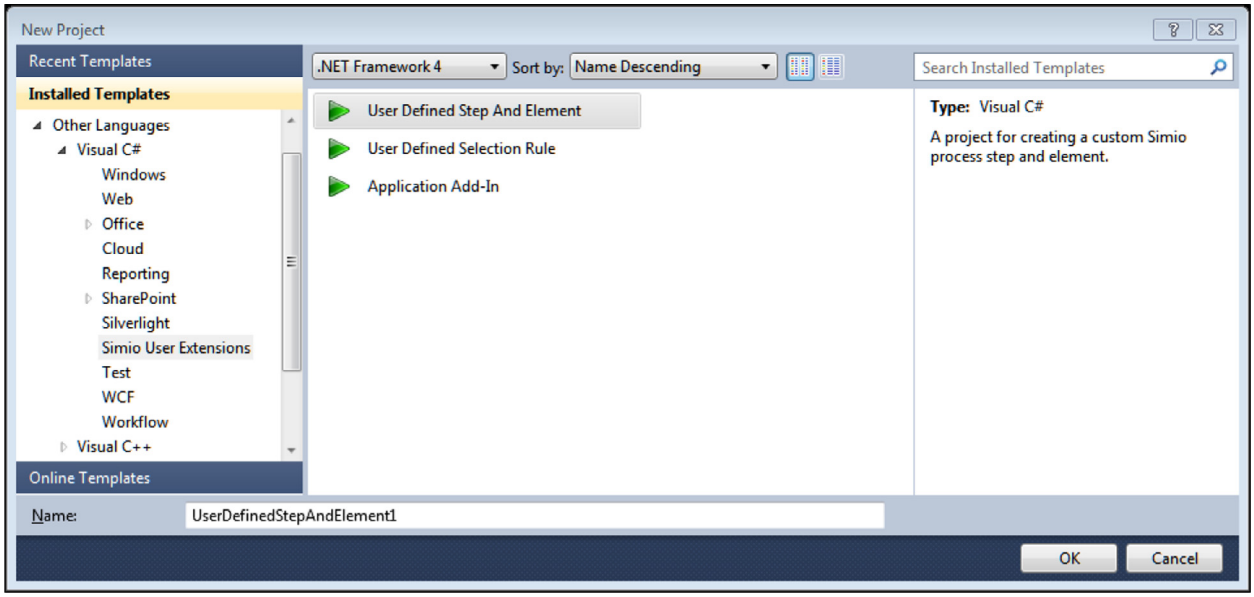

Fig. 11. Visual studio template for SIMIO.

```
/// <summary>
/// Method called that defines the property schema for the step.
/// </summary>
public void DefineSchema(IPropertyDefinitions schema)
{
    // Two IPropertyDefinition is needed to address "Function Name" and its "Folder Address"
    IPropertyDefinition pd1; //This is for MATLAB "Function Name"
    IPropertyDefinition pd2; //This is for MATLAB "Folder Address"

    //Function Name
    pd1 = schema.AddStringProperty("FunctionName", string.Empty);
    pd1.DisplayName = "Function Name";
    pd1.Description = "The MATLAB 'Function name' that you would like to call";
    pd1.Required = true;

    //Folder Address
    pd2 = schema.AddStringProperty("FolderAddress", string.Empty);
    pd2.DisplayName = "Folder Address";
    pd2.Description = "The 'Folder Address' that your MATLAB file is located";
    pd2.Required = true;

}
```

Fig. 12. Creating two properties for MATLAB 'Function Name' and 'Folder Address'.

By default, SIMIO provides a C# class template called *UserStep.cs* which enables programmers to develop their desired steps by coding their own logic. In order to provide more illustration for SIMIO programmers, a few important hints through creating this step are listed as follows.

- First of all, two different types of references are added to the project's references. The first type of references is related to SIMIO, which includes: *SIMIOAPI.dll* and *SIMIOAPI.Extension.dll*. These two references are available in SIMIO directory. The second required reference is for MATLAB which adds MATLAB COM to the project and is called *MATLAB Application* version 8.5. This reference is compatible with MATLAB R2015a version. In the case of using older or newer versions of MATLAB, this reference has to be substituted with a proper one.

- The file which is created by the template will automatically create a *Globally Unique IDentifier* (GUID) for the class. Each user-defined step of SIMIO has to have a unique GUID to digitally sign the file.
- SIMIO's API defines many interfaces, several of which are collections of other interfaces. For instance, *ISIMIOProject* has a Models property, of the type *IModels*, which is a collection of *IModel* interfaces, each one corresponding to a model in the project. Similarly, the *IProperties* interface is a collection of *IProperty* interfaces [52]. The user-defined step, *CallMATLAB*, requires two properties, *folder address*, that the MATLAB file is located, plus the MATLAB *function's name*, which is name of the file that source codes are saved in. Therefore, as is illustrated in Fig. 12, two *IPropertyDefinitions* are added to the coding, respectively *FunctionName* and *FileAddress*.
- After debugging the coded project, the DLL file needs to be added to the source directory of SIMIO (C:\Program Files (x86)\SIMIO\UserExtensions). This will add *CallMATLAB* step instance to the *User Defined* panel within the *Processes* window in the SIMIO environment. Similar to the existing step instances in SIMIO, the user could easily drag-and-drop *CallMATLAB* while building processes.

B. . UserStep.cs codes in C#

```
using System;
using System.Collections.Generic;
using System.Text;
using SIMIOAPI;
using SIMIOAPI.Extensions;
namespace CallMATLAB
{
  public class UserStepDefinition : IStepDefinition
  {
    #region IStepDefinition Members
    /// <summary>
    /// Property returning the full name for this type of step. The name should
contain no spaces.
    /// </summary>
    public string Name
    {
      get { return "CallMATLAB"; }
    }
    /// <summary>
    /// Property returning a short description of what the step does.
    /// </summary>
    public string Description
    {
      get { return "This step calls your matlab function. Simply, enter the “folder
address” and matlab “function name” that you would like to be called. Make sure
that your Matlab version is Matlab R2015a.Note that, in case of execution error, use
slow Run Speed Factor e.g. between 5 to 10. For further question or help contact
m.dehghani86@gmail.com - Mohammad Dehghani"; }
    }
    /// <summary>
    /// Property returning an icon to display for the step in the UI.
    /// </summary>
    public System.Drawing.Image Icon
    {
      get { return null; }
    }
    /// <summary>
    /// Property returning a unique static GUID for the step.
    /// </summary>
    public Guid UniqueID
    {
      get { return MY_ID; }
    }
    static readonly Guid MY_ID=new Guid("{73fb845f-4295-4e7e-b744-6668b6c94a0e}");
    /// <summary>
```

```
      /// Property returning the number of exits out of the step. Can return either 1 or
2.
      /// </summary>
      public int NumberOfExits
      {
        get { return 1; }
      }
      /// <summary>
      /// Method called that defines the property schema for the step.
      /// </summary>
      public void DefineSchema(IPropertyDefinitions schema)
      {
        // Two IPropertyDefinition is needed to address "Function Name" and its "Folder
Address"
        IPropertyDefinition pd1; //This is for MATLAB "Function Name"
        IPropertyDefinition pd2; //This is for MATLAB "Folder Address"
        //Function Name
        pd1=schema.AddStringProperty("FunctionName", string.Empty);
        pd1.DisplayName="Function Name";
        pd1.Description="The MATLAB 'Function name' that would be called";
        pd1.Required=true;
        //Folder Address
        pd2=schema.AddStringProperty("FolderAddress", string.Empty);
        pd2.DisplayName="Folder Address";
        pd2.Description="The 'Folder Address' that your MATLAB file is located, like
'C:\\MatlabFiles'";
        pd2.Required=true;
      }
      /// <summary>
      /// Method called to create a new instance of this step type to place in a
process.
      /// Returns an instance of the class implementing the IStep interface.
      /// </summary>
      public IStep CreateStep(IPropertyReaders properties)
      {
        return new UserStep(properties);
      }
      #endregion
    }
    class UserStep : IStep
    {
      IPropertyReaders _properties;
      IPropertyReader _FunctionNameProp;
      IPropertyReader _FolderAddressProp;
      public UserStep(IPropertyReaders properties)
      {
        _properties=properties;
        _FunctionNameProp=_properties.GetProperty("FunctionName");
        _FolderAddressProp=_properties.GetProperty("FolderAddress");
      }
      #region IStep Members
      /// <summary>
      /// Method called when a process token executes the step.
      /// </summary>
      public ExitType Execute(IStepExecutionContext context)
      {
        // Example of how to get the value of a step property.
        String FunctionNameStr=_FunctionNameProp.GetStringValue(context);
        String FolderAddressStr=_FolderAddressProp.GetStringValue(context);

        // Example of how to display a trace line for the step.
```

```
        context.ExecutionInformation.TraceInformation(String.Format("The MATLAB Function
'{0}' located at '{1}' is triggered.", FunctionNameStr, FolderAddressStr));
        //Using CallMtlab.cs in order to call Matlab function from its located folder
        CallMATLAB CallMATLAB=new CallMATLAB();
        FolderAddressStr="cd "+FolderAddressStr;
        CallMATLAB.CallMATLABFunction(FunctionNameStr, FolderAddressStr);
        return ExitType.FirstExit;
      }
      #endregion
    }
  }
```

C. . CallMATLABClass.cs codes in C#

```
using System;
using System.Collections.Generic;
using System.Text;
namespace CallMATLAB
{
  public class CallMATLAB
  {
    public void CallMATLABFunction(string FunctionName, String FolderAddress)
    {
      // Create the MATLAB instance
      MLApp.MLApp matlab=new MLApp.MLApp();
      matlab.Execute(FolderAddress);
      // Change to the directory where the function is located
      matlab.Execute(FunctionName);
    }
  }
}
```

References

[1] M.R. Galankashi, E. Fallahiarezoudar, A. Moazzami, N.M. Yusof, S.A. Helmi, Performance evaluation of a petrol station queuing system: a simulation-based design of experiments study, Adv. Eng. Software 92 (2016) 15–26.
[2] A.V. Barenji, R.V. Barenji, M. Hashemipour, Flexible testing platform for employment of RFID-enabled multi-agent system on flexible assembly line, Adv. Eng. Software 91 (2016) 1–11.
[3] M. Göçken, A.T. Dosdoğru, A. Boru, F. Geyik, (R,s,S) inventory control policy and supplier selection in a two-echelon supply chain: an optimization via simulation approach, in: Proceedings of the Winter Simulation Conference, California, USA, 2015.
[4] M. Dehghanimohammadabadi, T. Keyser, Does the Iranian national productivity and excellence award get leadership buy-in, in: IIE Annual Conference Proceedings, 2014, p. 2656.
[5] S.M.H. Bamakan, M. Dehghanimohammadabadi, A weighted Monte Carlo simulation approach to risk assessment of information security management system, Int. J. Enterp. Inf. Syst. (IJEIS) 11 (4) (2015) 63–78.
[6] K. Kianfar, S.F. Ghomi, A.O. Jadid, Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA, Eng. Appl. Artif. Intell. 25 (3) (2012) 494–506.
[7] F.G. Gonzalez, A simulation-based controller builder for flexible manufacturing systems, in: Proceedings of the 28th Conference on Winter Simulation, 1996, pp. 1068–1075.
[8] G. Wagner, Ontologies and rules for enterprise modeling and simulation, in: 2011 15th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011, pp. 385–394.
[9] J. Jackman, E. Johnson, The role of queueing network models in performance evaluation of manufacturing systems, J. Oper. Res. Soc. 44 (8) (1993) 797–807.
[10] J. Haddock, A simulation generator for flexible manufacturing systems design and control, IIE Trans. 20 (1) (1988) 22–31.
[11] C.D. Pegden, SIMIO: a new simulation system based on intelligent objects, in: Proceedings of the 39th Conference on Winter Simulation: 40 Years! The Best Is Yet to Come, 2007, pp. 2293–2300.
[12] D. Creighton, S. Nahavandi, Application of discrete event simulation for robust system design of a melt facility, Rob. Comput. Integr. Manuf. 19 (6) (2003) 469–477.
[13] J.S. Smith, R. Wysk, D.T. Sturrock, S.E. Ramaswamy, G.D. Smith, S.B. Joshi, others, Discrete event simulation for shop floor control, in: Simulation Conference Proceedings, 1994. Winter, 1994, pp. 962–969.
[14] M. Dehghanimohammadabadi, T.K. Keyser, Iterative simulation-optimization of a multi-state manufacturing system, in: Proceedings of the Poster Session and Student Colloquium Symposium, 2015, pp. 3–4.
[15] C.J. Paredis, T. Johnson, Using OMG's SysML to support simulation, in: Simulation Conference, 2008. WSC 2008. Winter, 2008, pp. 2350–2352.
[16] B. Shirazi, I. Mahdavi, N. Mahdavi-Amiri, iCoSim-FMS: an intelligent co-simulator for the adaptive control of complex flexible manufacturing systems, Simul. Modell. Pract. Theory 19 (7) (2011) 1668–1688.
[17] D. Subramanian, J.F. Pekny, G.V. Reklaitis, G.E. Blau, Simulation-optimization framework for stochastic optimization of R&D pipeline management, AIChE J. 49 (1) (2003) 96–112.
[18] G.J. Mejtsky, A metaheuristic algorithm for simultaneous simulation optimization and applications to traveling salesman and job shop scheduling with due dates, in: Proceedings of the 39th Conference on Winter Simulation: 40 Years! The Best Is Yet to Come, 2007, pp. 1835–1843.
[19] A.I. Sivakumar, Multiobjective dynamic scheduling using discrete event simulation, Int. J. Comput. Integra. Manuf. 14 (2) (2001) 154–167.

[20] N. Sadeghi, A.R. Fayek, N. Gerami Seresht, A fuzzy discrete event simulation framework for construction applications: improving the simulation time advancement, J. Constr. Eng. Manage. (2016) 4016071.
[21] G.A. Corona-Suárez, S.M. AbouRizk, S. Karapetrovic, Simulation-based fuzzy logic approach to assessing the effect of project quality management on construction performance, J. Qual. Reliab. Eng. 2014 (2014) 1–18.
[22] S.M. Hosseini, S. Jannat, and A. Al Khaled, *Integrating Fuzzy Expert System with Discrete Event Simulation to Determine Configuration of Resources Level for an Emergency Unit*. 2013.
[23] S.A. Ali, H. Seifoddini, Simulation intelligence and modeling for manufacturing uncertainties, in: Proceedings of the 38th Conference on Winter Simulation, 2006, pp. 1920–1928.
[24] S. Liang, X. Yao, Multi-level modeling for hybrid manufacturing systems using Arena and MATLAB, in: International Workshop on Modelling, Simulation and Optimization, 2008. WMSO'08, 2008, pp. 155–159.
[25] Z. Gao, X. Li, J. Yu, C. Liao, Integrating Simulink debugger with .NET to enhance interactivity: a case study for hydraulic pump system simulation, in: Seventh International Symposium on Instrumentation and Control Technology, 2008, p. 71271E.
[26] G. Mušič, D. Matko, Combined simulation for process control: extension of a general purpose simulation tool, Comput. Ind. 38 (2) (1999) 79–92.
[27] A. Aussem, D. Hill, Neural-network metamodelling for the prediction of Caulerpa taxifolia development in the Mediterranean sea, Neurocomputing 30 (1) (2000) 71–78.
[28] A. Ahmed, O. Ashour, Using artificial neural network as a meta-modeling technique in supply chains, in: IIE Annual Conference. Proceedings, 2015, p. 2221.
[29] A. Cherkassky, A neural network meta-model of roll-drafting process, J. Text. Inst. 103 (2) (2012) 166–178.
[30] L. Song, S.M. AbouRizk, Measuring and modeling labor productivity using historical data, J. Constr. Eng. Manage. 134 (10) (2008) 786–794.
[31] L. Rabelo, H. Eskandari, T. Shalan, M. Helal, Supporting simulation-based decision making with the use of AHP analysis, in: Proceedings of the 37th Conference on Winter Simulation, 2005, pp. 2042–2051.
[32] H. Eskandari, M. Riyahifard, S. Khosravi, C.D. Geiger, Improving the emergency department performance using simulation and MCDM methods, in: Proceedings of the 2011 Winter Simulation Conference (WSC), 2011, pp. 1211–1222.
[33] M. Andersson, H. Grimm, A. Persson, A. Ng, A web-based simulation optimization system for industrial scheduling, in: Simulation Conference, 2007 Winter, 2007, pp. 1844–1852.
[34] D.W. Huang, H. Diefes-Dux, P.K. Imbrie, B. Daku, J.G. Kallimani, Learning motivation evaluation for a computer-based instructional tutorial using ARCS model of motivational design, in: 34th Annual Frontiers in Education, 2004. FIE 2004, 2004, pp. T1E–T30.
[35] L.F. Silva, C. Olivete, P.H. Reis, R.S. Santos, R.C.M. Correia, R.E. Garcia, Supportive environment for teaching and learning digital image processing, in: Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE, 2015, pp. 1–7.
[36] R. Choy, A. Edelman, Parallel MATLAB: doing it right, in: Proceedings of the IEEE, 93, 2005, pp. 331–341.
[37] "MATLAB COM Integration - MATLAB & Simulink." [Online]. Available: http://www.mathworks.com/help/matlab/matlab_external/introducing-matlab-com-integration.html. (accessed 19.07.15).
[38] J.A. Joines, S.D. Roberts, Simulation Modeling with SIMIO: A Workbook, North Carolina State University, 2010.
[39] SIMIO Co., What is SIMIO simulation software? SIMIO, 2015. [Online]. Available: http://www.SIMIO.com/the-SIMIO-simulation-advantage/index.php. (accessed 15.07.15).
[40] G. Figueira, B. Almada-Lobo, Hybrid simulation–optimization methods: a taxonomy and discussion, Simul. Modell. Pract. Theory 46 (2014) 118–134.
[41] D. Ouelhadj, S. Petrovic, A survey of dynamic scheduling in manufacturing systems, J. Schedul. 12 (4) (2009) 417–431.
[42] M. Dehghanimohammadabadi, Iterative Optimization-based Simulation (IOS) with Predictable and Unpredictable Trigger Events in Simulated Time, Western New England University, 2016.
[43] M. Dehghanimohammadabadi, T. Keyser, Tradeoffs between objective measures and execution speed in iterative optimization-based simulation (IOS), in: Proceeding of Winter Simulation Conference, 2015.
[44] P.L. Rocha, M.G. Ravetti, G.R. Mateus, P.M. Pardalos, Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times, Comput. Oper. Res. 35 (4) (2008) 1250–1264.
[45] Y. Carson, A. Maria, Simulation optimization: methods and applications, in: Presented at the Proceedings of the 29th Conference on Winter Simulation, 1997, pp. 118–126.
[46] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, et al., Optimization by simmulated annealing, Science 220 (4598) (1983) 671–680.
[47] S. Rai, R.K. Ettam, Xerox Corporation, Webster, NY, USA, in: Simulation Conference (WSC), 2013 Winter, 2013, pp. 1097–1108.
[48] Y. Kang, H. Lu, J. He, A PSO-based genetic algorithm for scheduling of tasks in a heterogeneous distributed system, J. Software 8 (6) (2013) 1443–1450.
[49] M. Gen, L. Lin, Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey, J. Intell. Manuf. 25 (5) (2014) 849–866.
[50] C. Espinoza, J. Pascual, F. Ramis, D. Bórquez, J.A. Sepúlveda, Real-time simulation as a way to improve daily operations in an emergency room, in: Proceedings of the 2014 Winter Simulation Conference, 2014, pp. 1445–1456.
[51] D. Subramanian, J.F. Pekny, G.V. Reklaitis, A simulation—optimization framework for addressing combinatorial and stochastic aspects of an R&D pipeline management problem, Comput. Chem. Eng. 24 (2) (2000) 1005–1011.
[52] SIMIO Co., "SIMIO software reference guide version 7.119.12234.0," 2015.