GESTÃO & PRODUÇÃO

ISSN 0104-530X (Print)
ISSN 1806-9649 (Online)

ORIGINAL ARTICLE

Use of workload control in production planning and control: modeling on a simulation software

Disseminando o uso do controle de carga no planejamento e controle da produção: modelagem utilizando software de simulação

Bruno da Silva Barreto[1], Juliana Keiko Sagawa[1], Mateus Scalet Silva[1]

[1]Universidade Federal de São Carlos – UFSCar, Departamento de Engenharia de Produção, São Carlos, SP, Brasil. E-mail: juliana@dep.ufscar.br

How to cite: Barreto, B. S., Sagawa, J. K., & Silva, M. S. (2021). Use of workload control in production planning and control: modeling on a simulation software. *Gestão & Produção*, 28(2), e5744. https://doi.org/10.1590/1806-9649-2021v28e5744

Abstract: This paper aims at presenting a structure of implementation of the Workload Control (WLC) approach on a commercial discrete event simulation software by comprehensively demonstrating how to construct and simulate this approach. This research was developed considering the lack of studies fully describing this implementation on this kind of software, which hinders the dissemination and use of the WLC approach by managers. Initially, the logic was developed on a dedicated and structured language and then converted to the simulation software. A detailed description of the WLC implementation contributes to the business domain by facilitating its replication and application in other manufacturing environments, reducing the project time needed to develop the base model and allowing managers and/or researchers to focus directly on the adjustments of the model to the environment being modeled. In the academic domain, this paper addresses a gap in the WLC literature concerning the lack of tutorials for simulation and the lack of information, in the existing papers, regarding the development of the computational model.

Keywords: Workload control; Simulation; Software; Tutorial; Job shop; Make-to-order.

Resumo: Este artigo tem por objetivo apresentar um roteiro (tipo tutorial) de desenvolvimento de um modelo de simulação do controle de carga (WLC) em um software comercial de simulação de eventos discretos, demonstrando detalhadamente como construir e simular essa abordagem. Embora haja diversas pesquisas sobre essa abordagem, esta pesquisa se justifica uma vez que não há estudos ou tutoriais que descrevam detalhadamente o desenvolvimento do modelo computacional nesses softwares de simulação, sendo isto uma potencial barreira para que tal método seja conhecido e aplicado pelos gestores. O software adotado neste trabalho é um ambiente gráfico integrado de simulação que possui bibliotecas de elementos e programação não centralizada. A descrição detalhada do desenvolvimento da simulação do controle de carga contribui com o ambiente empresarial, pois facilita a replicação e aplicação dessa estrutura nos ambientes fabris, possibilitando reduzir o tempo do projeto para desenvolver o modelo base e permitindo aos gestores e/ou pesquisadores focarem diretamente nas adequações desse modelo ao ambiente específico que está sendo modelado. Em termos acadêmicos, o trabalho

Received Sept. 18, 2019 - Accepted Jan. 7, 2020
Financial support: This study was partially financed by the Coordination for the Improvement of Higher Education Personnel – Brazil (CAPES), by means of a scholarship.

busca suprir uma lacuna na área de WLC, relativa à escassez de trabalhos do tipo tutorial e relativa à falta de informações, nos trabalhos existentes de simulação, sobre como a modelagem computacional é realizada.

Palavras-chave: Controle de carga; Simulação; Software; Tutorial de implementação; Job shop; Make-to-order.

1 Introduction

The functioning of any company encompasses the structuring of a production system involving operations and management of products, services or the combination of both. Making a production system effective, that is, reaching the goals proposed by the company, requires a major managing effort and a constant pursuit of improvement.

Each production system has specific characteristics, and the classification of the system, especially in terms of flow type (e.g. flow shop, job shop, projects) and type of response to demand (e.g. MTS, MTO, ATO), is relevant to choose the proper production planning and control (PPC) approach.

This study emphasizes the Workload Control approach – WLC. According to Hendry et al. (1998), the WLC is a PPC system originally designed to control the queues in a production system with intermittent job shop configuration, in which the production orders may or may not have the same routing and the production strategy may be make-to-order (MTO). Thürer & Godinho (2012) also highlight that the WLC is known as a method with potential to improve the throughput time and the work in process (WIP) level, as well as the delivery punctuality.

Several studies, such as Hendry & Kingsman (1989), Hendry et al. (1998), Thürer et al. (2015), among others, investigate and analyze the WLC performance for MTO systems with job shop configuration, pointing out advantages, limitations, and conditions of use. Many of these studies apply the simulation method to assess the results that can be generated from the approach. However, it is difficult to replicate the simulation models in the existing studies, since for most of them the computer implementation is not presented, and the software or language used are not reported. Thürer et al. (2011) raised this information for the simulation models developed between 2000 and 2009: two of them were developed on SIMAN, two on Arena, and some models (of the same group of authors) on EM-Plant. However, in several other studies, like Oosterman et al. (2000), Kingsman & Hendry (2002), Missbauer (2002), Land (2006), among others, the software or language used is not informed. Generally, the authors do not provide information on the logic that support the simulation models developed, which hampers a reliable comparison of the results (Thürer et al., 2011). It is worth emphasizing that, with the emergence of open-source software programs such as *Simpy*, some studies on the WLC were simulated on these software programs, like in Thürer et al. (2015). However, these software programs do not have such user-friendly interface, easy to simulate and interpret the real-time results, like the commercial software programs do.

In this context of studies on the WLC, the general goal of this paper is to introduce a road map (tutorial) that comprehensively demonstrates how to model and simulate the WLC approach on a commercial simulation software for discrete events with decentralized programming. The simulation of an industrial environment using the WLC approach allows managers, consultants and students to evaluate and understand the results that it brings to the shop floor, stimulating its use. Additionally, simulation enables to calibrate the parameters of this approach for relevant performance indicators (such as throughput time and lateness/tardiness) to present close to optimum or desirable values. The definition of parameters for the WLC is encompassed by many studies in the literature (e.g. Land, 2006). In general, poorly defined parameters (like extremely tight values of workload) may generate worse results than in the cases in which the approach is not applied. Therefore, simulation is an important stage preceding the practical implementation of the approach. Additionally, the module of order release and the module of collection of indicators, which will be introduced in Section 4.2, can indeed benefit the practical implementation of the WLC, since these are the routines that should be programmed to use the approach in practice. The remaining modules constitute the simulation environment and must be replaced with actual data related to the arrival and execution of the orders in the shop floor. It is not the scope of this study to directly discuss aspects related to the approach's application per se. Such discussion can be found in Stevenson et al. (2011), for example.

It is relevant to mention the existence of papers or books in the literature focusing on tutorials of simulation or simulators related to other PCP systems, that is, regarding other systems of order release. For example, a simulator of MRP and DRP was developed by Suwamuji (2003) using electronic spreadsheets, VBA and Arena in an integrated manner. The simulator also allows to compare the performance of MRP/DRP with the reorder point system and with the Kanban system. In the case of pull systems, there are Kanban simulators in the literature (Seppanen, 1993; Williams et al., 2012), in some cases also associated with other techniques, like genetic algorithms, to optimize the number of cards (Köchel & Nieländer, 2002). The technique of value stream mapping (VSM) itself may be regarded as a type of tutorial for the application of pull production in the shop floor. A simulator of the DBR system (drum-buffer-rope) that can be manipulated by the user, as in a game, is fully reported by Goldratt (1996) not only regarding the simulation in several distinct scenarios, but gradually demonstrating the application of principles of the Theory of Constraints/DBR and the respective expected results.

Therefore, in addition to overcoming the gap in the literature, presenting a tutorial-type material, this research has practical implications, aiming to disseminate the workload control approach to practitioners and to facilitate the development of WLC simulation models.

For the computational modeling, a wide-ranging software of commercial simulation (FlexSim) (present in many countries) was used, which allows greater access to the software, contributing for the solution to be replicated.

2 Theoretical principles

2.1 Workload control – WLC

Workload control is a concept in the scope of Production Planing and Control (PPC) projected for complex environments of production, such as job shop and make-to-order – MTO systems. This method is based on the concept of input and output controls of Plossl (1985), in which input control refers to the decision of the quantity of orders to be released to the system and the output control relates to controlling the quantity of orders that leave the system.

The principles of the WLC concept include controlling the length of the queues of orders (workload) formed in each work station by deciding the release of orders and enhancing the delivery speed, maintaining and possibly improving the delivery reliability, reducing and stabilizing the throughput time of shop floor (Land & Gaalman, 1996; Thürer et al., 2012; Fernandes et al., 2016).

The order release is conditioned to a workload norm (maximum limit of quantity of work allowed per workstation in the shop floor) that will be used as parameter to determine whether an order can or cannot be released to the shop floor. The use of a proper norm can generate considerable reduction in the WIP without damaging other performance aspects (Land, 2006). In case the order has a processing time whose value exceeds the level of workloads established by the norm, such order is not released. In this case, this order is sent to the pool of non-released orders and waits for the moment of the next release at which the processing time does not exceed the level of workloads fixed by the norm (Fernandes et al., 2016; Oosterman et al., 2000).

The methods of order release can be classified as periodic (ex: beginning of each shift, day or week) or continuous (ex: at any given moment during the system operation) (Fernandes & Carmo-Silva, 2011).

The method of continuous release is based on the continuous monitoring of the workload in the shop floor to verify when and how much the workload is below the norm. When it occurs, the release of orders is assessed for all the orders contained in the pool (Fernandes & Carmo-Silva, 2011).

According to Land & Gaalman (1998), the methods of periodic release result in the decision of which order could be released from the pool to the shop floor at certain periods of time, considering the workload situation in the shop floor combined with the order processing time for not exceeding the norm value.

Land (2006) presents the algorithm of periodic release applied by Bertrand & Wortmann (1981), Tatsiopoulos (1983), Kingsman et al. (1989), Hendry (1989), and Hendry & Kingsman (1991), named Method B, which estimates the entry considering the aggregate workload L_{st}^{A}. This aggregate workload is constituted of the indirect workload (L_{st}^{U}) upstream the station *s*, that is, the sum of the processing time of all orders in the queues of other stations, but that will pass through the workstation *s* at a given moment, as well as the direct workload L_{st}^{D}, that is, the processing time of the order being processed in the station *s* or in the queue of the station. According to the aggregate workload approach, the direct and indirect workloads are added up, that is, $L_{st}^{A} = L_{st}^{U}$ + L_{st}^{D}, in which *s* and *t* indicate the station and time, respectively. For each workstation, the aggregate workload is subjected to the norm. The steps of the algorithm of periodic release considering the aggregate workload are presented below:

1. Eachorder *j* waiting to be released to the shop floor is sequenced based on the planned release date (t_{i}^{*R}), which is calculated based on the delivery date δ_j and the planned lead time (T_{s}^{*D}) for all stations in the routing of *j* (set S_j), that is: $t_{j}^{*R} = \delta_j - \sum_{s \in Sj} T_{s}^{*D}$.
2. Orders *j* with $t_{j}^{*R} \leq t + \theta$ are included in the set *J*, with θ as the threshold time frame.
3. Order $j \in J$ with the closest release date t_{j}^{*R} is considered first.
4. If the processing time p_{is} of the order under consideration added to the existing workload meet the workload norm (Λ_{s}^{A}), that is, if $L_{st}^{A} + p_{is} \leq \Lambda_{s}^{A} \forall s \in S_i$, the order is selected for the release and this contribution for the workload is included, that is, $L_{st}^{A} = L_{st}^{A} + p_{is} \; \forall s \in S_i$. Otherwise, the order must wait in stock until the next release period.

5. If set *J* has some order that is yet to be considered, it must return to step 3 considering the order with the closest release date, if not, the release procedure is completed, and the orders selected are released.

In addition to the aggregate workload approach, there is a probabilistic approach in which the indirect workload of the orders upstream a given workstation *s* is estimated using a factor of depreciation based on historical data. When an order is released, its processing time partially contributes to this workload estimate; the contribution enhances as the order advances downstream (until becoming a direct workload in the station *s*). The total, including the direct and indirect workloads estimated in this manner, is named converted workload (Bechte, 1994; Thürer et al., 2011).

Depending on the type of approach used (aggregate workload or probabilistic approach), different norm values must be applied. Additionally, the norms can establish upper, lower, or upper and lower limits, and may be rigid or flexible (Thürer et al., 2011). Other authors apply a workload balancing approach instead of a bounding one (Portioli-Staudacher & Tantardini, 2012).

Thürer et al. (2011) developed a systematic review of the literature on WLC published between 1980 and 2009. The papers are categorized as follows: conceptual research, analytical research, empirical research, and simulation-based research. By observing the evolution of the field, the authors concluded that the best-developed approaches (LUMS and LOMC) reached a conceptual maturity. Still according to the authors, the area of analytical modeling has been increasing since 2000 and efforts should be dedicated to develop simple heuristics and models to support managers in fast decision-making processes. In empirical terms, the recent scientific production has raised problems of implementation and matters related to their solutions. It is also necessary to conduct further action-research studies, develop an implementation road map, and investigate whether the positive results could be supported on a long-term basis.

It is important to highlight that simulation continues to represent the dominant method applied in the scope of Workload Control research. Thürer et al. (2011) identified four simulation-based research subgroups, involving: (1) assessment of the different combinations of rules used at each WLC stage (entry, release, and dispatching of orders) to determine the best combination; (2) development of new methods of release and performance comparison; (3) study of the influence of external parameters on performance; (4) analysis of the influence of internal parameters of WLC on performance. The initial focus of the research (80s and 90s) was directed to themes 1 and 2, however, since 2000, the development of subgroups 3 and 4 became dominant. There is certainly a gap related to the development of more realistic simulation models since most of them are hypothetical and do not reflect reality (Perona & Miragliotta, 2000; Thürer et al., 2011). The feedback of the simulation-based research with results from empirical studies can improve the applicability of the WLC theory.

As presented in the introduction, this paper aims to facilitate the development or replication of WLC simulation models to improve the current knowledge on this approach, adjust its parameters, and ultimately stimulate its application. Stevenson et al. (2009) is one of the most relevant studies in the literature and shares the same goal. The authors introduce an interactive tool of final user's training in the WLC approach application. The tool is based on the LUMS approach of the University of Lancaster and encompasses the stages of customer enquiry and job entry, in addition to the order release stage. The simulation is used to generate orders at the early stages mentioned and to reflect the variability of the throughput times in an actual manufacturing environment.

This tool provides training and support to the decision-making processes regarding the following aspects: definition of parameters; definition of delivery dates; acceptance/rejection of orders; release of orders and capacity management. This study differs from the mentioned work in terms of focus, since its focus is more restricted to the order release and shop floor behavior, in addition to prioritizing the computational modeling and implementation. Another relevant factor is that the training tool proposed by Stevenson et al. (2009) was developed on C++ based on an existing decision support system (DSS) designed by Stevenson (2006). Therefore, for users to access the tool, it is also required to access the mentioned system. The referenced authors do not provide computing details to allow the replication of the tool or the DSS. The proposal of this present work is that the computer model here presented can be replicated by any user/developer that can access the FlexSim software.

Other studies present some of the computer implementation of the WLC simulation models, but on a macro, non-detailed level. Moreira & Alves (2012) describe a scheme of decision-making in the scope of WLC on Arena, but do not specify its implementation. Kirchhof et al. (2008) report a scheme demonstrating the modules in their simulation model, also developed on Arena. In terms of structure, this scheme is relatively similar to what is described here, however, the authors do not comprehensively describe the development of the computational model.

In addition to the presented overview, it is understood that the literature on Workload Control continues to expand in diverse directions. Similarly to the implementation of pull production, Land (2009) introduces a system of cards for WLC application. Portioli-Staudacher & Tantardini (2012) propose a model of programming Mixed Integer Linear Programming (MILP) to optimize the workload balancing – a similar principle to workload control, in which violations of the norm (within the limits) are allowed for optimizing the balancing. Other authors modeled the Workload Control using multi-agent systems and closed-loop systems (Weng et al., 2008; Sagawa & Land, 2018, respectively). In the case of Weng et al. (2008), there is an agent for each level/module the WLC, that is, an agent for order entry (OEA), job release (JRA), routing and sequencing (RSA), and information feedback (IFA). In Sagawa & Land (2018), the principles of Workload Control are applied to a continuous model based on an analogy with a flow system, in which the release decisions are conducted automatically by means of a feedback loop (principles of the Classical Control Theory).

It is worth mentioning that the workload control approach had been originally proposed for application in job shop and MTO systems and most studies focus on this type of environment. However, the literature has introduced extensions to this method for other shop floor configurations, such as flow shop, for example, as in Portioli-Staudacher & Tantardini (2012), Thürer et al. (2017), and Costa et al. (2019).

3 Research method

This research is characterized as axiomatic descriptive since the presented implementation structure/tutorial is directed to an idealized problem and aims to describe the problem modeled, allowing a better understanding of the characteristics and functional relationships of the environment studied (Morabito & Pureza, 2012).

The modeling and simulation method was applied as research method. A pure job shop whose main variables were defined as probability distributions was modeled. For purposes of comparison and verification of the consistency of the proposed model, a system with the same characteristics and parameters proposed by Land (2006) was adopted, as shown in section 4.1. Initially, the algorithm introduced in section 2.1 was implemented in Matlab language, which follows a paradigm of structured programming. Such a preliminary computer model that did not contain a graphic representation of the order execution was developed to guarantee the correct implementation of the release logic before implementing the code of the FlexSim elements in a distributed manner.

The model proposed on FlexSim was divided in modules (see sections 4.2 and 4.3) to facilitate the implementation. The modules are relatively independent and the debugging of code and connections between the elements was carried out per module. Thus, successive versions of this model with increasing degree of complexity were generated. For each version, it was verified if the modules (or part of them) were functioning correctly. For example, by leaving the release module inactive (that is, making the release unconstrained), it is possible to verify the functioning of the order execution module (the shop floor simulator per se). It was also verified whether the simulation was in accordance with the WLC behavior (the form of periodic release according to the maximum norm allowed and the processing time in the stations). The software (FlexSim) enables to simulate and verify some indicators momentarily (as the utilization of machines, among others), and this feature was also applied as mechanism of verification/validation. Finally, the model was validated by comparing its results with some of the results obtained by Land (2006). After the validation, four other scenarios were simulated with varied norm levels to assess their respective effects on the performance measures (total throughput time and shop floor throughput time). Finally, the model and results of the simulation were documented.

4 Road map of modeling and simulation

4.1 Characteristics of the simulated model

Chart 1 presents the characteristics and data used in the WLC implementation based on the model described by Land (2006).

Chart 1. Characteristics of the modeled system used as reference.

Shop	Six stations, each with unique capacity
Routing length	Uniform discrete distribution of [1.6]
Sequence of stations in the routing	Randomly chosen, no re-entrant loops
Inter-arrival times	Exponential distribution (average: 0.648 time unit)
Due dates	Uniform distribution with values between 35 and 60 units of time
Operation processing times	2-Erlang distribution (average: 1 time unit)
Priority dispatching rule	First come, first served - FCFS

Source: adapted from Land (2006).

The modeled system considers six workstations and the orders that arrive to these stations have production routings that simulate a pure job shop. The size of the routing (quantity of operations per order) is random and can vary between one operation (being processed on only one workstation) and six operations (a routing that has all the stations of the system), which guarantees a higher variety of flow. The production routings do not consider the reentry into a given workstation.

An exponential distribution determines the interval time between arrivals with an average of 0.648 seconds, while a uniform distribution determines the delivery date varying from 35 to 60 seconds. In turn, operation processing time is determined by a 2-Erlang distribution with an average equal to 1. After the orders are released to shop floor, the dispatching to the machines follows the first come, first served rule.

4.2 Macro view of the simulated model – conceptual model

Figure 1 illustrates the flowchart of the simulation execution. The model is divided in five modules that execute the following macro functions: order creation, periodic opening of pool, order release, order execution, and collection of indicators.

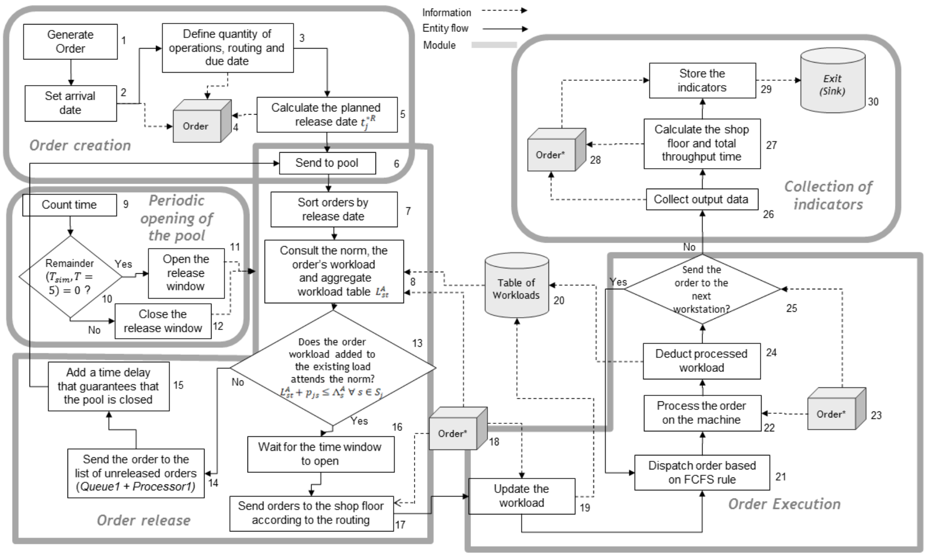

Information
Entity flow
Module
Generate Order 1
Set arrival date 2
Define quantity of operations, routing and due date 3
Order 4
Calculate the planned release date t_j^{*R} 5
Order creation
Send to pool 6
Sort orders by release date 7
Consult the norm, the order's workload and aggregate workload table L_{st}^A 8
Count time 9
Periodic opening of the pool
Remainder $(T_{sim}, T = 5) = 0$? 10
Yes
No
Open the release window 11
Close the release window 12
Does the order workload added to the existing load attends the norm? $L_{st}^A + p_{js} \leq \Lambda_s^A \; \forall s \in S_j$ 13
Add a time delay that guarantees that the pool is closed 15
Send the order to the list of unreleased orders (Queue1 + Processor1) 14
Wait for the time window to open 16
Send orders to the shop floor according to the routing 17
Order release
Order* 18
Update the workload 19
Table of Workloads 20
Dispatch order based on FCFS rule 21
Process the order on the machine 22
Order* 23
Deduct processed workload 24
Send the order to the next workstation? 25
Collect output data 26
Calculate the shop floor and total throughput time 27
Order* 28
Store the indicators 29
Exit (Sink) 30
Collection of indicators
Order Execution

Figure 1. Detailed flowchart of the model. Source: elaborated by the authors.

In fact, these macro functions compose a conceptual model of the represented system. The creation of orders encompasses the generation of an arrival date for the order, the definition of the quantity and sequence of operations in the routing (i.e. the sequence of machines in which the order must be processed), and the determination of a processing time to each operation. A due date for the order is also established. Such characteristics are generated by applying the probability distributions presented in Chart 1. As an additional operation to the module of order creation, the first stage of the basic algorithm indicated in section 2 is executed and the date of planned release (t_i^{*R}) for each order is calculated. Based on this parameter, the orders (jobs) are ordered (sequenced) in the job pool.

The second macro function refers to the periodic opening of the pool and encompasses only computer operations, using timers and counters. This function does not carry any conceptual meaning.

The function or module of order release executes the steps required to implement the concepts of workload control. After the ordering of the jobs according to their planned release date, the first job in the list is selected and the current workload of each machine in which the job is processed is assessed. In the proposed implementation, these workloads are stored in a "table of workloads" that is continuously updated along the simulation. The additional workload that would be generated by the job/order, if released, is added to the existing workload in each machine and compared with the respective workload norm. Depending on the result, the job follows to the list of release or non-release. In the implemented model, these lists are continuously updated. When the pool opens, the jobs in the list of release are sent to the shop floor. This macro function of the conceptual model executes steps 2, 3, 4, and 5 of the algorithm presented in section 2. The WLC was applied using the method of periodic release and considering the aggregate workload, that is, using method B.

The function of order execution corresponds to a shop floor simulator. Once released, the jobs are sent to each machine according to the routing or put to wait in the queue, according to the immediate availability of each machine. The queues belonging to the dispatching level of the WLC approach are organized according to the FCFS criterion. In the model implemented, each job has labels in which the fixed data of the production routing are stored. The transition from a machine to another is carried out based on the reading of these labels.

The last macro function of the model executes the calculation of the indicators (shop floor throughput time, total throughput time, and evolution of workload in the machines). The arrival, release and completion dates are stored in labels. The performance indicators and graphs are generated based on these labels, as will be further detailed over the next subsections.

The following paragraphs include the description of the modules/functions using the objects and terminology of the software applied to the modeling.

4.3 Macro view of the simulated model – computer model

Figure 1 illustrates the explanation of the computational model. The processes contained in the modules are represented by blocks, which were numbered and follow the flow indicated by a continuous arrow. Three other objects are represented in the model: order, table of workloads and exit – important elements in the information flow, which is indicated by a dotted arrow.

The first module, order creation, defines the initial attributes of the orders. Block 1 creates the orders (item *Box*) on the entity *Source* (entry). Block 2 defines the order arrival dates, which are used to calculate the total throughput time in the end of the model. Block 3 defines the quantity of operations in the production routing (varying from 1 to 6), the sequence of operations/machines, and order due date. Block 5 defines the processing time for each operation in the routing, as well as the planned release date of the order. Subsequently, the order is sent to the job pool.

The second module – order release – works alongside the module of periodic opening of the pool and starts as the orders arrive to the pool (block 6) and are organized in increasing order according to the planned release date, calculated through the difference between the due date and the sum of the processing times (block 7). In block 8, three kinds of data are gathered/read: 1) the workload generated by a given order to the system (i.e. the processing time of the order in each workstation); 2) workload norm defined for each workstation; 3) the existing aggregate workloads already allocated to each station. Data 2) and 3) is read from the workload table. These data allow to verify, for each workstation, whether the new projected load does not exceed the norm (block 13). For this purpose, the order processing times are added to the aggregate workloads of each workstation in the production routing, generating a value that is compared with the workload norm. In case the value found exceeds the norm, the order follows to block 14, which works as a list of non-released orders. The non-released order goes through a set composed by a *Queue* and a *Processor* (block 15) and returns to the pool. The function of this *Processor* is to generate a delay before the orders return to the pool, so that they come back to it only after the execution of the next release. If the norm is not exceeded, the orders remain in the pool and wait for the opening of the time window to be released to the shop floor.

The module of periodic opening of the pool contains the procedures that control the opening and closing of the time window of the pool. This module starts at the moment in which the program is executed. In the first procedure, block 9, time is accounted using the function *time* () in FlexSim, while at the next step, shown in block 10, it is verified whether the division of the *time* () function by 5 equals zero. This calculation is performed continuously. As the value of *time()* reaches the multiples of 5 (value defined for the interval between releases), the pool exit is opened, and the orders that did not exceed the norm are released to the shop floor. Similarly, whenever the value of *time*() is not a multiple of five, the pool exit is kept closed (blocks 11 and 12).

In case the value generated from the sum of the order processing time and the accumulated workload of the workstations in its routing does not exceed the norm, the order follows to next procedure, shown in block 16, in which it waits for the opening of the time window, according to previous explanation. After the window is opened, in the procedure of block 17, the orders are sent to the module of Order Execution, which simulates the shop floor by processing the orders in the workstations. Therefore, the program reads the labels of each order containing the number of operations in the routing and the sequence of these operations.

The module of order execution starts in block 19 with the reading of the processing times (from the labels of the entity *Box*) and their addition to the table of workloads, which is updated (procedure 20). In this model, the workload of a station is given by the sum of the processing time registered on the order labels. At the first moment in which the order is released to the shop floor and follows to the queue of the first machine in the routing, all workloads, either direct or indirect, are registered in the table of workloads (i.e. the processing time in the remaining machines downstream the order processing is registered). Each column in the table of workloads receives the value accumulated from the workload of a given machine.

Once in the processing queue (block 21), the orders are released according to the rule of First Come, First Served (FCFS).

In block 22, the reading of the order processing time for the machine contained on the item label is performed. The order is processed based on this processing time. Subsequently, the table of workloads is read according to the indication in block (24) and the value of the processed workload (processing time spent in that operation) is deducted from the table. The last step of the order execution module is related to block 25: the order is sent to the next machine in the routing or to the exit in case that the last operation had been executed. For such purpose, this procedure reads the next operation of the item on the label of the routing contained in the order. In case there are still operations to be executed, the order follows through the exit "yes" and returns to block 21; otherwise, it follows to the last module – collection of indicators.

This last module starts with block 26 – collection of output data – in which all indicators of time related to that order are collected, especially the time they enter and exit the machines and the overall system. In block 27 the throughput time is calculated based on these dates, while in block 28 the calculated indicators are saved in the labels of the *Sink* (i.e. the entity at the exit of the system) and the order is eliminated.

Chart 2 presents the basic procedures illustrated in Figure 1. It describes the variables modified by each event, their relation to the performance measures, main objects and respective activities or waiting periods, and the step of the WLC algorithm that is implemented. The description is distributed in five columns: modules and submodules, objects (from Flexsim), attributes and tabs of the objects, routines and commands, and line of the WLC general algorithm that is implemented. Chart 2 also indicates in brackets the nomenclature used on the Flexsim software for each component.

Chart 2. Conversion of the general workload control algorithm into a model implemented on commercial software (FlexSim).

Modules/ submodules	**Objects**	**Attributes/ tabs of the objects**	**Implemented routines/commands**	**Line of the algorithm of periodic release of the WLC classified as Method B (Land, 2006)**
Order creation	Generator of entities (*Source* "Chegada")	Tab *Source*	Definition of the arrival date	A_j
		Tab *Triggers*, *On Creation, Set label* + customized code	Creation of labels, definition of the number of operations, production routing, processing time, and due dates	S_j
				p_{js}
				δ_j
		Tab *Triggers, On Creation,* customized code	Calculation of the planned lead time, calculation of the planned release date	$T_s^{*D} = \sum_{s \in S_j} p_{js}$
				$t_j^{*R} = A_j + T_s^{*D}$
Periodic opening of the pool	*Source* TIC	Tab *Trigger*, Event: *OnExit*	Counter of time of 5 units (seconds)	T
			Verification the pool exit (open or closed)	
	Item *Cylinder*		Transitory item from *Source* TIC to *Sink* TAC	
Order release	*Job pool*/queue of orders (*Queue*)	Tab *Triggers – On entry – Sort by Expression*	Ascending ordination (*Ascending*, available on the software)	1. Ordination of the tasks based on the planned release date: $t_j^{*R} = \delta_j - \sum_{s \in Sj} T_s^{*D}$
		Tab *Flow*, *Send to Port*, customized code	Verification of the workloads in the Table of Workloads (table "Cargas").	Reading of L_{st}^A; Assess the projected workload in relation to the norm: $L_{st}^A + p_{js} \le \Lambda_s^A \forall s \in S_j$
			Assessing whether the processing time of the order to be	

Chart 2. Continued...

Modules/ submodules	Objects	Attributes/ tabs of the objects	Implemented routines/commands	Line of the algorithm of periodic release of the WLC classified as Method B (Land, 2006)
			released, added to the machine workload, exceeds the norm. In case it does not exceed the norm, it adds to the existing workload and directs to the shop floor. In case it exceeds the norm, it directs to the queue reentry.	
	Reentry of non-released orders, *Processor1*	Tab *Processor*, *Process time*	Generating a waiting period to guarantee that the release/pool exit is closed, allowing that the non-released orders return later to the *pool*.	
Order execution	Queue before each machine "Fila Est i" (Queue)	Tab *Trigger, On Entry*	Reading of processing time and updating the workload of the shop floor in function of the released orders	$L_{st}^{A} = L_{st}^{A} + p_{js} \ \forall s \in S_j$
	Machines (Processor)	Tab *Process*, on *Process time*	Execution of the corresponding operation	Operation with processing time p_{js}
		Tab *Trigger*, Event *On Entry*	Counter of processed objects per machine	
		Tab *Trigger*, Event On Exit	Update of the existing workload (in each machine), subtracting the processing time of the order already processed.	$L_{st}^{A} = L_{st}^{A} - p_{js}^{+} \ \forall s \in S_j$
	Sink Exit	Tab *Triggers*, Event *On Entry*	Registration of the moment at which the order exists the system	$T'_j = D_j - A_j$
			Calculation of the total throughput time and shop floor throughput time.	
Collection of indicators	*Sink* Exit	Tab *Triggers*, Event *On Entry*	Readings of total throughput time and of the processing/waiting time on each entity, on the *Source* and on the pool.	$\hat{w}(n) = \sum W_j$

The first column of Chart 2 describes the five modules/ submodules. The module of order creation has the object *Source* ("Chegada"), which creates the items (*Box*) to be processed in the model. For this purpose, the attributes either in the item and the object *Source* are configured.

The configuration of these attributes is performed on the screen of object properties. The tab *Source,* for example, includes the definition of arrival dates, which indicates the moment in which the object creates the items. The type of arrival/creation of an item can be generated based on a given interval of time, a fixed schedule or even a type of sequencing. In the last column, the variable or equation of the algorithm of order release referred in the literature and used to support the modeling on FlexSim is presented. Still regarding the first module ("Order creation"), in the tab *On Creation*, commands are implemented for the creation of labels and customized codes.

The second module, "Periodic Opening of the Pool", is responsible for controlling the pool opening to execute the release of the items present in the object *Queue (pool)*. For this purpose, this module is constituted of two objects (*Source TIC* and *Sink TAC*) connected to each other, in which the TIC is also connected to the pool. The *Source* TIC has the function of creating an item to be processed (called *Cylinder*), which is immediately destroyed on the TAC without generating any type of alteration in the material flow of the simulation (more detail in the subsection 4.4.2). This control is implemented on the tab *trigger* considering the time of periodic pool opening corresponding to five units of time, programmed using a customized code.

The third module – Order release – is configured in the object *Queue* (*pool*) and the set of objects *Queue 1* and *Processor 1* (representing the list of non-released orders). This module defines which items (orders) should be released to the shop floor and which should not be released, which must be sent to the set of objects *Queue 1* and *Processor 1*. After a period of time, the latter return to the *pool* for a further release verification. The release mechanism of this module considers the planned release date to order the jobs/orders. Subsequently, it is verified whether the sum of the workload in the machines and the order processing time per machine in the routing attends the norm. In case it does not attend, the order is sent to the list of non-released orders, which will return to the pool. In case the evaluated sum does attend the norm, the order waits for the periodic opening of the pool to be released to the shop floor. The last column in Chart 2 presents the equation that determines whether an order is or not released to the shop floor.

In the fourth module, "Order execution", the items released to the pool are processed. The objects used in this module correspond to six *Queues*, representing the queues in front of the machines, six *Processors*, representing the machines, and one object *Sink*, representing the exit. The orders are positioned in the queues according to the moment of arrival of each one (FCFS). When released to be processed, the orders are directed to the machines and processed based on the processing time registered on the labels of each order, which are read by the machines. In this module, the object *Sink* is responsible for the calculation of the total throughput time and shop floor throughput time of the orders.

The fifth and last module, "Order completion and data collection", includes the gathering of information calculated and registered in the orders and the register of total throughput time and shop floor throughput time of each order, in addition to the completion of the process by destroying the orders. The object composing this module is called *Sink* (exit). The implementations are performed on the window of properties, tab *triggers/On Entry*, in which the values of throughput time of shop floor and total throughput time are recorded in the table of orders.

4.4 Building the structure to implement the FlexSim tool

Figure 1 and Chart 2 illustrate the computational modeling on FlexSim based on the modules and stages of the flowchart, while Figure 2 illustrates the objects used on the software. It is possible to verify the presence of lines linking the objects indicating the relationship between entry and exit of information. The items enter the objects, are processed and transferred to one of the connected objects.

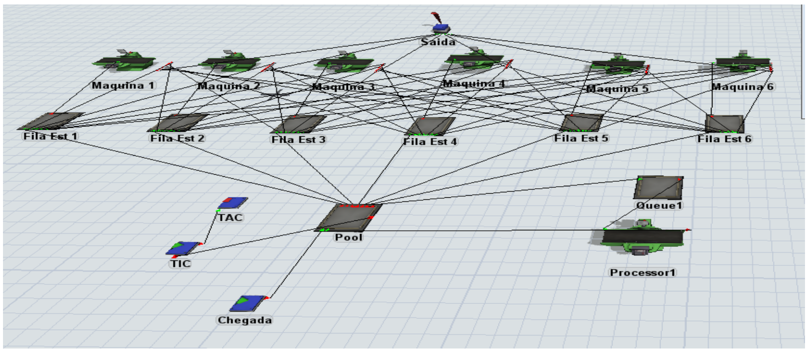
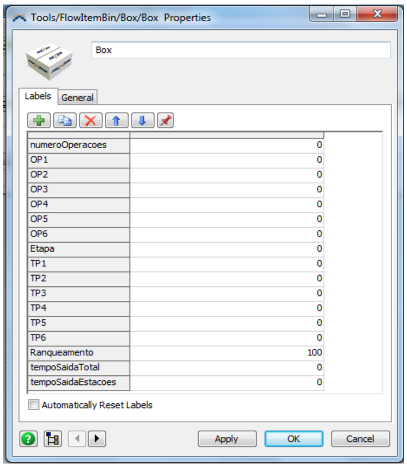


Figure 2. Overall view of the FlexSim model displaying the objects used. Source: elaborated by the authors.

Figure 2 represents the following objects: i) the *Source* “Chegada” for the entry of the order into the system; ii) the *pool*, and the reentry system into the *pool* (*Queue1* + *Processor1*), the time counter (*Source TIC* + *Sink TAC*) to execute the periodic opening of the pool; iii) the objects “Fila Est” 1–6, “Maquina” 1–6 to execute the order, and iv) the *Sink* “Saida” to complete the order and generate the indicators.

4.4.1 Order creation

Work order is represented on FlexSim by the item *Box*. A *label* was used to define the attributes of the order/job. The labels allow to save data in the orders for decision-making procedures in the model (Figure 3).



Box

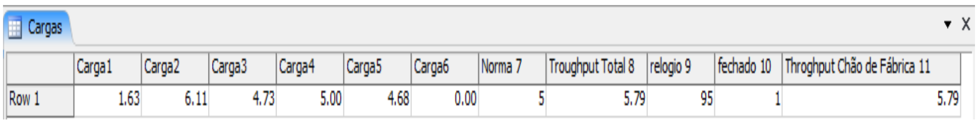
numeroOperacoes	0
OP1	0
OP2	0
OP3	0
OP4	0
OP5	0
OP6	0
Etapa	0
TP1	0
TP2	0
TP3	0
TP4	0
TP5	0
TP6	0
Ranqueamento	100
tempoSaidaTotal	0
tempoSaidaEstacoes	0

☐ Automatically Reset Labels

Figure 3. Labels of the item (order) used to register attributes. Source: elaborated by the authors

The label “numeroOperacoes” allows to save the quantity of operations in the routing of an order. The labels from OP1 to OP6 contain the information of which machine must process an order according to its sequence of operations. The label “Etapa” stores the operation number in which the order is at each moment, according to the sequence of operations in the routing. The labels from TP1 to TP6 store the processing time of order per operation, used to calculate the workloads. The label “Ranqueamento”, which starts with value 100, gives preference to an order when it cannot be released, follows to the waiting list, and returns to the pool. The labels “TempoSaidaTotal” and “TempoSaidaEstacoes” allow to store, respectively, the moment of order completion and the moment of exit from each station after processing.

The table “Cargas” (Figure 4), containing a single line and eleven columns, was created to store and visualize the system workload. The first six columns store the temporary workload of the machines used for decision making about the order release. The seventh column contains the workload norm for all machines. In the eighth column, the value of total throughput time of the items is saved, while the ninth column corresponds to a time counter to execute the simulation considering five units of time. Finally, the tenth column indicates whether the pool is open to release orders, while the eleventh column stores the value of shop floor throughput time.



	Carga1	Carga2	Carga3	Carga4	Carga5	Carga6	Norma 7	Troughput Total 8	relogio 9	fechado 10	Throghput Chão de Fábrica 11
Row 1	1.63	6.11	4.73	5.00	4.68	0.00	5	5.79	95	1	5.79

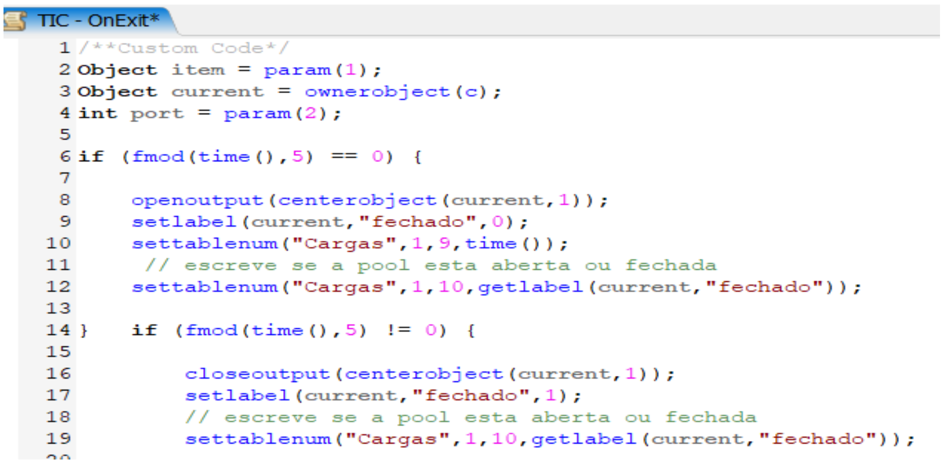
Figure 4. Detailed view of the table “Cargas”. Source: elaborated by the authors.

The orders exit the *Source* “Chegada” considering the inter-arrival time presented in Chart 1 and follow to the pool. The determination of the model attributes in the *Source* “Chegada” follows a sequence of internal events. For the event of simulation restart (“*OnReset*”), a value of zero is determined for the table controlling the workloads of the stations. For the event of item exit of the entity *Source*, a color for the item is randomly defined. The event of item creation (*OnCreation*) creates labels to store the attributes that will be used in the simulation.

4.4.2 Periodic opening of the pool

The mechanism created to control the periodic opening of the pool for the release of orders is the TIC-TAC (Figure 5), which is constituted of two objects (*Source* and *Sink*) linked to the object *queue*, which represents the job pool. TIC-TAC is programmed to open and close the door (exit) of the *pool* at an interval time by releasing the orders through the commands of *openoutput* and *closeoutput*, as indicated in the lines of the commands extracted from the model, tab *OnExit* of the object TIC.

The *Source* TIC assesses the remainder of the division of the time by five units, since this value was defined as the interval between releases (Figure 5, line 6). If the remainder of this division is zero, the *Source* opens its exit for the items created (Figure 5, line 8). It generates the label “fechado” (Figure 5, line 9), transfers the current simulated time to the workload table in the counter column (Figure 5, line 10), and writes the value of “fechado” on the workload table (Figure 5, line 11). These two last steps occur for verification of the periodic behavior. In case the time is different than the multiple of 5 units (Figure 5, line 14), the *Source* close the exit of orders (Figure 5, line 16), modifies the value of its label (Figure 5, line 17), and rewrites on the table of workloads that the pool is closed (Figure 5, line 19). Thus, the columns of the Workload Table called “Relogio 9” and “Fechado 10” allow to verify the functioning of the pool periodic opening to release the orders.



```
 1 /**Custom Code*/
 2 Object item = param(1);
 3 Object current = ownerobject(c);
 4 int port = param(2);
 5 
 6 if (fmod(time(),5) == 0) {
 7 
 8     openoutput(centerobject(current,1));
 9     setlabel(current,"fechado",0);
10     settablenum("Cargas",1,9,time());
11      // escreve se a pool esta aberta ou fechada
12     settablenum("Cargas",1,10,getlabel(current,"fechado"));
13 
14 }   if (fmod(time(),5) != 0) {
15 
16         closeoutput(centerobject(current,1));
17         setlabel(current,"fechado",1);
18         // escreve se a pool esta aberta ou fechada
19         settablenum("Cargas",1,10,getlabel(current,"fechado"));
```

Figure 5. Implementation of the periodic opening of the pool. Source: elaborated by the authors.

The released orders follow to the workstations defined in their routing.

4.4.3 Order release

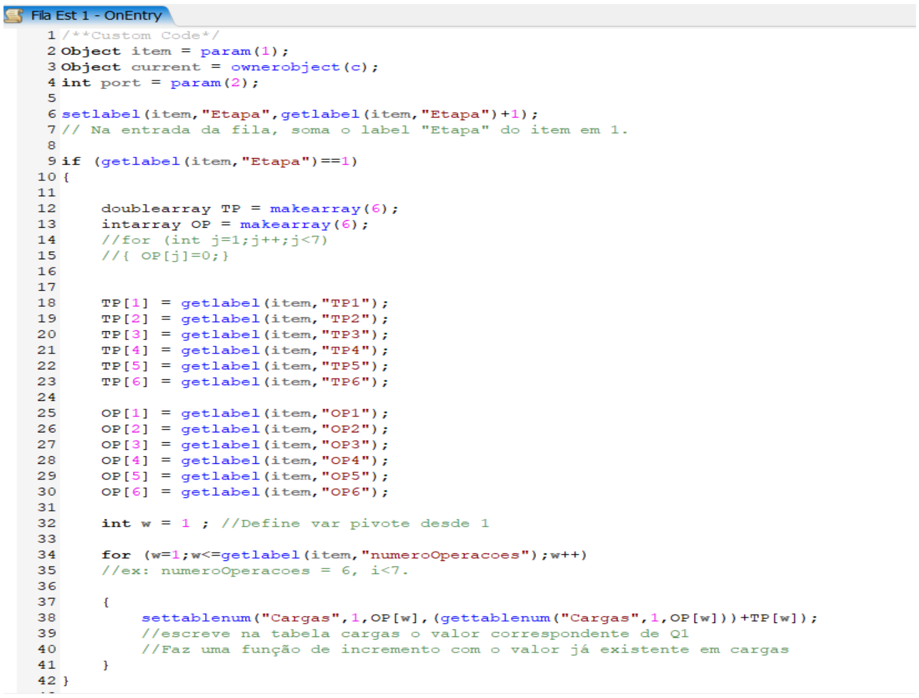
The assessment of the workloads for the release of orders is performed in the object *pool* according to the following steps: i) ordering of jobs/orders to define the sequence in which they are evaluated for release, ii) reading of the current workload (direct and indirect) of each machine, iii) decision of order release based on the currently existing workload, iv) storage of the time at the moment of release, and v) definition of the exit door for the order to follow the desired routing.

To proceed with the order release, the system checks the data of current workload of the machines (stored in a *Global Table* “Cargas”), as well as the routing and order processing time to be considered for release, sums the current workload of the machines with the processing times of the considered order and evaluates the result in relation to the norm.

4.4.4 Order execution and load update

Once the order is released, it is processed in the machines following its routing until leaving the shop floor. Each machine has its own waiting queue. Figure 6 details the steps executed in the waiting queue of a given machine. For example, in machine 1, the first step is to state that the order completes one more stage in the sequence by adding a unit value to the label of the stage, i.e. label “Etapa” (Figure 6, lines 6-8). Subsequently, in the waiting queue, it is possible to check whether an order is at the first stage/operation of its routing. This is important, since the update of the workloads (in the *Global Table* “Cargas”) is only conducted at the first operation of each order, when the order has just been released to the shop floor. At this moment, the order processing times are added to the workload of the respective machines. Thus, in case an order is at the first operation, two vectors are created to store the six processing times (TP) and the values of the six order operations (OP) (Figure 6, lines 18-30). Subsequently, a "*for*" loop is created to register in the *Global Table* of workloads (named "Cargas") all the processing times of a certain order in the respective machines, according to the production routing, by adding them to the existing workload displayed in each column of the table (Figure 6, lines 34-40). In this context, the order processing time referent to each machine in which it is inserted (or the queue in front of a machine) is the direct workload, while the remaining processing times of the order referent to the other machines is the indirect workload. In case the order is not ongoing its first operation when arriving in the queue (Figure 6, line 9), nothing is added to the *Global Table* of workloads and the order follows its programmed routing. The queues of the other machines follow the same pattern.

The last step is to remove the processing time of the order already processed from the workload table, which is performed in the machine exit (*OnExit*). With these data, the machine has its current workload – stored in the table of workloads – reduced, after the value of the processing time corresponding to this stage is subtracted. Subsequently, the order follows its established routing.



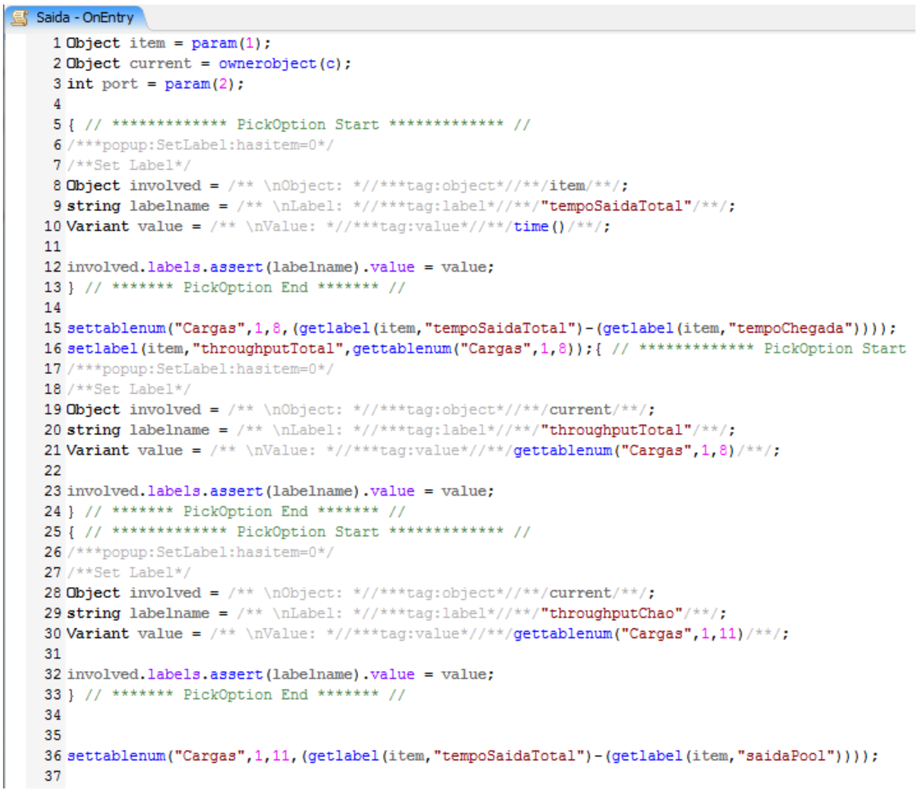
```
 1 /**Custom Code*/
 2 Object item = param(1);
 3 Object current = ownerobject(c);
 4 int port = param(2);
 5
 6 setlabel(item,"Etapa",getlabel(item,"Etapa")+1);
 7 // Na entrada da fila, soma o label "Etapa" do item em 1.
 8
 9 if (getlabel(item,"Etapa")==1)
10 {
11
12     doublearray TP = makearray(6);
13     intarray OP = makearray(6);
14     //for (int j=1;j++;j<7)
15     //{ OP[j]=0;}
16
17
18     TP[1] = getlabel(item,"TP1");
19     TP[2] = getlabel(item,"TP2");
20     TP[3] = getlabel(item,"TP3");
21     TP[4] = getlabel(item,"TP4");
22     TP[5] = getlabel(item,"TP5");
23     TP[6] = getlabel(item,"TP6");
24
25     OP[1] = getlabel(item,"OP1");
26     OP[2] = getlabel(item,"OP2");
27     OP[3] = getlabel(item,"OP3");
28     OP[4] = getlabel(item,"OP4");
29     OP[5] = getlabel(item,"OP5");
30     OP[6] = getlabel(item,"OP6");
31
32     int w = 1 ; //Define var pivote desde 1
33
34     for (w=1;w<=getlabel(item,"numeroOperacoes");w++)
35     //ex: numeroOperacoes = 6, i<7.
36
37     {
38         settablenum("Cargas",1,OP[w],(gettablenum("Cargas",1,OP[w]))+TP[w]);
39         //escreve na tabela cargas o valor correspondente de Q1
40         //Faz uma função de incremento com o valor já existente em cargas
41     }
42 }
```

Figure 6. Configuration of the queue before machine 1 (object *Queue*, "Fila Est 1"). Source: elaborated by the authors.

4.4.5 Order completion and collection of indicators

When an order completes its routing, it follows to the *Sink* "Saida" to finalize its processing in the simulation. The moment at which an order enters the *Sink*, the current value of the simulation time is stored on the label "TempoSaidaTotal" (Figure 7, lines 5-13).

The goal is to save the total completion time of the order on the labels of the item *Box*. Subsequently, the *Sink* calculates the total throughput time of the order and stores the value in the column 8 of the Workload Table (Figure 7, line 15), registers this value in the label of the item *Box* (Figure 7, lines 16) and in the labels of the *Sink* (Figure 7, lines 17-24). The *Sink* stores on its label the last value of shop floor throughput time (Figure 7, lines 25-33) and updates the same value in the column 11 of the *Global Table* "Cargas" (Figure 7, line 36). The columns 8 and 11 of this table allow to verify, at the moment of the simulation: 1) the values of total throughput, that is, the order throughput time from the moment at which it is created in the object *Source* to its exit, in the object *Sink*; 2) the values of shop floor throughput time, that is, the time from the moment at which the order is released by the pool to the shop floor to its exit, in the object *Sink*.



```
 1 Object item = param(1);
 2 Object current = ownerobject(c);
 3 int port = param(2);
 4
 5 { // ************* PickOption Start ************* //
 6 /***popup:SetLabel:hasitem=0*/
 7 /**Set Label*/
 8 Object involved = /** \nObject: *//***tag:object*//**/item/**/;
 9 string labelname = /** \nLabel: *//***tag:label*//**/"tempoSaidaTotal"/**/;
10 Variant value = /** \nValue: *//***tag:value*//**/time()/**/;
11
12 involved.labels.assert(labelname).value = value;
13 } // ******* PickOption End ******* //
14
15 settablenum("Cargas",1,8,(getlabel(item,"tempoSaidaTotal")-(getlabel(item,"tempoChegada"))));
16 setlabel(item,"throughputTotal",gettablenum("Cargas",1,8));{ // ************* PickOption Start
17 /***popup:SetLabel:hasitem=0*/
18 /**Set Label*/
19 Object involved = /** \nObject: *//***tag:object*//**/current/**/;
20 string labelname = /** \nLabel: *//***tag:label*//**/"throughputTotal"/**/;
21 Variant value = /** \nValue: *//***tag:value*//**/gettablenum("Cargas",1,8)/**/;
22
23 involved.labels.assert(labelname).value = value;
24 } // ******* PickOption End ******* //
25 { // ************* PickOption Start ************* //
26 /***popup:SetLabel:hasitem=0*/
27 /**Set Label*/
28 Object involved = /** \nObject: *//***tag:object*//**/current/**/;
29 string labelname = /** \nLabel: *//***tag:label*//**/"throughputChao"/**/;
30 Variant value = /** \nValue: *//***tag:value*//**/gettablenum("Cargas",1,11)/**/;
31
32 involved.labels.assert(labelname).value = value;
33 } // ******* PickOption End ******* //
34
35
36 settablenum("Cargas",1,11,(getlabel(item,"tempoSaidaTotal")-(getlabel(item,"saidaPool"))));
37
```

Figure 7. Configuration of the *Sink* "Saida" (calculation of the indicators – i.e. throughput time – and registration in the labels of the *Sink*). Source: elaborated by the authors.

5 Results

Exploratory simulations were performed for the model proposed aiming at verifying its feasibility. Five scenarios were simulated containing different norm values, in which the first scenario considers a large norm value, representing the infinite norm, in order to not obstruct the release of orders. The norm values for the remaining scenarios were: Scenario 2 = 8, Scenario 3 = 16, Scenario 4 = 24, Scenario 5 = 32.

A hundred replications with simulation time of 6,000 units of time (defined in the modeling as seconds) and warm-up time of 3,000 units of time were carried out. These parameters are commonly used to reduce the variation in similar WLC experiments (Land, 2006). The simulation was executed on a personal computer using an Intel i7 processor with 6GB of RAM – operational system MS® Windows 10.

The simulation considered the classic WLC model, that is: the order is always classified according to the date of planned release and there is no priority if it is returning from the loop (list of not released orders). In addition, it is only released to the shop floor if the sum of the workload of the machines and order processing time does not exceed the fixed norm.

The results of throughput time found in the first scenario were compared with the remaining scenarios.

The mean total throughput time (Figure 8) upon a very tight norm – as in the second scenario (norm of 8) – is much high (average of 79 time units) in relation to the first scenario with infinite norm (average of 27 time units). In scenarios 3, 4 and 5, in turn, a mean total throughput time of 32, 30, and 29 units of time was obtained, respectively.

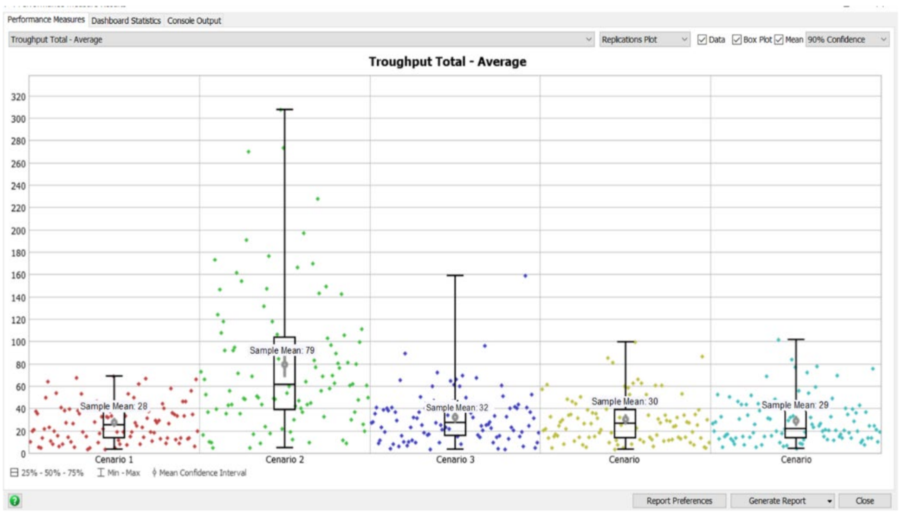


Figure 8. Simulation results: total throughput time. Source: elaborated by the authors.

The analysis of the shop floor throughput time (Figure 9) revealed that the second scenario presented an average of 28 units of time, registering an increment of 1 unit of time in relation to the first scenario (27 units of time). In turn, the scenarios 3, 4 and 5 had lower average values of throughput time for shop floor than the first scenario: 22, 23, and 24 units of time, respectively.

The average results of the shop floor throughput time found in the scenarios 3, 4 and 5 in relation to the first and second scenarios present differences that can be representative in a real situation.

In the comparison of scenario 3 and scenario 1, for example, the difference in the average value for each order is five units of time. Considering the simulation of 100 orders, this implies that scenario 3 processed these 100 orders in 500 units of time less than scenario 1. The result of this comparison leads to the conclusion that the shop floor in scenario 3 is more productive than in scenario 1 for being able to process the same quantity of orders at a shorter period. The orders pass through the shop floor more rapidly because of the occurrence of less queues (lower saturation), however, they spend more time waiting in the pool, which causes the total throughput time in scenario 3 to increase (32 versus 28 in scenario 1).

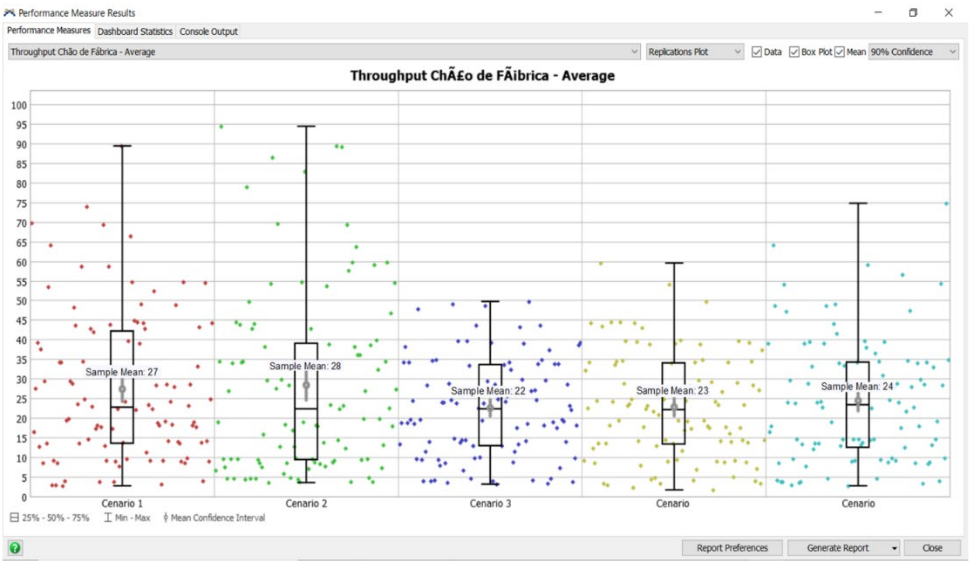


Figure 9. Simulation results: shop floor total throughput. Source: elaborated by the authors.

According to the results, the total throughput time obtained from the infinite norm (without workload control) were slightly better than those obtained in the scenarios 4 and 5, probably due to matters of adjustment of parameters, such as the values for workload norm and interval between releases. Such adjustments must be conducted in a way that the trade-off mentioned in the previous paragraph is advantageous, that is, the decrease in the throughput time of shop floor, in absolute values, is higher than the increase in the time waiting in the pool. Additionally, the efficiency of the workload control approach is also directly related to the levels of machine utilization. For not such high utilization levels, restricting the release does not generate significant results, which can also explain the results found.

Another important characteristic to be analyzed is the variability of the shop floor throughput time. It is possible to verify a high variability of these values for the scenarios 1, 2, and 5, as well as lower variability for the scenario 3, indicating that it is a more stable scenario comparing with the remaining ones (Figure 9).

The variation of norm values for order release in a discrete-event simulation environment allows to analyze the effects that it causes on the total and on the shop floor throughput time. It is also possible to analyze which norm value generates the lowest variability for the calculated time, thus enabling to choose a more stable scenario.

It is important to highlight that these simulations are only of exploratory nature since the emphasis of this study, as previously mentioned, is to present a detailed structure for the implementation of the WLC approach on a commercial software. Land (2006), among others (literature on this topic is relatively vast), presents a more thorough discussion on the definition of the workload control parameters and its impacts on the throughput time.

6 Final remarks

This paper presented an application of the workload control (WLC) approach on a commercial software of discrete-event simulation (i.e. on an integrated graphic simulation environment). The literature provides truly little emphasis on the model development per se and many simulation-based studies do not mention either the software programs or language used. Therefore, this study provides the details on the conversion of the basic algorithm of the workload control with periodic release to a simulation model developed on FlexSim software.

This paper demonstrated that the implementation of the WLC algorithm on a commercial software for discrete-event simulation is possible and viable. However, in an integrated graphic environment that uses objects (i.e. elements, entities) and a decentralized programming, such implementation is not trivial. The commands and sub-routines that implement the lines in the base algorithm of workload control are distributed in different objects and it is necessary to create coordination mechanisms between them for an execution in the correct order and moments. Examples of these mechanisms of interaction and integration between objects include the use of labels and tables, which are read at different moments of the simulation by different objects to gather updated information to execute the sub-routines or decisions on the order flow.

For the correct functioning of the workload control approach in the modeled system, it is necessary to implement either mechanisms to control the frequency of opening and closing of the pool and mechanisms representing the list of not released orders. When a routine with a structured and centralized programming without graphic representation is developed, such mechanisms require a few code lines comparing with the implementation on a simulation software for discrete events, which, in addition to codes lines, also requires to create connections between the objects and decentralized routines.

Even though it is not so common to implement the WLC using a commercial and decentralized simulation software, it is important to highlight some of the advantages involved, such as: descriptive capacity of this type of software, enabling the modeling and simulation of an industrial environment close to reality; possibility of visualizing the dynamics of the simulation and the functioning of the WLC approach in the modeled environment, since it is possible to visualize the trajectory of the orders from their creation to the system exit; and the possibility of clicking on the items and objects to visualize the information contained in them at the exact moment of the simulation, as well as modifications during the simulation period.

As contributions to further research, this model can support the representation of a WLC implementation in other shop floor configurations, contributing to reduce the developer's effort employed in modeling and programming, since it allows to set off a well-structured initial model for the creation and modeling of new industrial scenarios that use the WLC approach.

Among the limitations of this paper, we can mention the use of only one study as reference from the specific literature – Land (2006). Even though it represents a well-known model, expanding the comparison of this structure with other authors would generate a deeper analysis and allow to find different results, which is recommended for further research. Other suggestions include the application of other types of norms available in the literature, as well as alternative types of indirect workload accounting (i.e. probability approach), and the implementation of other types of release, such as the continuous release or the release based on the workload balancing mentioned in the literature review section.

The use of only one software of discrete-event simulation – FlexSim – is another research limitation, since similar software programs (Arena, ProModel, etc.), which require alternative strategies and resources to generate a WLC representation, are available. There is a known trade-off between the complexity of building an entire simulation model only by means of programming (e.g. using Python or SimPy, which are free software) and the use of software with user-friendly interface, like FlexSim, AnyLogic, Simul8, Arena etc., which requires license of use. This is a relevant limitation of this study and the publication of tutorials such as this one for languages and open-access software programs is an important contribution for further studies in the scope of this research area.

References

Bechte, W. (1994). Load oriented manufacturing control just in time production for job shops. *Production Planning and Control*, 5(3), 292-307. http://dx.doi.org/10.1080/09537289408919499.

Bertrand, J. W. M., & Wortmann, J. C. (1981). *Production control and information systems for component-manufacturing shops*. Amsterdam: Elsevier Scientific Publishing Company.

Costa, F., Portioli-Staudacher, A., Nisi, D., & Rossini, M. (2019). Integration of order release and output control with worker's allocation in a pure flow shop. In *Proceedings of the 9th IFAC Conference MIM 2019.* Berlin: International Federation of Automatic Control. http://dx.doi.org/10.1016/j.ifacol.2019.11.604.

Fernandes, N. O., & Carmo-Silva, S. (2011). Workload control under continuous order release. *International Journal of Production Economics*, 131(1), 257-262. http://dx.doi.org/10.1016/j.ijpe.2010.09.026.

Fernandes, N. O., Thürer, M., Silva, C., & Carmo-Silva, S. (2016). Improving workload control order release: incorporating a starvation avoidance trigger into continuous release. *International Journal of Production Economics*, 194, 181-189. http://dx.doi.org/10.1016/j.ijpe.2016.12.029.

Goldratt, E. M. (1996). *Production the TOC way: work book*. New Haven, CT: Avraham Y. Goldratt Institute.

Hendry, L. C. (1989). *A decision support system to manage delivery and manufacturing lead times in make-to-order companies* (Ph.D. thesis). University of Lancaster, Lancaster, UK.

Hendry, L. C., & Kingsman, B. G. (1989). Production planning systems and their applicability to make-to-order companies. *European Journal of Operational Research*, 40(1), 1-15. http://dx.doi.org/10.1016/0377-2217(89)90266-X.

Hendry, L. C., Kingsman, B. G., & Cheung, P. (1998). The effect of workload control (WLC) on performance in make-to-order companies. *Journal of Operations Management*, 16(1), 63-75. http://dx.doi.org/10.1016/S0272-6963(97)00011-9.

Hendry, L., & Kingsman, B. (1991). A decision support system for job release in make-to-order companies. *International Journal of Operations & Production Management*, 11(6), 6-16. http://dx.doi.org/10.1108/01443579110144655.Kingsman, B. G., & Hendry, L. C. (2002). The relative contributions of input and output controls on the performance of a workload control system in make-to-order companies. *Production Planning and Control*, 13(7), 579-590. http://dx.doi.org/10.1080/0953728021000026285.

Kingsman, B. G., Tatsiopoulos, I. P., & Hendry, L. C. (1989). A structural methodology for managing manufacturing lead times in make-to-order companies. *European Journal of Operational Research*, 40(2), 196-209. http://dx.doi.org/10.1016/0377-2217(89)90330-5.

Kirchhof, P., Meseth, N., & Witte, T. (2008, December). Simulation based evaluation of the workload control concept for a company of the automobile industry. In 2008 Winter Simulation Conference (pp. 1856-1862). IEEE. http://dx.doi.org/10.1109/WSC.2008.4736275.

Köchel, P., & Nieländer, U. (2002). Kanban optimization by simulation and evolution. *Production Planning and Control*, 13(8), 725-734. http://dx.doi.org/10.1080/0953728031000057334.

Land, M. (2006). Parameters and sensitivity in workload control. *International Journal of Production Economics*, 104(2), 625-638. http://dx.doi.org/10.1016/j.ijpe.2005.03.001.

Land, M. J. (2009). Cobacabana (control of balance by card-based navigation): a card-based system for job shop control. *International Journal of Production Economics*, 117(1), 97-103. http://dx.doi.org/10.1016/j.ijpe.2008.08.057.

Land, M. J., & Gaalman, G. J. C. (1998). The performance of workload control concepts in job shops: improving the release method. *International Journal of Production Economics*, 56, 347-364. http://dx.doi.org/10.1016/S0925-5273(98)00052-8.

Land, M., & Gaalman, G. (1996). Workload control concepts in job shops a critical assessment. *International Journal of Production Economics*, 46, 535-548. http://dx.doi.org/10.1016/S0925-5273(96)00088-6.

Missbauer, H. (2002). Aggregate order release planning for time-varying demand. *International Journal of Production Research*, 40(3), 699-718. http://dx.doi.org/10.1080/00207540110090939.

Morabito, R., No., & Pureza, V. (2012). Modelagem e simulação. In P. A. Cauchick Miguel (Ed.), *Metodologia de pesquisa em engenharia de produção e gestão de operações.* Rio de Janeiro: Elsevier.

Moreira, M. D. R. A., & Alves, R. A. F. (2012). Input-output control order release mechanism in a job-shop: how workload control improves manufacturing operations. *International Journal on Computer Science and Engineering*, 7(3), 214-223.

Oosterman, B., Land, M., & Gaalman, G. (2000). The influence of shop characteristics on workload control. *International Journal of Production Economics*, 68(1), 107-119. http://dx.doi.org/10.1016/S0925-5273(99)00141-3.

Perona, M., & Miragliotta, G. (2000, 21-25 February). Workload control: a comparison of theoretical and practical issues through a survey in field. In Conference Proceedings 11th International working seminar on production economics, Igls, Innsbruck (pp. 1-14).

Plossl, G. W. (1985). *Production and inventory control: principles and techniques* (2nd ed.). New Jersey: Prentice-Hall.

Portioli-Staudacher, A., & Tantardini, M. (2012). A lean-based ORR system for non-repetitive manufacturing. *International Journal of Production Research*, 50(12), 3257-3273. http://dx.doi.org/10.1080/00207543.2011.564664.

Sagawa, J. K., & Land, M. (2018). Representing workload control of manufacturing systems as a dynamic model. *IFAC-PapersOnLine*, 51(, 2), 825-830. http://dx.doi.org/10.1016/j.ifacol.2018.04.016.

Seppanen, M. S. (1993). Kanban simulator using siman and lotus 1-2-3. In *Proceedings of 1993 Winter Simulation Conference (WSC '93).* Piscataway: IEEE. http://dx.doi.org/10.1109/WSC.1993.718327.

Stevenson, M. (2006). Refining a workload control (WLC) concept: a case study. *International Journal of Production Research*, 44(4), 767-790. http://dx.doi.org/10.1080/00207540500338070.

Stevenson, M., Huang, Y., & Hendry, L. C. (2009). The development and application of an interactive end-user training tool: part of an implementation strategy for workload control. *Production Planning and Control*, 20(7), 622-635. http://dx.doi.org/10.1080/09537280903034313.

Stevenson, M., Huang, Y., Hendry, L. C., & Soepenberg, E. (2011). The theory and practice of workload control: a research agenda and implementation strategy. *International Journal of Production Economics*, 131(2), 689-700. http://dx.doi.org/10.1016/j.ijpe.2011.02.018.

Suwamuji, P. (2003). A simulation test bed for production and supply chain modeling. In *Proceedings of the 2003 Winter Simulation Conference*. Piscataway: IEEE. http://dx.doi.org/10.1109/WSC.2003.1261547.

Tatsiopoulos, I. P. (1983). *A micro-computer based interactive system for managing production and marketing in small component-manufacturing firms: using a hierarchical backlog control and lead time management methodology* (Ph.D. thesis). University of Lancaster, Lancaster, UK.

Thürer, M., & Godinho, M., Fo. (2012). Redução do lead time e entregas no prazo em pequenas e médias empresas que fabricam sob encomenda: a abordagem Worload Control (WLC) para o Planejamento e Controle da Produção (PCP). *Gestão & Produção*, 19(1), 43-58. http://dx.doi.org/10.1590/S0104-530X2012000100004.

Thürer, M., Land, M. J., Stevenson, M., Fredendall, L. D., & Godinho, M., Fo. (2015). Concerning workload control and order release: the pre-shop pool sequencing decision. *Production and Operations Management*, 24(7), 1179-1192. http://dx.doi.org/10.1111/poms.12304.

Thürer, M., Qu, T., Stevenson, M., Li, C. D., & Huang, G. Q. (2017). Deconstructing bottleneck shiftiness: the impact of bottleneck position on order release control in pure flow shops. *Production Planning and Control*, 28(15), 1223-1235. http://dx.doi.org/10.1080/09537287.2017.1362486.

Thürer, M., Stevenson, M., & Silva, C. (2011). Three decades of workload control research: a systematic review of the literature. *International Journal of Production Research*, 49(23), 6905-6935. http://dx.doi.org/10.1080/00207543.2010.519000.

Thürer, M., Stevenson, M., Silva, C., Land, M. J., & Fredendall, L. D. (2012). Workload control and order release: a lean solution for make-to-order companies. *Production and Operations Management*, 21(5), 939-953. http://dx.doi.org/10.1111/j.1937-5956.2011.01307.x.

Weng, M. X., Wu, Z., Qi, G., & Zheng, L. (2008). Multi-agent-based workload control for make-to-order manufacturing. *International Journal of Production Research*, 46(8), 2197-2213. http://dx.doi.org/10.1080/00207540600969758.

Williams, E. J., Ulgen, O. M., & Dewitt, C. (2012). An approach and interface for building generic manufacturing Kanban-systems models. In *Proceedings of the Winter Simulation*. Piscataway: IEEE. http://dx.doi.org/10.1109/wsc.2002.1166370.