Journal of Information Systems and Technology Management – Jistem USP
Vol. 18, 2021, e202118001
ISSN online: 1807-1775
DOI: 10.4301/S1807-1775202118001 Original Article

A DYNAMIC JOB SHOP MODEL FOR SCHEDULING TASKS IN A SOFTWARE DEVELOPMENT ENVIRONMENT

Sildenir A. Ribeiro[1,2] http://orcid.org/0000-0003-4808-1009
Eber A. Schmitz[2] http://orcid.org/0000-0002-4839-4606
Mônica F. da Silva[2] http://orcid.org/0000-0003-0951-6612
Antônio Juarez S. M. de Alencar[2] http://orcid.org/0000-0003-4791-0581

[1]Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, Rio de Janeiro, RJ, Brazil

[2]Federal University of Rio de Janeiro, UFRJ, Rio de Janeiro, RJ, Brazil

ABSTRACT

This work proposes a Dynamic Job Shop Scheduling (DJSS) model for scheduling the task in a software production environment. The aim is to organize the insertion of tasks in the shop in a precedence order previously defined that obeys the mapping of the critical path and the critical chain to identify constraints in the productive process. The complexity and dynamism of the software development environment require a mature and adjusted process that enables management throughout its extension. Therefore, the Unified Process (UP) was used in conjunction with the Theory of Constraints (TOC) in the Software Development Process (SDP). The UP requires an efficient model for the processing of activities, given by the set of independent variables involved in the process. For this reason, a model to programming the task based on dynamic scheduling was developed. The problem around the DJSS is to program the tasks in the shop in a way that allows identifying one or more production lines in the shop with constraints of capacity at runtime. A capacity constraint is any element that disturbs the productive process, causing L={Mi Aj} to have delivery (DA) of less than 100% of the artifacts, given by the set of jobs (j_n) scheduled. The resource with capacity constraint limits the production of the machines in the production line, leading to bottlenecks in the production process. A bottleneck, in turn, is denoted by the production limitation of a machine (people) caused by one or more capacity constraints that imply maximum production, which in this case is defined by D={Mi, Aj=100%}. In this work, the maximum production is given by the set of tasks processed in each of the phases of the UP and according to the productive calendar, which delimits the deadline of each delivery and establishes the budget of hours

Manuscript first received: Manuscript first received: 2020-09-29. Manuscript accepted: 2020-12-27

Address for correspondence:
Sildenir Alves Ribeiro, Federal Center for Technological Education of Rio de Janeiro (CEFET/RJ), Rio de Janeiro, RJ, Brazil,
Email:sildenir.ribeiro@cefet-rj.br
Eber Assis Schmitz, Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, RJ, Brazil,
Email: eber@nce.ufrj.br
Mônica Fereira da Silva, Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, RJ, Brazil,
Email: monica@nce.ufrj.br
Antônio Juarez S. M. de Alencar, Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, RJ, Brazil,
Email: juarezalencar@nce.ufrj.br

for the shop. The set of tasks scheduled and "rescheduling" and properly processed in all phases define the total production of each machine. The use of DJSS facilitated the organization and management of insertion and rescheduling operations in the shop and as result, produced data to measure the effort/time that allowed identifying the bottleneck of the software production process.

Keywords. Dynamic Job Shop Scheduling, Software Development Process, Software Development Environment, Theory of Constraints, Experimental Software Engineering.

1. INTRODUCTION

Job Shop Scheduling Problems (JSSP) are those that involve resource allocation over time, for perform a series of tasks, attending to an objective function, which can: minimize execution time, minimize cost, maximize quality, maximize production, etc. The optimal solution to these problems requires a great computational effort, once that the JSS problems is a particular case of the NP-Hard class (Blazewicz et al, 1996) qtd. in (Ribeiro et al, 2012). The JSSP is widely explored in the industry as an important tool for modeling and adjusting the process and optimizing the productive results (Araujo, 2006), (Zang et al 2017), (Souza et al, 2017), (Turker et al, 2019), (Mansouri et al, 2019) e (Dai et al, 2019). In the case of a software development environment, the application of a Job Shop Schedule (JSS) is not so trivial and so uncommon because the tasks/operations are performed by people, making the process more susceptible to internal and external disturbances. Besides, the software production process presents a fundamental difference concerning industrial production processes - the software production unit is organized to produce a single product at a time, and in principle, it must be reconfigured to make a new product. This scenario requires that the development process be delineated according to the specificity of each project/product, as stated in the software engineering literature (Pressman and Maxin, 2014), (Sommerville, 2011), (Schach, 2011).

However, the result of executing a software process is a product finished, usable, and applied for a particular purpose, as well as a manufactured product. Therefore, software development also requires a model, that together with its production process, allows managing, control, and make small adjustments to the construction of a reliable software product. But differently of the conventional production systems, where tasks have a linear order insertion and execution, with well-defined operations (one to one[1]), the Software Development Process (SDP) happens in a highly dynamic environment (Ribeiro, 2018), and this, affects the process and consequently the final product. The constraints in a software development environment are so many that discourage the application of a more rigid approach to SDP management and control. On the other hand, this research presents an important contribution with the multidisciplinary use of methodologies, methods, techniques, and concepts to the SDP in order to be able to measure the process and its entire development cycle and also to encourage the application of models and heuristics used in other areas in the SDP.

For this reason, we propose the use of a Dynamic Job Shop Schedule (DJSS) model to order the scheduling and rescheduling of tasks in a software development shop, thus allowing a time/effort mapping vs. total tasks of a set of 5 artifacts. Allied to the DJSS model, we made use of the Unified Process (UP) (Jacobson et al. 1999), supported by the Theory of Constraint (TOC) (Goldratt and Cox, 2014). The choice of UP is justified by the fact that the SDP requires a mature and robust process to guide the construction of the software in all PU-phases: design, design, coding, testing, and transition.

[1] One to one: for each entry, one output, i.e., a artifact produced.

2. EXECUTION ENVIRONMENT AND METHODOLOGY

To simulate the production process and applying the DJSS, we developing three quasi-experiments that characterize the shops of the software development environment. The tasks were scheduled in the shops according to the JSSP concepts, in their dynamic variation, called Dynamic Job Shop Scheduling (DJSS). The first shop was used only for observation and calibration of the process and the job shop model adopted. The second and third shop, we use the DJSS together with the PU to schedule the tasks in the shops and thus measure the effort (time consumption) and the production (quantity of tasks processed) by the machines.

The work plan was divided into four stages, presented in (Ribeiro, 2018). Stage 2 involves the environmental and human resources that configure the machines and production lines of each shop. The plan was based on theories, methods, and models (Wohlin et al, 2012) and (Wohlin, 2007), and available tools/templates, such as (Basili & Weiss, 1984) and (Schull et al. 2008). The work plan resulted in a quantitative mapping of the variables, indicators, and instruments generated for each shop. Table 1 shows the instruments, indicators, and units measured for each shop.

Table 1 – List and quantitated of instruments

Instruments	Indicators	Units
Count	Total of experiments (repetitions)	3 (Shop 1, 2 e 3)
Count	Total effort (time in minutes)	2 (Shop 2 e 3)
Count	Total of machines per production lines	>= 2, <= 4
Count	Total PL (shop 1, 2 e 3)	10, 9, 6
Count	Total of machines / partaker (shop 1, 2 e 3)	19, 28, 31
Count	Total of artifacts	5 per team / PL
Count	Total of tasks (activities)	26 per team / PL
Count	Total of numerical report	3 per shop.
Template & Process	Selection of sample variables	1 per shop
Support Software	Data repository	1 per shop
Model	Scheduling of tasks to be delivered	26 per shop (according to UP)

Total PL *(shop 1 = 10, shop 2 = 9 and shop 3 = 6).*
Total machines per shop *(shop 1 = 19, shop 2= 28 and shop 3 = 31).*

2.1 Literature Review

This work started from gaps and research opportunities published by (Ribeiro et al., 2017) originated from an extensive literature review shows by (Ribeiro et al., 2018) about bottlenecks identification in software development process.

3. THE JOB SHOP SCHEDULING PROBLEM (JSSP)

A classic JSSP is usually referenced as an n X m JSSP, where n is the number of jobs and m is the number of machines. A set of n jobs $J = \{J_1, J_2, ..., J_n\}$ must be processed on a set of available machines $M = \{M_1, M_2, ..., M_m\}$. Each job has a specific execution order between the machines, *i.e*, a job is composed of an ordered list of operations defined by the machine where the jobs were

scheduled and by the processing time of the operations of this machine (Klein, 2000) *qtd.in* (Ribeiro *et al*., 2006). In the classical model, we have a set ***J*** of jobs consisting of a set ***O*** of operations and ***k*** indexes, where ***O*** is the total number of operations for each job and ***k*** is the index of the machine to process the "***k-th***" operation ***O***.

According to Pinedo (2016), the following conditions must be considered for the problem.

1. All machines are different, and their processing speeds are constant.
2. Operations cannot be interrupted;
3. Each job is scaled only once;
4. Each machine can process only one operation at each instant of time;
5. Each job can only be processed on a single machine;
6. A job can be staggered on any machine;
7. Each job is produced by a known sequence of operations;
8. There is no precedence restriction between the operations of different jobs.

The most common purpose of a JSSP is to find a valid job stream that does not violate any of the above constraints and to complete all jobs with the shortest possible time. This is an aim known as makespan, and its purpose is to minimize the time of completion of operations (Ribeiro, 2006).

An example of a classic JSSP with 3 jobs being processed on three machines can be represented in Table 2.

Table 2 – Classic job shop 3X3 (Ribeiro, 2006)

A Job Shop Scheduling (3 X 3)			
Job	**Machine / (Time)**		
1	1(4)	2(6)	3(5)
2	1(3)	3(6)	2(7)
3	2(4)	1(5)	3(2)

Each job is scheduled at a given time point, and the values shown in parentheses indicate the total units of time consumed in the processing. The solution can be represented by a Gantt diagram as shown in Figure 1.

With the sequences of machines of each job fixed, the problem to be solved is to determine the sequences of the jobs in each machine, so that the elapsed execution time between the beginning of the first job until the end of the last job is minimized. The goal is to find a solution with smaller makespan (Ribeiro, 2006).

3.1 Dynamic Job Shop Scheduling (DJSS)

The JSSP is one of three types of classic scheduling problems. The other two types are (1) o Flow Shop Scheduling Problem (FSSP); and (2) Open Shop Scheduling Problem (OSSP) (Thörnblad,

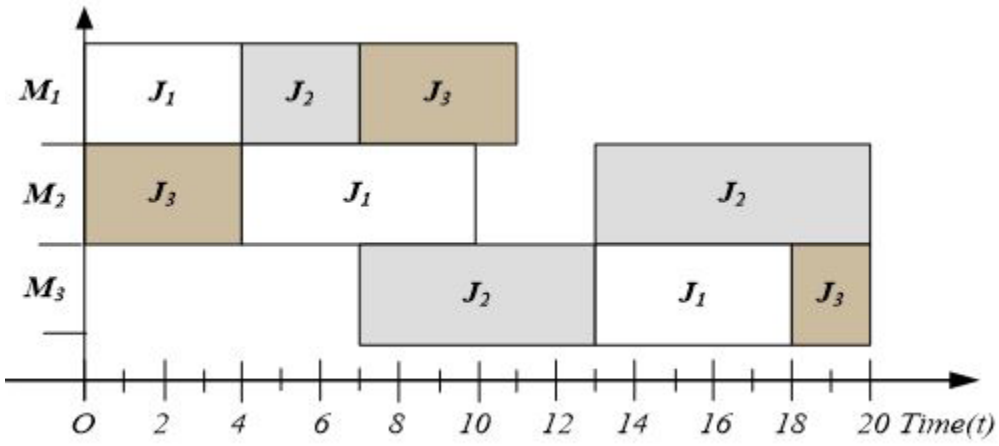

Figure 1 – A classic JSSP (Ribeiro, 2006)

2013). Scheduling methods of the JSSP type can be categorized as local and global (Casavant & Kuhl, 1988). The global method has two classifications: (1) static, that comprises the classic JSSP model; and (2) dynamics, that is based on the distribution of processes/tasks between machines. The dynamic classification may still be deterministic or nondeterministic.

The DJSS is defined by Rabelo & Klen (2000) qtd. in Araújo Jr. (2006), as an action to adapt the current scheduling, simultaneously with its execution in the function of the occurrence of unplanned events. This should be observed for both the factory floor schedules and those at the planning level (case of this work) so that the scheduling remains realistic, feasible, and achievable according to the temporal constraints of capacity and existing technologies.

According to Ouelhadj and Petrovic (2009), the scheduling problem in the presence of real-time events is called dynamic scheduling. Ouelhadj & Petrovic (2009) also affirm that the dynamic scheduling of tasks obeys three policies: periodic, eventual, and hybrid. Periodic and hybrid policies are also known as rolling time horizon policies, so the dynamic scheduling problem is decomposed into a series of static problems (Thörnblad, et al, 2013). In event-oriented policy, scheduling is fractionated and rescheduled in response to events during process execution. These characteristics make the DJSS a potential candidate for Software Development Environment (SDE), which has to deal with events disturbing the process in real-time and tasks being scaled and rescheduled until it is satisfactorily completed

3.2 Relating DJSS Features to the Software Development Environment

The software development context launches the process in a highly dynamic environment. Thus, the conditions of the classical JSSP presented by Pinedo (2016) and described in section 2 are not completely applicable, because, to the contrary that is established in the classic model, the dynamism of the software development environment determines the following scenario (Ribeiro, 2018).

1. The machines are different, and their processing speeds are not constant;
2. Operations can be interrupted and restarted whenever necessary;

3. A job can be rescheduled n times;
4. Each machine can process several operations at different time units;
5. One same job can be processed on different machines and different production lines (PL);
6. One same job can be partitioned and scheduled on two different machines from the same PL;
7. A job is necessarily scheduled on all production lines (typical case of the model and environment developed in this research);
8. Jobs are processed by a sequence of convenient operations or according to the objectives of the machines of each production line;
9. In some cases, exists precedence constraints between operations of different jobs;
10. The objective function is given by maximization of the production of a production line on shop, with $L=\{M_1A_1...A_n, M_2A_1...A_n, ... M_nA_1...A_n\} = 100\%$, where M and A are the machines (M_l) producing $(A_{1...n})$ artifacts.

Therefore, the choice of the DJSS is mainly justified by type of the productive environment and the "shop" prepared to run the trials, where: (1) the tasks performed on the machines are previously defined and determined; and (2) the order of execution is flexible, as it complies with the development plan elaborated for the shops/production line.

3.3 The DJSS model

The observation in this research necessarily passes through the execution environment of the experimental tests. However, is important to adopt an effective model for the processing of task, given by the set of independent variables involved in this type of study. Thus, after developing the methodology and modeling the process to execute the experiments, the dynamic job shop model was chosen as the best alternative to the scheduling of the tasks in the SDP.

3.4 Overview of the DJSS Model Adopted

The insertion of tasks in the machines is not fixed or follows a linear order. In addition, a task can be shared and/or rescheduled until it is completed or until it reaches the expected quality. The only pre-established rule is obedience to predecessor tasks.

The scheduling varies in function on the availability and capacity of the machines. Furthermore, each machine can establish what it wants to do and at what time according to the list of artifacts to be produced and delivered. However, the SDP imposes the prioritization of some tasks so that the product can be built. These tasks can be mapped through the critical path and critical chain methods and were listed as predecessor tasks ***(w)***.

Some tasks still depend on the enhancement of technical knowledge of the machines, which in our case are people. This is done through the insertion of the theoretical content of the Fundamentals Software Engineering (FES) discipline, which corroborates with the non-linear aspect of the scheduling of the shop tasks.

A job ***(j)*** scheduled on a machine can transition to another machine or even have shared processing, thus establishing dynamic processing of tasks. In virtue of these characteristics, the Dynamic Job Shop Scheduling (DJSS) model was chosen to model the problem. Figure 2, is possible to observe that Jobs ***(j)*** navigate between machines in a process of exchange (rescheduling) or sharing to complete the task.

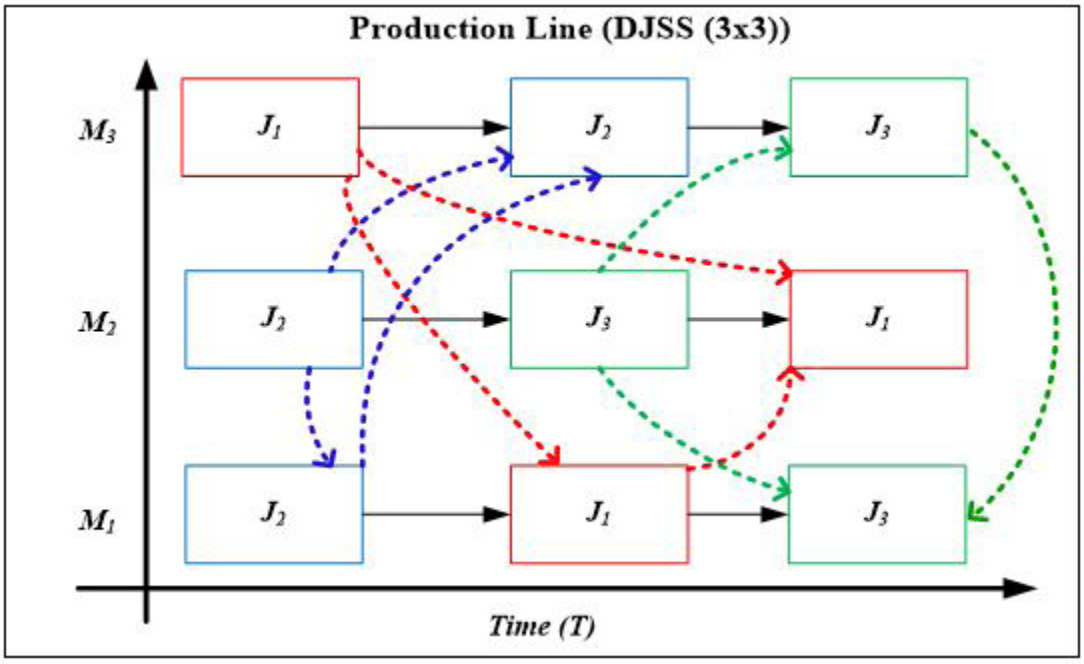

Figure 2 – Schematic model of the DJSS (3x3)

3.5 Dynamic Job Shop Scheduling Modeling

To represent the productive process and the scheduling of tasks in the production environment, a model based on the composition of a dynamic job shop system was developed with the following characteristics:

1. A shop ***S*** is composed of a set ***L*** of production lines;
2. A production line ***L*** is composed of an ***M***-set of machines;
3. Each machine M_i is represented by an individual ***i***;
4. A job ***j*** corresponds to a task, with ***j = 1..26*** tasks;
5. A relation ***W (a, p)*** denotes that activity ***a*** is the predecessor of ***p***;
6. A set of artifacts is denoted by ***A*** of ***j***, with ***A = {1..5}***;
7. An operation ***O*** is a quadruple denoted by ***O*** (M_i, A_j, ***t, int),*** where M_i is the machines of ***L*** by processing A_j initiated artifacts at a time ***t*** of an interval ***(int)***;

8. Each shop $\boldsymbol{S} \supset \boldsymbol{L}$ components processing $\boldsymbol{A}_j$ $\boldsymbol{Artifacts}_{tasks}$;
9. $\boldsymbol{D}_A = \{\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_n\}$, with $\boldsymbol{n = i..5}$, are the sets of artifacts produced and delivered in an $\boldsymbol{F}_f$, where $\boldsymbol{f}$ is one $\boldsymbol{F}$ of the PU phases;
10. Each $\boldsymbol{L}$ must produce and deliver a $\boldsymbol{DA}$ set of artifacts at the end of each PU phase;
11. A task $\boldsymbol{j}$ can be rescheduled until it is completed or until it reaches the expected quality, independent of the phase of the PU-Like.

From the definitions of the parameters and the variables that make up the dynamic job shop model, the problem can be modeled by the following objective function:

$$Max\ (D_A) = Max\ (Eval\ (A_j)),\ \forall\ j = 1..5 \tag{1}$$

Where: $\boldsymbol{Eval\ (A_j)}$ is the evaluation of the percentage $\boldsymbol{(j \rightarrow 0..100)}$ of the amount of Artifact type $\boldsymbol{(Aj)}$ delivered $\boldsymbol{(D_A)}$ in each of the phases $\boldsymbol{(F_f)}$ of the PU.

Subject to:

$$(A_i, A_j) \in W \rightarrow \forall\ t \in T \urcorner\ [(\exists\ (O\ (M_i, A_j, t, int)) \wedge \exists\ (O\ (M_k, A_j, t, int))] \tag{2}$$

$$t \in T \urcorner\ [(\exists\ (O\ (M_i, A_j, t, int)) \wedge \exists\ (O\ (M_i, A_k, t, int))] \tag{3}$$

$$T_{M_k} = \sum int_k < Tbud(k) \tag{4}$$

We assume that:

Equation (2): two machines $\boldsymbol{(M_1, M_2)}$ cannot work on the same activity of an artifact $\boldsymbol{(A_j)}$ at a given instant of time.

Equation (3): a machine cannot work on two distinct artifacts at any given instant of time.

Equation (4): The sum of the intervals $\boldsymbol{(int_k)}$ of time used for $\boldsymbol{M}_k$ are restricted to $\boldsymbol{Tbud\ (k)}$ of the time budget available.

3.6 DJSS: Problem Specification

The problem around DJSS is to identify one or more production lines in the shop with capacity constraints at run time. This is done by observing the duration (time/effort) applied to a scheduled-task on the machines of a production line.

A capacity constraint is any impeditive element that disrupts the productive process, causing an L to have a delivery $\boldsymbol{(D_A)}$ of less than 100% of the artifacts, given by the set of jobs $\boldsymbol{(j)}$ scheduled. Capacity-constrained resources limit the production of the $\boldsymbol{L}$, that when untreated imply a bottleneck in the production process.

A bottleneck is denoted by limitation of production of a machine caused by one or more capacity constraints that imply maximum production, defined by $\boldsymbol{D_A = \{(M_i A_j) = 100\%\}}$. In this specific model, the maximum production is given by the set of tasks processed in each of the phases according to the development based on the PU and according to the productive agenda, which delimits the term of each delivery and establishes the time budget for the shop.

The set of tasks scheduled and properly processed in all phases of the PU defines the total production of each machine and consequently the final product to be delivered.

3.7 DJSS: Simplest Case Specification

Figure 3 presents the simplest case of the job shop environment modeled for an ***L*** running with 2 machines ***L = {M1, M2}***, processing 2 tasks ***j***, scheduled at a given time ***T*** in a phase of the PU and according to the precedence relation ***W***.

Simplest case:

$$L = \{M_1, M_2\} \qquad T = T_{M_1} + T_{M_2}$$
$$W = \{(A_4, A_1), (A_3, A_1), (A_2, A_1), (A_5, A_2)\} \qquad F_f = \{1..4\}$$

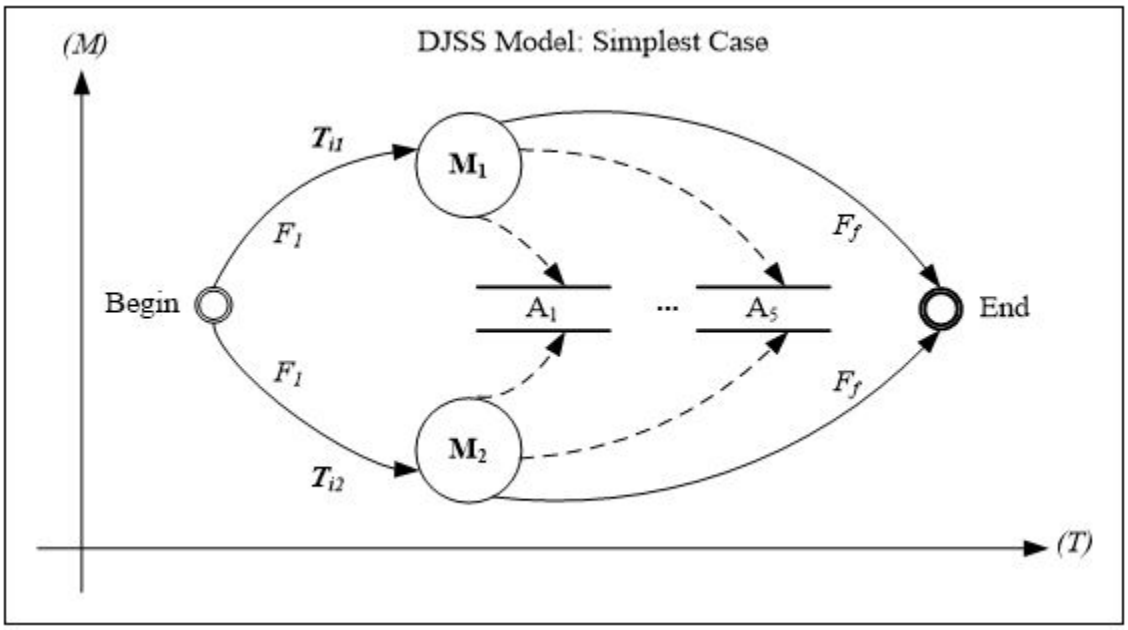

Figure 3 – Simplest case of DJSS model

The model of Figure 3 presents an simplest case of the DJSS in a shop composed of 2 machines ***(M_k)***, with ***k = 1***, ***2***, processing two tasks in a time interval $\boldsymbol{T_t}$. The sum of the intervals ***(int_k)*** of time for $\boldsymbol{M_k}$ is restricted by time budget ***Tbud(k)*** of time available for a given machine M_k. Dashed lines indicate a flow of partial deliveries per-formed by machines at run time.

3.8 DJSS: General case Specification

In the general case, ***L*** represents a production line of the shop and $\boldsymbol{M}_i$ are the machines of each production line, with ***(i ≥ 2 ≤ 4)***, and ***A = {A1, A2..A5}*** is the set of types of artifacts to be produced from a set ***j*** resulting from the execution of 26 tasks. Each ***L*** must produce and deliver $\boldsymbol{D}_A$ ***= 100%*** at the end of each round. This implies the optimal case.

Figure 4 is a representation of the general case of DJSS model proposed. To simplify and make the figure more intuitive, just as in the simplest case, the model represents an ***L*** with two machines ***M*** processing, a set of Artifacts ***A***, composed by $\boldsymbol{j}_n$ tasks, in the 4 phases of the PU, knowing that a Shop ***S*** is given by the set of production lines ***L.***

General case:

$$L = (M_i, A_j)$$
$$W = \{(A_4,A_1),(A_3,A_1),(A_2,A_1),(A_5,A_2)\}$$
$$T = T_{(M_1A_1} + T_{M_2A_1}(F_1) + T_{M_1A_2} + T_{M_2A_2}(F_2) + (\ldots) + T_{M_1A_n} + T_{M_2A_n}(F_4)$$
$$F_f = \{1..4\}$$

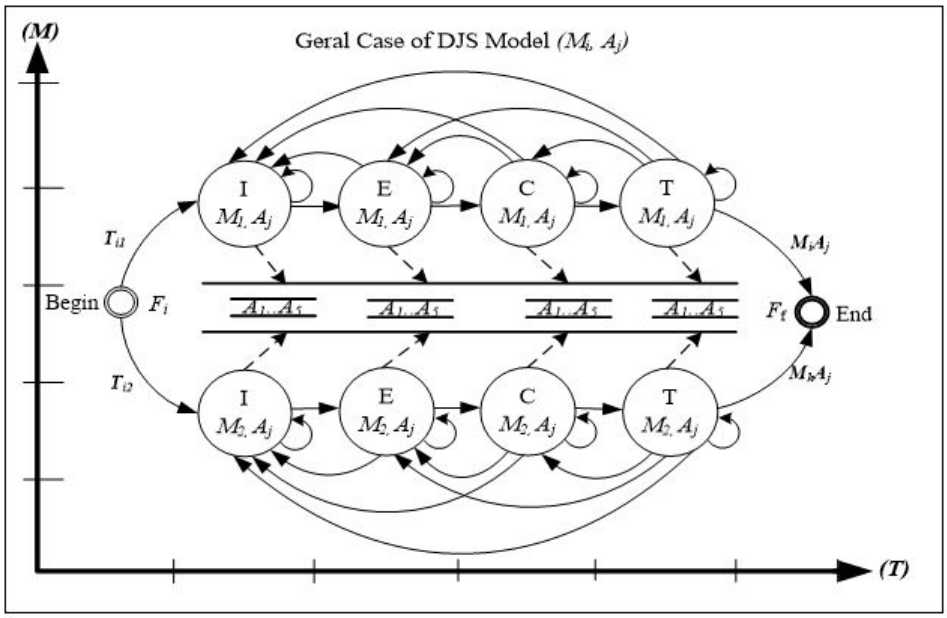

Figure 4 – General case of DJSS (Adapted from Ribeiro et al. 2017-b)

In this example, the shop contains a single L, composed of 2 machines $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$, processing $\boldsymbol{j}_n$ tasks at a time ***T***. The shop is instantiated with the tasks obeying the relation of precedence ***W***, where: *A1 = Architecture, A2 = Database, A3 = Coding, A4 = Documentation and A5 = Test*. ***S*** is feedback with the other tasks along the cycle and always on the basis of ***W***. The return flows indicate that the task has been rescheduled for reprocessing after the evaluation. This can occur in all phases of the PU (I = Inception, E = Elaboration, C = Construction and T = Transition). Dashed lines indicate a partial flow of deliveries according to PU-phases.

4. APPLICATIONS, ANALYSIS AND RESULTS

The scheduling scheme of tasks through a DJSS was developed in conjunction with the Theory of Constraints (TOC) (Goldratt e Cox, 1986) and SDP (PU) and was used to program the tasks in the shop according to the method TOC-DBR (Drum, Buffer and Rope).

The DJSS allows characterizing the productive environment by mapping and scheduling tasks throughout the development cycle and thus representing a manufacturing environment. In addition, it facilitates the allocation of tasks according to the order established by the disciplines and the PU-phases.

4.1 Application: Scheduling Tasks

Scheduling tasks in the shop is common to all production lines. The order of execution may vary depending on the development strategy adopted by the production line (development team).

The control of which machines will perform certain tasks, is also in charge of the production line, and a task can be scheduled on one machine and transferred to another machine, aiming at minimizing time consumption or maximizing quality. A task can also be scheduled by more than one machine at different times, and can also be rescheduled as often as needed to be completed according to construction requirements.

4.2 Applying the TOC: Subordinating and Synchronizing the Tasks in the Shop with the DBR

The Buffer is any contingency feature to protect the drum. It compensates for the variation of the process and makes the DBR Schedule very stable and immune to most problems (Woeppel, 2010) qtd. in (Ribeiro, 2018).

We assume that the Buffer is the set of tasks to be processed, plus the resources associated with them. Mainly the time resources of non-predecessor tasks, which have clearances or flexibility for shop schedule adjustments. In this way, these tasks are used to provide a resynchronization of the work through the subordination of the same to the machines with available resources. The Drum is the time available for each production line in the shop. The time budget is established by the timing (begin - end) of the processing of the tasks according to the PU-phases. The Rope is the synchronization mechanism of Shop production. It assists in scheduling by releasing tasks according to Drum rhythm and Buffer availability. We define the Rope in this research as the interactions and micro-interactions existing between the tasks and activities of the SDP, given by the PU and the order of precedence.

The Buffer has the effect of adding capacity in the production lines and eliminating the saturation of the scheduled tasks, thus allowing less robust scheduling in the shop and consequently smoothing the work of the lines. This is important in view of the dynamic characteristics of the shop

During the execution of the experiments, the tasks without restriction precedence and its resources have been used to adjust the schedule, allocating and “deallocating” them in the shop in a subordination process. The tasks release connected to the Drum depending on the availability of the Buffer. In this way the tasks are released in the same proportion as they are produced and delivered in a subordination and synchronization process, as can be observed in Figure 5: (a) (b) and (c).

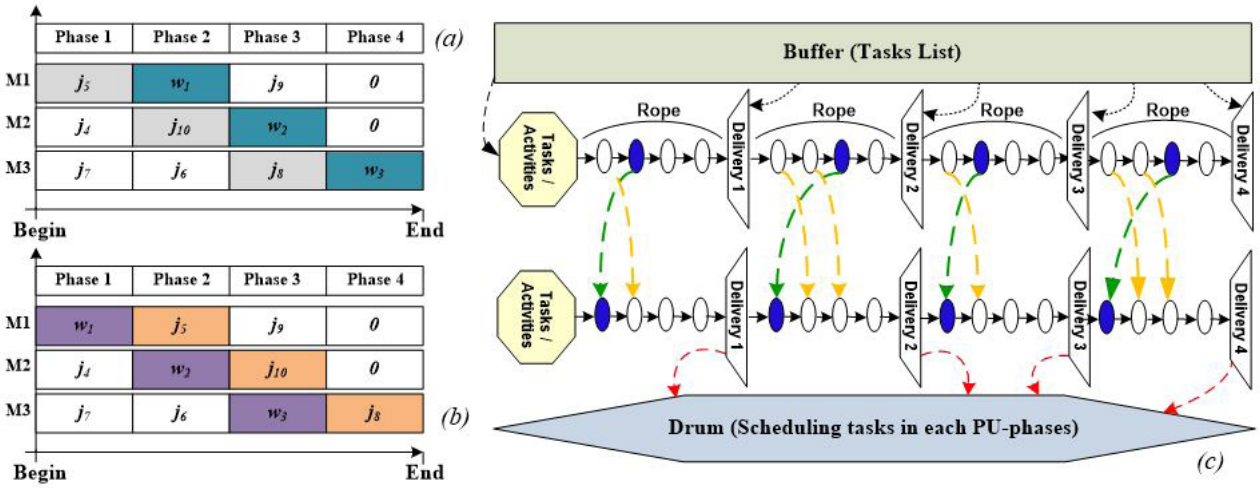

Figure 5 a, b, c – Scheduling, synchronization and subordination of tasks with TPC

Figure 5 (a) shows the scheduling of 10 jobs, where: w_1, w_2 e w_3 *are predecessor tasks scheduled on machine capacity constraints, at certain stages of the process. Machine 1 has capacity restriction with* $w_1 \rightarrow$ *phase 2*. The buffer releases task w_1 to phase 1 and task j_4 is reallocated in the next phase. The same happens with w_2 and w_3. Figure 5 (b) presents the new shop arrangement with the non-predecessor tasks allocated according to the capacity of the machines in their respective PU-phases. In Figure 5 (c), we have the DBR scheme with the change of the capacity-constrained predecessor tasks (blue ellipses) with non-predecessor tasks and without resource constraints (white ellipses). The green and yellow lines represent the process of allocation and relocation according to the release of the Buffer in the rhythm of the Drum, synchronized by the Rope, and according to the interactions of the PU.

4.3 Algorithm of Model: Preparation and Scheduling

We developed an algorithm divided into three parts; (1) Preparation and scheduling; (2) computation and parameterization, and (3) Data Analysis and evaluation. The first part of the algorithm, Figure 6, defines the initial steps to execute the process and involves the planning, the environment, the SDP, and the process of executing the experiment and the scheduling of the tasks, as well as the definition of the inputs and outputs. Due to the characteristics and layout of this text, only part 1 suppressed by the algorithm will be presented. For the complete reading of the algorithm, it is recommended to go to chapter 5 of (Ribeiro, 2018).

Algorithmic DJSS Model - Part 1: Preparation and Scheduling
Inputs: *Process (p), Tasks (j), Production Lines (L), Shop (s), PU-phases (f), Predecessor tasks (w).*
Output: *Scheduling Tasks.*

```
Begin
  Analysis and Evaluating Process(p); // Adjust of the process according to domain and environment
  Define Schedule (a);                // Define schedule and distribution of time.
  Prepare Shop(s);                    // Define team (LPs) and machines.
  Define production lines (L);        // Allocating machines on the L.
      L[Mi]=(⌊2⌋, ⌈4⌉);               // Attribution of machines to L (≥2, ≤4).
  PU-phases ← Time Budget;            // Programing phases in agree schedule.
  Define task (j) for each phase (f); // Define tasks of each PU-phase.
      W ← 0;  j ← 0;  f ← 0;          // Initialization of tasks w, j and PU-phase (f).
  For j ← 1 to 26                     // Routine of preparation to the scheduling.
    Define predecessor tasks (w);     // Mapping initial tasks (predecessors w).
    Mapping critical chain;           // Mapping of the critical chain.
    Do                                // Routine to scheduling task on the shop.
      For each L[Mi] on shop(s) do    // Routine to scheduling task on the shop by PU-phases.
        phase f([i]);                 // Identification of PU-phases.
        If phase (f) = 1 and task = w; // Scheduling Initialization (phase=1 of PU).
          Then Scheduling task (w) on shop (s);     // Scheduling tasks predecessors (w).
          Else Scheduling task (j) on shop (s);     // Scheduling task predecessors (j).
        End if;
      End For;
    While phase (f) = 4;
  End For;
End.
```

Figure 6: – Algorithm of model: preparation and scheduling

4.4 Analysis and Results

This section presents a descriptive analysis of the results obtained from the three indicators (Effort, Production, and Difficulty) used to evaluate the performance of the machines and consequently the production lines in Shops 2 and 3[2].

The difficulty measured is not a direct result of the scheduling of the tasks in the shop, and yes of a survey carried out with the machines of each production line, which were used to measure the degree of difficulty found in performing the tasks and to verify the impacts on the effort and production. For this reason, only we will show the general data about the difficulty of the production lines.

4.5 Descriptive Analysis

Firstly, we made an overview analysis of the productive results of the two cases (Shop 2 and 3). Next, three indicators were analyzed for each of the artifact types produced.

In order to map the performance of the production lines in each shop and to visualize a path to conduct the investigation to identify bottlenecks, the productive results were extracted for each indicator, where:

2 The shop 1 (experiment 1) was used for calibration of the process and scheduling model, for this reason, will not be displayed, once the data were not formally analyzed.

Effort: total time (expressed in minutes) consumed by the production lines to produce the software;

Production: measure of the percentage of the product constructed, delivered, evaluated, and accepted;

Difficulty: average value of the difficulty perceived by the machines for the production of the software, according to the items indicated in the survey.

Tables 3 and 4 presents the productive results and the descriptive statistics of shop 2 and shop 3.

Table 3: Productive results - Shop 2

Productive Results - Shop 2			
Production Lines (PL)	***Effort***	***Production***	***Difficulty***
PL1	17.590	58,33%	0,35%
PL2	8.718	76,08%	0,26%
PL3	7.560	61,11%	0,51%
PL4	17.889	80,56%	0,36%
PL5	8.386	59,26%	0,20%
PL7	19.233	62,96%	0,24%
PL8	15.222	76,36%	0,36%
PL9	13.583	80,52%	0,27%
Descriptive Statistic – *Shop* 2			
Statistic	***Effort***	***Production***	***Difficulty***
Mean	13.523	0,690	0,318
Median	14.403	0,698	0,310
Standard Deviation	4.723	0,105	0,097
Minimum	7.560	0,567	0,200
Maximum	19.233	0,807	0,510

Table 4: Productive results - Shop 3

Productive Results – Shop 3			
Production Lines (PL)	***Effort***	***Production***	***Difficulty***
PL1	16.759	87%	0,29%
PL2	9.128	53%	0,44%
PL4	16.069	90%	0,39%
PL5	21.236	78%	0,43%
Descriptive Statistic – *Shop* 3			
Statistic	***Effort***	***Production***	***Difficulty***
Mean	15.798	0,770	0,387
Median	16.414	0,822	0,410
Standard Deviation	5.001	0,167	0,068
Minimum	9.128	0,534	0,290
Maximum	21.236	0,904	0,440

The great variation of time consumption between the production lines exposes the dynamism of the SDP and evidence problems of the construct in the shops.

The logbook records showed that production lines with a high time consumption usually point to technical or behavioral problems, as presented [Ribeiro, 2017]. These problems (Technical Constraints (TC) and Behavioral Constraints (BC)), in most cases, are transformed into productive bottlenecks.

4.6 Effort by Type of Artifact (Shop 2 and Shop 3)

The effort applied by the PLs in the production of each type of artifact is interesting data because it allows visualizing which the artifacts where the PLs used more energy in the construction of the software.

Table 5 and 6 show the effort of the production lines of each shop (2 and 3) for each type of artifact along with its descriptive statistics.

Table 5 – Effort by type of artifact - Shop 2

Effort of Production Lines by Artifact – Shop 2					
Production Lines (PL)	***Architecture***	***Database***	***Coding***	***Documentation***	***Test***
PL1	1.359	1.760	4.958	7.336	2.177
PL2	441	884	2.836	3.140	1.417
PL3	56	684	3.626	3.121	73
PL4	920	1.247	9.989	5.124	609
PL5	175	2.048	5.082	773	308
PL7	1.125	4.033	7.459	6.011	605
PL8	1.449	2.965	8.049	2.519	240
PL9	1.459	1.033	3.432	2.677	4.982
Descriptive Statistic – Effort do *Shop* 2					
Statistic	***Architecture***	***Database***	***Coding***	***Documentation***	***Test***
Mean	873	1.832	5679	3.838	1.301
Median	1.021	1.504	5020	3.131	607
Standard Deviation	576	1.157	2.552	2.141	1.644
Minimum	56	684	2.836	773	73
Maximum	1.459	4.033	9.989	7.336	4.982

Table 6 – Effort by type of artifact - *Shop* 3

Effort of Production Lines by Artifact – Shop 3					
Production Lines (PL)	***Architecture***	***Database***	***Coding***	***Documentation***	***Test***
PL1	1.762	1.959	8.179	3.878	981
PL2	412	1.672	4.539	2.505	0
PL4	1.870	2.474	5.739	3.682	2.304
PL5	490	2.075	12.668	4.470	1.533
Descriptive Statistic – Effort do Shop 3					
Statistic	***Architecture***	***Database***	***Coding***	***Documentation***	***Test***
Mean	1.134	2.045	7.781	3.634	1.205
Median	1.126	2.017	6.959	3.780	1.257
Standard Deviation	790	332	3.593	824	969
Minimum	412	1.672	4539	2.505	0
Maximum	1.870	2.474	12.668	4.470	2.304

4.7 Production of the Lines by Type of Artifact (Shop 2 and Shop 3)

Production is an indicator that allows evaluating the performance of the production lines at run time (qualitatively) or (quantitatively) at the end of each phase of the PU. This is important because it can evidence the constraints or even the SDP-bottleneck even before the final delivery (concluded software).

Table 7 shows the total output of each artifact and the descriptive statistics for the software produced in Shop 2, while Table 8 shows the data corresponding to Shop 3.

Table 7 – Production of the lines by type of artifact - Shop 2

Production by Type of Artifact – Shop 2					
Production Lines (PL)	***Architecture***	***Database***	***Coding***	***Documentation***	***Test***
PL1	50%	88%	88%	63%	13%
PL2	75%	89%	75%	88%	50%
PL3	13%	75%	100%	100%	0%
PL4	100%	100%	100%	100%	13%
PL5	25%	100%	88%	63%	25%
PL7	63%	63%	38%	100%	38%
PL8	88%	100%	100%	100%	13%
PL9	63%	88%	88%	88%	75%
Descriptive Statistic – Production *Shop* 2					
Statistic	***Architecture***	***Database***	***Coding***	***Documentation***	***Test***
Mean	0,593	0,876	0,844	0,875	0,281
Median	0,625	0,880	0,875	0,939	0,187
Standard Deviation	0,297	0,134	0,209	0,164	0,248
Minimum	0,125	0,625	0,375	0,625	0,000
Maximum	1,000	1,000	1,000	1,000	0,751

Table 8 – Production of the lines by type of artifact - Shop 3

Production by Type of Artifact – Shop 2 – Shop 3					
Production Lines (PL)	***Architecture***	***Database***	***Coding***	***Documentation***	***Test***
PL1	88%	100%	100%	63%	100%
PL2	33%	75%	33%	50%	63%
PL4	88%	100%	75%	88%	100%
PL5	75%	88%	63%	75%	88%
Descriptive Statistic – Production *Shop* 3					
Statistic	***Architecture***	***Database***	***Coding***	***Documentation***	***Test***
Mean	0,708	0,906	0,677	0,688	0,875
Median	0,813	0,938	0,688	0,688	0,9375
Standard Deviation	0,257	0,120	0,277	0,161	0,177
Minimum	0,333	0,75	0,333	0,5	0,625
Maximum	0,875	1,00	1,00	0,875	1,00

The analysis of the percentage produced by the type of artifact allows evaluating the performance of PLs through the relationship (Effort X Production).

4.8 General Discussions

In this work, a simple and reliable measurement criterion was used to evaluate the production, as recommended by NBR/ISO/IEC [NBR/ISO/IEC 14598: 2001]. The measurement of the production was made from the values computed in each delivery/phase of the PU and punctuated in a ranking system as presented in [Ribeiro, 2018].

4.9 Correlation Between Indicators

The relationship between the indicators (Effort, Production, and Difficulty) was evaluated using the Spearman coefficients. For convenience, we will only present the correlation between the results totalized shown in Section 4.2.1.

Table 9 shows the correlation coefficients for both shop (2 and 3).

Tabela 9 – Correlation Matrix (Shops 2 e 3)

Correlation Matrix - Shop 2		
Indicadores	***Correlation Coefficient***	***P-value***
Effort X Production	0,214	0,619
Effort X Difficulty	-0,072	0,866
Production X Difficulty	0,204	0,629
Correlation Matrix - *Shop* 3		
Effort X Production	0,206	0,917
Effort X Difficulty	-0,415	0,750
Production X Difficulty	-0,802	0,333

The results of Shop 2 presented a low positive and statistically non-significant correlation between (Effort X Production). For (Effort X Difficulty) the results indicated a moderate negative and statistically non-significant correlation. Between (Production X Difficulty), the correlation was negative low and statistically non-significant. Concerning Shop 3, the results showed the following behavior: low positive correlation and statistically non-significant for (Effort X Production); moderate and statistically non-significant negative correlation between (Difficulty X Difficulty); and high and statistically significant negative correlation (Production X Difficulty).

In both cases, the statistical analysis showed that relationship effort and productivity have different weights and vary according to the performance of the machines in the production line. This implies that the use of greater effort does not necessarily indicate greater production. On the other hand, the difficulty demands more effort from the machines in the production line. But this also depends on the type of artifact being produced and the technical capacity of the machines.

4.10 Empirical Analysis

From the results presented in Section 4.2 (4.3.1 – 4.3.2) it was possible to analyze the behavior of PLs concerning Effort, Production, and Difficulty.

The empirical analysis of the data considers a percentage interval extracted from the maximum value obtained in each shop for Effort, Production, and Difficulty. Where: Low = (0 - 33%), Moderate = (34 - 66%) and High = (67 - 100%). This mapping is interesting because it facilitates the understanding of qualitative constraints and their impacts on quantitative results.

Another interesting fact about the general results obtained in the two shops is that greater effort does not always result in higher productivity. Regarding the difficulty, the two shops presented the following discrepancies: (1) a production line with great difficulty, applied a lot of effort, but produced little; (2) a production line with great difficulty and low effort and, consequently, produced little and (3) a production line with great difficulty, applied a lot of effort and produced satisfactorily.

5. CONSIDERATIONS AND FURTHER WORKS

Scheduling problems are difficult to implement due to its NP-hard nature, especially its dynamic version. On the other hand, they are important tools for programming and controlling any production environment. In this specific case, the main difficulty was to develop and apply a task scheduling model (DJSS) that would respond to variations and changes in the software development environment. However, this research has shown that this is possible and even facilitates the management of the process and of production as a whole, once the control and scheduling of the tasks in the environment return important numerical results to evaluate and measure the production and quality of the software product.

The conclusive analysis on the topic addressed in this thesis pointed to the Codification activity as the bottleneck of the Software Development Process, followed by the Documentation activity. The production of these artifacts required greater efforts of production lines and the productive results were not satisfactory in most cases (production lines).

5.1 Further Works

The continuity of this work is essential so that new contributions to the problems surrounding the software development environment and the bottlenecks in the SDP are produced. Therefore, it is intended to carry out new studies involving the topic addressed, since this research did not explore all points of the software development environment.

Thus, in the future work we will apply the DJSS model in two other environments: (1) In the software industry: replicating the experiment, with the same process and methodology, but with the use of professionals to represent the machines of the productive process; and (2) In an academic environment: In this second case, the study will be replicated with a significant change in software development methodology. The idea is to apply the model in an agile software development environment, exploring a new front beyond the structured model.

Another front that intends to explorer, in the next works is the application of heuristics and metaheuristics to better the process of scheduling, analysis, and measurement of data.

The replication of the method in distinct environments can open an opportunity for the creation of a repository of data or a benchmark to support new studies in the exploration of the SDP bottlenecks and with that contribute to the improvement of the software development process.

List of Symbols

L	Representation of Production line on the model.
M	Machine of an L.
J_j	Set of Job = J {1..26}.
$A_{1..5}$	Set of Artifacts.
D	Delivery of Artifacts.
T_t	Total Time, and t time unit accounted in minutes.
Tbud	Time Budge (Total time available to construct the software).
Eval	Evaluate of production (Effort and Production).
W(a, p)	Precedence Relation tasks between (a) artifact unit and (p) predecessor.
w	A predecessor tasks (w must be scheduled first).
F_f	UP-phases {I=Inception, E=Elaboration, C=Construction, T=Transition} or phase (f) 1, 2, 3 and 4 as used in this work.
k	Any constant of model.
S	Shop, denoted by a set of L.
i	Individual / Mi Processing a task/activity.
A	Set of Artifacts A= {1..5} (Documentation, Data Base, Architecture, Coding and test).
O	Operation eq. a machine Mi processing a task j in time Tt.
DA	Set of artifact delivered.
Max (DA)	Makespan (optimal results of a L).

Acknowledgements

The authors would like to thank the support provided by DCC – Department of Computer Science of the UFRJ – Federal University of Rio de Janeiro and everyone involved in this research.

Special thanks to students of Computer Science course of the discipline Fundamentals of Software Engineering (FES) of the Department of Computer Science (DCC) of the Institute of Computation (IC) of the Federal University of Rio de Janeiro (UFRJ), classes: 2014/2, 2015/1 and 2015/2, for accepting to participate in this research, for the commitment, for the effort, for the companionship and for the exchange of experience throughout this process.

Ethics Approval and Consent to Participate

The research was carried out in regular discipline of the computer science course and all participants agreed to participate in the development of the software. The research followed the ethical precepts for experimentation in software engineering proposed by SINGER , J., VINSON, N.G.: Ethical issues in empirical studies of software engineering. IEEE Transaction on Software Engineering; 2002.

Availability of Data and Materials

The data will not be shared at the moment due to the fact that other master's and doctoral researches are using the databases. However, if any researcher wants to have access to the data please contact the author and request access. Each case will be carefully analyzed.

Authors Contributions

Sildenir Alves Ribeiro: contributed to the development and conduct of research, monitoring of development teams, construction of scenarios, development of experimental essays, tabulation and analysis of data, development of method and algorithms; Eber Assis Schmitz: contributed to research guidance, data analysis and algorithm validation; Antonio Juarez S. M. de Alencar: contributed to the evaluation of materials and research methods; Monica Ferreira da Silva: contributed to the textual review, organization and standardization of the research.

REFERENCES

Araújo Jr., L. O.; Non-Deterministic Dynamic Job Shop Manufacturing Systems Programming Method; Doctoral Thesis; Mechanical Engineering Program; Polytechnic School; São Paulo University – USP; São Paulo – SP; 2006.

Basili, v. R.; weiss, d. M.; A Methodology for Collecting Valid Software Engineering Data; IEEE Transactions on Software Engineering; Vol. SE-10, No 6, pp 728-738; November-1984.

Blazewicz, j.; Ecker, k.; Schimidt g.; Weglarz, J.; Scheduling in Computer and Manufacturing Systems; Ed. Springer – Verlag; Berlin; 1996.

Casavant, t. L.; Kuhl, j. G.; A Taxonomy of Scheduling in General Purpose Distributed Computing Systems, IEEE Transactions on Software Engineering, Vol..14, no. 02; 1998.

Dai, M.; Tang, D.; Giret, A.; Salido, M.; A Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints; Robotics and Computer-Integrated Manufacturing, Volume 59, 2019, pp. 143-157. DOI: https://doi.org/10.1016/j.rcim.2019.04.006

Goldratt, e. M.; Cox, J.; The Goal: A Process of Ongoing Improvement; Ed. North River Press; Great Barrington, MA; 2014.

Hasgul, S.; Kartal, Z.; Analyzing A Drum-Buffer-Rope Scheduling System Executability Through Simulation; Proceedings in of the 2007 Summer Computer Simulation Conference (SCSC 2007), San Diego, California - US; 2007.

Jacobson, I.; Booch, G.; Rumbaugh, J. The Unified Software Development Process. Reading: Addison-Wesley, 1999(b).

Klein, R.; Scheduling of Resource Constrained Projects; Ed. Kluwer Academic Publisher; Norwell – Massachu-setts, 2000.

Mansouri, S. A.; Golmohammadi, d.; Miller, J.; The moderating role of master production scheduling method on throughput in job shop systems; International Journal of Production Economics, Volume 216, 2019, pp. 67-80; Elsevier. DOI: https://doi.org/10.1016/j.ijpe.2019.04.018

NBR/ISO/IEC 14598:2001; Software engineering - Product evaluation - Part 6: Documentation of evaluation modules; NBR/ISO/IEC; 2001; Reviewed and Confirmed in 2008.

Ouelhadj, D.; Petrovic, S.; A survey of dynamic scheduling in manufacturing systems, Journal of Scheduling, Vol. 12; 2009.

Özcan, U.; Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times; International Journal of Production Economics, Volume 213, 2019, pp. 81-96; Elsevier; DOI: https://doi.org/10.1016/j.ijpe.2019.02.023

Pinedo, M.; Scheduling – Theory, Algorithms and Systems; 5th Edition; Ed. Springer; New York-NY; 2016.

Pressman, R. S.; Maxin, B. R.; Software Engineering: A Practitioner's Approach; Ed., McGraw-Hill/Makron Books do Brasil, São Paulo, 2014.

Ribeiro, S. A.; Bottleneck Identification in Software Development Processes: A Proposal Based on the Principles of the Theory of Constraints; Doctoral Thesis; IM/PPGI/UFRJ; Rio de Janeiro-RJ; 2018.

Ribeiro, S. A.; Schmitz, E. A.; Alencar, A. J. S.; Silva, M. F; Bottlenecks Identification in Software Development Process: A Quali-Quantitative Approach; Proceedings of the XVI Brazilian Symposium on Human Factors in Computational Systems (IHC 2017); ACM Digital Library; ACM New York – NY-USA; DOI:10.1145/3160504.3160513; October 2017-a.

Ribeiro, S. A.; Schmitz, E. A.; Alencar, A. J. S.; Silva, M. F; Literature Review on the Theory of Constraints Applied in the Software Development Process; IEEE Latin America Transactions, vol. 16, no. 11, pp. 2447-2756; November 2018. DOI: 10.1109/TLA.2018.8795116;

Ribeiro, S. A.; Schmitz, E. A.; Alencar, A. J. S.; Silva, M. F; Research Opportunities on the Application of the Theory of Constraints to Software Process Development; Journal of Software vol. 12, no. 4, pp. 227-239, 2017. DOI: 10.17706/jsw.12.4.227-239; April 2017-b

Ribeiro, S. A.; Schmitz, E. A.; Alencar, A. J. S.; Silva, M. F; Scheduling of Tasks in a Software Development Environment Using a Dynamic Job Shop Scheduling (DJSS). Proceedings of the XLIX Brazilian Symposium of Operational Research (SBPO); Blumenau – SC; 2017. Anais SBPO/SOBRAPO – ISSN 1518-1731; August 2017-c.

Ribeiro; S. A.; Alvarenga, A. G.; Ahonen, H. T.; Artificial Immune System to Job Shop Scheduling Problem; Proceedings in Brazilian Production Engineering Symposium; XIX SIMPEP; 2012.

Ribeiro; S. A.; Artificial Immune System to Job Shop Scheduling Problem; Master Thesis; PPGI/IT; Federal University of Espírito Santo (UFES) – ES; 2006.

Schach, S. R.; Object-Oriented and Classical Software Engineering, 8th. Edition; Published by McGraw-Hill Press; New York – NY; 2011.

Singer, J., Vinson, N.G.; Ethical issues in empirical studies of software engineering; IEEE Transaction on Software Engineering; 2002.

Sommerville, I.; Software Engineering; 9th Edition; Ed. Addison-Wesley, Londres, 2011.

Sousa, R. A.; Varela, M. L. R.; Alves, C.; Machado, J.; Job shop schedules analysis in the context of industry 4.0; International Conference on Engineering, Technology and Innovation (ICE/ITMC)- 2017; pp-711-717; Funchal Portugal; Publisher: IEEE – IEEE Xplorer; 2018 Doi: 10.1109/ICE.2017.8279955

Thörnblad, K.; Mathematical Optimization in Flexible Job Shop Scheduling: Modelling, Analysis, and Case Studies; PhD Thesis; Department of Mathematical Sciences - Division of Mathematics; Chalmers University of Technology and University of Gothenburg; Göteborg; Sweden – 2013;

Turker, A. K.; AKTEPE, A.; INAL, A. F.; ERSOZ, O. O.; DAS, G. S.; BIRGOREN, B.; A Decision Support System for Dynamic Job-Shop Scheduling Using Real-Time Data with Simulation; pp. 01-19; Volume 7; Issue 3; Math-ematics; 2019. DOI: 10.3390/math7030278.

Wohlin, C.; "Empirical Software Engineering: Teaching Methods and Conducting Studies", in Empirical Soft-ware Engineering - Dagstuhl Seminar Proceedings (LNCS 4336), pp. 135-142, edited by V. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl and R. Selby, Springer Verlag; 2007.

Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A.; Experimentation in Software Engineering; Ed. Springer/Springer Science e Business; London-UK, 2012

Zhang, J.; Ding, G.; Zou, Y.; Qin, S.; Fu, J.; Review of job shop scheduling research and its new perspectives un-der Industry 4.0; Journal of Intelligent Manufacturing; Volume 30, Issue 4, pp 1809-1830; Springer; 2017; DOI: https://doi.org/10.1007/s10845-017-1350-2 Environments, R. A. Earnshaw and J. A. Vince, England, Academic Press Ltd., p. 449-460.