REBECA GOMES DE HOLLANDA CAVALCANTI

ANALYSIS AND IMPROVEMENT OF A STEEL PALLET MANUFACTURING PLANT USING SIMULATION

São Paulo

2016

REBECA GOMES DE HOLLANDA CAVALCANTI

ANALYSIS AND IMPROVEMENT OF A STEEL PALLET MANUFACTURING PLANT USING SIMULATION

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para a obtenção do Diploma de Engenheira de Produção

São Paulo

2016

REBECA GOMES DE HOLLANDA CAVALCANTI

ANALYSIS AND IMPROVEMENT OF A STEEL PALLET MANUFACTURING PLANT USING SIMULATION

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para a obtenção do Diploma de Engenheira de Produção

Orientador:

Prof°. Dr. Marco Aurélio de Mesquita

São Paulo

2016

FICHA CATALOGRÁFICA

Cavalcanti, Rebeca
ANALYSIS AND IMPROVEMENT OF A STEEL PALLET MANUFACTURING PLANT USING SIMULATION / R. Cavalcanti -- São Paulo, 2016.
121 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Produção.

1.SIMULAÇÃO DE SISTEMAS 2.CONTROLE DA PRODUÇÃO I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Produção II.t.

AKNOWLEDGEMENTS

First, I would like to thank my Family, who always supported me throughout my academic journey. Their love and care kept me going even on the hardest moments. I thank them for backing me up when needed, and for always nourishing my ambitions. They are and will always be my rock.

I also thank the university as an institution, but especially those who form it. During my years at USP I had the opportunity to meet the most amazing and inspiring people. Among them, my dear friends: Caio Giordano, Gabriel Francisco, Rodolfo Gandara, Priscilla Carneiro and Arthur Pires, whom I thank for always making me laugh, for their companionship, and for all our lunch dates (which will be missed). They will be forever in my heart.

Last but not least, my sincere acknowledgements to Prof. Marco A. de Mesquita for his patience and fruitful comments which helped enormously during the production of this thesis. Along with him I thank all my professors, the eternal mentors of my class. I thank them for their dedication, and for opening minds and touching hearts.

RESUMO

Empresas de manufatura precisam constantemente melhorar suas operações a fim de manter sua vantagem competitiva. Sendo assim, um ambiente de produção bem concebido, juntamente com um sistema de controle de processos eficaz é essencial para uma produção eficiente. Esta tese aborda conjuntamente ambas as questões, propondo uma reformulação no balanceamento de linha e um sistema de controle baseado na limitação de WIP que segue uma lógica CONWIP para um fabricante de paletes metálicos.

Atualmente a empresa apresenta um design de processo ineficiente que leva a tempos de ciclo maiores do que o *takt time* necessário. Para resolver o problema diferentes cenários de balanceamento são propostos e comparados. Realiza-se uma análise mais holística já que a continuidade do trabalho e a viabilidade de implantação são considerados. A melhor das soluções propostas resolve o problema dos tempos e melhora os fluxos internos.

Em um segundo momento é proposto um sistema de controle de estoque em processo que segue uma lógica CONWIP. As alterações apresentadas foram calibradas e testadas usando simulação de eventos discretos. Resultados mostram que, se implementadas, as mudanças poderiam aumentar a produção em quase 30%, além de reduzir significativamente o tempo médio de ciclo e os níveis de WIP.

Palavras-chave: Simulação de sistemas de produção. Controle de estoque em processo. CONWIP. Balanceamento de linha.

ABSTRACT

Manufacturing companies constantly need to improve their operations in order to maintain their competitive advantage. Therefore, a well-designed production environment along with an effective shop floor control system is essential for an efficient production process. This thesis addresses jointly both issues, proposing a redesign in process balance and WIP cap control system that follows a CONWIP logic for a steel pallet manufacturer.

Currently the company presents an inefficient process design which leads to cycle times larger than the required takt time. To address the problem different balancing scenarios are proposed and compared. Here a more holistic analysis is made once work continuity and feasibility are considered. The best solution proposition resolves the problem and improves internal flows.

On a second moment, a workload control system that follows a CONWIP logic is proposed. The presented changes were calibrated and tested using discrete event simulation. Results show that if implemented, the propositions could improve output by almost 30%, the levels of average cycle time and WIP would also be lowered significantly.

Keywords: System simulation. Shop floor control. CONWIP. Line balance.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVIATIONS AND ACRONYMS

ALB:	Assembly line balance
CONWIP:	Constant work in process
CT:	Cycle time
GALBP:	Generalized assembly line balance problem
JIT:	Jut-in-time
MRP:	Material requirement planning
MT:	Manufacturing task
SALBP:	Simple assembly line balance problem
SI:	Smoothness index
WIP:	Work in process

CONTENTS

1. INTRODUCTION

Manufacturing companies constantly need to improve their operations in order to maintain their competitive advantage. Under this scenario an efficient, effective and economical operation in a manufacturing unit of an organization is essential, especially for small companies, for which production represents a high percentage of their cost structure.

This paper presents a case study of a small metallurgical company in Poland which produces metallic logistic equipment. Their main clients are automotive industries from central and Eastern Europe. The company revealed many improvement opportunities, once different sorts of inefficiencies were identified. This thesis proposes a re-balance of the line, as well as a model for workload control. To address the presented issues, the project focuses on the production of a family of products, the steel pallets for auto part transportation, which is produced under a make to order strategy.

It is important to mention that the data used here was provided by Dr. Dorota Stadnicka, from the Politechnika Rzeszowska University. The author of this paper worked in partnership with her and Dr. Dario Antonelli from Politecnico di Torino to perform a dimensioning of workforce in the factory. The project originated an article which was submitted to publication analysis of Elsevier.

Due to geographical difficulties, this paper will follow a more theoretical approach, since the implementation of the solutions is unable. Gathering inputs was also a challenge, thus in some pointed out cases assumptions were made.

This introductory chapter presents an overview of the company, the motivation of the paper, the problem definition and objectives of the performed study.

1.1. The metallurgical company

The company, which from now on will be identified as Pallet Factory, is a small producer of different kinds of metallic goods such as containers, bins, trays, pallets etc, mostly dedicated to transportation and storage solutions. The firm is situated in Poland and its main clients are automotive industries, especially from Eastern Europe.

The plant nowadays counts with 25 workers, divided on the basis of product line. 14 workers are dedicated to boxes and cages, meanwhile the other 11 produce pallets.

The firm, globally speaking, has aspects that can be improved, however, this paper will emphasize the local optimization of the pallet production.

The Pallet Factory presents many inefficiencies, especially related to layout and line balance. Also, production planning improvements could be opportunities to diminish costs related to operations. The lack of formal structure or optimization logic behind the company's processes highlights the importance of improving and formalizing a production planning aiming cost reductions without affecting service levels.

1.2. Motivation and problem definition

This graduation thesis is motivated by the positive results its proposition may create to the metallurgical company, especially regarding one of their product lines. Since the company started as a small family business little to no effort on organizing the production process existed. Once the company grew those aspects began to make a difference on terms of efficiency and consequently cost.

As literature suggests, a formal and systematic production planning process can lead companies to achieve high efficiency and effectiveness levels (HOPP & SPEARMAN, 1997). Under that scenario, this thesis proposes experiments for changing the current production process and procedures. Those changes proposals are later tested using simulation. This tool arises as an important instrument allowing validation without incurring on costs.

Posterior sections will present, with a wide scope, the current state of the company. Though that analysis different issues and optimization opportunities are identified, each one with a specific best solution. However, given the time scope, this paper will focus solely on a few of them. Firstly, the production process is highly unbalanced, with workstations that have processing times higher than the required cycle time to satisfy the current average demand. A second issue here addressed regards the production control system. This paper proposes a scheme of workload control targeting improvements on the overall performance of the plant by best controlling the levels of intermediary inventory.

1.3. Objectives

The goal of this paper is to propose and test a CONWIP logic as a basis for a shop floor control in the Pallet Factory. But before addressing this issue, the balance of the plan must be improved. Hopp & Spearman (2008) state that to reasonably manage the implications of production control, it is important to consider a design standpoint, therefore, the balance issue will be addressed prior to the control decision itself. This could significantly decrease cycle time and improve work distribution among stations, providing a more stable environment to work upon.

In a second moment, the CONWIP introduction will be discussed, the Pallet Factory context will be simulated aiming to compare different control experiments in order to test which would be the most effective. This last approach purposes the improvement of performance by generating a leaner production system.

1.4. Relevance of the project

This paper intends to create a design with improved efficiency for the Pallet Factory through enlightening feasible changes regarding line balance and production control. Here the importance of simulation is highlighted, since it provides testing opportunities despite physical limitations that might be present.

This thesis proposes theoretical changes for the company to improve its overall performance by controlling its inventory levels. It is interesting to notice that two proposed approaches concern different levels of hierarchical production planning. Usually literature focuses on one single issue. Therefore, this paper may be useful to show positive interactions when control and balance flow are assessed together.

Due to geographical and communication difficulties, data gathering founded obstacles. Likewise, implementation was unable. Therefore, this project remains on a propositional level.

1.5. Outline of the thesis

This first chapter presented a brief introduction to the context, motivation and objectives assessed in this paper.

Chapter 2 contains the literature review of the main topics used in the assessment of the Pallet Factory re-design and production planning, focusing especially on line balancing, and production planning and control, more specifically the different workload control techniques.

Chapter 3 comprises an explanation of the product and the production process, as well as the diagnosis of its main problems. Based on the identified challenges, the later chapter will present a re-design proposition for the plant through the comparison of different line balance alternatives. Simulation is used to check if the proposed balance provides real improvements, and test its sensibility to random variation in processing times.

Once the redefinition of the line balance is set, chapter 5 will deal with the operational aspects, proposing a model to control order release according to WIP levels through the use of two propositions of CONWIP production system. Once again, simulation is used to test outputs for the proposed model.

Finally, chapter 6 contains the project conclusion, presenting its synthesis, the main results obtained and possibilities of future extension works.

2. LITERATURE REVIEW

This chapter contains a brief study of the existing literature regarding the topics applied on this paper.

The literature review focused on the rationale behind different aspects of production planning in order to set a framework suitable to identifying the company’s needs and opportunities for improvement regarding inventory control. Firstly, the paper focuses on line balancing. Then it enters more operational aspects, exploring different types of production systems logics. Those topics were selected because they turned out to be the most important improvement opportunities for the company. Inside each section, a special attention is given to the role of simulation in problem solving.

2.1. Assembly line balancing

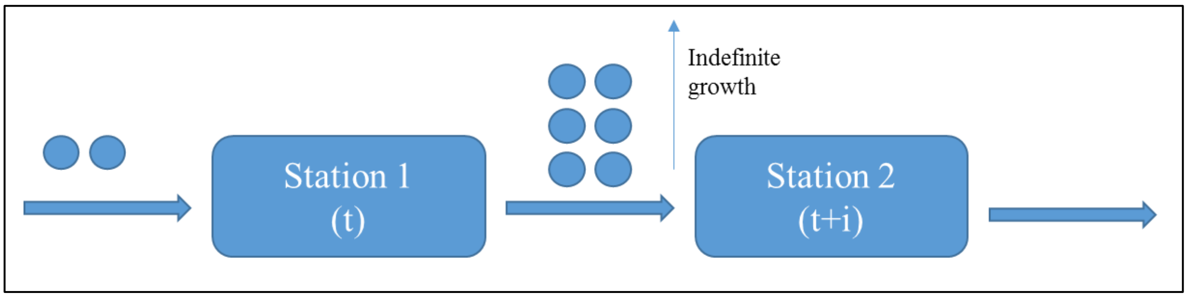
An assembly line is defined as an arrangement of machines, tools and workers in which a product is assembled by having each perform a specific, successive operation on a unit as it passes by in a series of stages (GHOSH & GAGNON, 1989)

The industrial revolution was the historical context for the development of production lines. Before that, goods were crafted, and no method nor standardization were employed. From that moment on processes evolved more and more, and principals such as the minimization of movements and times were introduced.

Assembly lines are still a crucial problem to get improvements on a production system. Under that assumption many studies have been performed approaching different aspects and with different final objectives. Aspects, such as task integration concept, or layout arrangement are common disruptions that affect production systems and decrease the line efficiency.

Here one important aspect that comes to light is the control of inventory in process, or work in process (WIP). The matter may be either a question of balancing the flows inside the plant to contain the indefinite increase of stock before the bottleneck station, or, it may be also a matter of production planning and control. This study will focus on the WIP control under both optics.

Figure 1: Line balance and the growth of WIP



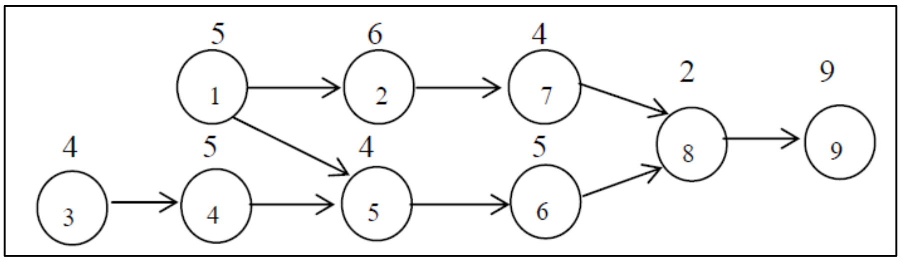
Source: Created by the author

Line balancing is a technique to solve or minimize problems occurred in assembly lines by trying to diminish unbalances between workstations and workload in order to get higher efficiency.

Assembly line balance (ALB) is a classic Operations Research optimization problem, it has been widely studied, and many algorithms have been proposed for the problem. Yet, despite the practical importance of the problem, little commercially available software is available to help the industry in optimizing their lines. Becker and Scholl (2006) show that there appear to be currently just two commercially available tools which have both a state of the art optimization algorithm and a user-friendly interface. Furthermore, one of those packages appears to handle only the “clean” formulation of the problem (Simple Assembly Line Balancing Problem, or SALBP). This situation appears to be controversial due to the amount of benefits and cost savings a well-balanced line can bring.

Assembly lines are one of the most common production technologies. Designed for the sequential organization of workers, workstations and materials. To balance an assembly line is to group work elements and assign them to stations in a systematic way, respecting applicable restrictions. Becker and Scholl (2006) defined the assembly line balance problem as splitting the total amount of work into tasks which have deterministic or stochastic cycle times (*tj*) depending on each task *j*, and allocating them to stations. Each task requires machines, workers or skills that may be task specific. Furthermore, there needs to be an optimal balance of task distribution between stations with respect to a final objective – which can be economic, technical, etc. – respecting constraints such as the precedence order of tasks. An example of a precedence order diagram is given in Figure 2.

Figure 2: Example of precedence diagram



Source: Adapted from Becker & Scholl (2006)

Another fundamental variable in order to perform a line balance is the takt time. It is the rate at which consumers buy products, thus it is equal to the rate at which those should be manufactured. Another nomenclature used is required cycle time, meaning the minimum amount of time

Davis (2001) describes steps to perform an ABL:

1. Specify the precedence relations between tasks;
2. Determine the needed cycle time
3. Determine the theoretical number of stations needed
4. Select primary and secondary allocation rules
5. Delegate tasks, one at a time, to pertinent station until the sum of all individual task times in the station is equal to the cycle time. Repeat in all stations
6. Use indicators to analyze line's performance.

Ong & Twentiarani (2012) mention different methods to solve a line balancing problem. Among them:

- Heigesson-Birnie Method,
- Moodie-Young Method,
- Immediate Update First-Fit heuristic,
- Rank-and-Assign heuristic.

Heigesson-Birnie method is commonly known as rank positional weight. The steps to perform such technique are:

1. Determine sequence of individual work elements.
2. Create a precedence matrix to show the relationship among those elements.
3. Calculating the positional weight of each element.

4. Listing all the positional weights in decreasing order of magnitude.
5. Assign the work elements into various workstations.

Moodie-Young's method follows a similar logic to Davis' iteration, however considering as a tie breaker the cycle time of tasks, instead of the number of successors.

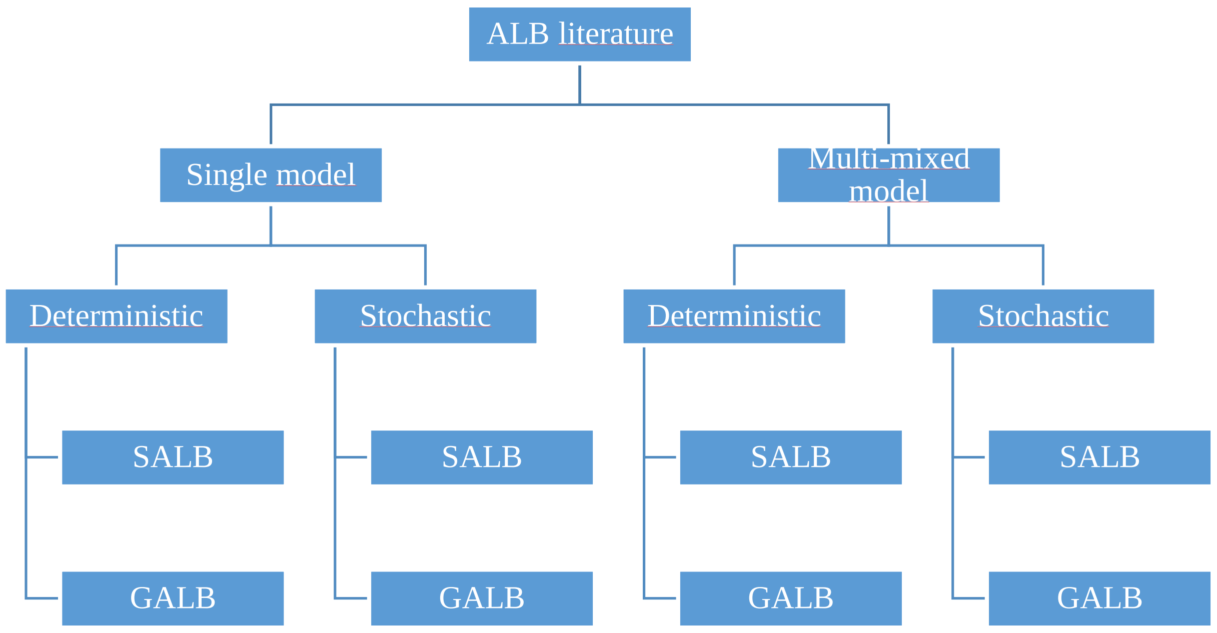
For Immediate Update First-Fit Heuristic, a fictional "cost function" is created. The author refers to it as a numerical score function n(x), which punctuate combinations in order to get a more efficient distribution. The steps for this method are the following:

1. Assign a numerical score n(x) to each task x.
2. Update the set of available tasks (tasks that don't have immediate predecessors or those which the predecessors have been assigned).
3. Among the available tasks, assign the task with the highest numerical score to the first station in which the capacity and precedence constraints will not be violated.
4. Repeat steps 2-3 until all tasks have been assigned to workstations.

Alternatively, to the approach above, an inverse relation can also be applied. Meaning that instead of allocating the task with highest score, if n(x) represents a cost function, the smaller value must be assigned.

The ALB problems can include an ample range of scopes. During the last decades, the number of problems related to ALB has increased considerably, different approaches that look more like the reality of industries were required so experts have been studying all types of models, creating branches and diversifying objectives. Since the range of problems is so wide, some kind of classification is needed, one of them was presented by Ghosh & Gagnon (1989):

Figure 3: Classification of ALB.



Source: Adapted from Ghosh & Gagnon (1989)

The simple assembly line balancing problem (SALB) is relevant for straight single product assembly lines where only precedence constraints between tasks are to be considered. Any problem that generalizes or removes assumptions from the SALB is considered a generalized line balancing problem (GALB). This classification embraces the most types of real case scenarios, including assignment restrictions, workers or equipment efficiency, etc.

Based on the above classification, Kriengkorakot & Pianthong (2007) summarized Ghosh & Gagnon's findings regarding which are the most common objectives criteria used in ALB problems. Their conclusions are shown in Figure 4.

Figure 4: ALB Objective Criteria

Type of objective criteria (order of frequency of use)
1. Technical
- Min number of workstations given a cycle time
- Min the cycle time given the number of stations
- Min the total idle time
- Min the balance delay
- Min the overall facility or line length
- Min the throughput time
- Min the probability that one or more stations will exceed the cycle time
2. Economic
- Min total cost of labor
- Min labor cost/unit
- Min total penalty costs for a number of inefficiencies
- Min the inventory, set-up or idle time cost
- Min total cost of WIP
- Max the net profit

Source: Adapted from Ghosh & Gagnon, (1989).

Exact optimization methods such as linear, integer or goal programming can be applied to solve an ALB problem. However, when the complexity of the system increases the feasibility of such solutions becomes limited. Assembly balance problems are considered NP-Hard, for that reason, heuristics are commonly applied to solve in less time the problem. Although heuristic methods do not necessarily find the optimal solution, they present simplicity, close to optimal answers and are a popular alternative due to the possibility of comparison among scenarios (SOROUSH; SAJADI & ARABZAD, 2014)

Another point that justifies the gap between literature and practice is that most mathematical approaches used to model ALB problems tend to oversimplify real scenarios to enable their solutions given the existing tools. Therefore, many aspects of the real life environment of the industries are ignored.

Despite the existing abysm in between theory and practice, a few studies have tried to employ operations research and simulation to solve practical problems. Regarding the need of workers in the line, for example, in literature there is a problem known as “assembly line worker assignment and balancing problem” (ALWABL) (BLUM & MIRALLES, 2010). In this formulation, tasks must be assigned to workers who have to be assigned to stations. Here, processing times are worker specific, some take longer than others to perform a task, and there may be cases in which a worker cannot perform a certain task. The problem was first introduced by Miralles *et al.* (2007), who proposed the inclusion of assembly lines in shelters for disable people. He solved the problem by setting the main objective as the reduction of total cycle time, respecting the different skills of the workers. Instead of allocating tasks to workstations as the normal ABL problems, he introduced an extra step by allocating tasks to workers and these to stations.

The problem was further explored in 2009, when Costa & Miralles included job rotation in their analysis. The problem turned out to be a two-phase optimization. The first phase was already done in 2007, and the second one maximized the number of jobs performed during a cycle rotation. For the latter, the authors created a total rotation time T subdivided in rotation sub periods t. The constraints guaranteed that the first phase constraints were respected at each sub period, and that the new final cycle time had to be smaller or equal to the one obtained in the first phase. Their results seemed to be successful, as they were able to propose a job rotation scenario without affecting the overall productivity.

Later, Blum & Miralles (2010) introduced another iterative two phased algorithm based on beam search (tree search method) in which the first phase consisted in finding the lower boundary for the cycle time, which is the smallest feasible one and the second phase considered the allocation of tasks to workers and is focused on optimizing the cycle time.

A second type of worker allocation problem considers as an objective the minimization of the total number of operators. Such a situation was studied by Corominas Pastor & Plums (2006) who presented a motorcycle firm which had to hire temporary employees due to the increase in sales during the summer season. He considered as objective function the minimization of the number of operators, subject to the constraint that unskilled workers had to work along with skilled ones, and the precedence relations of the assembly process had to be respected. Here, once again the processing times were worker dependent since an unskilled operator takes more time to perform a task than a skilled one.

2.2. Push and pull production systems

Since the introduction of JIT the concepts of pull and lean have become more popular. However, as Hopp and Spearman (2004) point out in their article "To pull or not to pull: What is the question?" much confusion exists regarding the real meaning of each control strategy. The authors define the concepts based strictly on WIP control. They state: *"A pull production system, is one that explicitly limits the amount of work in process that can be in the system. By default, this implies that a push system is one that has no explicit limit on the amount of work in process that can be in the system"* (HOPP & SPEARMAN, 2004, p.142).

However, it is important to mention that there are no pure push or pull systems, since in practice, blocking occurs naturally due to physical space limitation, for example. The point is that pull systems help diminishing variability in the system by introducing negative correlation between the WIP levels at different stations. Meanwhile, although possibly restricted, WIP levels are independent in push systems (HOPP & SPEARMAN, 2008).

The next sections explain the particularities of each control system, focusing specifically in two types of pull systems, Kanban and CONWIP. Inside each section, special attention is given to the importance of simulation in shop floor control.

2.2.1. Push production systems and MRP

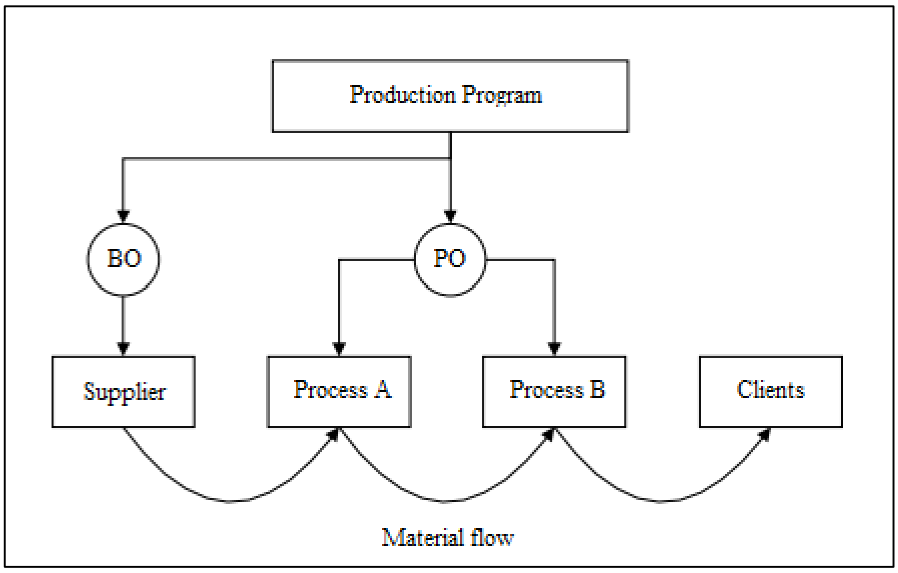
There are different models of production, broadly we can classify them into push or pull systems. The former is characterized by a production controlled by the fulfillment of delivery orders, the MRP and MRPII are considered push systems (MESQUITA & GOLDEMBERG, 2006). *Kanban*/Just in time, Drum-Buffer-Rope and Constant WIP (CONWIP) are different examples of pull production systems.

Defined by Spearman, Woodruff & Hopp (1990) as systems in which production jobs are scheduled, a push system starts its production before an actual demand order. A MRP system (Material Requirement Planning) controls what should be produced and when, and one production unit singly performs an order and pushes it to the next unit. This model appeared in the beginning of the industrial revolution, when quality was not an issue, demand was high and the effects of competition were barely felt.

The MRP system was introduced by Joseph Orlick in 1975, and during the 80s it became the most important production control system, later evolving to the MRPII that increased the scope out of the production area, embracing, as well, marketing and finance.

In the MRP system the demand forecast orients the Production Program which controls and liberates the production orders (PO) and buying orders (BO) as represented on Figure 5 (TUBINO, 2007)

Figure 5: Scheme for MRP system



Source: Adapted from Fernandes (2013).

The inputs for the MRP are the demand forecast, the inventory registers, and the bill of materials. As an output, the program produces buying orders, production orders, and materials plan. The parameters necessary for the implementation of a MRP are the lead times, the security levels for inventories, and the size of batches.

The definition of the batches size is affected by the set-up times, inventory costs and idleness of stations.

Each process in the MRP was compared to a black box by Graves (1988). At a given time t a job will enter the box and at $t + l$ it will come out, l being the lead time. Since most processes have random lead times, l is normally a pessimistic value, so, theoretically, most jobs should finish earlier, resulting in high levels of inventory.

Finally, the objective of the safety stock is to reduce uncertainties related to suppliers lead time and demand, which have a probabilistic nature, therefor, are subjected to variations (FERNANDES, 2013)

The push production has its advantages and disadvantages, explicitly:

- Higher centralization of information under the PPC

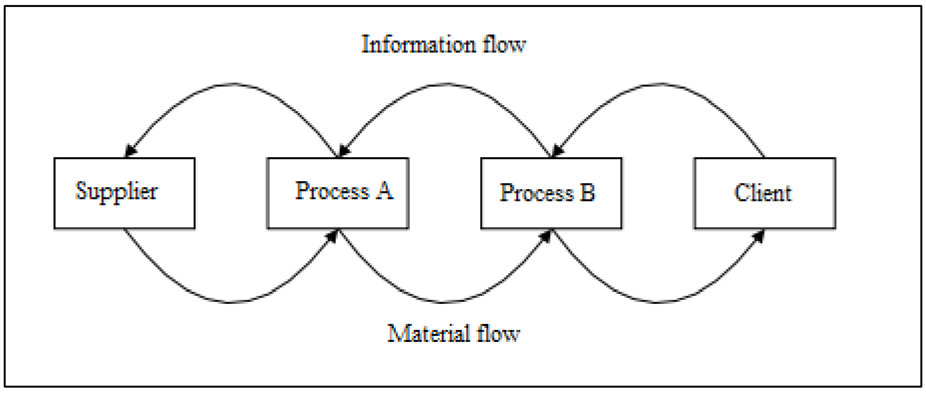
- Clear definition of delivery dates due to the presence of inventory and the control of lead times.
- Higher certainty that demand will be attended.
- Higher levels of inventory for raw materials, WIP.
- Excess of production may happen.
- Higher operational costs.
- Lower quality levels, once the presence of intermediary buffers delays the identification of failures and defective items.

2.2.2. The importance of workload control

Traditional push production system is not very responsive to changing customer demands. Although it relies on demand forecasting to create the scheduling, when demand is volatile the release of work into the system may not meet real necessities. These systems often have excess inventory, high WIP levels, and longer lead-times from order to delivery. On the other hand, just-in-time production is based upon real demand triggering the release of work into the system, and "pulling" work station after station to satisfy demand.

The pull system has a backward flow of information when compared to the push system. It does not schedule the start of production, but instead the demand for a given good authorizes production (SPEARMAN; WOODRUFF & HOPP 1990).

Figure 6: Information and material flows in a pull system.



Source: Adapted from Fernandes (2013)

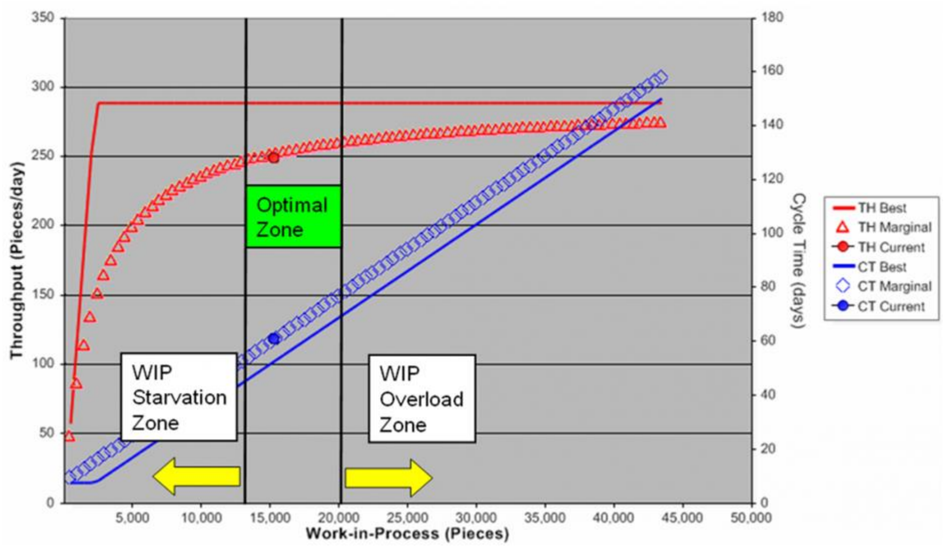
Literature often suggests that pull systems are easier to control than push systems. However, a very important point is that, in reality, they are conceptually harder than the latter. For a good functioning of a pull system, dimensioning its parameters is essential, since the benefits of a pull environment are due to the fact that WIP is limited. (MAREK, ELKINS, & SMITH, 2001).

When a production system has too much WIP flowing in the process, flexibility is lost. Possible design or engineering changes can be costly to apply. Even scheduling changes are pricier in systems with high volumes of WIP. Conversely, a restricted WIP capacity limits the amount released in the process, thus, orders remain on paper rather than as physical material into the system. By controlling WIP levels the amount of scrap or rework due to any sort of change is minimized.

Another benefit of restricting WIP derives from Little's Law, which states that $WIP = Cycle\,Time\ \times Throughput\,rate$. Little's law along with other principles presented in the book Factory Physics (HOPP & SPEARMAN, 1997) prove that there are three regions of performance related to WIP for any system, showing that both too little or too high WIPs are prejudicial.

The first region is called "The WIP overload zone", it happens when the system has so much WIP that throughput becomes independent of WIP, here an increase in WIP generates an increase in cycle time having no effect on throughput. The second region is called the "Starvation Zone", it is characterized by a small quantity of WIP for the system which results in shorter cycle times but throughput is also reduced. The last zone is the "Optimal WIP zone", here the levels of WIP provide the best results with minimal cycle time and maximum throughput. Defining the region in which the system is at is important because it helps defining if improvement opportunities are in WIP and cycle time or if there needs to be variability reduction is the system. Figure 7 presents a graphical representation of the three WIP performance regions.

Figure 7: Three WIP performance regions.



Source: Factory Physics Inc (2016)

Based on the above-mentioned principles, it can be said that limiting WIP in the right dimension reduces cycle time variability while allowing the pull system to still achieve the same throughput level with less WIP if compared to a push system.

There are different methodologies that rely upon restricted work in progress. This study focuses on two pull techniques: Kanban and CONWIP. Both are later explained.

2.2.3. Kanban

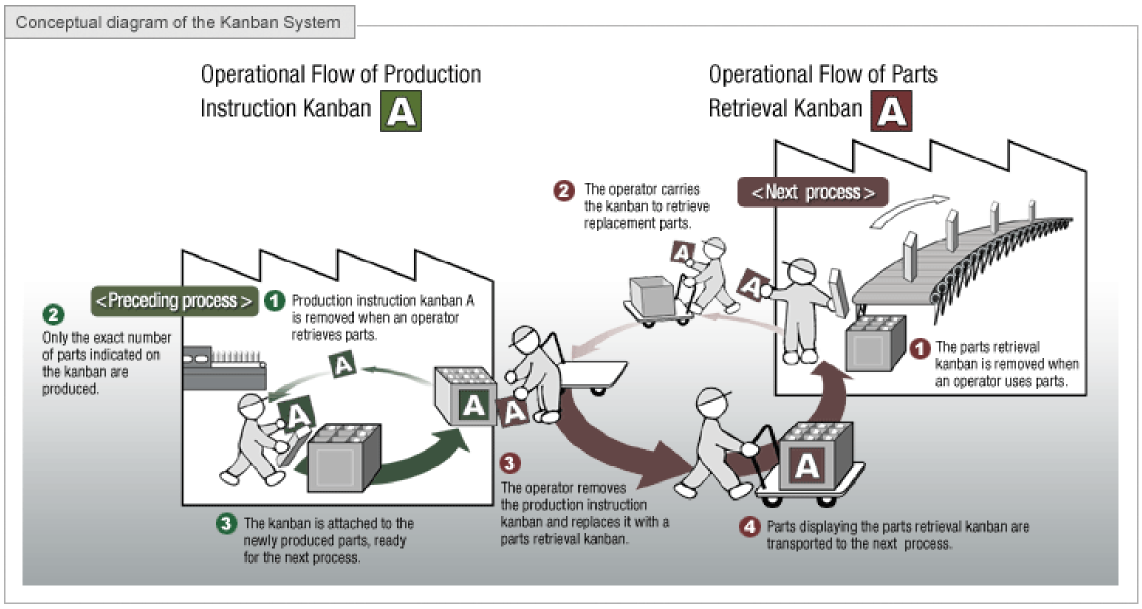
First introduced in Japan by Taichi Ohno, *Kanban* is often referred to as Just in time (JIT) or zero inventory (ZI). However, it is actually a tool for lean manufacture. Inspired by the functioning of a supermarket, in which products are restored after consumption, Ohno's main goals were to reduce "wastes", and to create a more dynamic environment, which could be more responsive to changes.

Kanban which is the Japanese word for card or marker, is a system based on the use of signalization (cards, charts, containers etc.) to activate production or internal flows (TUBINO, 2007). Wokman & Jones (1998) characterize the *Kanban* as a system in which production is determined by the subsequent stage, where quantities are limited by the downstream necessity.

The application of a *Kanban* system may be established based on any kind of signalization. The use of cards is the most common system, however, electronic *Kanban* is becoming more popular year after year. The *Kanban* cards signal the movement of materials within the factory and trigger the replenishment of used goods. That is why the card creates a demand-driven flow: there needs to be consumption before production activates.

Hopp and Spearman (1997) explain that the process begins by the movement of a container at an outgoing stock point authorized by a transportation card. A production card is then removed from the container. The production can start when there is a production card and an idle station available. The first thing the worker needs to do is to place the transportation card in the container and then begin the process. When an authorization *kanban* card is received, material is pulled downstream through the process chain. The described process is represented on Figure 8.

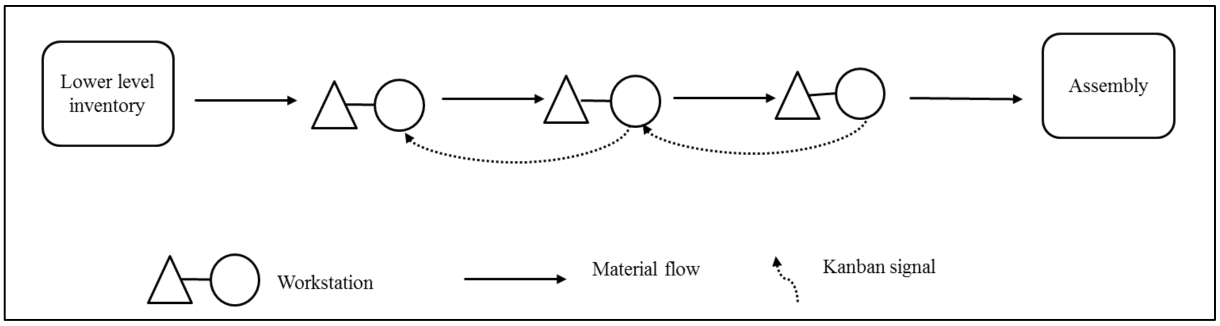
Figure 8: Conceptual diagram of the Kanban system.


Conceptual diagram of the Kanban System
Operational Flow of Production Instruction Kanban A
Operational Flow of Parts Retrieval Kanban A
<Preceding process>
<Next process>
1 Production instruction kanban A is removed when an operator retrieves parts.
2 Only the exact number of parts indicated on the kanban are produced.
3 The kanban is attached to the newly produced parts, ready for the next process.
1 The parts retrieval kanban is removed when an operator uses parts.
2 The operator carries the kanban to retrieve replacement parts.
3 The operator removes the production instruction kanban and replaces it with a parts retrieval kanban.
4 Parts displaying the parts retrieval kanban are transported to the next process.
A

Source: Toyota Global website(2016)

The *kanban*s in the factory control production as consumption happens. Consumption including external (final clients) and internal clients (subsequent processes). External and internal *kanban*s follow basically the same described process. By including all the productive chain, *Kanban* can connect all stages of a value stream, synchronizing processes according to the demand.

Figure 9: Kanban schematic representation.



Source: Adapted from Hopp & Spearman (2008)

Each production station possesses entry and exit inventory points, being controlled by two types of cards, production and transport (also known as withdraw). The former signalizes the quantity that has to be produced, avoiding excesses. The latter is often present in systems in which production cells are distant from each other, and it signalizes the quantity that has to be taken and transported.

Monden (1984) points out some features that a *Kanban* system should respect in order to be successfully implemented:

- Quality control must happen in every stage, and no defective item can be sent downstream.
- Downstream workers are responsible for transporting pieces to their station.
- Internal consumption must withdraw solely the necessary number of items, and only at the moment when they will be used.
- Production can only happen if there is a card authorizing it. Also, quantities need to respect the exact amount specified by the card.
- The production process should be standardized and stable.
- The number of cards should be minimized under the *kaizen* context.

Although the application of *Kanban* can significantly reduce costs and inventory levels, its downside is that *Kanban* is typically restricted to repetitive manufacturing. Large variations in volume or product mix destroy the flow and weaken the system’s performance.

Another important point is that the right implementation and dimensioning of *Kanban* is determinant for its success. If there is too much WIP, the goal of minimizing WIP in the system is not achieved, and financial flexibility is lost. If there is too little WIP, throughput goals cannot be accomplished. The dimensioning of a *Kanban* system is later on explained.

2.2.3.1. Kanban dimensioning

An important aspect for proper functioning of the *kanban* system is te determination of the number of *kanban* cards. A large number of cards may reduce the number of non-served clients. On the other hand, if the number is too small inventory levels will also be limited, increasing the chance of disruptions.

The original formula for calculating the number of *kanban* cards used in intermediary inventories was proposed by Monden (1984), and it is defined as:

$$K = \frac{d}{A} \times L_d \times (1 + \alpha)$$

Where,

- K is the number of cards
- D is the average demand for pieces per unit of time
- A is the batch size (units in the container)
- L_d is the lead time, including transport, processing and other times that may affect the delivery
- α is a safety coeficient to minimize the effects of possible sources of variation

Meanwhile, the dimensioning of containers for a process that produces one type of product was suggested by Tubino (1997) as:

$$C = \frac{E_\alpha + D \times L_d}{\frac{Q_c}{A}} + 1$$

Being:

- C the number of containers
- D the average demand
- Q_c the consumed quantity
- A the batch size
- E_α the safety inventory
- L_d the lead time

If multi-products are processed the above formula must be incremented by the minimum fabrication batch (L_{min}) divided by the quantity of pieces per container. The formula then, becomes:

$$C = \frac{L_{min} + E_{\alpha} + D \times L_d}{\frac{Q_c}{A}} + 1$$

The goal is to obtain the right amount of WIP in the system so that cycle time is not affected by the excess of inventory, nor the system suffers with blocking and starvation due to too little WIP (PETTERSON & SEGERSTEDT, 2009)

To obtain the leanest system possible, production may happen in small lots, and setups and lead times may be as small as possible. Also, sources of variability need to be identified and solved in order to reduce the safety coefficient, reducing overall inventory levels.

Another tool commonly used is the *Kanban* chart. It indicates priorities based on a color signalization. Normally, green, yellow and red indicate in crescent order the urgency of an action. Each color can have one or more *Kanban* cards, accordingly to the established calculations.

As containers are consumed, the *Kanban* cards are put on the charts, first occupying the green area, which indicates that there is enough inventory, thus, production does not have to be immediate. When the green area is full, the cards are **placed** on the yellow area, which is the first alert sign that production should start. Finally, there is the red area, which indicates urgency in production.

Each color area determines one piece of the presented card calculation for machines that need set ups.

- Green Area: $\frac{L_{min}}{\frac{Q_c}{A}}$ represents the stock level considering setups.
- Yellow Area: $\frac{E_{\alpha}}{\frac{Q_c}{A}}$ represents the safety stock level.
- Red Area: $\frac{D \times L_d}{\frac{Q_c}{A}} + 1$ also known as regular stock, it represents the demand during the cycle time.

Figure 10 presents an example of *kanban* chart.

Figure 10: example of Kanban Chart

Product A	**Product B**	**Product C**	**Product D**	**Product E**
🟩 Kanban card	🟩 Kanban card	🟩 Kanban card	🟩 Kanban card	🟩
🟩 Kanban card	🟩 Kanban card	🟩	🟨 Kanban card	🟩
🟩	🟨 Kanban card	🟩	🟨	🟩
🟨	🟥 Kanban card	🟩	🟥	🟨
🟨		🟨	🟥	🟨
🟥		🟥		🟥
		🟥		

Source: Adapted from FERNANDES (2013)

The use of the kanban chart must follow similar rules of the use of cards. Items must solely be produced if there is a card on the chart. Otherwise different activities can be performed, such as preventive maintenance, cleaning, training, etc.

Another important issue is observing the general pattern that appears on the chart. Items that are frequently on the red zone or even never out of the normal zone, may indicate miscalculations of the number of cards.

2.2.3.2. The use of simulation to model Kanban

The drawback of the use of formulas for dimensioning the system is that the results may be simplistic and lack flexibility. More complex mathematical formulations may be applied; however, they are only feasible for small systems. Another unpractical downside is that the use of deterministic values may lead to unrealistic results.

To overcome those obstacles, the use of simulation for system dimensioning appears as a good decision making tool. Simulation allows the creation and comparison of different scenarios, enabling verification without cost implications of real tryouts (KONDO & MESQUITA, 2008).

Simulation models have been used to explore the relationship between different system parameters on the performance of Kanban systems as well as optimization of performance measures. The parameters that are usually used are: line balance, variability in processing times and demand, station utilization, number of Kanban cards, etc. Some of the performance indicators include: WIP volumes, cycle time, utilization, and throughput (SAHU, 2012).

There are many software that enable the modeling of such problems, examples are: Pro-Model, SIMAN, ARENA, SIMULINK, Q-GERT, GPSS and even Excel Spreadsheets.

Some authors have used simulation for modeling *kanban*. Here a few modeling strategies and their results are presented.

Detty and Yingling (2010), have used discrete event simulation to assist the decision to implement lean manufacturing in an assembly process for a consumer electronic product. A model was developed for the total system, including warehousing, inventory management, transportation, and production. Among the benefits presented, there was significant reduction on equipment and man power utilization, order lead time and variability in supplier demand.

Regarding the dimensioning of the *Kanban* system, Sahu (2012) presented a simulation model that compared three methodologies to find the near optimal number of *kanban*s for an assembly line. She compared the Toyota Production System model, the histogram model and a cost minimization model. The simulation was held in Arena. The study concluded that the two latter models presented superior results, once method one failed to adjust the *kanban*s when the system changed.

Still on the topic of Kanban dimensioning, Marek, Elkins & Smith (2001) introduced a heuristic to reduce the number of cards in the system. He starts with the arithmetic result from Monden’s formula, once the system has been adjusted with those values, the station with higher utilization rate (bottleneck) is identified. Then, he reduces the number of cards for that station until the initial throughput rate is no longer achieved. The process is then iterated for the next bottleneck station, and so on until all stations have been analyzed.

Mesquita & Goldemberg (2006) created a spreadsheet on Excel that simulated a production line with four machines on series to work upon one single product. The study starts off with a perfectly balanced line, and from there they evaluate the effects of unbalance. They also compared different scenarios for the number of cards in the system, their scenarios were chosen randomly. Their study was then used by Kondo & Mesquita (2008) to create a version on Pro-Model to evaluate the effects of the number of cards in the system.

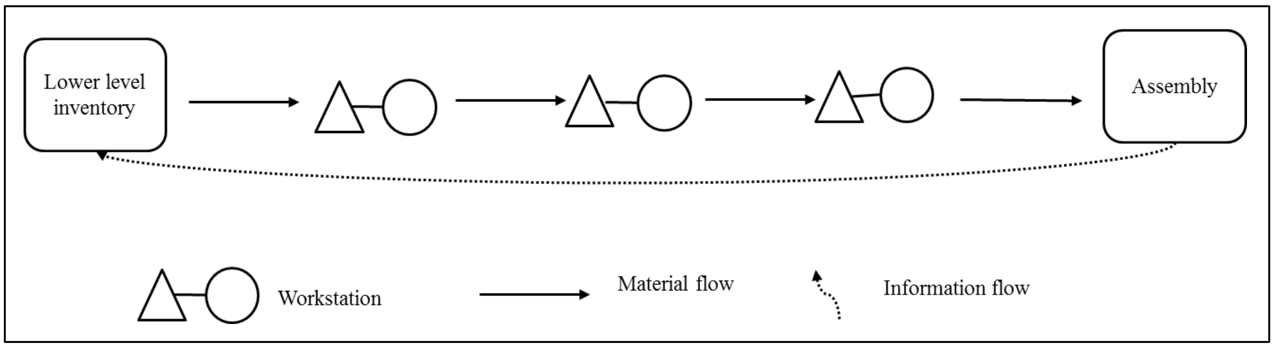
2.2.4. Constant Work in Process (CONWIP)

First introduced by Spearman, Woodruff & Hopp (1990), the CONWIP was an attempt to present another pull based system more flexible than the already known Kanban paradigm. Although Spearman was the first author to use the name CONWIP, others had already introduced similar concepts. Framinan Gonzales & Ruiz-Usano (2003) points identical or similar systems previously presented such as the 'Workload control' system by Bertrand (1983), the 'C-WIP' system by Glassey & Resende (1988), the 'Long pull' system by Lambrecht and Segaert (1990) and the 'Globally flexible line' by So (1990).

Since its introduction more than a decade ago, the CONstant Work in Process (CONWIP) production control system has received a great deal of attention. Many authors present it as a system which has the benefits of pull systems but is easy and democratic to implement as is the MRP.

Although Spearman considered it a pull system, some may present it as a mix of push and pull systems (MESQUITA & GOLDEMBERG, 2006). The production works under a push strategy, meanwhile the trigger that gives a production order respects a pull system. In other words, the CONWIP is a one stage *Kanban*, in which the whole production system is controlled by the card system.

Figure 11: Representation of CONWIP flows.



Source: Adapted from Marek, Elkins & Smith (2001)

The main difference between CONWIP and *Kanban* is that in the former implements a card system that controls the whole production, internally the processes work in a push way. In the latter however, there is one card system for each part of the product, thus, if a particular process possesses many different parts, the implementation of *Kanban* is unpractical.

Kanban, works well in high quantity low variety systems, that means it is more adapted to suit repetitive manufacturing. Meanwhile, CONWIP is more robust to changes in product mix (HOPP & SPEARMAN 2008). Some may interpret that difference as connected to the production environments MTS or MTO. However, the distinction is orthogonal to the concept of make to order r to stock (HOPP & SPEARMAN 2004).

As the Kanban system, CONWIP is also based on signalization, which is usually done by cards, as the previous. Here a card is attached to the job entering the system, once it has been completed at the end of the process the card is released and sent again to the beginning of the system. There it will be attached to another job, and only then another task may enter the process (FRAMINAN, LEZ, & RUIZ-USANO, 2003)

The difference between both systems is that card control in a *Kanban* may cause a workstation to become idle, even if it has raw material to process. The absence of cards cause a blocking after service which explains the idleness. Card control at the individual workstation level introduces an additional level of dependence between the workstations. On the other hand, CONWIP systems do not introduce the additional workstation dependency nor cause blocking after service. Since the CONWIP system works as a push system, each workstation will remain processing as long as there is sufficient input. WIP levels will be higher on the queue upstream the bottleneck. However, since CONWIP follows a restricted WIP logic, there will be no queue explosion like in the push systems (MAREK, ELKINS, & SMITH, 2001)

On different versions of applications, the cards may be linked to an actual consumption by external clients, which reduces final inventory levels, and only then the card may go back to the start of the process.

On the original version by Spearman, the materials requirements are controlled by a *backlog,* which contains the specific quantity necessities for each sub-part or piece. Since the intermediate processes push forward their output, the machines that run faster than the bottleneck will simply push their production which will remain blocked in the queue waiting for the bottleneck. That means that the slowest machine will have the only significant amount of stock in the process, assuring that the bottleneck works at its highest capacity. According to the author this is a good thing, since the bottleneck defines the speed of the entire line.

According to Framinan’s reviews the most common aspects presented on literature regarding the theme concern:

- The operation of CONWIP: including the decisions to be taken to operate the system on the light of given performance indicators.

- Applicability of CONWIP: authors use real implementation scenarios or simulation to present results of the application of CONWIP.
- Comparisons of CONWIP with different production systems.

Although the above features have been intensively studied, in practice there is a shortage of CONWIP installation guidelines (PETTERSON & SEGERSTEDT, 2009)

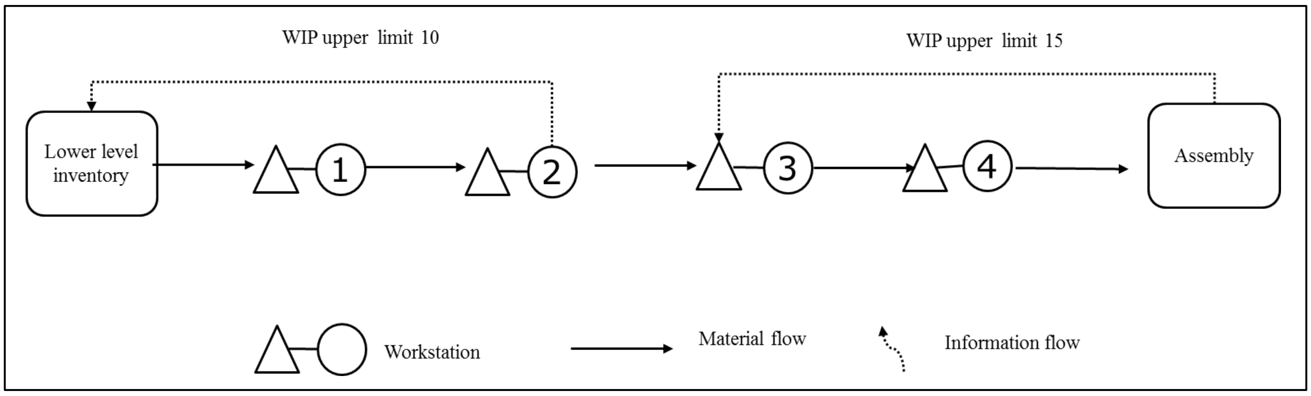
The parameter involved with CONWIP are (SPEARMAN, WOODRUFF & HOPP 1990):

1. The card count, which defines the maximum WIP level of the line,
2. The production target for a period
3. The maximum work ahead amount
4. A capacity shortage trigger.

These parameters are essentially tactical, and a fifth one can be added, job sequencing. The latter more related to hierarchical planning is only necessary for cases different than the one product make to stock process. In cases which different products are manufactured there needs to be a decision regarding which product can enter the system. Here, two approaches can be selected. The first relies on differentiating the card types for each item. The second fixes a repetitive schedule that reflects the long-term demand of the different items.

A different CONWIP design was proposed by Yang, Fu & Yang (2007). The study presents a simulation for a case in which the number of loops in the CONWIP is larger than one. When the segmentation of the process in loops is equal to the number of processes in the system, it constitutes the Kanban system.

Figure 12: Multi-CONWIP configuration.



Source: Adapted from Yang, Fub, & Yang, 2007

2.2.4.1. Dimensioning CONWIP

On Framinan's review he points out the existence of two methods for calculating the number of cards in the system: card setting and card controlling. The first establishes a trade-off relationship among factors (i.e. cost function, throughput levels, etc.) and the number of cards is selected based on the attendance of performance indicators. On this case the number of cards in the system is fixed. The second method foresees iterative rules to change or not the card levels.

Still according to the author, despite being the most important parameter on CONWIP application, no clear guidelines have been stabilished to quantify the number of cards in the system. Many different authors have created different methods to perform the calculation. Among them, the formula $N = D\ x\ T + S$, where N is the number of cards, D is the periodic demand, T is the total cycle time given as a percentage of the total demand period, and S is a security coefficient.

Hopp and Spearman (1997) present a formula for system throughput that can be used to estimate the number of cards needed to control a CONWIP system. They define throughput as a function of the number of cards as:

$$TH(w) = \frac{wr_b}{w + W_0 - 1}$$

Being:

- TH the throughput
- w the number of cards
- $r_b$the rate of the bottleneck station given in jobs per minute

- $W_0$the WIP level achieved for a line with maximum throughput operating at the bottleneck rate.

The critical WIP level can be defined as

$$W_0 = r_b \times T_0$$

where T_0 is the sum of the average processing times of the workstations, ignoring any processing time variability, blocking, machine failures, or any other type of disruption.

The card controlling category, also known as adaptive CONWIP has gained importance recently due to the increase in competitive pressure and dynamism of the production environment.

Among the authors who have studied adaptive CONWIP, Hopp and Roof (1998) developed a statistical throughput control (STC) to adjust WIP levels by changing the number of cards in the system. They considered a single product line and created the algorithm below:

Step 0. Set initial card count m and warm-up period n.
Step 1. Set target production rate λ ¸ and maximum cycle time CTmax.
Step 2. Clear statistics and wait until n jobs have been output.
Step 3. After each job completion, calculate:
(a) Average interoutput time μ.
(b) Standard deviation of interoutput time σ.
(c) Average cycle time CT.
Step 4. If
(a) CT > CTmax then revise capacity and/or λ; Go to step 1.
(b) μ >1 / λ + 3 σ then m receives m + 1; Go to step 2.
(c) μ < 1 / λ - 3 σ then m receives m - 1; Go to step 2.

Although changes in the number of cards can generate response to changing environments, they may imply on organizational changes that are normally not considered on card controlling studies. Under some situations cards are implemented on a simple way, so increasing or decreasing cards may be simple. However, on other cases management or operational changes may be required, generating costs that aren't evaluated on regular models.

2.2.4.2. The use of simulation to model CONWIP

As mentioned, mathematical modeling of complex systems may oversimplify the problems or offer too complex, non-feasible solutions. To overcome those issues simulation arises as a simpler and more economical alternative.

Regarding the three most studied aspects of CONWIP, this paper presents a few simulation based studies that addressed the issues.

Many author have compared the CONWIP systems with the *Kanban* systems. Among them, Marek, Elkins & Smith (2001) have introduced a methodology for simulating CONWIP and *Kanban* using the discrete simulation software Arena. For both cases a heuristic for card level estimation is presented. Regarding CONWIP, the study starts off with Hopp and Spearman's formula, and then follow the same iteration used for Kanban, although simplified.

Yang, Fu, & Yang, (2007) applied the multi-CONWIP problem to an integrated circuit packaging company. They also used the single level CONWIP as well as the Kanban as "special cases" of multi-CONWIP to enable comparison. They attempted to evaluate the loop segmentation as well as dimensioning the cards. Their proposed multi-stage CONWIP outperformed the both extremes (single loop CONWIP and Kanban). Pettersen & Segerstedt, (2009) simulated a small job shop with five machines in series that have stochastic operation times. Their results show that CONWIP is to prefer over Kanban since it presents higher utilization rate and it needs less storage room. However, they point out that there are no guidelines available for a practical application.

Huang, Wang & Ip (1998) also used simulation to compare control systems. They assessed the influences of Kanban, CONWIP and push in a cold rolling plant, which is a semi-continuous manufacturing system. They, as Petterson, concluded that CONWIP was the control solution that presented the least amount of WIP, and therefore of inventory costs. Despite of that, throughput rates and average utilization were higher than the other two scenarios.

2.3. Literature review synthesis

This literature review presented an overview of the topics that will be further explored in this thesis. Starting with the concept of assembly line balance, a key process design tool which is essential for processing time variability reduction among different stations. In order to implement an effective control system in the Pallet Factory, one needs to, firstly, improve its process balance so that a steadier environment is observed.

Secondly, the three most common shop floor control systems were introduced: Push, Kanban and CONWIP. It was seen that push systems release work into the system based on demand forecast, meanwhile pull systems present a WIP cap which controls production. This is actually the so called “magic of pull systems”, according to Hopp and Spearman (2008), because the WIP cap better attends to Little’s law principle. So, pull systems reduce average cycle time, reduce variability and also improve throughput.

Different studies here presented have proven the superiority of CONWIP over the other two control systems. When compared to the push system, CONWIP requires smaller WIP levels given the same throughput. Regarding Kanban, the latter is more complex since it requires more card handling, it is also less flexible to variability.

The next chapters present more details about the methodology used to perform this study, and about the Pallet Factory and its process. After that, the balance of the plant will be revised so that the study of shop floor control is facilitated. Finally, simulation will be used to test for possible improvements that a CONWIP control logic can bring to the plant.

3. METODOLOGY

This project aims at proposing improvement opportunities for the Pallet Factory, regarding line balance and the workload control. To do so, the methodology applied started by mapping the company's current state, through that analysis different inefficiency sources that are attacked here were identified.

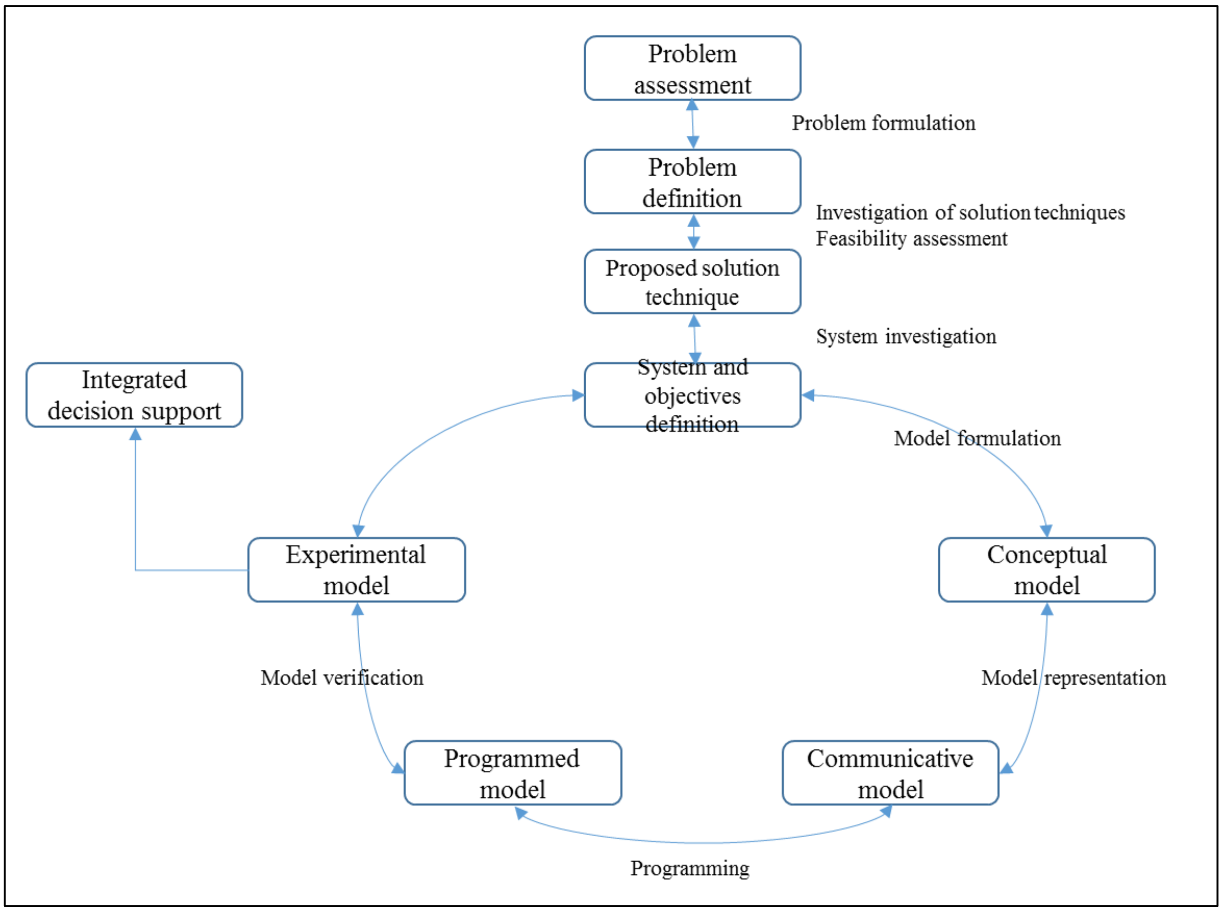
After defining the problem to be resolved, different scenario propositions were created and compared based on different performance measurements. Also, simulation modeling was used as a decision support tool to allow a better understanding of how the system would perform under the influence of the proposed changes and randomness. Since simulation modelling is an essential part of this paper, it is important to define it and to follow a modelling methodology. This paper chose the one proposed by Ernest H. Page (1994), which will be later explained.

Banks *et al.* (2001) defined simulation as the process of designing a model from a real system and make experiments with it with the aim of recognizing the system's behavior or evaluating different scenarios for its function. Computer simulation is used to study the behavior of real world complex systems by applying computer programs that replicate the system. The simulation model acts as a central element in a manufacturing decision support system which could address a wide range of decisions in planning, operations, and control. Computer simulation aids management decision making by allowing one to visualize how a system works, and analyze various configurations and its possible effects before they are implemented. Simulation software are becoming very popular over recent decades for their affordability, versatility and ability to deal with complex models of complicated systems. This is one of the primary reasons why simulation was chosen to perform the analysis of this research.

As reviewed, literature introduces some operation research approaches to deal with questions addressed in this paper. However, all the mentioned solutions apply to "closed form systems", such as single flow lines or reliable identical parallel machines. Since the problem here presented concerns a real company with many intricacies, the mathematical modeling is unable firstly because the model is very complex with many variables and interacting components, secondly because the relationships among the variables are non-linear. The solution found to overcome these obstacles is the use of simulation. With simulation, it is possible to replicate the actual system without recurring to simplifying assumptions that may distort the results.

Regarding the modelling methodology applied, Earnest H. Page proposed in 1994 a methodology for discrete event simulation modelling. Such approach was used in this study. His contributions state that all systems initiate from some problem or requirement. Therefore, this paper initiates from assessing and defining the current state of the Pallet Factory, through that analysis, improvement points are identified.

Figure 13: Modeling methodology



Source: Adapted from Ernest H.Page (1994)

Then, an investigation of different possible solutions is made and an initial draft of the model is created. That "draft" in here is presented as the data flow diagrams and are the conceptual model which preset the logic behind the functionalities of the different system propositions.

The next stage is the programming itself, that means translating the designed logic to a computer language. In order to perform our study, the Arena® discrete event simulation platform by Rockwell Software, Inc. was selected for building up the simulation model. The software is based on discrete event simulation and uses graphical visual tools so no coding is necessary.

The main advantages of the use of Arena are:

- It allows to validate and optimize discrete systems such as production lines. Facilitating the evaluation of different scenarios and maximizing their potential output and benefits.
- It is based on the known SIMAN simulation language - is well suited for modeling shop floors of production systems in which each a part follows a manufacturing path through production resources.
- It is a visual tool which possesses error diagnostics, facilitating modeling.

Once the model was complete the validation and verification stages begun. Due to a lack of available data to compare the model to the real system given the same initial inputs, the model was validated through the examination of its logics, and by the application of tests using inputs simpler inputs for which the outputs were known. Also, throughout the process, the tutor of this thesis, who is specialist in simulation model helped with the verification.

Finally, the model ran and the results for the different specified inputs were compared to help with the decision-making process, proofing if the propositions were indeed effective or not.

It is noticeable that here implementation was not considered. This is due to difficulties regarding the location of the plant, which is in Poland, and because the flow of information throughout the process was indirect. For all those reasons implementation was unable.

4. THE COMPANY’S CURRENT STATE

This paper aims to achieve improvements on the Pallet Factory by controlling work in process through the use of balance and a leaner operational system. The objective is to evaluate different scenarios and compare their performance using simulation models.

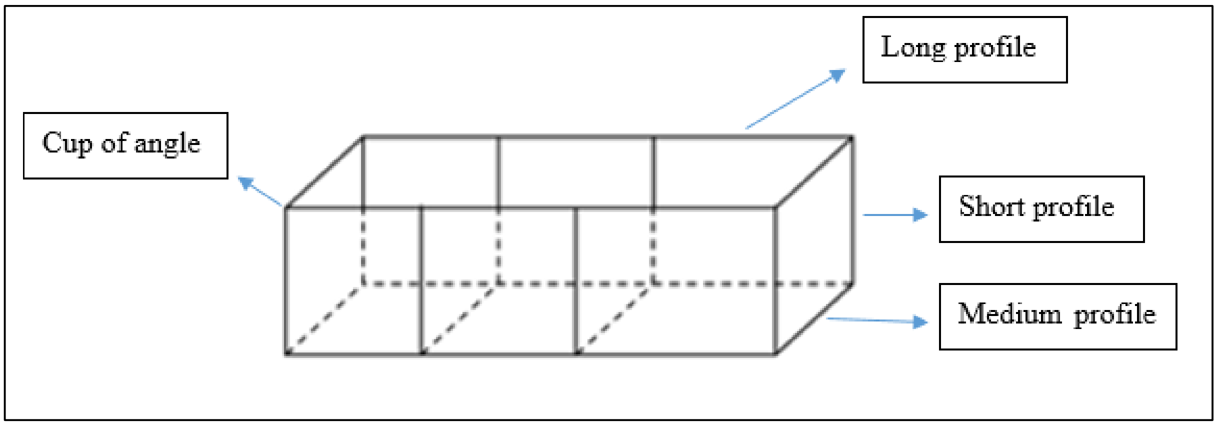
The further chapter of this paper presents the inputs of the considered problem. In a fist moment, the production process is introduced, there is a brief description of the product, including the product tree and the process task sequence. The second section points out possible improvement points, which will be later evaluated.

4.1. The product and its production process

The Pallet Factory is a small producer of different kinds of metallic goods such as containers, bins, trays, pallets etc, mostly dedicated to logistics and storage solutions. The firm is situated in Poland and began its operations on early 2000s. It started as a family business, but the plant nowadays counts with 25 workers, divided based on product line. Being 11 dedicated to the pallet production.

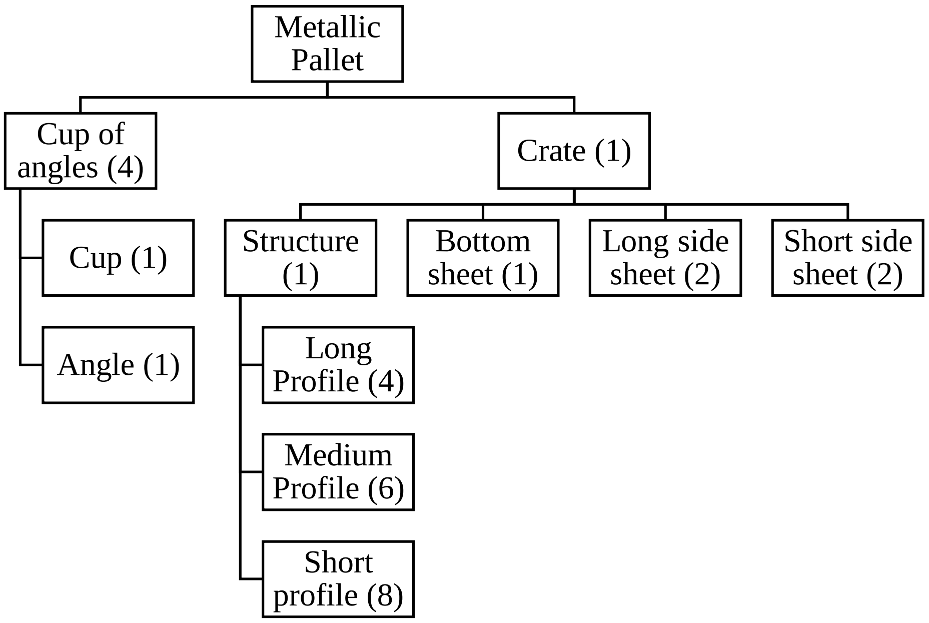
The pallets are made of an ordinary cold-rolled metallic sheet having a thickness of 0.7 mm and with sections made of carbon steel. The firm produces the same design of pallets in three different sizes, big medium and small. The first is made from metallic sheets measuring 540 x 2310 mm. For the others, sheets measuring 360 x 2310mm are required. However, as it is common in the industry, customization may be required. A schematic representation of the pallet as well as the bill of materials are presented in Figures 14 and 15.

Figure 14: Schematic representation of the product



Source: (Created by the author)

Figure 15: Product tree



Source: Created by the author

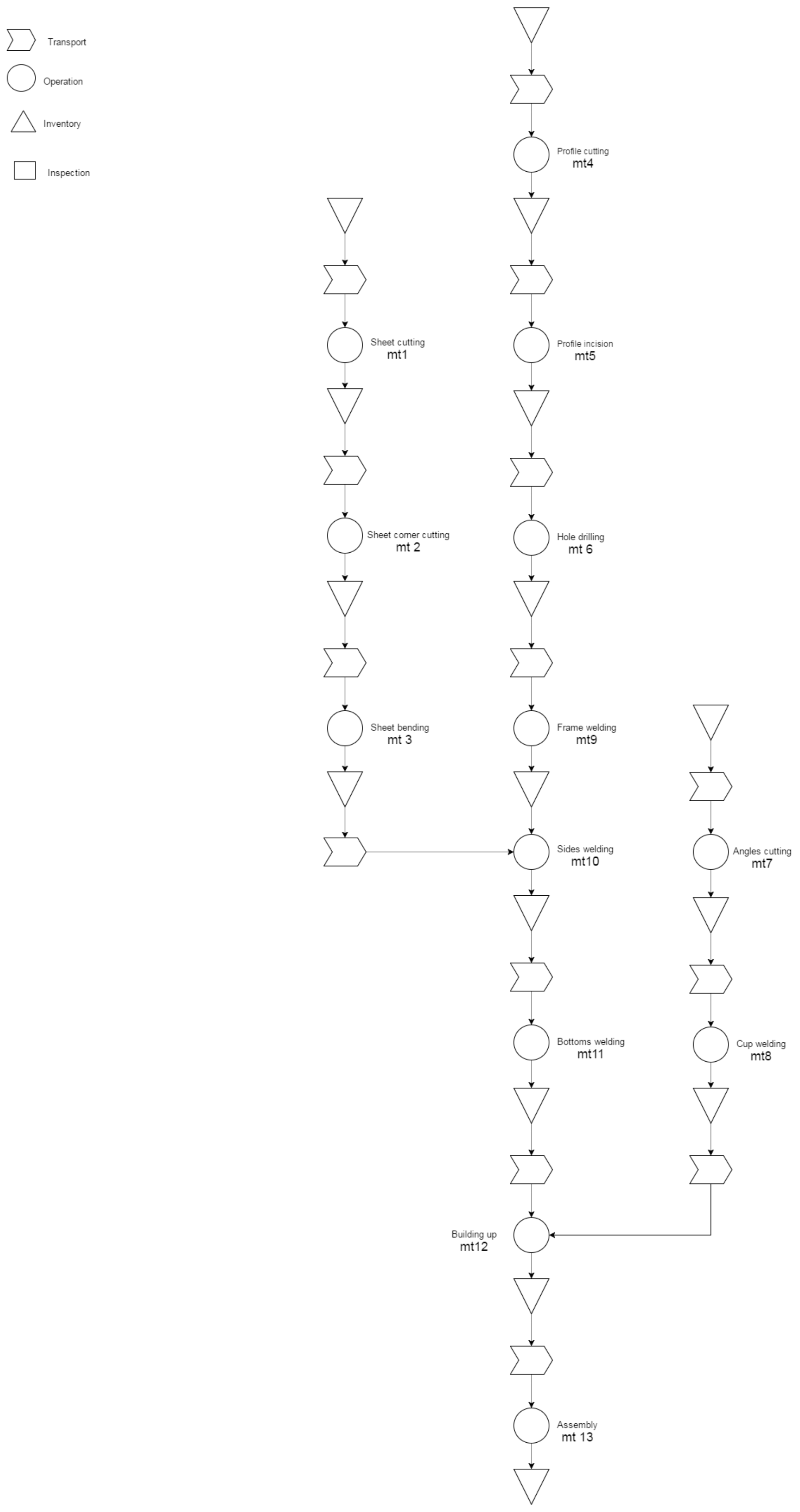
The plant has two different rooms dedicated to the storage of raw materials for the pallets, one stores profiles, the other sheets and angles. The initial stocks are big, since many items are shared among other products as well. The plant also counts with a finished products inventory and a deposit for scrap steel.

The production of metallic pallets involves 13 tasks. Each task can employ one or two operators. The manufacturing tasks as well as their duration, the physical resources applied (machines) and number of workers employed are presented on Table 4.1. Since only one pallet design is produced, although in different sizes, all pallets must pass through all operations, which requires approximately the same amount of time no matter the size. Figure 16 represents a process flowchart.

Table 1: Manufacturing tasks

Manufacturing task (mt)	**Description of manufacturing task**	**Task duration (s)**	**Resource**	**Numbers of workers needed to perform a task**
mt 1	Sheet cutting	40	Mechanical guillotine	2
mt 2	Sheet corners cutting	652	Angle grinder	2
mt 3	Sheet bending	624	Sheet press	2
mt 4	Profile cutting	349	Saw frame	1
mt 5	Profile incision	504	Bench grinder	1
mt 6	Holes drilling	1026	Drilling machine	1
mt 7	Angles cutting	16	Press for cutting angles	1
mt 8	Cup welding	192	Welding station	1
mt 9	Angle welding	304	Welding station	1
mt 10	Profile welding	416	Welding station	1
mt 11	Bottoms welding	120	Welding station	1
mt 12	Folding	1187	-	2
mt 13	Assembly	1212	-	2

Figure 16: Process flowchart



Source: Created by the author

Bellow there is a description of the work division among operators.

Operators 1 and 2 are responsible for cutting sheets (mt1). Although formally the raw materials are stored in a dedicated room, in reality they leave an amount of pieces beside the guillotine. Loading, processing and unloading one sheet on the mechanical guillotine takes 10 seconds. Since they must process batches of at least four sheets to pass forward, the total time dedicated to the activity sums 40s/pallet. Those workers are also responsible for the transport to the downstream resource. There, the same two workers will smooth the corners of the cut sheet using a grinder. This activity requires, on average, 163 s per sheet, totalizing 652 for the whole batch. Finally, right beside the grinder there is a press where the sheets are bend. Bending the batch takes on average 624 seconds.

Operator 3 is responsible for transporting profiles from the storage to the saw frame, there, he cuts the required number of items, which takes 349s. He also moves the angles from the initial storage until the press where they are cut (task mt7). After that, he is also responsible for the transportation of the cut angles to the welding station. Those activities take 25, 16 and 20 seconds, respectively.

The next worker, operator 4, transports the cut profiles to a second bench grinder where those are polished, this task takes around 504 seconds. The profiles are then put on a table right beside the grind, and the same operator moves to the drilling machine on another side of the table and drills the profiles. Drilling takes about 1026 s.

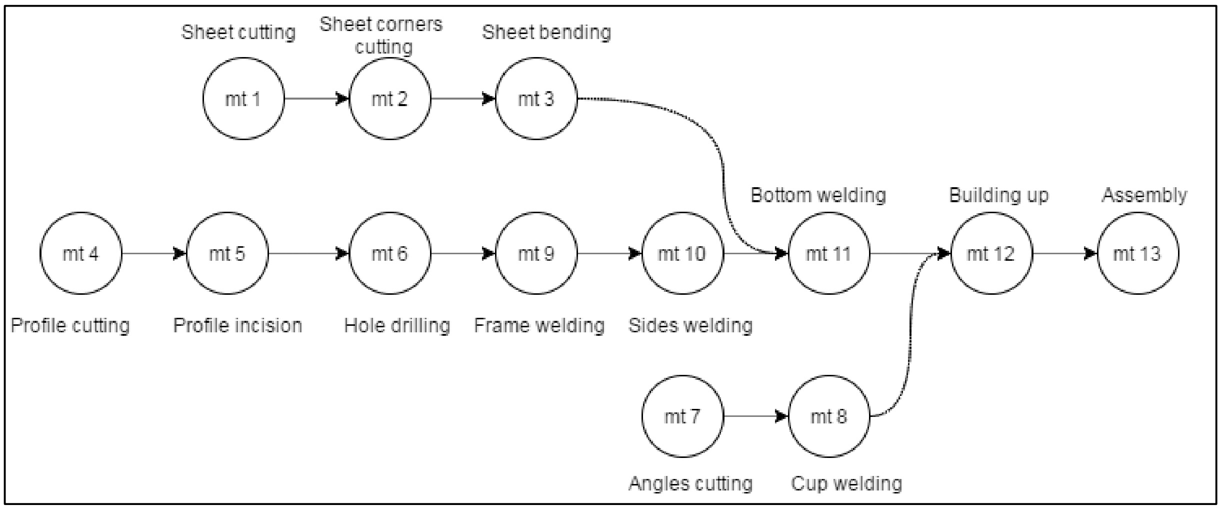
All processing lines converge in the welding sector. There are three welders in the plant (referred as operators 5, 6 and 7), as well as three dedicated welding stations. The materials that need to be welded are not necessarily allocated to the closest welding station. Operators normally talk between themselves and arrange the task distribution. On average, welding sockets take 192 seconds, welding skeleton pallets with profiles takes 304 s and welding bottoms and sides 536 s. Since the welders arrange the activities, they are also responsible for picking up the inputs for the activity on the upstream location. Therefore, transporting angles, sheets and profiles from the upstream to the downstream is their responsibility; those activities vary in time, taking from 10 to 20 s.

The final tasks are bringing together all the pieces to the right position and assembling them together. Both require more than one operator, since now the pieces have increased in weight and size. There are two workers dedicated to the building up and two others dedicated to the assemble activity. They require 1187 and 1212 seconds, respectively.

In addition to the 11 line operators, the plant counts with one manager to control all the factory's production. For the pallet line production follows a weekly schedule, established based on the demand forecast. The manager is also responsible for ordering raw materials.

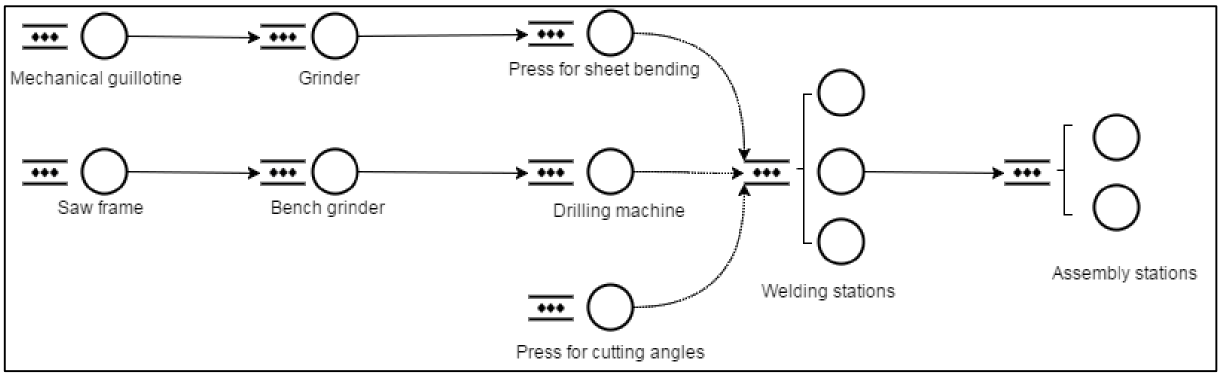
Figure 17 presents the task sequence for the process.

Figure 17: Task sequence



Source: Created by the author

Figure 18: Process diagram



Source: Created by the author

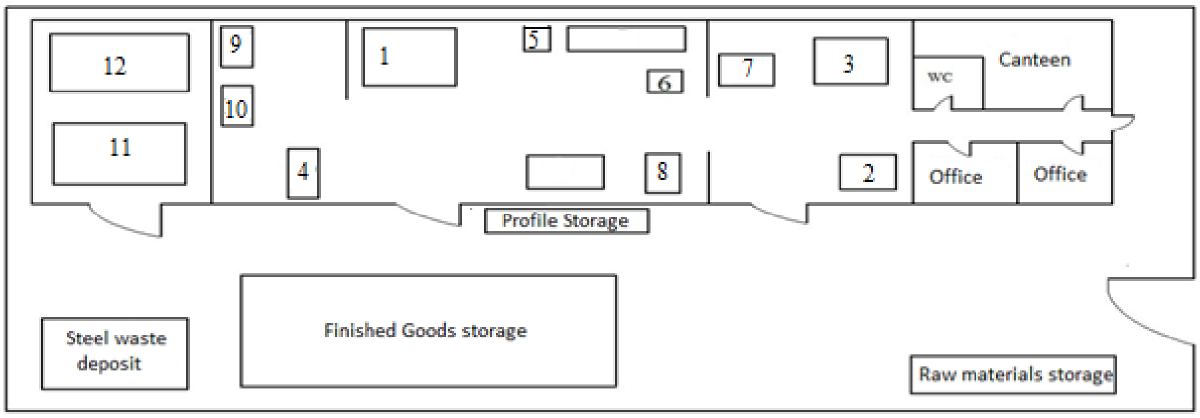
Through the description of the task sequence it is possible to observe that the process is relatively simple, the machinery involved in the process has no automation, so there is always the need of an operator to perform tasks. In addition, the transport of pieces and all internal flows are manually performed.

The machinery in the factory is basically formed by two presses, one guillotine, two grinders, one drilling machine, a saw frame and welding equipment. The administration of the firm intended to buy a plasma-cutting machine in the future; however, although there is already some space dedicated to it, no action has been taken, so we can assume that no changes will happen in a near future.

The transport of materials to the inventories is performed with the help of a four-wheel transport cart. However, its capacity is very limited, so, batching a large amount of items before transporting is difficult.

The layout of the plant, as well as the position of the machinery is presented in Figure 19. The picture is out of scale, and each workstation numbered is named below.

Figure 19: Plant Layout



Source: Created by the author

(1) Mechanical guillotine
(2) Grinder
(3) Sheet press
(4) Saw frame for profiles
(5) Bench grinder
(6) Drilling machine
(7) Press for cutting angles
(8) Welding station
(9) Welding station

(10) Welding Station
(11) Assembly station
(12) Assembly station

Currently there is no technique applied for the realization of a line balance. The grouping of workers and their allocation to tasks does not change and it was defined internally based on their perception of capabilities and competences.

Since there are no supervisors, the identification of bottlenecks takes longer to happen, which decreases productivity. In addition, when an operator exits momentarily his position (bathroom, coffee or smoking breaks), he is not substituted, therefore, the unbalance of the line becomes more accentuated. Each worker uses the method that he prefers to perform tasks; consequently, individual performance presents big dispersions.

4.2. Statement of the problem

Previously, the working process to produce the steel pallet was presented. This chapter consists on a definition of the problem to be solved. It is important to consider the time available for the solution proposition, as well as other physical and non-physical limitations.

The first noticeable point is that the separation of tasks does not follow any optimization logic, consequently, the resulting balance is inefficient. The required cycle time to supply for the average demand is 1363s (around 22 minutes), however, in one of the existing stations processing time surpasses the theoretical cycle time. That negatively affect the efficiency of the plant, since extra working hours are frequent.

Another consequence of the unbalance is a high level of idleness observed in a few stations. Since the difference in processing times among stations is significant, some workstations present very high utilization rates, meanwhile others have a lot of free time. The absence of supervisors contributes with a disorganized environment. The workers, especially those with idler time, tend to chat and have more coffee and smoke breaks than the others, instead of covering up or helping with more demanding tasks.

The unbalance is accentuated when an operator from a station with higher producing time takes a break. There is no rotation system foreseen to minimize disruptions caused by these situations. The shifts are composed of eight hours, however, there are two fifteen minute breaks during the day, so, practically, only seven and a half hours are productive.

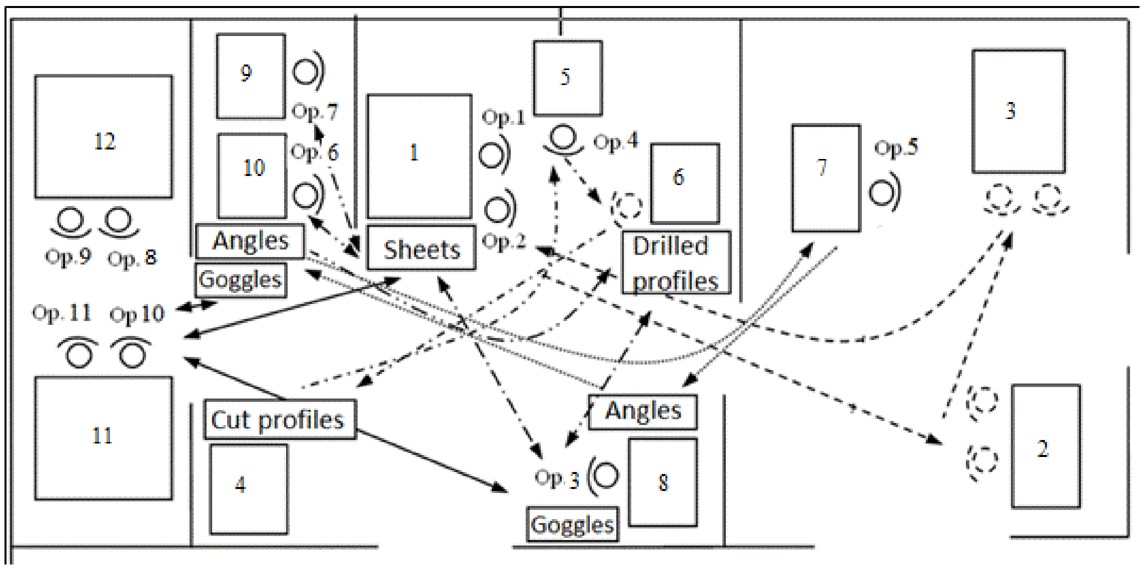
Another problem brought by the unbalance is the uneven amount of work in process. Bottleneck stations tend to accumulate a lot of inventory upstream.

Currently the pallets are produced in a push system. They normally produce three different pallet sizes, however, switching from one product type to another is not a limitation since all three sizes follow the same design so set up times are quite small.

Nowadays, there are no indicators that allow the identification of bottlenecks, quality checks, nor productivity. The only ratio controlled is the number of worked hours over available hours.

Another problem noticed is the layout of the plant. Figure 20 represents a flow diagram for the workers. It can be seen that the travelling paths are very disordered. Some consecutive stations are not close together, which increases transport times. Also, since the distribution of tasks was not planed, internal flows tend to take longer than if the tasks were programed.

Figure 20: Spaghetti chart for operators flows



Source: Created by the author

Another important point is the existence of large intermediary stocks in the plant. The guillotine (represented on the layout by the number 9), is far from the initial sheet inventory. Therefore, it is a common practice to store a pile of raw material by the machine. In addition, the unbalance of the line tends to form large batches before machines that have larger processing time.

The ergonomic factor might as well be pointed out. It is common for some workers to perform repetitive movements. Moreover, although there is a cart to help with the transporting activity, sometimes workers just lift and carry the pieces, which may cause lower back problems due to weight lifting.

We can summarize the presented problems using the following bullets:

- Absence of balance pattern in the line
- Low control of productivity
- Not-optimal layout
- Bad ergonomic conditions
- Lack of supervision
- High levels of intermediary inventory

Each one of the presented problems has a near to optimal solution. However, given the reality of the company, its financial limitations, and the time frame it is hard to solve them all. So, this study presents a theoretical attempt to improve productivity of the pallet line given the mentioned constraints.

In face of the context previously described, the project was motivated by the impact that a structured production planning can bring to the company. As academic literature suggests, a good resource allocation can significantly improve a system's performance. So, under the context of a small un-automatized organization simple measures can make a difference by reducing idleness, and consequently costs.

It is worth mentioning that there are different issues that can be improved inside the Pallet Factory. However, given the already mentioned geographical difficulties, this thesis remains as a propositional work. With that being said, the project starts with a wide-scope topic which is to improve the company's process performance. To do so, first a balance proposition is presented, which should improve idleness, and the overall process. Then, moving on to a more operational matter, the paper tries to simulate a leaner production system as an attempt to further improve productivity. So, a CONWIP system is simulated.

5. LINE BALANCE ANALYSIS

The scope of this chapter is to re-design the Pallet Factory line balance, improving the overall performance of the line. To do so, different balancing alternatives are compared to select a best alternative for the described process. Then, a simulation model is created to compare the current balance with the proposed one based on average utilization of resources and output rate and also to test how the proposed balance reacts to variability.

As reviewed in the bibliographic study, literature introduces some operation research approaches to deal with questions addressed in this paper. However, all the mentioned solutions apply to "closed form systems", such as single flow lines or reliable identical parallel machines. Since the problem here presented concerns a company with intricacies, the mathematical modeling must be then analyzed in a more qualitative way. Mathematical solutions for balancing are not perfect once they do not consider real limitations such as logical sequence, crew work continuity, number of employed crews, work crew size, project resources, work progress rate, workflow direction and learning phenomenon.

Therefore, the next sections present different option of balance and then a qualitative analysis of the above-mentioned limitations is performed to identify the best solution proposition. Then, a simulation model compares the effects of the proposed balance on the pallet process.

The compared scenarios are:

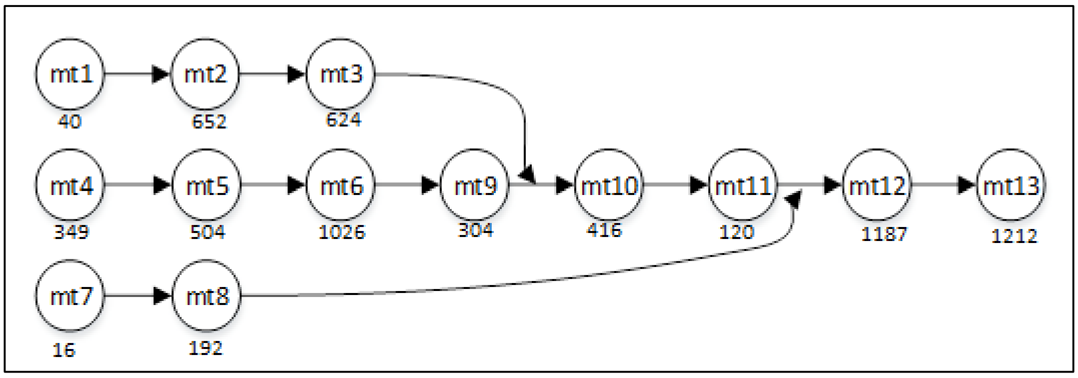
- Scenario 1 is the current state described on the previous section
- Scenario 2 is a balance that has as an objective function the minimization of the sum of idle times
- Scenario 3 groups together tasks based on the entity processed. In practice three "separate" lines are considered.
- Scenario 4 groups tasks based on the type of activity performed.

The scenarios are then compared based on idleness, utilization rate for the operators, flow of materials and people and workflow continuity.

The scenarios model the steel pallet process as a single model with deterministic variables (simple SALB) accordingly to Gosh & Gagnon's classification. Then, a sensitivity analysis is performed when variation on processing times is included. Although the pallets may be produced in different sizes, negligible setup times are considered, thus, the system can be treated as a single line model.

As a simplifying assumption, the transportation times were not considered, this is a fair assumption once the total transport time accounts for less than 2% of the total production time. The balance of the line as mentioned in Chapter 2 has as an input the precedence relations and the required tasks time. For the pallet, the task diagram is represented on Figure 21. This picture will be used as reference to explain station divisions for all scenarios.

Figure 21: Task sequence



5.1. *Scenario 1: the current state*

The current line balance presents a mix of logics. The tasks related to the sheet processing are grouped together, but the other tasks are grouped based on the logic of competences, meaning that all welding tasks, for example, are performed in a single station.

To evaluate the actual scenario, the required cycle time for the line must be used as a comparison parameter. Given a demand forecast, to find the required cycle time, one must take the available production time and divide it by the desired production rate. The resulting number must not be smaller than the highest task time, once it limits the speed of the line.

$$CT = \frac{Available\ time}{Expected\ demand}$$

Based on the historic demand, the average demand is used as a parameter, once no trends are observed. The plant must produce at least 394 pallets per month, the given demand rate is 19,8 pallets per working day, considering 5 working days per week and 4 weeks per month. One working day is composed of 8 nominal productive hours, excluding half an hour due to breaks. Therefore, we have:

$$CT = \frac{7{,}5\, h/day}{19{,}8\, pallets/day} \cong 0{,}38\, h$$

which is equivalent to 1363 seconds.

For the actual scenario, the resulting distribution of tasks as well as the times are represented on Figure 22 and 23.

Figure 22: Distribution of tasks in stations

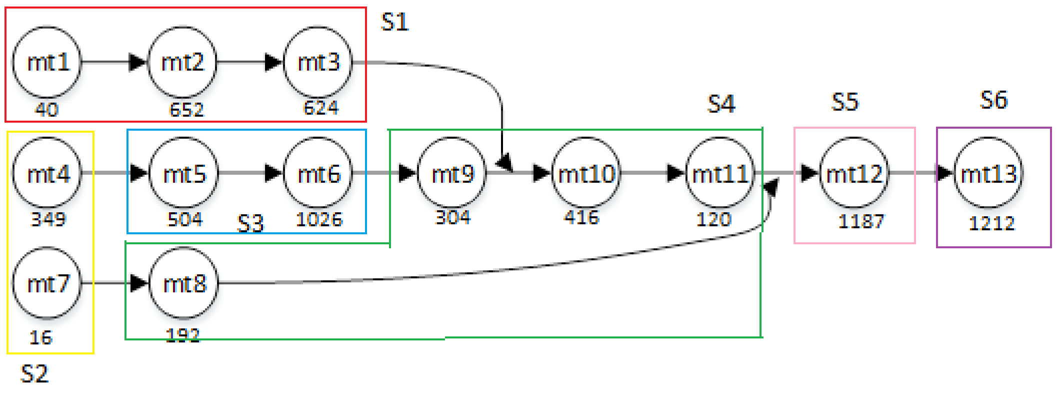


Figure 23: Original line balnace

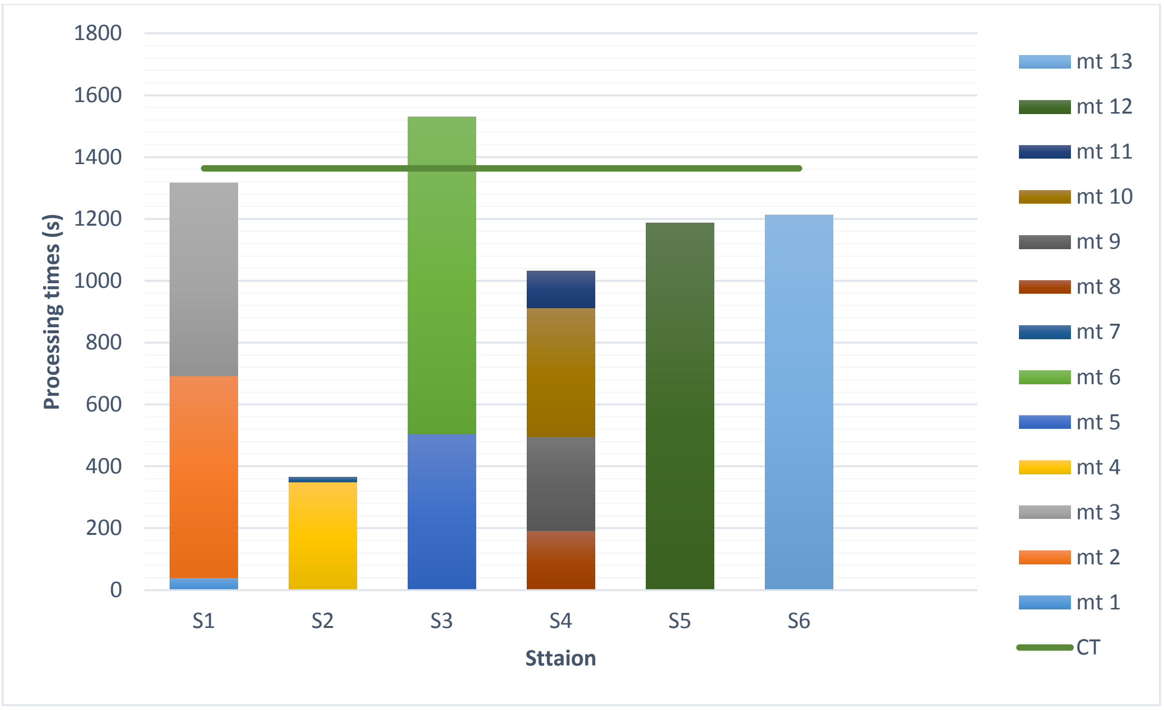


Table 2: Station processing and idle times for scenario 1

Station	Station processing time	Idle time
S1	1316	214
S2	365	1165
S3	1530	0
S4	1032	498
S5	1187	343
S6	1212	318

Based on the above balance we can infer that the real cycle time for the plant is 1530s, which is larger than the required cycle time, given by the expected demand.

Another important indicator used to define the scenario is idleness. Hax & Candea (1984), defined the idleness d of a line balance as:

$$d = Nc - \sum_{i=1}^{N} p_i$$

Where c is the real cycle time, and p_i (i=1,…, N which is the number of stations) is the processing time for station i, including receiving, loading, processing and unloading times.

With the above metric, it is possible to define an idleness coefficient according to the equation:

$$d\% = \frac{Nc - \sum_{i=1}^{N} p_i}{Nc} \times 100$$

Applying the presented idleness formulas on the system studied we get $d = 2538\,s$ and a $d\% = 28\%$ for the original scenario.

Another similar indicator for line balance efficiency is the smoothness index (SI). It is similar to a standard deviation calculation, being defined as:

$$SI = \sqrt{\sum_{1}^{N} (CT - p_i)^2}$$

For the current state the smoothness indicator of the line is 1367s.

5.2. Scenario 2: minimize idleness

The second scenario uses an optimization function that reduces idleness. To perform the line balance two input variables are needed. The first is the required cycle time (CT) already defined as 1363s. The other is the required number of stations. To calculate its value, the sum of task times is divided by the calculated cycle time, and the resulting value is rounded up.

$$Number\ of\ stations = \frac{\sum Task\ time}{CT} = \frac{6642\,sec}{1363\,sec} \cong 5\ stations$$

Table 3 presents the inputs for our optimization model.

Table 3: optimization model inputs

Daily working hours	7,5
Max Production (units/day)	22,3
Weekly Demand (units)	99
Daily demand (units)	19,8
Cycle time (s/unit)	1363
Theoretical number of stations	5

Once the model has all desired inputs. The problem is modeled using an objective function for minimizing the difference between the required cycle time and actual processing times for all stations. To solve the problem a 0-1 linear programming tool is used. As mentioned, the model must minimize the sum of stations' idle times satisfying three main constraints:

1. Each task must be allocated only once
2. The sum of times per station must not be larger than the calculated cycle time
3. All the precedencies must be respected.

Considering *CT* as the calculated cycle time, *i* the index for tasks, *j* the index for stations, x_{ij} a binary variable that assumes 1 if task *i* is allocated to station *j* otherwise it is 0. The time of each individual task is t_i. Another artificial variables created to enable the model is the matrix <u,v> to help organizing the precedence relations. Consequently, we have the objective function:

$$Min \sum_{i=1}^{n} CT - t_i x_{ij}$$

Subject to:

$$\sum_{i=1}^{n} x_{ij}\, t_i \leq CT$$

$$\sum_{j=1}^{k} x_{ij} \leq 1$$

$$\sum_{j=1}^{k} x_{uj}\, j \leq \sum_{j=1}^{k} x_{vj}\, j$$

To solve the problem above, the SIMPLEX algorithm was applied, through the Solver tool of Excel.

The first trial of the problem, using as an input the calculated CT as well as the theoretical number of stations, didn't present a feasible solution. The theoretical number of stations does not consider the extra restrictions related to the precedence relations. So, to solve the problem the number of stations was increased by 1 in order to relax the constraints. With the new input the problem could be answered.

Figure 24: Line balance for scenario 2

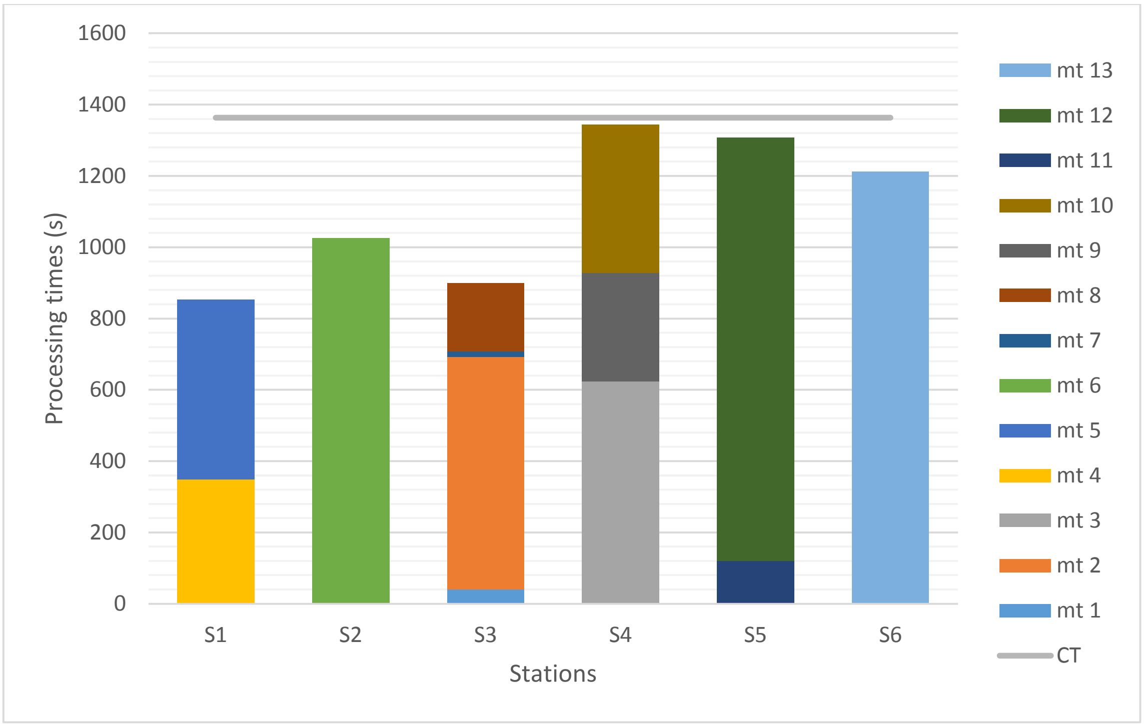
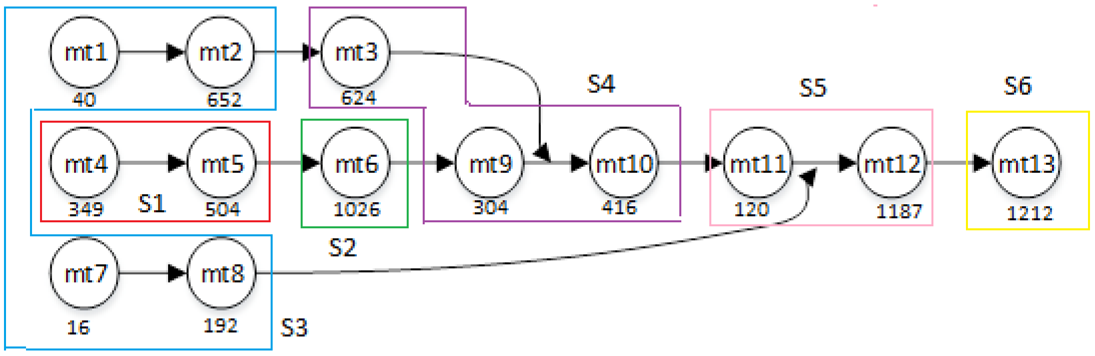


Figure 25: Line Balance for scenario 2



The proposed balance presents $d = 1422\ s$ and $d\% = 17\%$, which represents a significant improve in terms of idleness. The smoothness of the line also improves reaching 747s. However, the resulting balance seems to be inefficient regarding workers and material flows. On figure 25 it can be seen that the allocation of tasks doesn't seem efficient once tasks from different lines are shared by the same station.

For scenario 2, the suggested work organization must respect a few constraints. First of all, stations that perform any welding activity must have at least one operator specialized in welding (Op. 5, 6 or 7). Secondly tasks mt1 to mt3 and mt11 and 12 require at least two workers to be performed.

5.3. Scenario 3: material flow separation

A second approach considered was to separate production in different lines, depending on the material processed. So, there would be three product lines to be balanced, one that processes sheets, the other angles and one for profiles.

Such approach was considered because specializing workers according to material flows diminishes the amount of time dedicated to transport activities, thus reducing total time and costs. Besides, it can improve the distribution of flows presented on the spaghetti chart.

To perform such balancing the required cycle time remains the same, 1363s. However, this time the number of stations was calculated based on the sum of times for the line considered. Therefore, for the line that processes sheets only one station is needed, since the sum of times is smaller than the required cycle time. The same happens for the line that processes angles.

For the line that processes profiles the sum of task times is equal to 5118s, which divided by CT results on a rounded up total of 4 stations. Here, once again the for the arithmetic number calculated the problem presented no feasible solution. So the increment of one station relaxed the problem.

Once the number of stations per line was defined, the line balance can be solved on the bases of the same previously presented constraints. The results are shown on Figure 26.

Figure 26: Line balance scenario 3

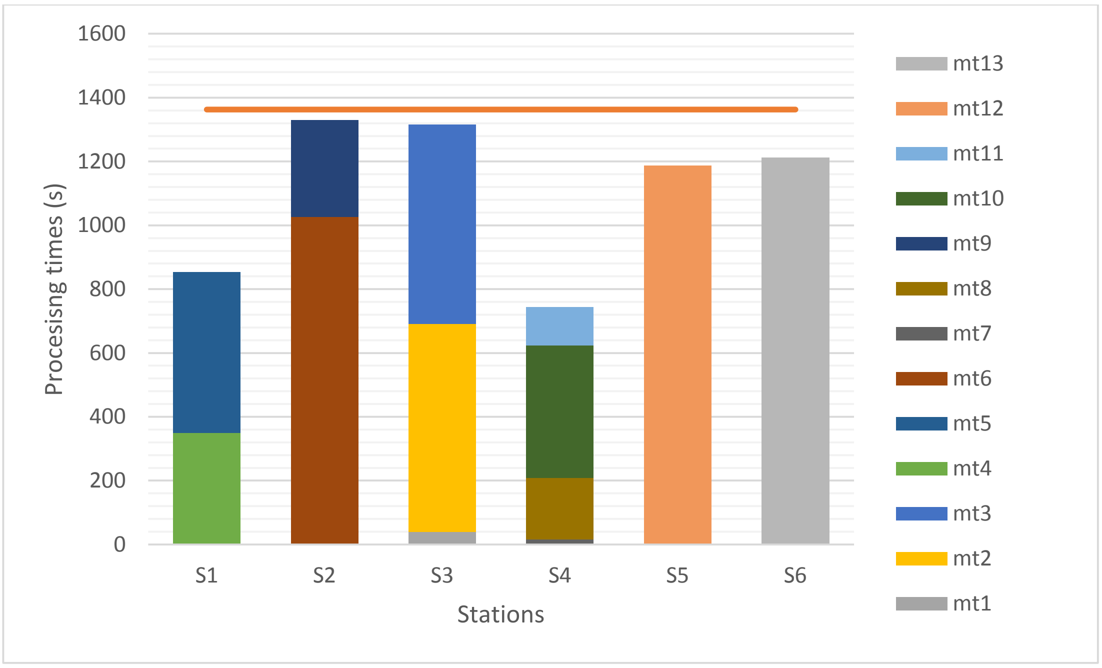
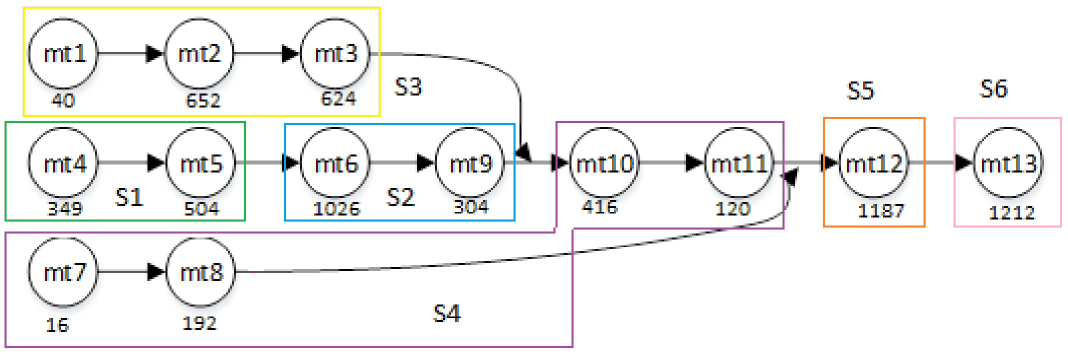


Figure 27: Line Balance scenario 3



As mentioned this proposition focuses on improving the internal flow of materials in the plant. Results show the new d = 1338s and the d(%) factor is now 17%. Regarding smoothness, SI=778s.

5.4. *Scenario 4: grouping by competence*

The final scenario performs the line balance by grouping tasks which require the same competence/set of machines. Here a different kind of optimization is needed. There needs to be a fictional “cost control” that penalizes inefficiencies.

To do so, a matrix of scores n(x,y) was created, where each relation among tasks x and y are established. If x and y are correlated, a low “cost” is set, otherwise a high cost is set. After the creation of the cost matrix, the algorithm follows basically the update first fit heuristic, where among the available tasks (whose immediate predecessor have been assigned) the one with lowest cost is allocated.

The resulting balance is given by Figure 28.

Figure 28: Line balance scenario 4

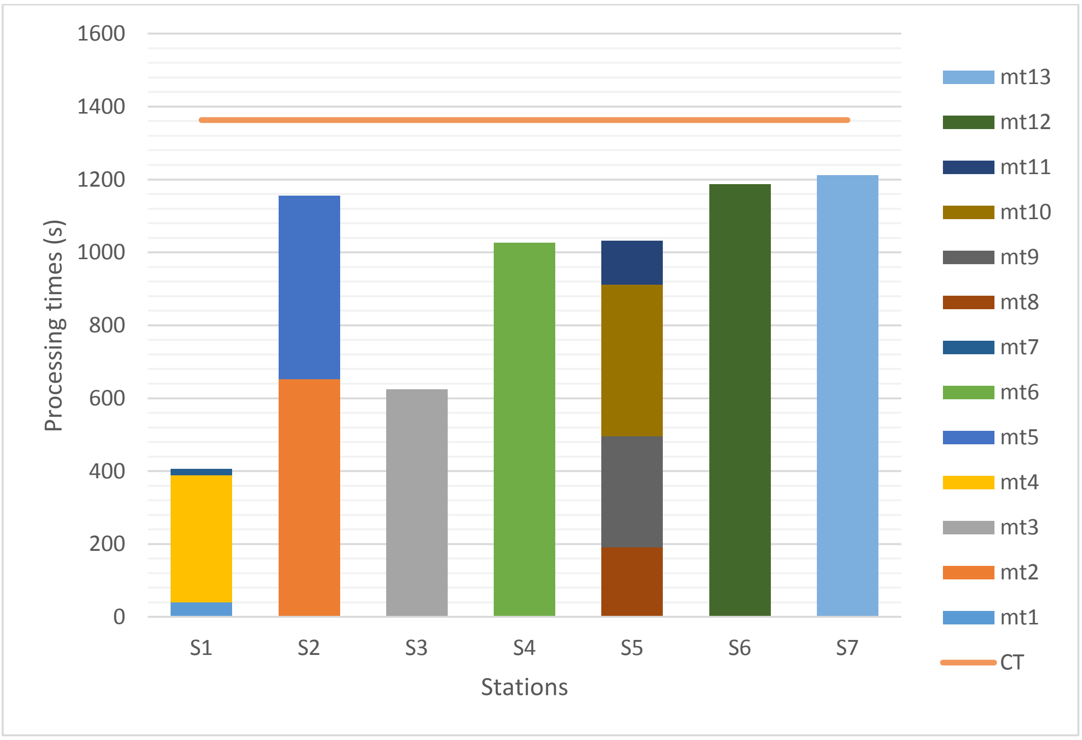
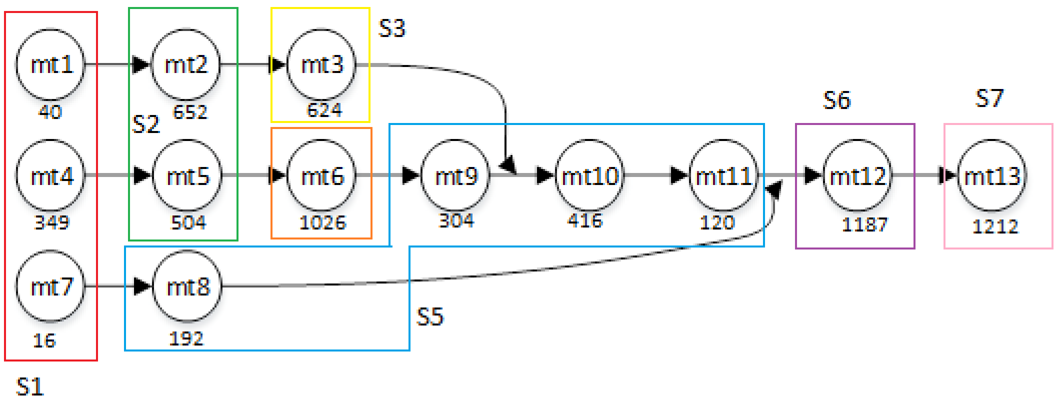


Figure 29: Line Balance scenario 4



Here d = 1842, d% = 32% and SI= 1033. Results show a significant decrease in indicators once more stations were created, and the fictional cost function was prioritized over idleness.

5.5. *Comparison of scenarios*

Once the scenarios are defined they will now be compared based on the already set performance measurements. The analysis considers also qualitative matters such as work continuity.

As seen before, the original scenario lacks in performance since the required cycle time to satisfy the average demand is not achieved. So, currently there is the need to employ extra hours in production, which decrease the line efficiency. Among the scenarios this is also the one that presents the highest level of idleness, and the worst distribution of times.

The second scenario presents significant improves in terms of real cycle time and idleness. Here CT decreases by 12%, enabling demand covering without the requirement of extra hours. However, this scenario presents the downside related to flows. Here sheets profiles and angles sometimes share the same station. So, there is no work continuity, affecting material and operators' flows. Therefore, the real application of such solution is unable.

The third scenario, when compared to the current, also presents enhancements regarding indicators. Although the smoothness index indicates that times distribution is not as homogeneous as scenario 2, this one presents better levels for all idleness indicators. Another advantage is that it reduces the problem of workers dealing with different materials. Here for each station one type of raw material is solely processed, creating fluidity.

The final scenario is the one that presents smaller cycle time. On the other hand, that indicator is reached through the increase in the number of stations. Although this scenario probably minimizes operators flows, it presents higher levels of idleness and is less homogeneous in terms of time distribution than the previous.

Based on the given performance indicators it can be conclude that the third scenario presents the best levels of performance and of work continuity. Representing a 13% improvement in terms of cycle time, as well as a reduction in idleness.

Figure 30: Comparison of line balance proposition

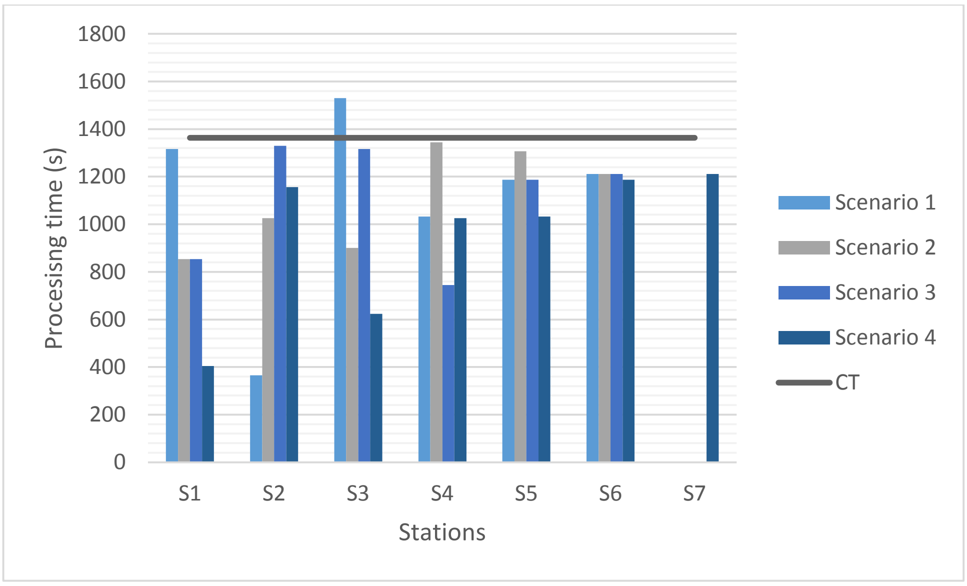


Table 4: Scenarios comparison

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Real CT (s)	1530	1344	1330	1212
d (s)	2538	1422	1338	1842
d(%)	28%	18%	17%	22%
SI (s)	1367	747	778	1033
Stations	6	6	6	7

The numbers above indicate that the line would improve significantly from the current condition. However, the proposed cannot be applied in the actual line, thus it is important to perform a simulation to test for additional consequences.

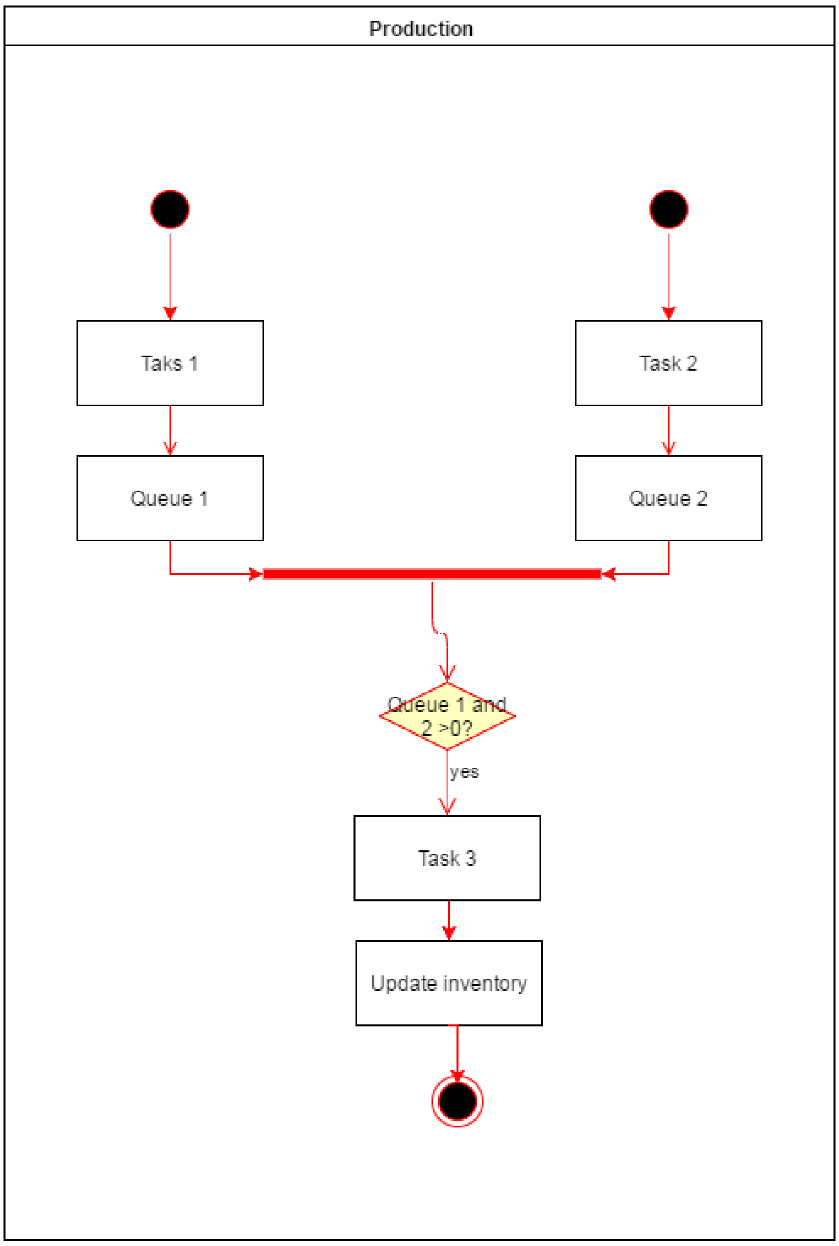
5.6. Simulation model

This section describes the simulation created to model the Pallet Factory using the two different line balances, the current scenario, and the chosen proposition.

The model contains only one thread which regards production. Items are created and processed, going from one task to the next, waiting in intermediary queues when needed. If the next task is an assemble operation, the two precedent queues must be non-negative to continue with the processing. After the item has passed through all tasks on its path it goes to the final inventory, which is updated with the new value in stock.

The described relations were processed for all workstations of the pallet factory, however, for sake of simplicity to illustrate the relations among variables a simple diagram for a three processes assembly line is illustrated on Figure 31.

Figure 31: Conceptual model diagram for a three stages system following a push logic.



Regarding the Arena logic, there are three create modules, one for each type of entity (sheets, profiles and angles). The entities created enter processing modules that follow a “seize delay release” logic. That means that for each task, resources are seized for the time duration of that process, and cannot be used for any other action. Also, the entity is delayed by the same amount of time. After the time has passed both resources and entity can be released, and the entity will flow to the next process.

For each process the human resource (operator) as well as the machine, if applicable, are bound to the module, which can only operate if all resources are available. The seizing time counts as value added time.

Each process module has a queue in which entities wait if the capacity is occupied. The queues follow a first-in-first-out logic. This logic can be applied once setup times for changing between product types are considered zero on the assumption that the products are similar.

For assembling activities, when two different entity types are bound together, the model uses a match module, which brings together a specified number of entities waiting in different queues. The match may be accomplished when there is at least one entity in each of the desired queues. After the match, there is an assign module which changes the entity type so that assembling activities can be traced.

When all tasks have been performed, there is an update in the final inventory, and statistics are computed.

The full model code containing all used specifications and inputs is on Appendix A.

To compare the proposed line balances, the resources bounded to each task changed to follow the balancing logic. It is important to mention that many tasks cannot be performed by one operator due to the weight of the material, or other reasons. Therefore, one station may not necessarily relate to one operator, but to a par of workers.

5.6.1. Verification of the system

Verification of a model is the process of confirming that it is correctly implemented with respect to the conceptual idea. Here two types of verification were used: firstly, the model was analyzed by the tutor of this project, who is an expert in simulation modeling. Secondly, the project assessed the execution of the model with simple inputs, which were not related to the real data. With such tests, it was possible to observe if the model was replicating the expected behavior, given the predictability of the inputs. Performing such tests also help with the visualization of the interaction between the workstations, entities and variables.

5.6.2. Validation of the model

As mentioned earlier, the line-balancing model was developed in the software Arena, which offers limited functionality in its student version for building and executing training-size models.

With respect to building the model, the software's student version proved to be sufficient to represent the production line in terms of flowchart blocks and variables, since the Pallet Factory is relatively simple in terms of phases and resources. However, the student version of the software constrains the number of simulation entities running in the model.

So, in the regular push system, the simulation could run only for a limited amount of time before the number of entities was reached. To solve this issue a type of entity "recycling" was applied. After each branch of the model before the assemble, a SEPARATE module was introduced. The separate creates duplicates of the entering entities and sends them to the beginning of the process. Along with that the amount of creations on the CREATE modules were limited.

This measure solves the problem of limited running time but disorders the inventory levels for the push system, since it creates a virtual limit on inventory. This aspect creates differences among the model and the real system. Another characteristic that prevented the validation was the difficulty for gathering the company's data once the communication channel became distant.

This situation points out a relevant restriction for the use of simulation, the high dependence on paid software. There are free alternatives in the market, however they present limitations in terms of capacity, simplicity to use and availability of features.

Some authors, to mention Mesquita & Goldemberg (2006) used a simple and well known toll to simulate production systems: Excel Spreadsheets. That is a feasible solution for smaller problems. However, managing time control through VBA for more complex systems is not an easy task. This paper first attempted to do so, but the number of variables required was too big creating additional unnecessary difficulty.

5.7. Results of line balance proposition

Results prove that the proposed scenario 3 introduces significant improvements to the line. In terms of utilization of operators, the average utilization rate increases 10%, thus the plant is able to produce more units given the same simulation time.

Figure 32 represents the average workers' utilization for the simulation running time, that is a total of 100 days, including warm-up time.

Figure 32: Average utilization rate for workers

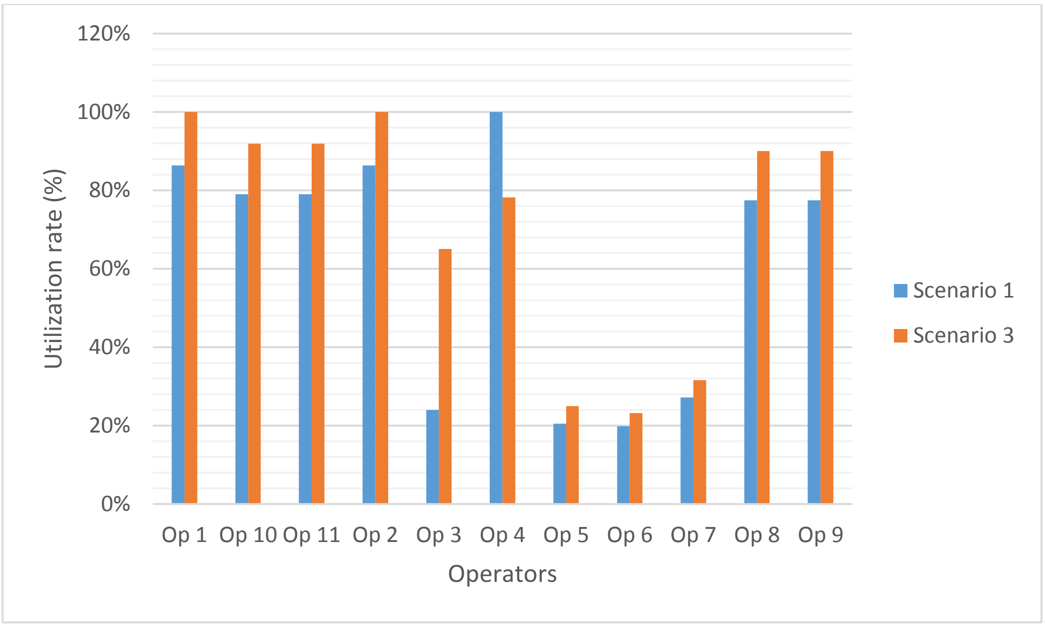


Table 5: Comparison of workers' average utilization rate

	Scenario 1	**Scenario 3**
Op 1	86%	100%
Op 10	79%	92%
Op 11	79%	92%
Op 2	86%	100%
Op 3	24%	65%
Op 4	100%	78%
Op 5	20%	25%
Op 6	20%	23%
Op 7	27%	32%
Op 8	77%	90%
Op 9	77%	90%
Average	62%	72%
Std. Deviation	30%	29%

For the current scenario, the average utilization for all workers is 62% with standard deviation of 30%. Meanwhile for the proposed line balance utilization increases to 72% and standard deviation doesn't change so much, becoming 29%.

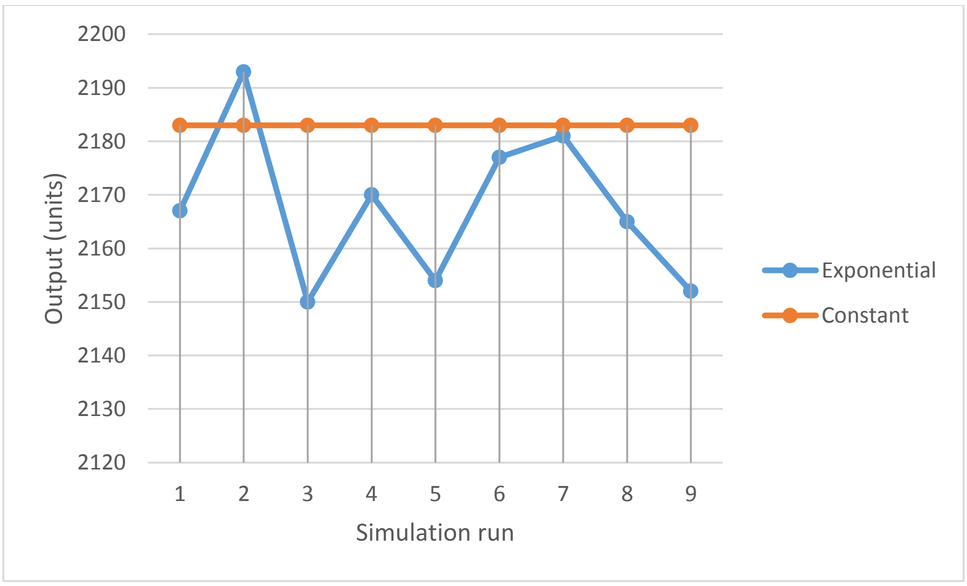
Such increase in workers' utilization is reflected on total production. In the first scenario, the total output after 100 days was 1878 pallets. For the proposed line balance the number increased by 16% reaching a total production of 2183 units. Such improvement is also reflected on the time between products. Currently around 25 minutes pass between the exit of two consecutive pallets. If the change is implemented simulation points out that the time could reach a little less than 21 minutes,

The tradeoff of the above-mentioned improvements is that a similar increase is observed in terms of WIP. That is natural and coherent with Little's Law.

The analysis performed so far used the mean of the processing times as deterministic values. A second step on the evaluation of the scenario proposition is to perform a sensibility analysis when variation is introduced in the process. To do so, in the simulation model, the values of the processing times were changed first to fit an exponential distribution with lambda equals to the inverse of the previous used mean.

As expected, the output levels, on average, decreased when variation was introduced. However, the total production after the 100 days of simulation presented only a 0,7% drop.

Figure 33: Sensitivity analysis for changing processing times distribution



The most significant change regards WIP levels. When the times were constant WIP remained stable around 32 units. As expected, for the exponential distribution of times the levels of WIP presented a much larger variation, and its average increased to 49 units. This enlightens the importance of the application of an inventory control.

6. WORKLOAD CONTROL PROPOSAL

Once the proposed line balance could provide a feasible solution for cycle time reduction, we now focus on evaluating improvements regarding the operations. To address the problems related to workload two control systems based on restricted work-in-progress are tested and compared to the current scenario. The first proposition is a single stage CONWIP system, and then a multi-loop CONWIP.

Those two options were considered in opposition to the Kanban because in the literature review it was seen that many studies have already proven the superiority of CONWIP over Kanban, e.g. Pettersen & Segerstedt (2009) and Hopp & Spearman (2008). Not to mention that in the context of the Pallet Factory the simplicity of CONWIP could be an advantage since it requires setting less stages of cards. Besides, Kanban is more appropriate for MTS systems, since the factory is an MTO, CONWIP is considered more adequate.

Consequently, this paper decided to compare the actual system (push), a single loop CONWIP and a multi CONWIP system in order to see which one could be more efficient given the Pallet Factory scenario. Simulation is used to test and calibrate the control systems. The software used was Arena, a discrete event simulation software by Rockwell Automation.

The models contain the following assumptions:

- The plant does not run out of raw material
- Demand for the product is infinity so that we can evaluate the full potential of the scenarios.
- Machine breakdowns are not considered
- The model assumes that there is no scrap or rework.
- One-piece flow, meaning that material movements occur in batches made of only one piece.
- The setup times for changing between product types are considered zero on the assumption that the products are quite similar.

The total running time used to perform this comparison was arbitrarily set to 500 hours, warm up or start up times of 50 hours are included in the total running time.

Our study does not consider the use of cards, instead it controls directly WIP levels. So, the modeled system is not technically a CONWIP, instead inventory levels between machines are designed to follow a CONWIP approach.

The different scenarios were compared based on five performance measures:

- Flow time: the amount of time an entity takes between entering and exiting the system
- Cycle time: time between two finished goods exit the system
- Throughput: The number of items produced during the total time of the simulation
- WIP level: Total number of entities in the system
- Average utilization rate: the amount of time a machine is adding value. (Idleness is a symmetric indicator).

The simulation held for the push system is identical to that used to perform the line balance on the previous chapter. Therefore, to be succinct, the following sections introduce only the simulation models for the two different configurations of CONWIP, the single and multi-loop. Then all three are compared on the results analysis.

6.1. CONWIP production system model

As explained in the literature review, the CONWIP is a methodology where the work-in-process (WIP) is not constrained at every operation or machine, instead the level of WIP in a total production flow is constrained. The production flow may consist of several operations or machines and not just one machine.

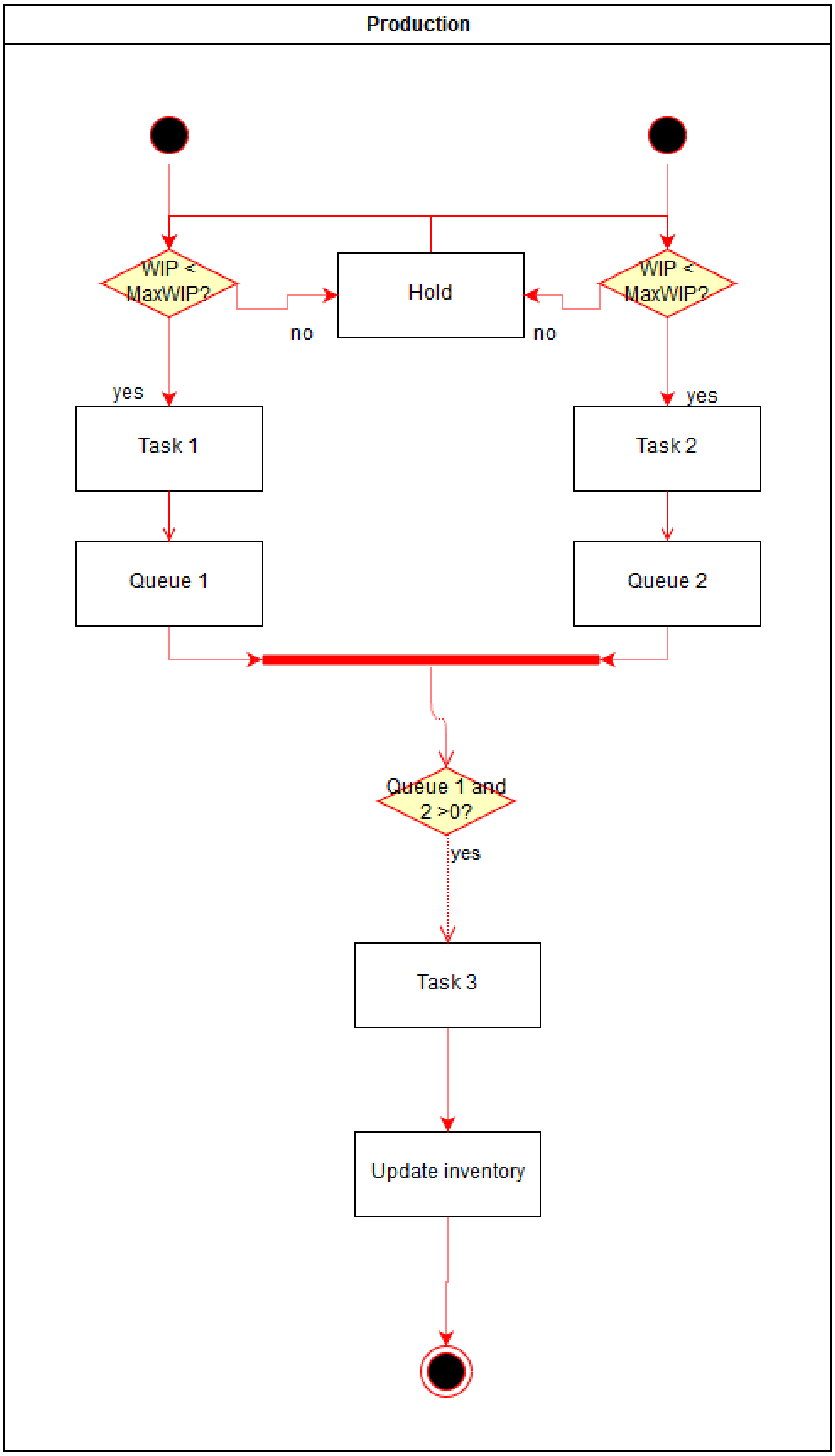
To simulate the system following a CONWIP production, additional relations among the workstations are necessary. As mentioned, the model does not simulate a card control system. In its place inventory levels are controlled. So, technically the system is not a CONWIP, but inventory is controlled based on a CONWIP logic. With that being said, the level MaxWIP is the controlled variable which represents the restricted maximum amount of WIP in the system.

It is worth mentioning that here WIP is measured in units (or entities in the system), since the case in study produces one product. In multiproduct systems, however, it can be measured by time in the system, or even in monetary units. Hopp & Spearman, (2008).

Once the total WIP inside the system falls below the set $MaxWIP$, an authorization is given for liberating more work inside the line. New created items will only enter the process if this condition is satisfied. Besides that, the line behaves as a normal push system, so it is expected that WIP will accumulate before the bottleneck.

Figure 34 shows a conceptual model with three workstations that follow the CONWIP logic.

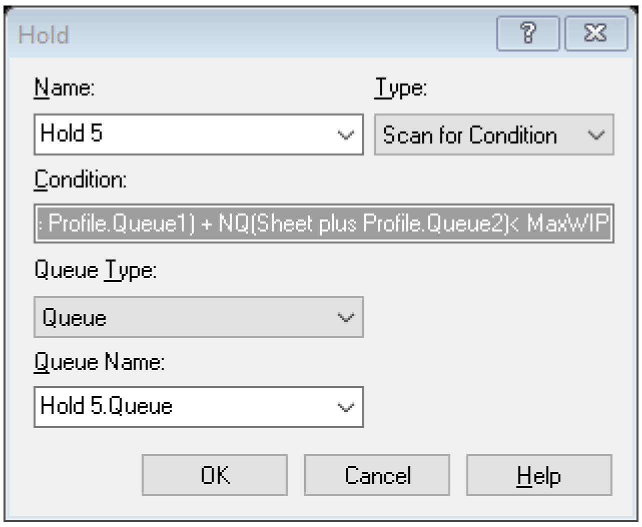
Figure 34: Model diagram for a three stages CONWIP system.



Source: Created by the author

The application of the above logic on Arena starts with CREATE modules, there is one for each entity type, profiles, sheets and angles. Then, to guarantee that created items don’t enter the system before authorized HOLD modules are present after the creation modules. They are governed by an expression equivalent to Total WIP < MaxWIP. If such condition is observed the module liberates entities into the system. Being the MaxWIP a variable which represents the WIP cap. Figure 35 shows a HOLD module property window in Arena.

Figure 35: HOLD module control painel



Name: Hold 5
Type: Scan for Condition
Condition: : Profile.Queue1) + NQ(Sheet plus Profile.Queue2)< MaxWIP
Queue Type: Queue
Queue Name: Hold 5.Queue

Source: Screen shot taken from Arena 14.7

In addition to the HOLD modules after each CREATE, which control the liberation of orders into the system, the production process itself follows the same logic as the push system.

The full model code containing all used specifications, inputs and the organization of the modules is on Appendix B.

Once the model is set the CONWIP system must be dimensioned. That means that the maximum amount of WIP in the system must be defined. To do so, two methodologies have been tested. The first is based on the strategy adopted by Pettersen & Segerstedt, (2009). They incrementally increased WIP levels and evaluated the resulting lead time and throughput.

Following the same logic, for the model here presented the variable MaxWIP was incrementally changed, and the resulting number of outputs and time between finished goods were computed. By analyzing the results it can be seen that the highest output level, or symmetrically the lowest time between products is given for a MaxWIP of 30.

Table 6: Obtained results for incrementing MaxWIP

MaxWIP	**Output**	**Time between products (h)**
5	340	1,4534
10	365	1,3746
15	977	0,5100
20	1110	0,4496
25	1106	0,4513
30	1176	0,4242
40	1193	0,4178
50	1174	0,425
75	1170	0,4264
100	1181	0,421

Figure 36: Variation of output for changing MaxWIP

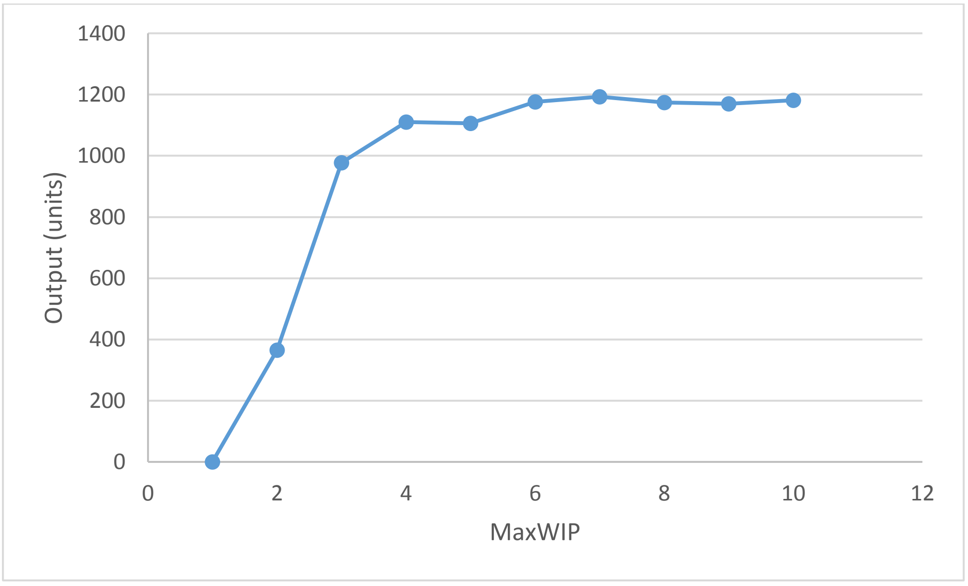
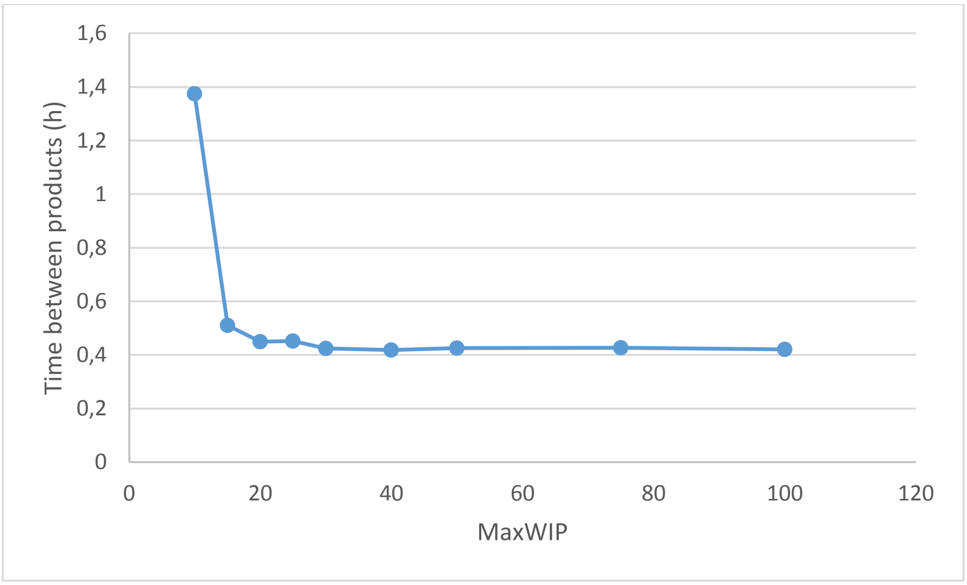


Figure 37: Time between products for changing MaxWIP



The second method used to test the quantification of the restricted WIP was based on the one used by Marek, Elkins, & Smith, (2001). They start with Hopp & Sparman's formula

$$TH(w) = \frac{wr_b}{w + W_0 - 1}$$

Which defines throughput as a function of the number of cards (in our case pieces) in the system. The number of cards w can be obtained by inverting the given formula, resulting:

$$w = \frac{TH\ (1 - W_0)}{TH - r_b}$$

The variable r_b is the rate of the bottleneck workstation in jobs per unit of time. After running the model, the workstation with highest processing time (or value added time VA) is the assembly with an average processing time of 0,34h/piece, as indicated on Table 7 Thus the bottleneck rate r_b is given by ${}^{1}/_{0,34} = 2{,}94$ pieces/hour.

Table 7: Values of average time per workstation

	VA Time	**Wait Time**
Angles cutting	0,00	0,00
Angles welding	0,08	0,00
Assembly	0,34	0,02
Bottoms welding	0,03	0,01
Building up	0,33	0,13
Cup welding	0,05	0,00
Hole drilling	0,28	0,78
Profile cutting	0,10	0,02
Profile incision	0,14	0,79
Profile welding	0,12	0,00
Sheet bending	0,17	0,14
Sheet corners	0,18	0,18
Sheet cutting	0,01	0,15

The next step is to define the critical WIP W_0, which is given by $W_0 = r_b T_0$, being T_0 the sum of average processing times for the critical path, ignoring any processing time variability. In our case, for the pallet factory, this number is 5118s, or 1,42h. So, W_0 is 4,18.

The throughput must be equal to or larger than the one observed in the push system, besides it need to cover the average weekly demand of 99 pallets. Therefore, the chosen value for the throughput was 2,5 pallets/hour, which is the rate that satisfies both conditions.

Using Hopp and Spearman's formula and solving for the unknown value of w the value of restricted WIP can be estimated. The resulting number gives us 18.

The two methodologies presented slightly different results. This paper chose to follow the first method, since it is more tailored to fit the Pallet Factory's specificities.

6.2. Multi-CONWIP production system

The multi- CONWIP is an extension of the regular single loop CONWIP, here, more than one control loop is created. On the edge, when the number of loops reaches the number of workstations a Kanban equivalent control system is achieved. So, in practice, the multi-CONWIP is in between the two other control methods.

This paper chose to control the three different intermediary lines plus the assembly stage as loops for the multi-CONWIP, that is, one control loop for profile, another for sheets, one for angles and a last one for assembling. To do so, after entities are created, there is a control which checks if the amount of WIP inside the branch is smaller than a defined $MaxWIP_i$ for that control loop. If the current value is smaller, the created entity is authorized to enter the system, else it remains on hold until the condition is satisfied. Regarding the assembly loop the mechanism is analog, but the HOLD module bocks the subparts from entering the assembly stage.

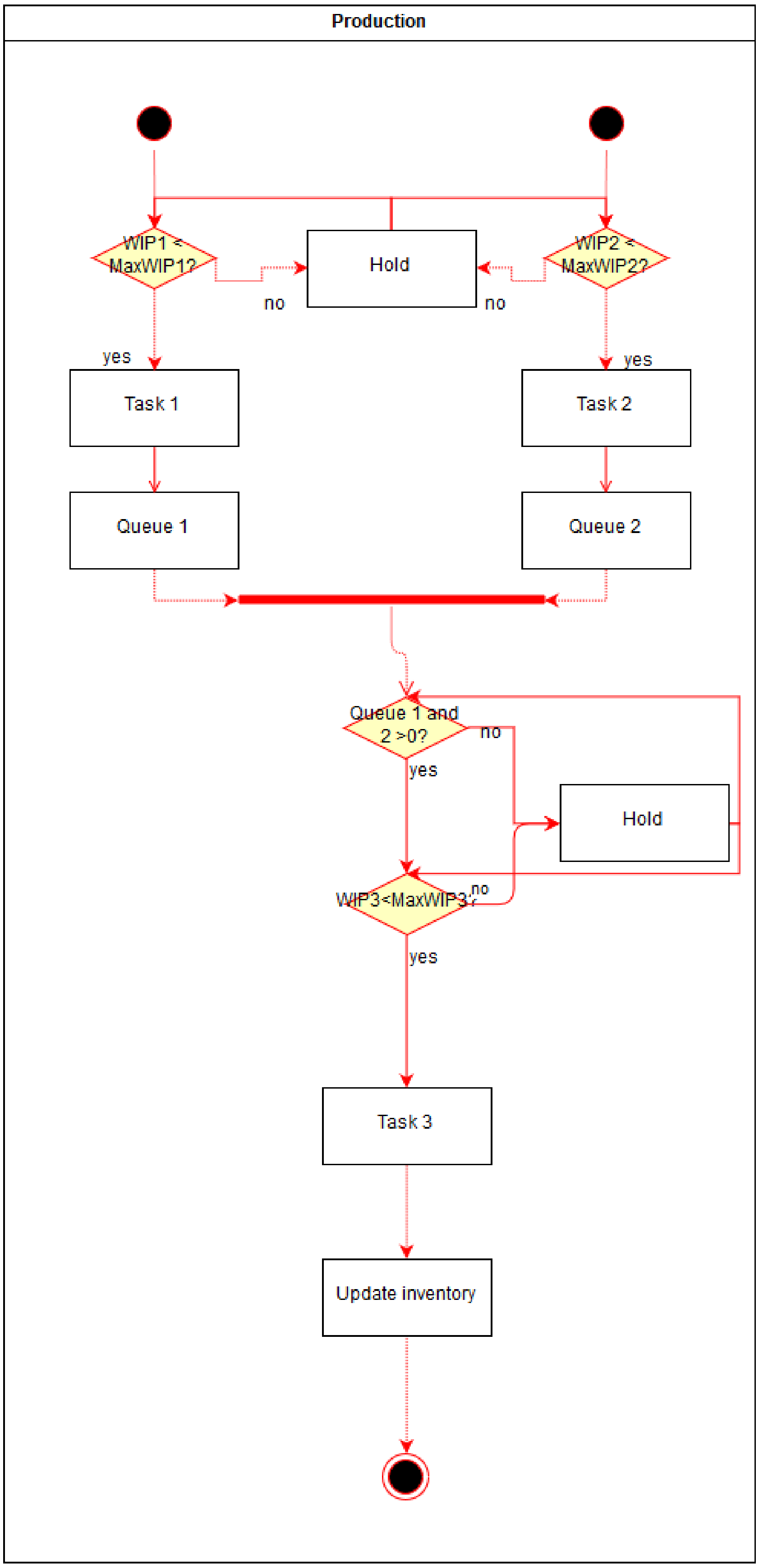
The main difference between the multi-CONWIP and the single-CONWIP is that meanwhile the latter presents the same equation controlling all CREATE modules, the former has four different control equations, which are:

- $Sheet\ bending.WIP + Sheet\ corners\ cutting.WIP + Sheet\ cutting.WIP + NQ(Sheet\ plus\ Profile.Queue1) < MaxWIP1$
- $Angles\ welding.WIP + Hole\ drilling.WIP + Profile\ cutting.WIP + Profile\ incisision.WIP + NQ(Sheet\ plus\ Profile.Queue2) < MaxWIP2$
- $Angles\ cutting.WIP + Cup\ welding.WIP + NQ(Final\ welding.Queue2) < MaxWIP3$
- $Profile\ welding.WIP + Bottoms\ welding.WIP + NQ(Final\ welding.Queue1) + Assembly.WIP + Building\ up.WIP < MaxWIP4$

Being, WIP the sum of units in process mode and waiting in the queue, NQ are the number in queue for the MATCH modules, which the software doesn’t count as WIP but must be considered.

Regarding the Arena logic, the multi-CONWIP design is similar to the previous, the only changes are the equations that control the initial HOLD modules. The conceptual model for a multi-CONWIP with three stages is presented on Figure 38. (The full model code is on Appendix C.)

Figure 38: Conceptual model for multi-CONWIP



Once the system was defined, the amount of WIP must be divided among the chosen branches or loops. As previously justified, the total WIP for the Pallet Factory will start off as 30. Now, that amount must be divided among the three loops. The methodology here used to optimize the WIP level will be the one used by Marek, Elkins & Smith (2001)

Such heuristic is applied for the dimensioning of Kanban. This paper, however, will treat each loop of the CONWIP as a Kanban station. The heuristic to reduce the number of cards in the system, starts with the arithmetic result from Monden's formula. This paper, as justified before, will stick to the value of 30, obtained by the incremental increase test. That amount will firstly be divided among the loops. Once the system has been adjusted with those values, the loop with higher utilization rate (bottleneck) is identified. Then, the value for the MaxWIP variable of that branch is incrementally reduced until the initial throughput rate is no longer achieved. The process is then iterated for the next bottleneck loop, and so on until all have been analyzed.

Table 8: Dimensioning iteration for multi-CONWIP

Value of MaxWIP	8-8-8-8	8-8-8-7	8-7-8-8	7-8-8-8	8-8-7-8
Output	1311	1236	1307	1281	1275

Table 8 presents the variation of output given the changes in the variable $MaxWIP_i$. The values are presented in the order 1, 2, 3, 4 that is, the sheets loop, profiles, angles and lastly assembly. The best result observed refers to the initial value of 8 for all loops. Higher amounts of MaxWIP were also tested, not positively influencing output levels.

6.3. Results of control systems comparison

This section presents a comparison of the results obtained with the simulation of the three different control systems, as well as a few observations concerning the relation of changing WIP for the scenarios in which it is restricted. The performance of the scenarios will be evaluated based on output, time between products, average queue waiting time and utilization rate.

The push system presented the worst results. Its average output given an exponential distribution of processing times was 1064 units after 500 hours running time. The value is 11% lower than the CONWIP output and 23% smaller than the multi-CONWIP value which is 1311 pallets, as shown in Table 9. The difference, as expected, is symmetric on terms of time between products.

Table 9: Comparison of experiment results

	Pushed	**CONWIP**	**Multi-CONWIP**
Output (units)	1064	1176	1311
Time between products (h)	0,4687	0,4242	0,3794
Cycle time (h)	0,4699	0,4252	0,3814
WIP (units)	49	23	31

An additional important change concerns the possible gains in terms of average waiting time on queue. Since for the push system no control is applied, queue upstream bottleneck stations tend to increase endlessly. Consequently, the average waiting time on queue for workstations with higher processing times increases significantly. On the MATCH module “Final welding”, for example, the time that cups remain waiting for profiles surpasses 8h.

When any type of WIP control is introduced the scenario changes. Since there are less items inside the production process, the average waiting time decreases. More importantly, the standard deviation declines, which indicates that distribution of WIP among the workstations is more uniform. This is also proven by the average units waiting on queue shown on Figure 40.

Figure 39: Average waiting time on queue

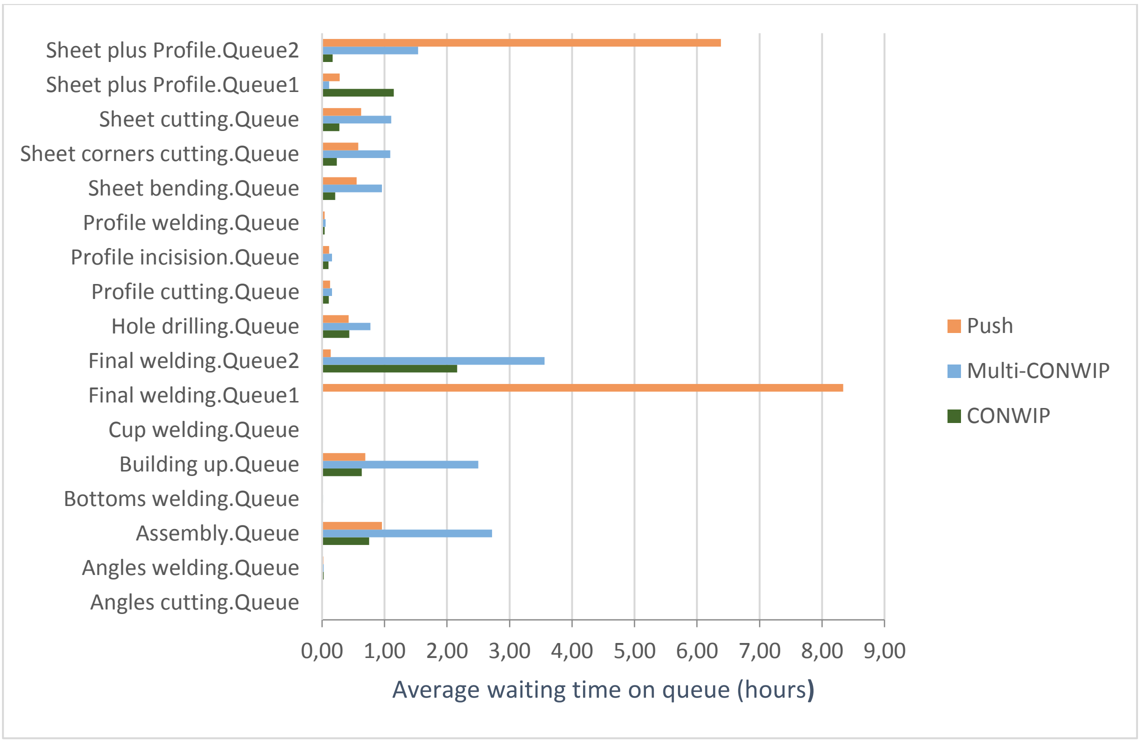
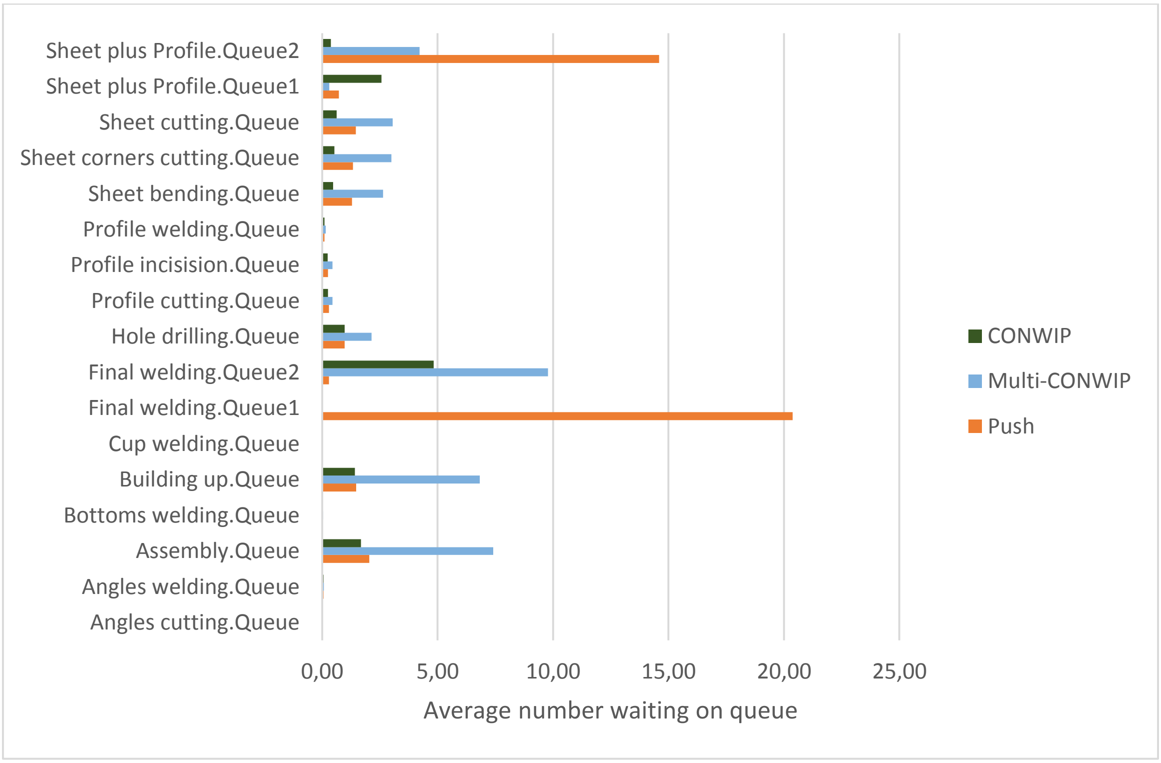


Figure 40: Average number waiting on queue



Hopp & Spearman (2007) present a corollary regarding CONWIP's efficiency when compared to push systems that is in line with what is here presented. They state that "*for a given level of throughput, a push system will have more WIP on average than an equivalent CONWIP system*". Therefore, according to Little's law, for an unaltered throughput given higher levels of WIP, the push system must present higher average cycle times.

Another perceived effect is that for the single loop restricted WIP models an increase in one extra unit on the variable MaxWIP increases outputs and decreases time between jobs out. However, the marginal change in throughput decreases with higher total MaxWIP, and after a given level it reaches zero. This finding is in line with other studies presented in the literature review, such as the article presented by Pettersen & Segerstedt (2009) and Hopp & Spearman's book Factory Physics (2008).

Regarding the marginal change in throughput, the curves present different behaviours given the processing time distributions. When those times are deterministic (have a constant distribution), the results are maximum for any given level of MaxWIP. The introduction of times which follow an exponential distribution generates a graph that tangents below the first one. For the latter, the marginal change in output is also reached later, as seen in Figures 41 and 42.

Figure 41: Variation of output for different processing time distributions

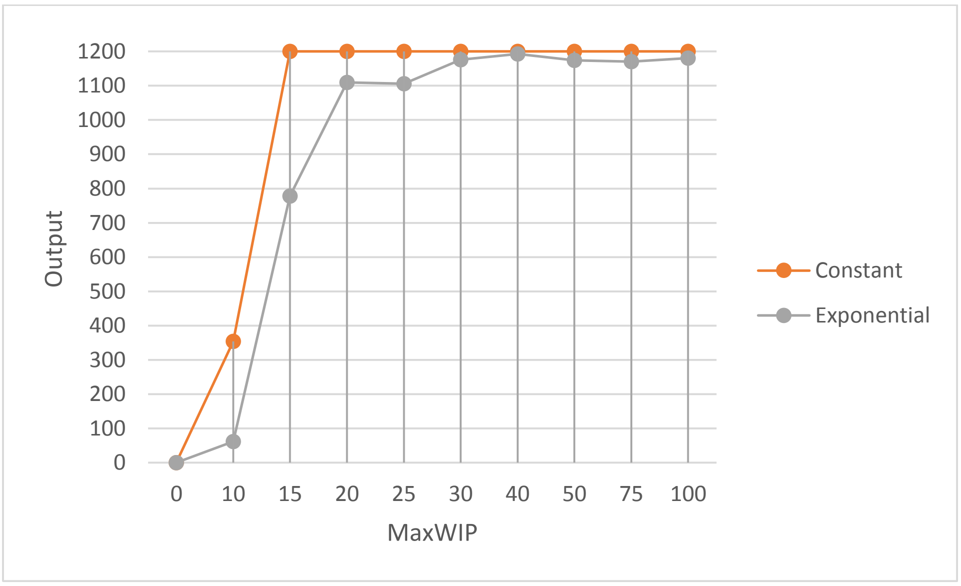
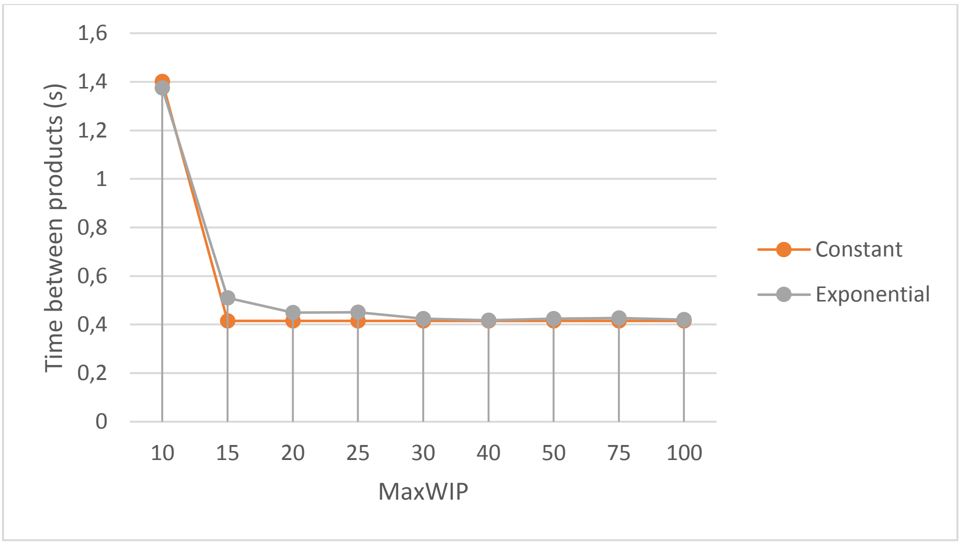


Figure 42: Variation of time between products for different processing time distributions



Concerning the levels of WIP, the practical use of CONWIP or multi-CONWIP cannot explicitly control average WIP, since it varies with processing time of operations, for example. The variable MaxWIP does not represent a CONstant level of WIP, as suggested by the name. Instead the variable represents an upper bound constraint, as can be seen in the WIP graph represented by Figure 43. When times have an exponential distribution, variation in WIP levels increase, but the value is constrained by the variable MaxWIP.

For the multi-CONWIP the levels of average WIP are higher. So, in practice, fixed or sunk costs are higher, besides, more control and storage room may be needed. Under that point of view the CONWIP system is to be preferred. However, in terms of output the multi-CONWIP presented higher levels given the same simulation run time.

7. CONCLUSION AND STUDY EXTENSIONS

This thesis aimed at proposing changes for a Pallet Factory in order to improve its overall performance by controlling its inventory levels. To this end, the project approached two different fronts, the first related to the line balance, and a second more operational, associated to the inventory control system. It is interesting to notice that there are few papers that consider jointly process design and control system analysis to assess and improve a company's current state.

The project began by assessing the Pallet Factory current state, specially focusing on the production process. Then, improvement opportunities were identified. On a first moment, different balancing scenarios were created and compared to the current plant so that an initial improvement in terms of cycle time could be achieved. Then, restricted work in process experiments were simulated to test if different control systems could make the plant leaner, further improving its overall performance.

The experiments here presented showed the importance of controlling work in process levels, not allowing it to increase out of control as in a push system. Those findings are not new, the context of Little's law has already presented that too much WIP, as well as too little, affect throughput. Hopp & Spearman's (1997) have shown that after WIP has reached a certain level, output remains unchanged, so additional units may only increase average time in queue, therefore costs, without affecting throughput levels.

The same finding is proven true for the context of the Pallet Factory. The dimensioning of the CONWIP and multi-CONWIP systems are extremely important for the good function of the system. The simulation confirmed that an increase in maximum WIP could improve the system's performance but only until a certain point. After that the marginal change in throughput equals to zero. So, if WIP goes further than the established number, costs increase without an output rise.

For the Pallet Factory production process, the use of a multi-CONWIP control system presented the best results in terms of output. Therefore, it can be concluded that it is to be preferred over the others. Simulation has shown a possible increase of 27% in terms of total production. However, a real application of such condition is not clear. Although literature regarding the comparison of CONWIP and multi-CONWIP with other systems is extensive, no attention is given to its application on practical situations. (FRAMINAN, LEZ, & RUIZ-USANO, 2003)

It is important to mention that the improvement would not have been so significant without the implementation of a proper balance for the Pallet Factory. Balance flow implicates a lot on the levels of WIP since it helps to contain the indefinite increase of stock before the bottleneck station. A practical difficulty here presented was that the mathematical models commonly applied do not consider important qualitative real matters, such as logical sequence, work continuity, project resources, work progress rate, workflow direction and learning phenomenon. With that being said, this thesis proposed different balancing scenarios for the Pallet Factory. They were compared on the bases of both qualitative and quantitative performance indicators so that a feasible one could be selected.

The proposed process configuration presented solved the problem of the cycle time being larger than the takt time, so that an average demand could be satisfied without the necessity of extra hours. Another improvement was the level of idleness that decreased from 27 to 18%. The increase in utilization was reflected on output, which increased by 16% when compared to the current state.

Left for future studies and extensions are the inclusion of failure modes in the control system models, so that the system may be evaluated when disruptions are introduced. Also, the introduction of a demand tread in the model can include further performance measures such as demand attendance and lateness. The latter option was unable due to the lack of information regarding the company's customers' orders and because the software used only allowed a limited simulation run time.

Another extension possibility regards the practical implementation of the CONWIP or multi-CONWIP. Such matter has been neglected by literature. Pettersen & Segerstedt (2009) point out that Kanban application in practical situations are widely spread. CONWIP, on the other hand, is not obvious. They point out questions about which level of WIP should be adopted and in what way. Another point made by the authors is that existing computerized Enterprise Resource Planning systems (ERP) do not contain CONWIP features.

A final remark for future studies concerns the layout optimization. Although the introduction of a better line balance proofed its efficiency, improving the distribution of the machines in the Pallet Factory could refine its functioning even further, by decreasing transporting times, and facilitating internal flows.

8. REFERENCES

BANKS, J.; CARSON, J.; NELSON, B.; NICOL, D. **Discrete-Event System Simulation**. Pearson, 2001.

BLUM, C.; MIRALLES, C. On solving the assembly line worker assignment and balancing problem via beam search. **Computers and Operations Research**, v. 38, n. 1, p. 328-339, 2011.

COROMINAS, A.; PASTOR, R.; PLANS, J. Balancing assembly line with skilled and unskilled workers. **Omega**, v. 36, n. 6, p. 1126–1132, 2008.

COSTA, A. M.; MIRALLES, C. Job rotation in assembly lines employing disabled workers. **International Journal of Production Economics**, v. 120, n. 2, p. 625-632, 2009.

DETTY; YINGLING, Quantifying benefits of conversion to lean manufacturing with discrete event simulation: A case study. **International Journal of Production Research,** v.38 n.2, 2010.

FACTORY PHYSICS INC. Flow Benchmarking. Available on: <https://factoryphysics.com/flow-benchmarking>. Accessed on november 6th, 2016.

FERNANDES, G. W. **A UTILIZAÇÃO DO KANBAN E MRP EM UMA INDÚSTRIA ELETRÔNICA COM SISTEMA HÍBRIDO DE PRODUÇÃO**. Undergraduate Work;

FRAMINAN, J. M.; GONZÁLEZ, P. L.; RUIZ-USANO, R. The CONWIP production control system: Review and research issues. **Production Planning & Control**, v. 14, n. 3, p. 255-265, 2003.

GHOSH, S.; GAGNON, R. J. A Comprehensive Literature Review and Analysis of the Design, Balancing and Scheduling of Assembly Systems. **International Journal of Production Research**, v. 27, n. 4, p. 637-670, 1989.

HAX, A. CANDEA, D. **Production and inventory management**. New Jersey: Prentice-Hall, Englewood Cliffs, 1984.

HOPP, W. J.; ROOF, M. L. Setting WIP levels with statistical throughput control (STC) in CONWIP production lines. **International Journal of Production Research**, v. 36, n. 4, p. 867-882, 1998.

HOPP, W. J.; SPEARMAN, M. L. **Factory Physics**. New York: McGraw-Hill, 1997.

HOPP, W. J.; SPEARMAN, M. L. **Factory Physics**. New York: McGraw-Hill, 2008.

HOPP, W. J.; SPEARMAN, M. L. To pull or not to pull: What is the question? **Manufacturing & Service Operations Management** 6(2):133-148. 2004

HUANG, M. WANG, D. IP, W.H. A simulation and comparative study of the CONWIP, Kanban and MRP production control systems in a cold rolling plant. **Production Planning and Control**, v. 9, p. 803-812, 1998

KONDO, D., & MESQUITA, M. A. Simulação de sistemas Kanban. 2008

KRIENGKORAKOT, N.; PIANTHONG, N. The Assembly Line Balancing Problem: Review articles. **KKU Engineering Journal**, v. 34, n. 2, p. 133-140, 2007.

MAREK, R. P.; ELKINS, D. A.; SMITH, D. R. Understanding the fundamentals of Kanban and Conwip pull systems using simulation. **Proceedings of the 2001 Winter Simulation Conference**. 2001.

MESQUITA, M. A.; GOLDEMBERG, D. Simulação em Planilhas de Linhas de Produção Puxada. In: XXVI ENEGEP - Fortaleza, CE, Brasil, 9 a 11 de outubro de 2006. **Anais do XXVI Encontro Nacional de Engenharia de Produção**, 2006

MIRALLES, C.; GARCÍA-SABATER, J. ; ANDRÉS, C. ; CARDOS, M. Advantages of assembly lines in Sheltered Work Centres for Disabled. **International Journal of Production Economics**, v. 110, n. 1-2, p. 187-197, 2007.

MONDEN, Y. **Produção sem estoques: uma abordagem prática do sistema de produção Toyota**. São Paulo: IMAM, 1984.

ONG, J. O.; TWENTIARANI, Y. Increasing Line Efficiency by using Time Study and Line Balancing. **Proceedings of the Asia Pacific Industrial Engineering & Management Systems Conference 2012**, 2012.

PAGE, E. H. **SIMULATION MODELING METHODOLOGY: PRINCIPLES AND ETIOLOGY OF DECISION SUPPORT**. 1994. 294 p. Doctoral Dissertation - Virginia Polytechnic Institute and State University, Blacksburg, 1994.

PATTERSON, J.; SEGERSTEDT, A. Restricted work-in-process: A study of differences between Kanban and CONWIP. **International Journal of Production Economics**, v. 118, n. 1, p. 199-207, 2009.

SAHU, P. **A Simulation Study of Kanban Levels for Assembly Lines and Systems**. 2012; 105 p. Masters Thesis – Arizona State University, Arizona, 2012.

SCHOLL, A.; BECKER, C. A survey on problems and methods in generalized assembly line balancing. **European Journal of Operational Research**, v. 168, n. 3, p. 694-715, 2006.

SOROUSH, H.; SAJADI, S. M.; ARABZAD, S. M. Efficiency analysis and optimization of a multi-product assembly line using simulation. **International Journal of Productivity and Quality Management**, v. 13, n. 1, p. 89-104, 2014.

SPEARMAN; WOODRUFF & HOPP. Conwip: a pull alternative to Kanban. **International Journal of Production Research.** v.28, n.1, p.879-894, 1990.

TOYOTA MOTORS CORPORATION. Just-in-Time: Philosophy of complete elimination of waste. Available on: < http://www.toyota-global.com/company/vision_philosophy/toyota_production_system/just-in-time.html > . Accessed on October 3rd, 2016.

TUBINO, D. F. **Manual de Planejamento e Controle de Produção**. São Paulo: Atlas, 1997.

TUBINO, D. F. **Planejamento e controle da Produção: teoria e prática**. São Paulo: Atlas, 2007.

YANG, T.; FU, H.; YANG, K. An evolutionary-simulation approach for the optimization of multi-constant work-in-process strategy—A case study. **International Journal of Production Economics**, v. 107, n. 1, p. 104-114, 2007

APPENDIX A – PUSH MODEL

```
Model statements for module:  BasicProcess.Create 1 (Sheet storage)
;
33$           CREATE,
1,SecondstoBaseTime(0.0),Sheet:SecondstoBaseTime(EXPO(1600)):NEXT(34$);

34$           ASSIGN:        Sheet storage.NumberOut=Sheet storage.NumberOut +
1:NEXT(30$);
;
;     Model statements for module:  BasicProcess.Assign 5 (Sheet entity picture)
;
30$           ASSIGN:        Picture=Picture.Red Ball:NEXT(0$);

;     Model statements for module:  BasicProcess.Process 1 (Sheet cutting)
;
0$            ASSIGN:        Sheet cutting.NumberIn=Sheet cutting.NumberIn + 1:
                             Sheet cutting.WIP=Sheet cutting.WIP+1;
66$           STACK,         1:Save:NEXT(40$);

40$           QUEUE,         Sheet cutting.Queue;
39$           SEIZE,         2,VA:
                             Op 1,1:
                             Op 2,1:
                             Mechanical guillotine,1:NEXT(38$);
38$           DELAY:         SecondsToBaseTime(EXPO( 40 )),,VA:NEXT(81$);
81$           ASSIGN:        Sheet cutting.WaitTime=Sheet cutting.WaitTime +
Diff.WaitTime;
45$           TALLY:         Sheet cutting.WaitTimePerEntity,Diff.WaitTime,1;
47$           TALLY:         Sheet cutting.TotalTimePerEntity,Diff.StartTime,1;
71$           ASSIGN:        Sheet cutting.VATime=Sheet cutting.VATime +
Diff.VATime;
72$           TALLY:         Sheet cutting.VATimePerEntity,Diff.VATime,1;
37$           RELEASE:       Op 1,1:
                             Op 2,1:
                             Mechanical guillotine,1;
86$           STACK,         1:Destroy:NEXT(85$);
85$           ASSIGN:        Sheet cutting.NumberOut=Sheet cutting.NumberOut + 1:
                             Sheet cutting.WIP=Sheet cutting.WIP-1:NEXT(1$);
;     Model statements for module:  BasicProcess.Process 2 (Sheet corners cutting)
;
1$            ASSIGN:        Sheet corners cutting.NumberIn=Sheet corners
cutting.NumberIn + 1:
                             Sheet corners cutting.WIP=Sheet corners cutting.WIP+1;
117$          STACK,         1:Save:NEXT(91$);

91$           QUEUE,         Sheet corners cutting.Queue;
90$           SEIZE,         2,VA:
                             Op 1,1:
                             Op 2,1:
                             Grinder,1:NEXT(89$);
89$           DELAY:         SecondsToBaseTime(EXPO( 652 )),,VA:NEXT(132$);
132$          ASSIGN:        Sheet corners cutting.WaitTime=Sheet corners
cutting.WaitTime + Diff.WaitTime;
96$           TALLY:         Sheet corners
cutting.WaitTimePerEntity,Diff.WaitTime,1;
98$           TALLY:         Sheet corners
cutting.TotalTimePerEntity,Diff.StartTime,1;
122$          ASSIGN:        Sheet corners cutting.VATime=Sheet corners
cutting.VATime + Diff.VATime;
123$          TALLY:         Sheet corners cutting.VATimePerEntity,Diff.VATime,1;
88$           RELEASE:       Op 1,1:
                             Op 2,1:
                             Grinder,1;
```

```
137$          STACK,         1:Destroy:NEXT(136$);

136$          ASSIGN:        Sheet corners cutting.NumberOut=Sheet corners
cutting.NumberOut + 1:
                             Sheet corners cutting.WIP=Sheet corners cutting.WIP-
1:NEXT(2$);
;     Model statements for module:  BasicProcess.Process 3 (Sheet bending)

2$            ASSIGN:        Sheet bending.NumberIn=Sheet bending.NumberIn + 1:
                             Sheet bending.WIP=Sheet bending.WIP+1;
168$          STACK,         1:Save:NEXT(142$);

142$          QUEUE,         Sheet bending.Queue;
141$          SEIZE,         2,VA:
                             Op 1,1:
                             Op 2,1:
                             Press for sheet bending,1:NEXT(140$);
140$          DELAY:         SecondsToBaseTime(EXPO( 624 )),,VA:NEXT(183$);
183$          ASSIGN:        Sheet bending.WaitTime=Sheet bending.WaitTime +
Diff.WaitTime;
147$          TALLY:         Sheet bending.WaitTimePerEntity,Diff.WaitTime,1;
149$          TALLY:         Sheet bending.TotalTimePerEntity,Diff.StartTime,1;
173$          ASSIGN:        Sheet bending.VATime=Sheet bending.VATime +
Diff.VATime;
174$          TALLY:         Sheet bending.VATimePerEntity,Diff.VATime,1;
139$          RELEASE:       Op 1,1:
                             Op 2,1:
                             Press for sheet bending,1;
188$          STACK,         1:Destroy:NEXT(187$);

187$          ASSIGN:        Sheet bending.NumberOut=Sheet bending.NumberOut + 1:
                             Sheet bending.WIP=Sheet bending.WIP-1:NEXT(7$);
;     Model statements for module:  AdvancedProcess.Match 1 (Sheet plus Profile)
;
7$            QUEUE,         Sheet plus Profile.Queue1:DETACH;
10$           QUEUE,         Sheet plus Profile.Queue2:DETACH;
              MATCH:         7$,24$:
                             10$,25$;
;     Model statements for module:  BasicProcess.Assign 1 (Welded profile entity
picture)
;
25$           ASSIGN:        Entity.Type=Welded profile:
                             Picture=Picture.Blue Ball:NEXT(14$);
;     Model statements for module:  BasicProcess.Process 10 (Profile welding)
;
14$           ASSIGN:        Profile welding.NumberIn=Profile welding.NumberIn + 1:
                             Profile welding.WIP=Profile welding.WIP+1;
219$          STACK,         1:Save:NEXT(193$);

193$          QUEUE,         Profile welding.Queue;
192$          SEIZE,         2,VA:
                             Op 7,1:
                             Welding Station 3,1:NEXT(191$);

191$          DELAY:         SecondsToBaseTime(EXPO( 416 )),,VA:NEXT(234$);

234$          ASSIGN:        Profile welding.WaitTime=Profile welding.WaitTime +
Diff.WaitTime;
198$          TALLY:         Profile welding.WaitTimePerEntity,Diff.WaitTime,1;
200$          TALLY:         Profile welding.TotalTimePerEntity,Diff.StartTime,1;
224$          ASSIGN:        Profile welding.VATime=Profile welding.VATime +
Diff.VATime;
225$          TALLY:         Profile welding.VATimePerEntity,Diff.VATime,1;
190$          RELEASE:       Op 7,1:
                             Welding Station 3,1;
239$          STACK,         1:Destroy:NEXT(238$);
```

```
238$          ASSIGN:        Profile welding.NumberOut=Profile welding.NumberOut +
1:
                             Profile welding.WIP=Profile welding.WIP-1:NEXT(15$);
;     Model statements for module:  BasicProcess.Process 11 (Bottoms welding)
;
15$           ASSIGN:        Bottoms welding.NumberIn=Bottoms welding.NumberIn + 1:
                             Bottoms welding.WIP=Bottoms welding.WIP+1;
270$          STACK,         1:Save:NEXT(244$);
244$          QUEUE,         Bottoms welding.Queue;
243$          SEIZE,         2,VA:
                             Welding Station 1,1:
                             Op 6,1:NEXT(242$);
242$          DELAY:         SecondsToBaseTime(EXPO( 120 )),,VA:NEXT(285$);
285$          ASSIGN:        Bottoms welding.WaitTime=Bottoms welding.WaitTime +
Diff.WaitTime;
249$          TALLY:         Bottoms welding.WaitTimePerEntity,Diff.WaitTime,1;
251$          TALLY:         Bottoms welding.TotalTimePerEntity,Diff.StartTime,1;
275$          ASSIGN:        Bottoms welding.VATime=Bottoms welding.VATime +
Diff.VATime;
276$          TALLY:         Bottoms welding.VATimePerEntity,Diff.VATime,1;
241$          RELEASE:       Welding Station 1,1:
                             Op 6,1;
290$          STACK,         1:Destroy:NEXT(289$);

289$          ASSIGN:        Bottoms welding.NumberOut=Bottoms welding.NumberOut +
1:
                             Bottoms welding.WIP=Bottoms welding.WIP-1:NEXT(16$);
;     Model statements for module:  AdvancedProcess.Match 2 (Final welding)
;
16$           QUEUE,         Final welding.Queue1:DETACH;
19$           QUEUE,         Final welding.Queue2:DETACH;
              MATCH:         16$,27$:
                             19$,32$;
;     Model statements for module:  BasicProcess.Dispose 3 (Dispose 3)
;
32$           ASSIGN:        Dispose 3.NumberOut=Dispose 3.NumberOut + 1;
292$          DISPOSE:       Yes;
;     Model statements for module:  BasicProcess.Assign 2 (Almost pallet entity
picture)
;
27$           ASSIGN:        Entity.Type=Almost pallet:
                             Picture=Picture.Box:NEXT(21$);
;     Model statements for module:  BasicProcess.Process 12 (Building up)
;
21$           ASSIGN:        Building up.NumberIn=Building up.NumberIn + 1:
                             Building up.WIP=Building up.WIP+1;
322$          STACK,         1:Save:NEXT(296$);

296$          QUEUE,         Building up.Queue;
295$          SEIZE,         2,VA:
                             Op 9,1:
                             Op 8,1:
                             Assembly station 1,1:NEXT(294$);
294$          DELAY:         SecondsToBaseTime(EXPO( 1187 )),,VA:NEXT(337$);
337$          ASSIGN:        Building up.WaitTime=Building up.WaitTime +
Diff.WaitTime;
301$          TALLY:         Building up.WaitTimePerEntity,Diff.WaitTime,1;
303$          TALLY:         Building up.TotalTimePerEntity,Diff.StartTime,1;
327$          ASSIGN:        Building up.VATime=Building up.VATime + Diff.VATime;
328$          TALLY:         Building up.VATimePerEntity,Diff.VATime,1;
293$          RELEASE:       Op 9,1:
                             Op 8,1:
                             Assembly station 1,1;
342$          STACK,         1:Destroy:NEXT(341$);

341$          ASSIGN:        Building up.NumberOut=Building up.NumberOut + 1:
                             Building up.WIP=Building up.WIP-1:NEXT(22$);
;     Model statements for module:  BasicProcess.Process 13 (Assembly)
```

```
;
22$           ASSIGN:        Assembly.NumberIn=Assembly.NumberIn + 1:
                             Assembly.WIP=Assembly.WIP+1;
373$          STACK,         1:Save:NEXT(347$);

347$          QUEUE,         Assembly.Queue;
346$          SEIZE,         2,VA:
                             Op 11,1:
                             Op 10,1:
                             Assembly station 2,1:NEXT(345$);
345$          DELAY:         SecondsToBaseTime(EXPO( 1212 )),,VA:NEXT(388$);
388$          ASSIGN:        Assembly.WaitTime=Assembly.WaitTime + Diff.WaitTime;
352$          TALLY:         Assembly.WaitTimePerEntity,Diff.WaitTime,1;
354$          TALLY:         Assembly.TotalTimePerEntity,Diff.StartTime,1;
378$          ASSIGN:        Assembly.VATime=Assembly.VATime + Diff.VATime;
379$          TALLY:         Assembly.VATimePerEntity,Diff.VATime,1;
344$          RELEASE:       Op 11,1:
                             Op 10,1:
                             Assembly station 2,1;
393$          STACK,         1:Destroy:NEXT(392$);
392$          ASSIGN:        Assembly.NumberOut=Assembly.NumberOut + 1:
                             Assembly.WIP=Assembly.WIP-1:NEXT(31$);
;     Model statements for module:  BasicProcess.Record 3 (Time between products)
;
31$           TALLY:         Time between products,BET,1:NEXT(26$);
;     Model statements for module:  BasicProcess.Record 1 (Count output)
;
26$           COUNT:         Count output,1:NEXT(23$);
;     Model statements for module:  BasicProcess.Dispose 1 (Dispose 1)
;
23$           ASSIGN:        Dispose 1.NumberOut=Dispose 1.NumberOut + 1;
395$          DISPOSE:       Yes;
;     Model statements for module:  BasicProcess.Dispose 2 (Dispose 2)
;
24$           ASSIGN:        Dispose 2.NumberOut=Dispose 2.NumberOut + 1;
396$          DISPOSE:       Yes;
;     Model statements for module:  BasicProcess.Create 2 (Profile storage)
;

397$          CREATE,
1,SecondstoBaseTime(0.0),Profile:SecondstoBaseTime(EXPO(1500)):NEXT(398$);

398$          ASSIGN:        Profile storage.NumberOut=Profile storage.NumberOut +
1:NEXT(29$);
;     Model statements for module:  BasicProcess.Assign 4 (Profile entity picture)
29$           ASSIGN:        Picture=Picture.Green Ball:NEXT(3$);
;     Model statements for module:  BasicProcess.Process 4 (Profile cutting)
;
3$            ASSIGN:        Profile cutting.NumberIn=Profile cutting.NumberIn + 1:
                             Profile cutting.WIP=Profile cutting.WIP+1;
430$          STACK,         1:Save:NEXT(404$);

404$          QUEUE,         Profile cutting.Queue;
403$          SEIZE,         2,VA:
                             Op 3,1:
                             Saw frame,1:NEXT(402$);
402$          DELAY:         SecondsToBaseTime(EXPO( 349)),,VA:NEXT(445$);
445$          ASSIGN:        Profile cutting.WaitTime=Profile cutting.WaitTime +
Diff.WaitTime;
409$          TALLY:         Profile cutting.WaitTimePerEntity,Diff.WaitTime,1;
411$          TALLY:         Profile cutting.TotalTimePerEntity,Diff.StartTime,1;
435$          ASSIGN:        Profile cutting.VATime=Profile cutting.VATime +
Diff.VATime;
436$          TALLY:         Profile cutting.VATimePerEntity,Diff.VATime,1;
401$          RELEASE:       Op 3,1:
                             Saw frame,1;
450$          STACK,         1:Destroy:NEXT(449$);
```

```
449$          ASSIGN:        Profile cutting.NumberOut=Profile cutting.NumberOut +
1:
                             Profile cutting.WIP=Profile cutting.WIP-1:NEXT(4$);
;     Model statements for module:  BasicProcess.Process 5 (Profile incisision)
;
4$            ASSIGN:        Profile incisision.NumberIn=Profile
incisision.NumberIn + 1:
                             Profile incisision.WIP=Profile incisision.WIP+1;
481$          STACK,         1:Save:NEXT(455$);

455$          QUEUE,         Profile incisision.Queue;
454$          SEIZE,         2,VA:
                             Op 3,1:
                             Grinder for profiles,1:NEXT(453$);
453$          DELAY:         SecondsToBaseTime(EXPO( 504 )),,VA:NEXT(496$);
496$          ASSIGN:        Profile incisision.WaitTime=Profile
incisision.WaitTime + Diff.WaitTime;
460$          TALLY:         Profile incisision.WaitTimePerEntity,Diff.WaitTime,1;
462$          TALLY:         Profile
incisision.TotalTimePerEntity,Diff.StartTime,1;
486$          ASSIGN:        Profile incisision.VATime=Profile incisision.VATime +
Diff.VATime;
487$          TALLY:         Profile incisision.VATimePerEntity,Diff.VATime,1;
452$          RELEASE:       Op 3,1:
                             Grinder for profiles,1;
501$          STACK,         1:Destroy:NEXT(500$);

500$          ASSIGN:        Profile incisision.NumberOut=Profile
incisision.NumberOut + 1:
                             Profile incisision.WIP=Profile incisision.WIP-
1:NEXT(5$);
;     Model statements for module:  BasicProcess.Process 6 (Hole drilling)
;
5$            ASSIGN:        Hole drilling.NumberIn=Hole drilling.NumberIn + 1:
                             Hole drilling.WIP=Hole drilling.WIP+1;
532$          STACK,         1:Save:NEXT(506$);
506$          QUEUE,         Hole drilling.Queue;
505$          SEIZE,         2,VA:
                             Op 4,1:
                             Drilling machine,1:NEXT(504$);
504$          DELAY:         SecondsToBaseTime(EXPO( 1026 )),,VA:NEXT(547$);
547$          ASSIGN:        Hole drilling.WaitTime=Hole drilling.WaitTime +
Diff.WaitTime;
511$          TALLY:         Hole drilling.WaitTimePerEntity,Diff.WaitTime,1;
513$          TALLY:         Hole drilling.TotalTimePerEntity,Diff.StartTime,1;
537$          ASSIGN:        Hole drilling.VATime=Hole drilling.VATime +
Diff.VATime;
538$          TALLY:         Hole drilling.VATimePerEntity,Diff.VATime,1;
503$          RELEASE:       Op 4,1:
                             Drilling machine,1;
552$          STACK,         1:Destroy:NEXT(551$);

551$          ASSIGN:        Hole drilling.NumberOut=Hole drilling.NumberOut + 1:
                             Hole drilling.WIP=Hole drilling.WIP-1:NEXT(13$);
;     Model statements for module:  BasicProcess.Process 9 (Angles welding)
;
13$           ASSIGN:        Angles welding.NumberIn=Angles welding.NumberIn + 1:
                             Angles welding.WIP=Angles welding.WIP+1;
583$          STACK,         1:Save:NEXT(557$);

557$          QUEUE,         Angles welding.Queue;
556$          SEIZE,         2,VA:
                             Op 5,1:
                             Welding Station 2,1:NEXT(555$);

555$          DELAY:         SecondsToBaseTime(EXPO( 304 )),,VA:NEXT(598$);
```

```
598$          ASSIGN:        Angles welding.WaitTime=Angles welding.WaitTime +
Diff.WaitTime;
562$          TALLY:         Angles welding.WaitTimePerEntity,Diff.WaitTime,1;
564$          TALLY:         Angles welding.TotalTimePerEntity,Diff.StartTime,1;
588$          ASSIGN:        Angles welding.VATime=Angles welding.VATime +
Diff.VATime;
589$          TALLY:         Angles welding.VATimePerEntity,Diff.VATime,1;
554$          RELEASE:       Op 5,1:
                             Welding Station 2,1;
603$          STACK,         1:Destroy:NEXT(602$);
602$          ASSIGN:        Angles welding.NumberOut=Angles welding.NumberOut + 1:
                             Angles welding.WIP=Angles welding.WIP-1:NEXT(10$);
;     Model statements for module:  BasicProcess.Create 3 (Angles storage)

605$          CREATE,
1,SecondstoBaseTime(500.0),Angles:SecondstoBaseTime(EXPO(1700)):NEXT(606$);

606$          ASSIGN:        Angles storage.NumberOut=Angles storage.NumberOut +
1:NEXT(28$);
;     Model statements for module:  BasicProcess.Assign 3 (Angles entity picture)
;
28$           ASSIGN:        Picture=Picture.Yellow Ball:NEXT(6$);
;     Model statements for module:  BasicProcess.Process 7 (Angles cutting)
;
6$            ASSIGN:        Angles cutting.NumberIn=Angles cutting.NumberIn + 1:
                             Angles cutting.WIP=Angles cutting.WIP+1;
638$          STACK,         1:Save:NEXT(612$);
612$          QUEUE,         Angles cutting.Queue;
611$          SEIZE,         2,VA:
                             Op 6,1:
                             Press for cutting angles,1:NEXT(610$);
610$          DELAY:         SecondsToBaseTime(EXPO( 16 )),,VA:NEXT(653$);
653$          ASSIGN:        Angles cutting.WaitTime=Angles cutting.WaitTime +
Diff.WaitTime;
617$          TALLY:         Angles cutting.WaitTimePerEntity,Diff.WaitTime,1;
619$          TALLY:         Angles cutting.TotalTimePerEntity,Diff.StartTime,1;
643$          ASSIGN:        Angles cutting.VATime=Angles cutting.VATime +
Diff.VATime;
644$          TALLY:         Angles cutting.VATimePerEntity,Diff.VATime,1;
609$          RELEASE:       Op 6,1:
                             Press for cutting angles,1;
658$          STACK,         1:Destroy:NEXT(657$);

657$          ASSIGN:        Angles cutting.NumberOut=Angles cutting.NumberOut + 1:
                             Angles cutting.WIP=Angles cutting.WIP-1:NEXT(12$);
;     Model statements for module:  BasicProcess.Process 8 (Cup welding)
;
12$           ASSIGN:        Cup welding.NumberIn=Cup welding.NumberIn + 1:
                             Cup welding.WIP=Cup welding.WIP+1;
689$          STACK,         1:Save:NEXT(663$);

663$          QUEUE,         Cup welding.Queue;
662$          SEIZE,         2,VA:
                             Op 6,1:
                             Welding Station 1,1:NEXT(661$);
661$          DELAY:         SecondsToBaseTime(EXPO( 192 )),,VA:NEXT(704$);
704$          ASSIGN:        Cup welding.WaitTime=Cup welding.WaitTime +
Diff.WaitTime;
668$          TALLY:         Cup welding.WaitTimePerEntity,Diff.WaitTime,1;
670$          TALLY:         Cup welding.TotalTimePerEntity,Diff.StartTime,1;
694$          ASSIGN:        Cup welding.VATime=Cup welding.VATime + Diff.VATime;
695$          TALLY:         Cup welding.VATimePerEntity,Diff.VATime,1;
660$          RELEASE:       Op 6,1:
                             Welding Station 1,1;
709$          STACK,         1:Destroy:NEXT(708$);

708$          ASSIGN:        Cup welding.NumberOut=Cup welding.NumberOut + 1:
                             Cup welding.WIP=Cup welding.WIP-1:NEXT(19$);
```

APPENDIX B – CONWIP MODEL

```
Model statements for module:  BasicProcess.Create 7 (Create 7)
38$           CREATE,
1,SecondstoBaseTime(0.0),Sheet:SecondstoBaseTime(EXPO(1500)),5:NEXT(39$);

39$           ASSIGN:        Create 7.NumberOut=Create 7.NumberOut + 1:NEXT(31$);

;     Model statements for module:  AdvancedProcess.Hold 3 (Hold 3)
;
31$           QUEUE,         Hold 3.Queue;
              SCAN:
                             Angles cutting.WIP + Angles welding.WIP + Assembly.WIP
+ Bottoms welding.WIP +Building up.WIP + Cup welding.WIP + Hole drilling.WIP +
Profile cutting.WIP + Profile incisision.WIP + Profile welding.WIP + Sheet
bending.WIP + Sheet corners cutting.WIP + Sheet cutting.WIP+ NQ(Final
welding.Queue1) + NQ(Final welding.Queue2) + NQ(Sheet plus Profile.Queue1) +
NQ(Sheet plus Profile.Queue2)< MaxWIP
                             :NEXT(29$);
;
;     Model statements for module:  BasicProcess.Assign 5 (Sheet entity picture)
;
29$           ASSIGN:        Picture=Picture.Red Ball:NEXT(0$);
;     Model statements for module:  BasicProcess.Process 1 (Sheet cutting)
;
0$            ASSIGN:        Sheet cutting.NumberIn=Sheet cutting.NumberIn + 1:
                             Sheet cutting.WIP=Sheet cutting.WIP+1;
71$           STACK,         1:Save:NEXT(45$);

45$           QUEUE,         Sheet cutting.Queue;
44$           SEIZE,         2,VA:
                             Op 1,1:
                             Op 2,1:
                             Mechanical guillotine,1:NEXT(43$);

43$           DELAY:         SecondsToBaseTime(EXPO( 40 )),,VA:NEXT(86$);

86$           ASSIGN:        Sheet cutting.WaitTime=Sheet cutting.WaitTime +
Diff.WaitTime;
50$           TALLY:         Sheet cutting.WaitTimePerEntity,Diff.WaitTime,1;
52$           TALLY:         Sheet cutting.TotalTimePerEntity,Diff.StartTime,1;
76$           ASSIGN:        Sheet cutting.VATime=Sheet cutting.VATime +
Diff.VATime;
77$           TALLY:         Sheet cutting.VATimePerEntity,Diff.VATime,1;
42$           RELEASE:       Op 1,1:
                             Op 2,1:
                             Mechanical guillotine,1;
91$           STACK,         1:Destroy:NEXT(90$);

90$           ASSIGN:        Sheet cutting.NumberOut=Sheet cutting.NumberOut + 1:
                             Sheet cutting.WIP=Sheet cutting.WIP-1:NEXT(1$);

;     Model statements for module:  BasicProcess.Process 2 (Sheet corners cutting)
;
1$            ASSIGN:        Sheet corners cutting.NumberIn=Sheet corners
cutting.NumberIn + 1:
                             Sheet corners cutting.WIP=Sheet corners cutting.WIP+1;
122$          STACK,         1:Save:NEXT(96$);

96$           QUEUE,         Sheet corners cutting.Queue;
95$           SEIZE,         2,VA:
                             Op 1,1:
                             Op 2,1:
                             Grinder,1:NEXT(94$);
```

```
94$           DELAY:         SecondsToBaseTime(EXPO( 652)),,VA:NEXT(137$);

137$          ASSIGN:        Sheet corners cutting.WaitTime=Sheet corners
cutting.WaitTime + Diff.WaitTime;
101$          TALLY:         Sheet corners
cutting.WaitTimePerEntity,Diff.WaitTime,1;
103$          TALLY:         Sheet corners
cutting.TotalTimePerEntity,Diff.StartTime,1;
127$          ASSIGN:        Sheet corners cutting.VATime=Sheet corners
cutting.VATime + Diff.VATime;
128$          TALLY:         Sheet corners cutting.VATimePerEntity,Diff.VATime,1;
93$           RELEASE:       Op 1,1:
                             Op 2,1:
                             Grinder,1;
142$          STACK,         1:Destroy:NEXT(141$);

141$          ASSIGN:        Sheet corners cutting.NumberOut=Sheet corners
cutting.NumberOut + 1:
                             Sheet corners cutting.WIP=Sheet corners cutting.WIP-
1:NEXT(2$);
;     Model statements for module:  BasicProcess.Process 3 (Sheet bending)
;
2$            ASSIGN:        Sheet bending.NumberIn=Sheet bending.NumberIn + 1:
                             Sheet bending.WIP=Sheet bending.WIP+1;
173$          STACK,         1:Save:NEXT(147$);

147$          QUEUE,         Sheet bending.Queue;
146$          SEIZE,         2,VA:
                             Op 1,1:
                             Op 2,1:
                             Press for sheet bending,1:NEXT(145$);

145$          DELAY:         SecondsToBaseTime(EXPO( 624)),,VA:NEXT(188$);

188$          ASSIGN:        Sheet bending.WaitTime=Sheet bending.WaitTime +
Diff.WaitTime;
152$          TALLY:         Sheet bending.WaitTimePerEntity,Diff.WaitTime,1;
154$          TALLY:         Sheet bending.TotalTimePerEntity,Diff.StartTime,1;
178$          ASSIGN:        Sheet bending.VATime=Sheet bending.VATime +
Diff.VATime;
179$          TALLY:         Sheet bending.VATimePerEntity,Diff.VATime,1;
144$          RELEASE:       Op 1,1:
                             Op 2,1:
                             Press for sheet bending,1;
193$          STACK,         1:Destroy:NEXT(192$);

192$          ASSIGN:        Sheet bending.NumberOut=Sheet bending.NumberOut + 1:
                             Sheet bending.WIP=Sheet bending.WIP-1:NEXT(7$);
;     Model statements for module:  AdvancedProcess.Match 1 (Sheet plus Profile)
;
7$            QUEUE,         Sheet plus Profile.Queue1:DETACH;
10$           QUEUE,         Sheet plus Profile.Queue2:DETACH;
              MATCH:         7$,31$:
                             10$,24$;
;     Model statements for module:  BasicProcess.Assign 1 (Welded profile entity
picture)
;
24$           ASSIGN:        Entity.Type=Welded profile:
                             Picture=Picture.Blue Ball:NEXT(14$);
;     Model statements for module:  BasicProcess.Process 10 (Profile welding)
;
14$           ASSIGN:        Profile welding.NumberIn=Profile welding.NumberIn + 1:
                             Profile welding.WIP=Profile welding.WIP+1;
224$          STACK,         1:Save:NEXT(198$);

198$          QUEUE,         Profile welding.Queue;
197$          SEIZE,         2,VA:
                             Op 7,1:
```

```
                             Welding Station 3,1:NEXT(196$);

196$          DELAY:         SecondsToBaseTime(EXPO( 416)),,VA:NEXT(239$);

239$          ASSIGN:        Profile welding.WaitTime=Profile welding.WaitTime +
Diff.WaitTime;
203$          TALLY:         Profile welding.WaitTimePerEntity,Diff.WaitTime,1;
205$          TALLY:         Profile welding.TotalTimePerEntity,Diff.StartTime,1;
229$          ASSIGN:        Profile welding.VATime=Profile welding.VATime +
Diff.VATime;
230$          TALLY:         Profile welding.VATimePerEntity,Diff.VATime,1;
195$          RELEASE:       Op 7,1:
                             Welding Station 3,1;
244$          STACK,         1:Destroy:NEXT(243$);

243$          ASSIGN:        Profile welding.NumberOut=Profile welding.NumberOut +
1:
                             Profile welding.WIP=Profile welding.WIP-1:NEXT(15$);
;     Model statements for module:  BasicProcess.Process 11 (Bottoms welding)
;
15$           ASSIGN:        Bottoms welding.NumberIn=Bottoms welding.NumberIn + 1:
                             Bottoms welding.WIP=Bottoms welding.WIP+1;
275$          STACK,         1:Save:NEXT(249$);

249$          QUEUE,         Bottoms welding.Queue;
248$          SEIZE,         2,VA:
                             Welding Station 1,1:
                             Op 6,1:NEXT(247$);

247$          DELAY:         SecondsToBaseTime(EXPO( 120)),,VA:NEXT(290$);

290$          ASSIGN:        Bottoms welding.WaitTime=Bottoms welding.WaitTime +
Diff.WaitTime;
254$          TALLY:         Bottoms welding.WaitTimePerEntity,Diff.WaitTime,1;
256$          TALLY:         Bottoms welding.TotalTimePerEntity,Diff.StartTime,1;
280$          ASSIGN:        Bottoms welding.VATime=Bottoms welding.VATime +
Diff.VATime;
281$          TALLY:         Bottoms welding.VATimePerEntity,Diff.VATime,1;
246$          RELEASE:       Welding Station 1,1:
                             Op 6,1;
295$          STACK,         1:Destroy:NEXT(294$);

294$          ASSIGN:        Bottoms welding.NumberOut=Bottoms welding.NumberOut +
1:
                             Bottoms welding.WIP=Bottoms welding.WIP-1:NEXT(16$);
;     Model statements for module:  AdvancedProcess.Match 2 (Final welding)
;
16$           QUEUE,         Final welding.Queue1:DETACH;
19$           QUEUE,         Final welding.Queue2:DETACH;
              MATCH:         16$,26$:
                             19$,35$;
;     Model statements for module:  AdvancedProcess.Hold 5 (Hold 5)
;
35$           QUEUE,         Hold 5.Queue;
              SCAN:
                             Angles cutting.WIP + Angles welding.WIP + Assembly.WIP
+ Bottoms welding.WIP +Building up.WIP + Cup welding.WIP + Hole drilling.WIP +
Profile cutting.WIP + Profile incisision.WIP + Profile welding.WIP + Sheet
bending.WIP + Sheet corners cutting.WIP + Sheet cutting.WIP+ NQ(Final
welding.Queue1) + NQ(Final welding.Queue2) + NQ(Sheet plus Profile.Queue1) +
NQ(Sheet plus Profile.Queue2)< MaxWIP
                             :NEXT(27$);
;     Model statements for module:  BasicProcess.Assign 3 (Angles entity picture)
;
27$           ASSIGN:        Picture=Picture.Yellow Ball:NEXT(6$);
;     Model statements for module:  BasicProcess.Process 7 (Angles cutting)
;
6$            ASSIGN:        Angles cutting.NumberIn=Angles cutting.NumberIn + 1:
```

```
                              Angles cutting.WIP=Angles cutting.WIP+1;
326$          STACK,          1:Save:NEXT(300$);

300$          QUEUE,          Angles cutting.Queue;
299$          SEIZE,          2,VA:
                              Op 6,1:
                              Press for cutting angles,1:NEXT(298$);

298$          DELAY:          SecondsToBaseTime(EXPO( 16)),,VA:NEXT(341$);

341$          ASSIGN:         Angles cutting.WaitTime=Angles cutting.WaitTime +
Diff.WaitTime;
305$          TALLY:          Angles cutting.WaitTimePerEntity,Diff.WaitTime,1;
307$          TALLY:          Angles cutting.TotalTimePerEntity,Diff.StartTime,1;
331$          ASSIGN:         Angles cutting.VATime=Angles cutting.VATime +
Diff.VATime;
332$          TALLY:          Angles cutting.VATimePerEntity,Diff.VATime,1;
297$          RELEASE:        Op 6,1:
                              Press for cutting angles,1;
346$          STACK,          1:Destroy:NEXT(345$);

345$          ASSIGN:         Angles cutting.NumberOut=Angles cutting.NumberOut + 1:
                              Angles cutting.WIP=Angles cutting.WIP-1:NEXT(12$);
;     Model statements for module:  BasicProcess.Process 8 (Cup welding)
;
12$           ASSIGN:         Cup welding.NumberIn=Cup welding.NumberIn + 1:
                              Cup welding.WIP=Cup welding.WIP+1;
377$          STACK,          1:Save:NEXT(351$);

351$          QUEUE,          Cup welding.Queue;
350$          SEIZE,          2,VA:
                              Op 6,1:
                              Welding Station 1,1:NEXT(349$);

349$          DELAY:          SecondsToBaseTime(EXPO( 192)),,VA:NEXT(392$);

392$          ASSIGN:         Cup welding.WaitTime=Cup welding.WaitTime +
Diff.WaitTime;
356$          TALLY:          Cup welding.WaitTimePerEntity,Diff.WaitTime,1;
358$          TALLY:          Cup welding.TotalTimePerEntity,Diff.StartTime,1;
382$          ASSIGN:         Cup welding.VATime=Cup welding.VATime + Diff.VATime;
383$          TALLY:          Cup welding.VATimePerEntity,Diff.VATime,1;
348$          RELEASE:        Op 6,1:
                              Welding Station 1,1;
397$          STACK,          1:Destroy:NEXT(396$);

396$          ASSIGN:         Cup welding.NumberOut=Cup welding.NumberOut + 1:
                              Cup welding.WIP=Cup welding.WIP-1:NEXT(19$);
;     Model statements for module:  BasicProcess.Assign 2 (Almost profile entity
picture)
;
26$           ASSIGN:         Entity.Type=Almost pallet:
                              Picture=Picture.Box:NEXT(21$);
;     Model statements for module:  BasicProcess.Process 12 (Building up)
;
21$           ASSIGN:         Building up.NumberIn=Building up.NumberIn + 1:
                              Building up.WIP=Building up.WIP+1;
428$          STACK,          1:Save:NEXT(402$);

402$          QUEUE,          Building up.Queue;
401$          SEIZE,          2,VA:
                              Op 9,1:
                              Op 8,1:
                              Assembly station 1,1:NEXT(400$);

400$          DELAY:          SecondsToBaseTime(EXPO( 1187)),,VA:NEXT(443$);
```

```
443$          ASSIGN:        Building up.WaitTime=Building up.WaitTime +
Diff.WaitTime;
407$          TALLY:         Building up.WaitTimePerEntity,Diff.WaitTime,1;
409$          TALLY:         Building up.TotalTimePerEntity,Diff.StartTime,1;
433$          ASSIGN:        Building up.VATime=Building up.VATime + Diff.VATime;
434$          TALLY:         Building up.VATimePerEntity,Diff.VATime,1;
399$          RELEASE:       Op 9,1:
                             Op 8,1:
                             Assembly station 1,1;
448$          STACK,         1:Destroy:NEXT(447$);

447$          ASSIGN:        Building up.NumberOut=Building up.NumberOut + 1:
                             Building up.WIP=Building up.WIP-1:NEXT(22$);
;     Model statements for module:  BasicProcess.Process 13 (Assembly)
;
22$           ASSIGN:        Assembly.NumberIn=Assembly.NumberIn + 1:
                             Assembly.WIP=Assembly.WIP+1;
479$          STACK,         1:Save:NEXT(453$);

453$          QUEUE,         Assembly.Queue;
452$          SEIZE,         2,VA:
                             Op 11,1:
                             Op 10,1:
                             Assembly station 2,1:NEXT(451$);

451$          DELAY:         SecondsToBaseTime(EXPO( 1212)),,VA:NEXT(494$);

494$          ASSIGN:        Assembly.WaitTime=Assembly.WaitTime + Diff.WaitTime;
458$          TALLY:         Assembly.WaitTimePerEntity,Diff.WaitTime,1;
460$          TALLY:         Assembly.TotalTimePerEntity,Diff.StartTime,1;
484$          ASSIGN:        Assembly.VATime=Assembly.VATime + Diff.VATime;
485$          TALLY:         Assembly.VATimePerEntity,Diff.VATime,1;
450$          RELEASE:       Op 11,1:
                             Op 10,1:
                             Assembly station 2,1;
499$          STACK,         1:Destroy:NEXT(498$);

498$          ASSIGN:        Assembly.NumberOut=Assembly.NumberOut + 1:
                             Assembly.WIP=Assembly.WIP-1:NEXT(30$);
;     Model statements for module:  BasicProcess.Record 3 (Time between products)
;
30$           TALLY:         Time between products,BET,1:NEXT(25$);
;     Model statements for module:  BasicProcess.Record 1 (Count output)
;
25$           COUNT:         Count output,1:NEXT(37$);
;     Model statements for module:  BasicProcess.Record 6 (Statistics)
;
37$           TALLY:         Entity,0,1:NEXT(23$);
;     Model statements for module:  BasicProcess.Dispose 1 (Dispose 1)
;
23$           ASSIGN:        Dispose 1.NumberOut=Dispose 1.NumberOut + 1;
501$          DISPOSE:       Yes;
;     Model statements for module:  BasicProcess.Create 8 (Create 8)
;

502$          CREATE,
1,SecondstoBaseTime(0.0),Profile:SecondstoBaseTime(EXPO(1500)):NEXT(503$);

503$          ASSIGN:        Create 8.NumberOut=Create 8.NumberOut + 1:NEXT(33$);
     Model statements for module:  AdvancedProcess.Hold 4 (Hold 4)
;
33$           QUEUE,         Hold 4.Queue;
              SCAN:
                             Angles cutting.WIP + Angles welding.WIP + Assembly.WIP
+ Bottoms welding.WIP +Building up.WIP + Cup welding.WIP + Hole drilling.WIP +
Profile cutting.WIP + Profile incisision.WIP + Profile welding.WIP + Sheet
bending.WIP + Sheet corners cutting.WIP + Sheet cutting.WIP+ NQ(Final
welding.Queue1) + NQ(Final welding.Queue2) + NQ(Sheet plus Profile.Queue1) +
NQ(Sheet plus Profile.Queue2)< MaxWIP
                             :NEXT(28$);
;     Model statements for module:  BasicProcess.Assign 4 (Profile entity picture)
;
28$           ASSIGN:        Picture=Picture.Green Ball:NEXT(3$);
;     Model statements for module:  BasicProcess.Process 4 (Profile cutting)
;
3$            ASSIGN:        Profile cutting.NumberIn=Profile cutting.NumberIn + 1:
                             Profile cutting.WIP=Profile cutting.WIP+1;
535$          STACK,         1:Save:NEXT(509$);

509$          QUEUE,         Profile cutting.Queue;
508$          SEIZE,         2,VA:
                             Op 3,1:
                             Saw frame,1:NEXT(507$);

507$          DELAY:         SecondsToBaseTime(EXPO( 349)),,VA:NEXT(550$);

550$          ASSIGN:        Profile cutting.WaitTime=Profile cutting.WaitTime +
Diff.WaitTime;
514$          TALLY:         Profile cutting.WaitTimePerEntity,Diff.WaitTime,1;
516$          TALLY:         Profile cutting.TotalTimePerEntity,Diff.StartTime,1;
540$          ASSIGN:        Profile cutting.VATime=Profile cutting.VATime +
Diff.VATime;
541$          TALLY:         Profile cutting.VATimePerEntity,Diff.VATime,1;
506$          RELEASE:       Op 3,1:
                             Saw frame,1;
555$          STACK,         1:Destroy:NEXT(554$);

554$          ASSIGN:        Profile cutting.NumberOut=Profile cutting.NumberOut +
1:
                             Profile cutting.WIP=Profile cutting.WIP-1:NEXT(4$);
;     Model statements for module:  BasicProcess.Process 5 (Profile incisision)
;
4$            ASSIGN:        Profile incisision.NumberIn=Profile
incisision.NumberIn + 1:
                             Profile incisision.WIP=Profile incisision.WIP+1;
586$          STACK,         1:Save:NEXT(560$);

560$          QUEUE,         Profile incisision.Queue;
559$          SEIZE,         2,VA:
                             Op 3,1:
                             Grinder for profiles,1:NEXT(558$);

558$          DELAY:         SecondsToBaseTime(EXPO( 504)),,VA:NEXT(601$);

601$          ASSIGN:        Profile incisision.WaitTime=Profile
incisision.WaitTime + Diff.WaitTime;
565$          TALLY:         Profile incisision.WaitTimePerEntity,Diff.WaitTime,1;
567$          TALLY:         Profile
incisision.TotalTimePerEntity,Diff.StartTime,1;
591$          ASSIGN:        Profile incisision.VATime=Profile incisision.VATime +
Diff.VATime;
592$          TALLY:         Profile incisision.VATimePerEntity,Diff.VATime,1;
557$          RELEASE:       Op 3,1:
                             Grinder for profiles,1;
606$          STACK,         1:Destroy:NEXT(605$);

605$          ASSIGN:        Profile incisision.NumberOut=Profile
incisision.NumberOut + 1:
                             Profile incisision.WIP=Profile incisision.WIP-
1:NEXT(5$);
;     Model statements for module:  BasicProcess.Process 6 (Hole drilling)
;
5$            ASSIGN:        Hole drilling.NumberIn=Hole drilling.NumberIn + 1:
                             Hole drilling.WIP=Hole drilling.WIP+1;
637$          STACK,         1:Save:NEXT(611$);
```

```
611$          QUEUE,         Hole drilling.Queue;
610$          SEIZE,         2,VA:
                             Op 4,1:
                             Drilling machine,1:NEXT(609$);

609$          DELAY:         SecondsToBaseTime(EXPO( 1026)),,VA:NEXT(652$);

652$          ASSIGN:        Hole drilling.WaitTime=Hole drilling.WaitTime +
Diff.WaitTime;
616$          TALLY:         Hole drilling.WaitTimePerEntity,Diff.WaitTime,1;
618$          TALLY:         Hole drilling.TotalTimePerEntity,Diff.StartTime,1;
642$          ASSIGN:        Hole drilling.VATime=Hole drilling.VATime +
Diff.VATime;
643$          TALLY:         Hole drilling.VATimePerEntity,Diff.VATime,1;
608$          RELEASE:       Op 4,1:
                             Drilling machine,1;
657$          STACK,         1:Destroy:NEXT(656$);

656$          ASSIGN:        Hole drilling.NumberOut=Hole drilling.NumberOut + 1:
                             Hole drilling.WIP=Hole drilling.WIP-1:NEXT(13$);
;     Model statements for module:  BasicProcess.Process 9 (Angles welding)
;
13$           ASSIGN:        Angles welding.NumberIn=Angles welding.NumberIn + 1:
                             Angles welding.WIP=Angles welding.WIP+1;
688$          STACK,         1:Save:NEXT(662$);

662$          QUEUE,         Angles welding.Queue;
661$          SEIZE,         2,VA:
                             Op 5,1:
                             Welding Station 2,1:NEXT(660$);

660$          DELAY:         SecondsToBaseTime(EXPO( 304)),,VA:NEXT(703$);

703$          ASSIGN:        Angles welding.WaitTime=Angles welding.WaitTime +
Diff.WaitTime;
667$          TALLY:         Angles welding.WaitTimePerEntity,Diff.WaitTime,1;
669$          TALLY:         Angles welding.TotalTimePerEntity,Diff.StartTime,1;
693$          ASSIGN:        Angles welding.VATime=Angles welding.VATime +
Diff.VATime;
694$          TALLY:         Angles welding.VATimePerEntity,Diff.VATime,1;
659$          RELEASE:       Op 5,1:
                             Welding Station 2,1;
708$          STACK,         1:Destroy:NEXT(707$);

707$          ASSIGN:        Angles welding.NumberOut=Angles welding.NumberOut + 1:
                             Angles welding.WIP=Angles welding.WIP-1:NEXT(10$);
;     Model statements for module:  BasicProcess.Create 9 (Create 9)
;

710$          CREATE,
1,SecondstoBaseTime(0.0),Angle:SecondstoBaseTime(EXPO(1500)),5:NEXT(711$);

711$          ASSIGN:        Create 9.NumberOut=Create 9.NumberOut + 1:NEXT(35$);
```

APPENDIX C – MULTI-LOOP CONWIP MODEL

```
Model statements for module:  BasicProcess.Create 7 (Create 7)
;

45$           CREATE,
1,SecondstoBaseTime(0.0),Sheet:SecondstoBaseTime(EXPO(1500)),10:NEXT(46$);

46$           ASSIGN:        Create 7.NumberOut=Create 7.NumberOut + 1:NEXT(33$);
;     Model statements for module:  AdvancedProcess.Hold 3 (Hold 3)
;
33$           QUEUE,         Hold 3.Queue;
              SCAN:
                             Sheet bending.WIP + Sheet corners cutting.WIP + Sheet
cutting.WIP +  NQ(Sheet plus Profile.Queue1)< MaxWIP1
                             :NEXT(40$);
;     Model statements for module:  BasicProcess.Separate 1 (Separate 1)
;
40$           DUPLICATE,     100 - 50:
                             1,51$,50:NEXT(50$);

50$           ASSIGN:        Separate 1.NumberOut Orig=Separate 1.NumberOut Orig +
1:NEXT(31$);

51$           ASSIGN:        Separate 1.NumberOut Dup=Separate 1.NumberOut Dup +
1:NEXT(33$);
;     Model statements for module:  BasicProcess.Assign 5 (Sheet entity picture)
;
31$           ASSIGN:        Picture=Picture.Red Ball:NEXT(0$);
;     Model statements for module:  BasicProcess.Process 1 (Sheet cutting)
;
0$            ASSIGN:        Sheet cutting.NumberIn=Sheet cutting.NumberIn + 1:
                             Sheet cutting.WIP=Sheet cutting.WIP+1;
81$           STACK,         1:Save:NEXT(55$);

55$           QUEUE,         Sheet cutting.Queue;
54$           SEIZE,         2,VA:
                             Op 1,1:
                             Op 2,1:
                             Mechanical guillotine,1:NEXT(53$);

53$           DELAY:         SecondsToBaseTime(EXPO( 40 )),,VA:NEXT(96$);

96$           ASSIGN:        Sheet cutting.WaitTime=Sheet cutting.WaitTime +
Diff.WaitTime;
60$           TALLY:         Sheet cutting.WaitTimePerEntity,Diff.WaitTime,1;
62$           TALLY:         Sheet cutting.TotalTimePerEntity,Diff.StartTime,1;
86$           ASSIGN:        Sheet cutting.VATime=Sheet cutting.VATime +
Diff.VATime;
87$           TALLY:         Sheet cutting.VATimePerEntity,Diff.VATime,1;
52$           RELEASE:       Op 1,1:
                             Op 2,1:
                             Mechanical guillotine,1;
101$          STACK,         1:Destroy:NEXT(100$);

100$          ASSIGN:        Sheet cutting.NumberOut=Sheet cutting.NumberOut + 1:
                             Sheet cutting.WIP=Sheet cutting.WIP-1:NEXT(1$);
;     Model statements for module:  BasicProcess.Process 2 (Sheet corners cutting)
;
1$            ASSIGN:        Sheet corners cutting.NumberIn=Sheet corners
cutting.NumberIn + 1:
                             Sheet corners cutting.WIP=Sheet corners cutting.WIP+1;
132$          STACK,         1:Save:NEXT(106$);
```

```
106$          QUEUE,         Sheet corners cutting.Queue;
105$          SEIZE,         2,VA:
                             Op 1,1:
                             Op 2,1:
                             Grinder,1:NEXT(104$);

104$          DELAY:         SecondsToBaseTime(EXPO (652)),,VA:NEXT(147$);

147$          ASSIGN:        Sheet corners cutting.WaitTime=Sheet corners
cutting.WaitTime + Diff.WaitTime;
111$          TALLY:         Sheet corners
cutting.WaitTimePerEntity,Diff.WaitTime,1;
113$          TALLY:         Sheet corners
cutting.TotalTimePerEntity,Diff.StartTime,1;
137$          ASSIGN:        Sheet corners cutting.VATime=Sheet corners
cutting.VATime + Diff.VATime;
138$          TALLY:         Sheet corners cutting.VATimePerEntity,Diff.VATime,1;
103$          RELEASE:       Op 1,1:
                             Op 2,1:
                             Grinder,1;
152$          STACK,         1:Destroy:NEXT(151$);

151$          ASSIGN:        Sheet corners cutting.NumberOut=Sheet corners
cutting.NumberOut + 1:
                             Sheet corners cutting.WIP=Sheet corners cutting.WIP-
1:NEXT(2$);
;     Model statements for module:  BasicProcess.Process 3 (Sheet bending)
;
2$            ASSIGN:        Sheet bending.NumberIn=Sheet bending.NumberIn + 1:
                             Sheet bending.WIP=Sheet bending.WIP+1;
183$          STACK,         1:Save:NEXT(157$);

157$          QUEUE,         Sheet bending.Queue;
156$          SEIZE,         2,VA:
                             Op 1,1:
                             Op 2,1:
                             Press for sheet bending,1:NEXT(155$);

155$          DELAY:         SecondsToBaseTime(EXPO( 624 )),,VA:NEXT(198$);

198$          ASSIGN:        Sheet bending.WaitTime=Sheet bending.WaitTime +
Diff.WaitTime;
162$          TALLY:         Sheet bending.WaitTimePerEntity,Diff.WaitTime,1;
164$          TALLY:         Sheet bending.TotalTimePerEntity,Diff.StartTime,1;
188$          ASSIGN:        Sheet bending.VATime=Sheet bending.VATime +
Diff.VATime;
189$          TALLY:         Sheet bending.VATimePerEntity,Diff.VATime,1;
154$          RELEASE:       Op 1,1:
                             Op 2,1:
                             Press for sheet bending,1;
203$          STACK,         1:Destroy:NEXT(202$);

202$          ASSIGN:        Sheet bending.NumberOut=Sheet bending.NumberOut + 1:
                             Sheet bending.WIP=Sheet bending.WIP-1:NEXT(7$);
;     Model statements for module:  AdvancedProcess.Match 1 (Sheet plus Profile)
;
7$            QUEUE,         Sheet plus Profile.Queue1:DETACH;
10$           QUEUE,         Sheet plus Profile.Queue2:DETACH;
              MATCH:         7$,25$:
                             10$,26$;
;     Model statements for module:  BasicProcess.Assign 1 (Welded profile entity
picture)
;
26$           ASSIGN:        Entity.Type=Welded profile:
                             Picture=Picture.Blue Ball:NEXT(43$);
;     Model statements for module:  AdvancedProcess.Hold 6 (Hold 6)
;
43$           QUEUE,         Hold 6.Queue;
```

```
              SCAN:
                             + Profile welding.WIP +Bottoms welding.WIP +NQ(Final
welding.Queue1) + Assembly.WIP +Building up.WIP< MaxWIP4
                             :NEXT(14$);
;     Model statements for module:  BasicProcess.Process 10 (Profile welding)
;
14$           ASSIGN:        Profile welding.NumberIn=Profile welding.NumberIn + 1:
                             Profile welding.WIP=Profile welding.WIP+1;
234$          STACK,         1:Save:NEXT(208$);

208$          QUEUE,         Profile welding.Queue;
207$          SEIZE,         2,VA:
                             Op 7,1:
                             Welding Station 3,1:NEXT(206$);

206$          DELAY:         SecondsToBaseTime(EXPO( 416 )),,VA:NEXT(249$);

249$          ASSIGN:        Profile welding.WaitTime=Profile welding.WaitTime +
Diff.WaitTime;
213$          TALLY:         Profile welding.WaitTimePerEntity,Diff.WaitTime,1;
215$          TALLY:         Profile welding.TotalTimePerEntity,Diff.StartTime,1;
239$          ASSIGN:        Profile welding.VATime=Profile welding.VATime +
Diff.VATime;
240$          TALLY:         Profile welding.VATimePerEntity,Diff.VATime,1;
205$          RELEASE:       Op 7,1:
                             Welding Station 3,1;
254$          STACK,         1:Destroy:NEXT(253$);
253$          ASSIGN:        Profile welding.NumberOut=Profile welding.NumberOut +
1:
                             Profile welding.WIP=Profile welding.WIP-1:NEXT(15$);
;     Model statements for module:  BasicProcess.Process 11 (Bottoms welding)
;
15$           ASSIGN:        Bottoms welding.NumberIn=Bottoms welding.NumberIn + 1:
                             Bottoms welding.WIP=Bottoms welding.WIP+1;
285$          STACK,         1:Save:NEXT(259$);

259$          QUEUE,         Bottoms welding.Queue;
258$          SEIZE,         2,VA:
                             Welding Station 1,1:
                             Op 6,1:NEXT(257$);

257$          DELAY:         SecondsToBaseTime(EXPO( 120)),,VA:NEXT(300$);

300$          ASSIGN:        Bottoms welding.WaitTime=Bottoms welding.WaitTime +
Diff.WaitTime;
264$          TALLY:         Bottoms welding.WaitTimePerEntity,Diff.WaitTime,1;
266$          TALLY:         Bottoms welding.TotalTimePerEntity,Diff.StartTime,1;
290$          ASSIGN:        Bottoms welding.VATime=Bottoms welding.VATime +
Diff.VATime;
291$          TALLY:         Bottoms welding.VATimePerEntity,Diff.VATime,1;
256$          RELEASE:       Welding Station 1,1:
                             Op 6,1;
305$          STACK,         1:Destroy:NEXT(304$);

304$          ASSIGN:        Bottoms welding.NumberOut=Bottoms welding.NumberOut +
1:
                             Bottoms welding.WIP=Bottoms welding.WIP-1:NEXT(16$);
;     Model statements for module:  AdvancedProcess.Match 2 (Final welding)
;
16$           QUEUE,         Final welding.Queue1:DETACH;
19$           QUEUE,         Final welding.Queue2:DETACH;
              MATCH:         16$,28$:
                             19$,24$;
;     Model statements for module:  BasicProcess.Dispose 2 (Dispose 2)
;
24$           ASSIGN:        Dispose 2.NumberOut=Dispose 2.NumberOut + 1;
307$          DISPOSE:       Yes;
```

```
;     Model statements for module:  BasicProcess.Assign 2 (Almost profile entity
picture)
;
28$           ASSIGN:        Entity.Type=Almost pallet:
                             Picture=Picture.Box:NEXT(21$);
;     Model statements for module:  BasicProcess.Process 12 (Building up)
;
21$           ASSIGN:        Building up.NumberIn=Building up.NumberIn + 1:
                             Building up.WIP=Building up.WIP+1;
337$          STACK,         1:Save:NEXT(311$);

311$          QUEUE,         Building up.Queue;
310$          SEIZE,         2,VA:
                             Op 9,1:
                             Op 8,1:
                             Assembly station 1,1:NEXT(309$);
309$          DELAY:         SecondsToBaseTime(EXPO( 1187)),,VA:NEXT(352$);
352$          ASSIGN:        Building up.WaitTime=Building up.WaitTime +
Diff.WaitTime;
316$          TALLY:         Building up.WaitTimePerEntity,Diff.WaitTime,1;
318$          TALLY:         Building up.TotalTimePerEntity,Diff.StartTime,1;
342$          ASSIGN:        Building up.VATime=Building up.VATime + Diff.VATime;
343$          TALLY:         Building up.VATimePerEntity,Diff.VATime,1;
308$          RELEASE:       Op 9,1:
                             Op 8,1:
                             Assembly station 1,1;
357$          STACK,         1:Destroy:NEXT(356$);

356$          ASSIGN:        Building up.NumberOut=Building up.NumberOut + 1:
                             Building up.WIP=Building up.WIP-1:NEXT(22$);
;     Model statements for module:  BasicProcess.Process 13 (Assembly)
;
22$           ASSIGN:        Assembly.NumberIn=Assembly.NumberIn + 1:
                             Assembly.WIP=Assembly.WIP+1;
388$          STACK,         1:Save:NEXT(362$);

362$          QUEUE,         Assembly.Queue;
361$          SEIZE,         2,VA:
                             Op 11,1:
                             Op 10,1:
                             Assembly station 2,1:NEXT(360$);

360$          DELAY:         SecondsToBaseTime(EXPO( 1212)),,VA:NEXT(403$);

403$          ASSIGN:        Assembly.WaitTime=Assembly.WaitTime + Diff.WaitTime;
367$          TALLY:         Assembly.WaitTimePerEntity,Diff.WaitTime,1;
369$          TALLY:         Assembly.TotalTimePerEntity,Diff.StartTime,1;
393$          ASSIGN:        Assembly.VATime=Assembly.VATime + Diff.VATime;
394$          TALLY:         Assembly.VATimePerEntity,Diff.VATime,1;
359$          RELEASE:       Op 11,1:
                             Op 10,1:
                             Assembly station 2,1;
408$          STACK,         1:Destroy:NEXT(407$);

407$          ASSIGN:        Assembly.NumberOut=Assembly.NumberOut + 1:
                             Assembly.WIP=Assembly.WIP-1:NEXT(32$);
;     Model statements for module:  BasicProcess.Record 3 (Time between products)
;
32$           TALLY:         Time between products,BET,1:NEXT(27$);
;     Model statements for module:  BasicProcess.Record 1 (Count output)
;
27$           COUNT:         Count output,1:NEXT(39$);
;     Model statements for module:  BasicProcess.Record 6 (St. dev)
;
39$           TALLY:         St. dev,TSTD(Time between products),1:NEXT(23$);
;     Model statements for module:  BasicProcess.Dispose 1 (Dispose 1)
;
23$           ASSIGN:        Dispose 1.NumberOut=Dispose 1.NumberOut + 1;
```

```
410$           DISPOSE:       Yes;
;     Model statements for module:  BasicProcess.Dispose 3 (Dispose 3)
;
25$            ASSIGN:        Dispose 3.NumberOut=Dispose 3.NumberOut + 1;
411$           DISPOSE:       Yes;
;     Model statements for module:  BasicProcess.Create 8 (Create 8)
;

412$           CREATE,
1,SecondstoBaseTime(0.0),Profile:SecondstoBaseTime(EXPO(1500)),10:NEXT(413$);

413$           ASSIGN:        Create 8.NumberOut=Create 8.NumberOut + 1:NEXT(35$);
;     Model statements for module:  AdvancedProcess.Hold 4 (Hold 4)
;
35$            QUEUE,         Hold 4.Queue;
               SCAN:
                              Angles welding.WIP+ Hole drilling.WIP + Profile
cutting.WIP + Profile incisision.WIP + NQ(Sheet plus Profile.Queue2) < MaxWIP2
                              :NEXT(41$);
;     Model statements for module:  BasicProcess.Separate 2 (Separate 2)
;
41$            DUPLICATE,     100 - 50:
                              1,418$,50:NEXT(417$);

417$           ASSIGN:        Separate 2.NumberOut Orig=Separate 2.NumberOut Orig +
1:NEXT(30$);

418$           ASSIGN:        Separate 2.NumberOut Dup=Separate 2.NumberOut Dup +
1:NEXT(35$);
;     Model statements for module:  BasicProcess.Assign 4 (Profile entity picture)
;
30$            ASSIGN:        Picture=Picture.Green Ball:NEXT(3$);
;
;     Model statements for module:  BasicProcess.Process 4 (Profile cutting)
;
3$             ASSIGN:        Profile cutting.NumberIn=Profile cutting.NumberIn + 1:
                              Profile cutting.WIP=Profile cutting.WIP+1;
448$           STACK,         1:Save:NEXT(422$);

422$           QUEUE,         Profile cutting.Queue;
421$           SEIZE,         2,VA:
                              Op 3,1:
                              Saw frame,1:NEXT(420$);
420$           DELAY:         SecondsToBaseTime(EXPO( 349 )),,VA:NEXT(463$);

463$           ASSIGN:        Profile cutting.WaitTime=Profile cutting.WaitTime +
Diff.WaitTime;
427$           TALLY:         Profile cutting.WaitTimePerEntity,Diff.WaitTime,1;
429$           TALLY:         Profile cutting.TotalTimePerEntity,Diff.StartTime,1;
453$           ASSIGN:        Profile cutting.VATime=Profile cutting.VATime +
Diff.VATime;
454$           TALLY:         Profile cutting.VATimePerEntity,Diff.VATime,1;
419$           RELEASE:       Op 3,1:
                              Saw frame,1;
468$           STACK,         1:Destroy:NEXT(467$);

467$           ASSIGN:        Profile cutting.NumberOut=Profile cutting.NumberOut +
1:
                              Profile cutting.WIP=Profile cutting.WIP-1:NEXT(4$);
;     Model statements for module:  BasicProcess.Process 5 (Profile incisision)
;
4$             ASSIGN:        Profile incisision.NumberIn=Profile
incisision.NumberIn + 1:
                              Profile incisision.WIP=Profile incisision.WIP+1;
499$           STACK,         1:Save:NEXT(473$);

473$           QUEUE,         Profile incisision.Queue;
472$           SEIZE,         2,VA:
```

```
                             Op 3,1:
                             Grinder for profiles,1:NEXT(471$);
471$          DELAY:         SecondsToBaseTime(EXPO( 504 )),,VA:NEXT(514$);

514$          ASSIGN:        Profile incisision.WaitTime=Profile
incisision.WaitTime + Diff.WaitTime;
478$          TALLY:         Profile incisision.WaitTimePerEntity,Diff.WaitTime,1;
480$          TALLY:         Profile
incisision.TotalTimePerEntity,Diff.StartTime,1;
504$          ASSIGN:        Profile incisision.VATime=Profile incisision.VATime +
Diff.VATime;
505$          TALLY:         Profile incisision.VATimePerEntity,Diff.VATime,1;
470$          RELEASE:       Op 3,1:
                             Grinder for profiles,1;
519$          STACK,         1:Destroy:NEXT(518$);

518$          ASSIGN:        Profile incisision.NumberOut=Profile
incisision.NumberOut + 1:
                             Profile incisision.WIP=Profile incisision.WIP-
1:NEXT(5$);
;     Model statements for module:  BasicProcess.Process 6 (Hole drilling)
;
5$            ASSIGN:        Hole drilling.NumberIn=Hole drilling.NumberIn + 1:
                             Hole drilling.WIP=Hole drilling.WIP+1;
550$          STACK,         1:Save:NEXT(524$);

524$          QUEUE,         Hole drilling.Queue;
523$          SEIZE,         2,VA:
                             Op 4,1:
                             Drilling machine,1:NEXT(522$);

522$          DELAY:         SecondsToBaseTime(EXPO( 1026 )),,VA:NEXT(565$);

565$          ASSIGN:        Hole drilling.WaitTime=Hole drilling.WaitTime +
Diff.WaitTime;
529$          TALLY:         Hole drilling.WaitTimePerEntity,Diff.WaitTime,1;
531$          TALLY:         Hole drilling.TotalTimePerEntity,Diff.StartTime,1;
555$          ASSIGN:        Hole drilling.VATime=Hole drilling.VATime +
Diff.VATime;
556$          TALLY:         Hole drilling.VATimePerEntity,Diff.VATime,1;
521$          RELEASE:       Op 4,1:
                             Drilling machine,1;
570$          STACK,         1:Destroy:NEXT(569$);

569$          ASSIGN:        Hole drilling.NumberOut=Hole drilling.NumberOut + 1:
                             Hole drilling.WIP=Hole drilling.WIP-1:NEXT(13$);
;     Model statements for module:  BasicProcess.Process 9 (Angles welding)
;
13$           ASSIGN:        Angles welding.NumberIn=Angles welding.NumberIn + 1:
                             Angles welding.WIP=Angles welding.WIP+1;
601$          STACK,         1:Save:NEXT(575$);

575$          QUEUE,         Angles welding.Queue;
574$          SEIZE,         2,VA:
                             Op 5,1:
                             Welding Station 2,1:NEXT(573$);
573$          DELAY:         SecondsToBaseTime(EXPO( 304)),,VA:NEXT(616$);

616$          ASSIGN:        Angles welding.WaitTime=Angles welding.WaitTime +
Diff.WaitTime;
580$          TALLY:         Angles welding.WaitTimePerEntity,Diff.WaitTime,1;
582$          TALLY:         Angles welding.TotalTimePerEntity,Diff.StartTime,1;
606$          ASSIGN:        Angles welding.VATime=Angles welding.VATime +
Diff.VATime;
607$          TALLY:         Angles welding.VATimePerEntity,Diff.VATime,1;
572$          RELEASE:       Op 5,1:
                             Welding Station 2,1;
621$          STACK,         1:Destroy:NEXT(620$);
```

```
620$          ASSIGN:        Angles welding.NumberOut=Angles welding.NumberOut + 1:
                             Angles welding.WIP=Angles welding.WIP-1:NEXT(10$);
;     Model statements for module:  BasicProcess.Create 9 (Create 9)
;

623$          CREATE,
1,SecondstoBaseTime(0.0),Angle:SecondstoBaseTime(EXPO(1500)),10:NEXT(624$);

624$          ASSIGN:        Create 9.NumberOut=Create 9.NumberOut + 1:NEXT(37$);
;
;     Model statements for module:  AdvancedProcess.Hold 5 (Hold 5)
;
37$           QUEUE,         Hold 5.Queue;
              SCAN:          Angles cutting.WIP + Cup welding.WIP+NQ(Final
welding.Queue2) < MaxWIP3:NEXT(42$);
;     Model statements for module:  BasicProcess.Separate 3 (Separate 3)
;
42$           DUPLICATE,     100 - 50:
                             1,629$,50:NEXT(628$);
628$          ASSIGN:        Separate 3.NumberOut Orig=Separate 3.NumberOut Orig +
1:NEXT(29$);
629$          ASSIGN:        Separate 3.NumberOut Dup=Separate 3.NumberOut Dup +
1:NEXT(37$);
;     Model statements for module:  BasicProcess.Assign 3 (Angles entity picture)
29$           ASSIGN:        Picture=Picture.Yellow Ball:NEXT(6$);
;     Model statements for module:  BasicProcess.Process 7 (Angles cutting)
6$            ASSIGN:        Angles cutting.NumberIn=Angles cutting.NumberIn + 1:
                             Angles cutting.WIP=Angles cutting.WIP+1;
659$          STACK,         1:Save:NEXT(633$);
633$          QUEUE,         Angles cutting.Queue;
632$          SEIZE,         2,VA:
                             Op 6,1:
                             Press for cutting angles,1:NEXT(631$);
631$          DELAY:         SecondsToBaseTime(EXPO( 16)),,VA:NEXT(674$);
674$          ASSIGN:        Angles cutting.WaitTime=Angles cutting.WaitTime +
Diff.WaitTime;
638$          TALLY:         Angles cutting.WaitTimePerEntity,Diff.WaitTime,1;
640$          TALLY:         Angles cutting.TotalTimePerEntity,Diff.StartTime,1;
664$          ASSIGN:        Angles cutting.VATime=Angles cutting.VATime +
Diff.VATime;
665$          TALLY:         Angles cutting.VATimePerEntity,Diff.VATime,1;
630$          RELEASE:       Op 6,1:
                             Press for cutting angles,1;
679$          STACK,         1:Destroy:NEXT(678$);
678$          ASSIGN:        Angles cutting.NumberOut=Angles cutting.NumberOut + 1:
                             Angles cutting.WIP=Angles cutting.WIP-1:NEXT(12$);
;     Model statements for module:  BasicProcess.Process 8 (Cup welding)
;
12$           ASSIGN:        Cup welding.NumberIn=Cup welding.NumberIn + 1:
                             Cup welding.WIP=Cup welding.WIP+1;
710$          STACK,         1:Save:NEXT(684$);
684$          QUEUE,         Cup welding.Queue;
683$          SEIZE,         2,VA:
                             Op 6,1:
                             Welding Station 1,1:NEXT(682$);
682$          DELAY:         SecondsToBaseTime(EXPO( 192)),,VA:NEXT(725$);
725$          ASSIGN:        Cup welding.WaitTime=Cup welding.WaitTime +
Diff.WaitTime;
689$          TALLY:         Cup welding.WaitTimePerEntity,Diff.WaitTime,1;
691$          TALLY:         Cup welding.TotalTimePerEntity,Diff.StartTime,1;
715$          ASSIGN:        Cup welding.VATime=Cup welding.VATime + Diff.VATime;
716$          TALLY:         Cup welding.VATimePerEntity,Diff.VATime,1;
681$          RELEASE:       Op 6,1:
                             Welding Station 1,1;
730$          STACK,         1:Destroy:NEXT(729$);
729$          ASSIGN:        Cup welding.NumberOut=Cup welding.NumberOut + 1:
                             Cup welding.WIP=Cup welding.WIP-1:NEXT(19$);
```