

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES
--T L E M C E N--

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-



المدرسة العليا في العلوم التطبيقية
École Supérieure en
Sciences Appliquées

Engineer Senior Project

For obtaining the Engineering Diploma

Speciality: Industrial Engineering
Option: Industrial Management and Logistics

Presented by :

Rym BOUALI
Chawki Houssemeddine BOUKOFFA

Theme

**Performance Calculation of
manufacturing systems**

Publicly defended, on 07/14/2021, in front of a jury composed of:

Mr Mohammed BENNEKROUF	LCB	ESSA. Tlemcen	President
Mr Zaki SARI	Professor	ESSA. Tlemcen	Thesis director
Mr Fouad MALIKI	LCB	ESSA. Tlemcen	Co-Thesis director
Mrs Amina OUHOUD	LCB	ESSA. Tlemcen	Examiner 1
Mr Mohammed Adel HAMZAOU	Doctor	ESSA. Tlemcen	Examiner 2

College year : 2020 /2021

Dedication

I dedicate this modest work:

*To my parents to whom I owe everything and supported me
during all my school years*

To my brother who always had my back and cherish me

*To my sister who from 6000 miles always encourage me in
everything that I do and always has been there for me*

*And last but not least my dear fiancé who has always been
patient with me and caring*

Thanks for believing me even when I don't

Rym Bouali.

Dedication

I dedicate this work to those who believed in me:

My parents, Youssra, Fatima and Mrs BOUDISSA

You should be proud of me!

Chawki Houssemeddine BOUKOFFA

Thanks.

We would like to thank all of those who contributed to the success of our project and internship and who helped us in the preparation of this Senior Project.

First of all we would like to thank the members of our jury, Mr. Bennakrouf Mohamed president of the jury and teacher in the industrial engineering department at the Higher School of Applied Science – Tlemcen (ESSAT) and Mrs. Ouhoud Amina Teacher in Industrial Engineering also at ESSAT and Mr. Adel Hamzaoui PhD in Industrial Engineering also at ESSAT, who made us the honor of being present on our defence day to evaluate our work and share with us their expertise in the field concerned.

We would like to express our gratitude to our two coaches Mr. Zaki Sari, Professor of Industrial Engineering at the Tlemcen School of Applied Sciences (ESSAT), Mr Maliki Fouad, Head of Department Industrial engineering for their patience, their availability and above all their wise advice, which contributed to nourish our reflection and sharpen our critical mind in order to achieve our senior project.

We also thank the entire teaching team of the Tlemcen School of Applied Sciences (ESSAT) in particular Mr Maliki Fouad, Head of Department Industrial engineering for the training we have acquired during our course and the patience he had with all his students

Abstract.

Manufacturing systems are becoming more and more complex therefore different methods and approaches are used to understand and analyze their behavior in order have insights about their natural tendencies and gain the ability to understand, analyze and evaluate their performance. « Factory Physics® » -a systematic description of the underlying behavior of manufacturing systems as described by the authors Wallace J. Hopp and Mark L. Spearman- is one of the most complete approaches used to understand manufacturing systems.

The main goal of this project is to develop a desktop application that includes most of the equations and mathematical models described in the « Factory Physics® » book, them in a systematic approach use it to calculate and perform an extensive analysis of manufacturing system performance.

Résumé.

Les systèmes de production deviennent de plus en plus complexes, c'est pourquoi différentes méthodes et approches sont utilisées pour comprendre et analyser leur comportement afin d'avoir un aperçu de leurs tendances naturelles et d'acquérir la capacité de comprendre, analyser et évaluer leurs performances. "Factory Physics® est "une description systématique du comportement sous-jacent des systèmes de production". -comme décrit par les auteurs Wallace J. Hopp et Mark L. Spearman- est l'une des approches les plus complètes utilisées pour comprendre les systèmes de production.

L'objectif principal de ce projet est de développer une application de bureau qui inclut la plupart des équations et des modèles mathématiques décrits dans le livre " Factory Physics® ", les mettre en œuvre dans une approche systématique et les utiliser pour calculer et effectuer une analyse approfondie des performances des systèmes de production.

ملخص

أصبحت أنظمة التصنيع أكثر تعقيداً، وهذا هو السبب في استخدام منهجيات ومقاربات مختلفة لفهم وتحليل سلوكها من أجل اكتساب نظرة معمقة على ميولها الطبيعية واكتساب القدرة على فهم وتحليل وتقييم أداءها.

"Factory Physics®" - وصف منهجي للسلوك الأساسي لأنظمة التصنيع كما وصفه المؤلفان Wallace J. Hopp و Mark L. Spearman - هو أحد الأساليب الأكثر شمولاً المستخدمة لفهم تصنيع الأنظمة.

الهدف الرئيسي من هذا المشروع هو تطوير تطبيق سطح مكتب يتضمن معظم المعادلات والنماذج الرياضية الموضحة في كتاب "Factory Physics®" واستعمالها بطريقة منهجية واستخدامها لحساب وإجراء تحليل متعمق لـ أداء أنظمة التصنيع.

Summary.

General Introduction:	14
Chapter 1: Factory physics	17
Introduction:	17
1. Basic factory physics	18
1.1 Example : Hal company:	18
1.2 Simple Relationships:	19
1.2.1 Penny Fab One:	19
1.2.2 Best Case Performance :	19
Little's Law:	20
1.2.2.1 Law (Best-Case Performance):.....	20
1.2.3 Worst case performance:	21
1.2.3.1 Law (Worst-Case Performance):.....	21
1.2.4 Practical Worst-Case Performance :	21
1.2.4.1 State of a system:	22
1.2.4.2 Law (Practical Worst-Case Performance):	24
1.2.5 Internal Benchmarking:	24
1.2.6 Labor constrained:	25
1.2.6.1 Ample Capacity Case :	25
1.2.6.1.1 Law (Labor Capacity):.....	26
1.2.6.2 Full Flexibility Case:	26
1.2.6.3 CONWIP Lines with Flexible Labor	27
1.2.6.3.1 Law (CONWIP with Flexible Labor):.....	28
1.3 Conclusion:	28
2. Variability basics:	28
2.1 Introduction:	28
2.1.1 Variability and randomness:.....	29
2.2 Process Time Variability:	29
2.2.1 Measures and Classes of Variability :	29
2.2.1.1 Classes of Variability :	30
2.2.1.1.1 Low and Moderate Variability.....	30
2.2.1.1.2 Highly Variable Process Times :	31
2.2.2 Causes of Variability :	32
2.2.2.1 Natural Variability:.....	32
2.2.2.2 Variability from Preemptive Outages (Breakdowns):	33
2.2.2.3 Variability from Nonpreemptive Outages	36

2.2.2.4	Variability from Recycle:.....	38
2.3	Flow Variability:.....	38
2.3.1	Characterizing Variability in Flows	38
2.4	Variability Interactions—Queueing:.....	41
2.5	Effects of blocking:	43
2.6	Variability Pooling:	45
2.7	Conclusion:	45
3.	The Corrupting Influence of Variability:	46
3.1	Good and Bad Variability:.....	46
3.2	Variability laws:	46
3.3	Flow laws:	47
3.4	Batching laws:.....	47
3.5	The Cycle Time:	49
3.6	Performances and variability:	49
3.7	Diagnostic and improvements:	49
3.8	Conclusion:	51
4.	Push and Pull production systems:.....	51
4.1	Push VS Pull Systems:.....	51
4.2	The magic of Pull:	52
4.3	CONWIP:.....	52
4.4	Conclusion:	53
Chapter 2 : Application Modelling		55
1.	UML :	55
1.1	User Case Diagram:	55
1.2	Sequence Diagram:	56
1.3	Class Diagram:	57
2.	Programmation of the Application:.....	59
2.1	Python:	59
2.1.1	Why did we choose Python?	59
2.2	Visual Studio Code:.....	59
2.2.1	Why did we choose Visual Studio Code?	60
2.3	Code:	60
2.3.1	The general functioning:	60
2.3.2	The detailed functioning:	61
2.3.2.1	Cascading the stations:.....	62
2.3.2.2	The induction of line and factory parameters:	64

Chapter 3: Interface Desktop Application	68
1. User interface:	68
1.1 Definition:.....	68
1.2 GUI Programming in Python.....	68
1.3 The desktop application GUI:	68
2. Examples of Application:	72
2.1 Cardboard papers:	72
2.2 White Paper Company :	75
General Conclusion:	79
Bilbiographical & Webographic Research	80
Annex:	81
Annex 1: Screenshot of the result Excel File - Line Sheet - Example 1.....	81
Annex 2: Screenshot of the result Excel File - Product Sheet - Example 1.....	81
annex 3: Screenshot of the result Excel File - Factory Sheet - Example 1.....	82
Annex 4: Screenshot of the result Excel File - Line Sheet - Example 2.....	82
Annex 5: Screenshot of the result Excel File - Product Sheet - Example 2.....	83
Annex 6: Screenshot of the result Excel File - Station Sheet - Example 2.....	83
Annex 7: Screenshot of the result Excel File - Data Sheet - Example 2.....	84
Annex 9: Screenshot of the result Excel File - Data Sheet - Example 1.....	84

Table List.

Table 1: Hal Example	18
Table 2: Summary of the results about Penny Fab One	20
Table 3: Possible State for a System with 4 machines & 3 Jobs	22
Table 4: Classes of Variability	30
Table 5 : Parameters of Tortoise 2000 & Hare X19	33
Table 6: Push System VS Pull System	51
Table 7: Parameters of the workshop - Example1	72
Table 8: Parameters of the Workshop - Example 2	76

Figure List.

Figure 1: Throughput versus WIP in HAL example	25
Figure 2 : A low-variability distribution	30
Figure 3 : Low- and moderate-variability distribution.....	31
Figure 4 : Comparison of high- and low-variability distributions	32
Figure 5: High and Low CV arrivals	39
Figure 6: Cascade Equations.....	40
Figure 7: User Case	55
Figure 8: Sequence Diagram.....	57
Figure 9: Class Diagram	58
Figure 10: Screenshot of Application Program – Class	61
Figure 11: Screenshot of Application Program - Station Class	61
Figure 12: Screenshot of Application Program - Function of Class Station	61
Figure 13: Screenshot of Application Program -Creation of Station Object.....	62
Figure 14: Cascade Equations.....	62
Figure 15: Screenshot of Application Program – Cascade Equations.....	64
Figure 16: Screenshot of Application Program - Functions	64
Figure 17: Screenshot of Application Program - Functions Implementations.....	65
Figure 18: Screenshot of Application Program - Product Type Attribute.....	65
Figure 19: Screenshot of Application Program - Product Type Attribute in Line Class	66
Figure 20: Screenshot of Application Program - Performance Indicators.....	66
Figure 21:Screenshot of Application Program - Data Frames	66
Figure 22: Screenshot of the Interface Application- Main Window.....	69
Figure 23: Screenshot of the Excel File with the Import Data.....	70
Figure 24: Screenshot of the Interface Application -Station Window.....	70
Figure 25: Screenshot of the Interface Application- Lines Window	70
Figure 26: Screenshot of the Interface Application- Products Window	71
Figure 27: Screenshot of the result Excel File - Station Sheet	71
Figure 28: Screenshot of the result Excel File - Line Sheet.....	72
Figure 29: Screenshot of the result Excel File - Product Sheet.....	72
Figure 30: Screenshot of the Main Window –Example1	73
Figure 31: Screenshot of the Stations Window Part 1 –Example1	73
Figure 32: Screenshot of the Station Window Part 2 –Example1.....	74
Figure 33: Screenshot of the Lines Window –Example1	74
Figure 34: Screenshot of the Product Window –Example1.....	74
Figure 35: Screenshot of Open Excel File Window –Example1	75

Figure 36: Screenshot of the result Excel File - Station Sheet - Example1	75
Figure 37: Screenshot of the Main Window –Example 2	77
Figure 38: Screenshot of the Excel File –Example 2	77
Figure 39: Screenshot of the result Excel File - Station Sheet - Example 2	78

List of Abbreviations.

r_e : Average rate of the station.

r_a : rate of arrivals in jobs per unit time to station. In a serial line without yield loss or rework, $r_a = TH$ at every workstation.

r_d : Departure rate, measured in jobs per unit time ($1/t_d$).

t_a : Mean time between arrivals ($1/r_a$).

t_e : Mean effective process time, the rate (capacity) of the workstation is given by $r_e = m/t_e$.

t_d : Mean time between departures

c_a : Coefficient of variation (CV) of the inter-arrival times.

c_e : Coefficient of variation (CV) of the process time.

c_d : Coefficient of variation (CV) of the inter-arrival times.

σ_a : Standard deviation of the time between arrivals.

u : Utilization

m : number of parallel machines at station

N_t : number of demands (arrivals) in period t , a random variable. We assume demand is stationary over time, so that N_t has same distribution for each period t ; we also assume the period demands are independent.

μ_n : $E[N_t]$ = expected number of demands per period (in units).

σ_n : standard deviation of the number of demands per period (in units).

b : buffer size (i.e., maximum number of jobs allowed in system).

- The performance measures we will focus on are:

P_n : probability there are n jobs at station.

CT_q : expected waiting time spent in queue.

CT : expected time spent at station (i.e., queue time plus process time).

WIP : average WIP level (in jobs) at station.

WIP_q: expected WIP (in jobs) in queue.

A: the distribution of inter-arrival times.

B: the distribution of process times.

m: the number of machines at the station.

b: the maximum number of jobs that can be in the system.

D: constant (deterministic) distribution

M: exponential (Markovian) distribution

G: completely general distribution (e.g., normal, uniform)

r_a: the rate of potential arrival.

b: units in the system.

WIP_{nb}: the expected WIP in the system without any blocking.

ρ: “corrected” utilization.

k: serial batch size

t: time to process a single part.

s: time to perform a setup.

c_e: CV for batch (parts+setup).

c_a: CV for batch arrivals.

r_a: arrival rate for parts.

k: parallel batch size

t: time to process a batch

c_e: CV for batch

r_a: arrival rate for parts

c_a: CV of batch arrivals

B: maximum batch size

u_j(w): utilization of station j in CONWIP line with WIP level w

CT_j(w): cycle time at station j in CONWIP line with WIP level w

CT(w) = $\sum CT_j(w)$: cycle time of CONWIP line with WIP level w

TH(w): throughput of CONWIP line with WIP level w

WIP_j(w): average WIP level at station j in CONWIP line with WIP level w

GENERAL INTRODUCTION:

A deeper and a more insightful understanding of the dynamics of manufacturing systems has become an absolute necessity as their complexity is rising with every market change and evolution.

Quality, time, costs and responsiveness are the key words when it comes to companies competing as these parameters determine a big part of whether they will thrive in their markets or not. However, without using quantified and data driven approaches to understand its own manufacturing systems, no company is going to be able to make the right decisions, improve or at least understand their performance.

« Identify opportunities for improving existing systems, Design effective new systems and Make the trade-offs needed to coordinate policies from disparate areas » that was Wallace J. Hopp and Mark L. Spearman authors of « Factory Physics® » answer to the question: What is Factory Physics, and why should one study it?

These exact same words represent our motivation for this project which is a part of our engineering training at the Higher School of Applied Sciences Tlemcen.

Our main goal in this project was to make use of the « Factory Physics® » mathematical models and implement them in a systematic approach by trying to link equations that focused on calculations for an individual workstation with those that focused on calculation for a whole line and eventually using all the previous work to perform a general calculation for the whole manufacturing unit or factory.

The result is a desktop application that enable the user to calculate performance indicators for different workstations, different lines and routings for a Jobshop or a Flowshop layout and the whole manufacturing unit or factory by entering data using exclusively « Factory Physics® » models and equations.

In the process of linking the different equations, some slight modifications were made on the already existing equations and new equation were elaborated in order to keep the overall calculation coherent, all the changes that were made are documented in the chapters and their consistency was proved in calculation.

The 1st chapter of this thesis is a summary of the core chapters of the « Factory Physics® » third edition book, it includes all the equations and models implemented in our application, we tried to keep the essence of the authors words in supporting their equation and avoided over-reformulating it.

In the 2nd chapter we presented our Oriented Object approach in modeling the application using UML Diagrams. We also showed parts of our source code to explain the overall functioning as well as code lines that relate the divided equations to demonstrate the consistency of our work.

Finally, we used real data from real companies to perform calculations through our application and presented the results as well as the interface in the 3rd chapter.

Chapter

1

Factory Physics

In this chapter we will resume the core chapter of the book « Factory Physics » and present all the mathematical equations needed.

CHAPTER 1: FACTORY PHYSICS

INTRODUCTION:

Over the last years, manufacturing systems have become more and more complex. A “good” modeling approach is required in order to explain and understand the behavior of the current systems.

Analytic models include relationships between components of a system in a predefined structure with some detail omitted. Solution of such models typically yield information about long term or steady state average behavior. Analytic models and their solutions provide at least a starting point for gaining valuable information about system structure and behavior even if additional information is required. These models help in gaining mathematical insight into the cause and effect relationships that can govern or at least influence, the behavior of a system.

Factory Physics is a systematic description of the underlying behaviour of manufacturing systems.

1. BASIC FACTORY PHYSICS

The authors argued that manufacturing management needs a science of manufacturing because it offers a number of uses in this context such as: **precision**, **intuition** and **synthesis**. Chapter 7 is the beginning of a process to establish such a science. To motivate the measures and mechanics focused on a realistic example of HAL Company was given.

1.1 EXAMPLE : HAL COMPANY:

Hal is a computer company which manufactures Printed-Circuit Board Line. The basic process and the best capacity estimates are summarized in the table below

Process	Rate(time/hour)	Time(hour)
Lamination	191.5	1.2
Machining	186.2	5.9
Circuitize	150.5	6.9
Optical test/repair	157.8	5.6
Drilling	185.9	10.0
Copper plate	136.4	1.5
Procoat	146.2	2.2
Sizing	126.5	2.4
EOL test	169.5	1.8

Table 1: Hal Example

These values are averages, which account for the different types of PCBs manufactured by HAL and also the different routings. They also account for "detractors," such as machine failures, setup times, and operator efficiency. As such, the process rate gives an approximation of how many panels each process could produce per hour if it had unlimited inputs. The process time represents the average time a typical panel spends being worked on at a process, which includes time waiting for detractors but does not include time waiting in queue to be worked on.

The main performance measures emphasized by HAL are **throughput**, **cycle time**, and **work in process** and customer service. Over the past several months, throughput has averaged about **1,100** panels per day, or about **45.8** panels per hour (HAL works a 24-hours a day). WIP in the line has averaged about **37,009** panels, and manufacturing cycle time has been roughly **34** days, or **816** hours. Customer service has averaged about **75** percent.

In order to answer the question "How is Hal doing?" a comparison baseline must be established to be able to compare the actual performance with what is theoretically possible for this facility. Among this chapter the authors examined the extremes of behavior that are possible for simple idealized production lines, and used the resulting models to develop a scale with which to rate actual facilities. They returned to the HAL example and used this scale to evaluate the performance of its PCB line.

1.2 SIMPLE RELATIONSHIPS:

In the pursuit of a science of manufacturing, the fundamental question is “What are the relationships among WIP, throughput, and cycle time in a single production line?” The answer will depend on the assumptions made about the line. In this section, the authors gave a quantitative description of the range of possible behavior. This will serve to sharpen intuition about how lines perform and will provide a scale on which to benchmark actual systems.

To analyze and understand the behavior of a line under the best possible circumstances,

Namely, when process times are absolutely regular, a simulation of Penny Fab One was used.

1.2.1 PENNY FAB ONE:

Penny Fab One consists of a simple production line that makes giant one-cent pieces. The line consists of four machines in sequence that uses well-known, stable processes. Each machine takes exactly **two hours** to perform its operation. After each penny is processed, it is moved immediately to the next machine. The line runs 24 hours per day, with breaks, lunches, etc., covered by spare operators. For our purposes, the market for giant pennies can be assumed to be unlimited; thus, more **throughput** is unambiguously better for this system. The capacity of each machine is the same and equals one-half part per hour. Hence, any of the four machines can be regarded as the bottleneck and **$rb = 0.5$ penny per hour**. Such a line is said to be **balanced**, since all stations have equal capacity. Next, note that the raw process time is simply the sum of the processing times at the four stations, so **$T_0 = 8$ hours**. The critical **WIP** level is given by

$$W_0 = rb \cdot T_0 = 0.5 \times 8 = 4 \text{ pennies}$$

The line is operating under a **CONWIP** (constant WIP) protocol.

1.2.2 BEST CASE PERFORMANCE :

The simulation of Penny Fab One was started with one penny; the result was one penny coming out of the line every 8 hours. Then a second penny was added (starting both at the front of the line) and so on. The behavior of the line is summarized in the table below:

WIP	CT	%T	TH	%rb
1	8	100	0.125	25
2	8	100	0.250	50
3	8	100	0.375	75
4	8	100	0.5	100
5	10	125	0.5	100
6	12	150	0.5	100
7	14	175	0.5	100
8	16	200	0.5	100
9	18	225	0.5	100
10	20	250	0.5	100

Table 2: Summary of the results about Penny Fab One

LITTLE'S LAW:

Close examination of the table above reveals an interesting, and fundamental, relationship among WIP, cycle time, and throughput. At every WIP level, WIP is equal to the product of **throughput** and **cycle time**. This relation is known as **Little's law** (named for John D. C. Little, who provided the mathematical proof) and represents the first *factory physics* relationship:

$$\text{Law (Little's Law): } WIP = TH \times CT$$

Little's law is quite useful in that it can be applied to a single station, a line, or an entire plant. As long as the three quantities are measured in consistent units, the above relationship will hold over the long term. This makes it immensely applicable to practical situations.

1.2.2.1 LAW (BEST-CASE PERFORMANCE):

Generalizing the results shown in Table and gives a precise summary of the relationship between WIP and throughput for a "best-case" line. Applying Little's law extends this to describe the relationship between WIP and cycle time. Since these relationships were derived for perfect lines with no **variability**, the following expressions indicate the **maximum throughput** and **minimum cycle time** for a given WIP level for any system having parameters **rb** and **To**. The resulting equations are the next **Factory Physics law**.

$$CT_{\text{best}} = \begin{cases} T_o & \text{if } w \leq W_o \\ w/rb & \text{otherwise} \end{cases}$$

The maximum throughput for a given WIP level w is given by

$$TH_{best} = \begin{cases} w / T_o & \text{if } w \leq W_o \\ R_b & \text{otherwise} \end{cases}$$

In addition to the best case, considered above, two other scenarios will be treated, the worst case and the practical worst case.

1.2.3 WORST CASE PERFORMANCE:

Instead of imagining the best possible behavior of a line, the worst is considered. Specifically, the maximum **cycle time** and minimum **throughput** possible for a line with bottleneck rate r_b and raw process time T_o .

A simulation of Penny Fab One is used but instead of all jobs requiring two hours at each station jobs on pallet 1 require eight hours, while jobs on pallets 2, 3, and 4 require zero hours.

The line operates under **CONWIP**.

We still have $r_b = 0.5 \text{ job per hour}$ and $T_o = 8 \text{ hours}$.

The cycle time for this system is

$$8 + 8 + 8 + 8 = 32 \text{ hours}$$

Or $4T_o$, and since four jobs are output each time pallet 1 finishes on station 4, the throughput is

$$4/32 = 1/8 \text{ job per hour}$$

Or $1/T_o$ jobs per hour. Notice that the product of throughput and cycle time is $(1/8) \times 32 = 4$,

Which is the WIP level, so, as always, **Little's law** holds.

Let us summarize these results for a general line as the next **factory physics law**.

1.2.3.1 LAW (WORST-CASE PERFORMANCE):

The worst-case cycle time for a given WIP level w is given by:

$$CT_{worst} = wT_o$$

The worst-case throughput for a given WIP level w is given by:

$$TH_{worst} = 1 / T_o$$

1.2.4 PRACTICAL WORST-CASE PERFORMANCE :

Both the best-case and worst-case performances occur in systems with no randomness. There is **variability** in the worst-case system, since jobs have different process times; but there is no **randomness**, since all process times are completely predictable. Virtually no real-world line behaves literally according to either the best case or the worst case. Therefore, to better understand the behavior between these two extreme cases, it is instructive to consider an intermediate case that, unlike the previous two, involves **randomness**. In fact, it represents the "maximum randomness" case" the practical worst case".

1.2.4.1 STATE OF A SYSTEM:

The state of the system is a complete description of the jobs at all the stations: how many there are and how long they have been in process. Under special conditions, assumed here, the only information needed is the number of jobs at each station. Hence, it is possible to give a concise summary of a state by using a vector with as many elements as there are stations in the line.

Possible States for a System with Four Machines and Three Jobs:

State	Vector	State	Vector
1	(3, 0, 0, 0)	11	(1, 0, 2, 0)
2	(0, 3, 0, 0)	12	(0, 1, 2, 0)
3	(0, 0, 3, 0)	13	(0, 0, 2, 1)
4	(0, 0, 0, 3)	14	(1, 0, 0, 2)
5	(2, 1, 0, 0)	15	(0, 1, 0, 2)
6	(2, 0, 1, 0)	16	(0, 0, 1, 2)
7	(2, 0, 0, 1)	17	(1, 1, 1, 0)
8	(1, 2, 0, 0)	18	(1, 1, 0, 1)
9	(0, 2, 1, 0)	19	(1, 0, 1, 1)
10	(0, 2, 0, 1)	20	(0, 1, 1, 1)

Table 3: Possible State for a System with 4 machines & 3 Jobs

Depending on the specific assumptions about the line, not all states will necessarily occur. For instance, if all processing times in the four-station, three-job system are one hour and it behaves according to the best case, then only four states (1, 1, 1, 0), (0, 1, 1, 1), (1, 0, 1, 1), and (1, 1, 0, 1)-will be repeated. Similarly, if it behaves according to the worst case, then four different states-(3, 0, 0, 0), (0, 3, 0, 0), (0, 0, 3, 0), and (0, 0, 0, 3)-will be repeated. Because both of these systems have no randomness, other states are never reached.

When randomness is introduced into a line, more states become possible. For instance, suppose the processing times are deterministic, but every once in a while a machine may break down for several hours. Then most of the time we will observe "spread out" states, but occasionally we will see "clumped up" states. If there is only a little randomness (e.g., machine failures are very rare), then the frequency of the

spread-out states will be very high, whereas if there is a lot of randomness (e.g., machines are failing right and left), then all the states may occur quite often. Hence, we define the maximum randomness scenario to be that which causes every possible state to occur with equal frequency.

In order for all states to be equally likely, three special conditions are required:

1. The line must be balanced (i.e., all stations must have the same average process times).
2. All stations must consist of single machines. (This assumption also allows us to avoid the complexities of parallel processing and jobs passing one another.)
3. Process times must be random and occur according to a specific probability distribution known as the **exponential distribution**.

Suppose there are N (single machine) stations, each with average processing times of t , and a constant level of w jobs in the line. Thus, the raw process time is $T_o = Nt$, and the bottleneck rate is $rb = 1/t$ for this line. Since the above three conditions guarantee that all states are equally likely, then, from your vantage point on a pallet, you would expect to see on average the $w - 1$ other jobs equally distributed among the N stations each time you arrive at a station. So the expected number of jobs ahead of you upon arrival is $(w - 1)/N$. Since the average time you spend at the station will be the time for the other jobs to complete processing plus the time for your job to be processed, we can write

Average time at a station = Time for other jobs + Time for your job

$$\text{Average time at a station} = ((w - 1)/N)t + t$$

$$\text{Average time at a station} = (1 + (w - 1)/N)t$$

By assuming that the $(w-1)/N$ jobs ahead of you require an average of $[(w - 1)/N]t$ Time to complete, we are ignoring the fact that the job in process at the station was partially finished when you arrived. It is the memoryless property of the exponential distribution that enables us to do this.

Finally, since all stations are assumed identical, we can compute the average cycle time by simply multiplying the average time at each station by the number of stations N , to get

$$\begin{aligned} CT &= N(1 + (w - 1)/N)t \rightarrow CT = Nt + (w - 1)t \\ &\rightarrow CT = Nt + (w - 1)t \\ &\rightarrow CT = T_o + (w - 1)/rb \end{aligned}$$

To get the corresponding throughput, we simply apply Little's law:

$$TH = \frac{WIP}{CT} \rightarrow TH = w / (T_o + (w - 1)/rb)$$

$$\rightarrow TH = w / (Wo / rb + (w - 1) / rb)$$

$$\rightarrow TH = (w / Wo + w - 1) rb$$

1.2.4.2 LAW (PRACTICAL WORST-CASE PERFORMANCE):

The practical worst-case (PWC) cycle time for a given WIP level w is given by

$$CT_{pwc} = To + (w - 1) / rb$$

The PWC throughput for a given WIP level w is given by

$$TH_{pwc} = (w / (Wo + w - 1)) rb$$

1.2.5 INTERNAL BENCHMARKING:

At this point, the tools necessary to reconsider the HAL example are developed. It is now possible to compare the PCB line's actual performance with the best, the worst and the practical worst case.

The bottleneck is simply the process with the smallest capacity. This is sizing with $rb = 126.5$ panels per hour. The raw process time is simply the sum of the process times in, which is $To = 33.1$ hours. Hence, the critical WIP for the line is:

$$Wo = rb \times To = 126.5 \times 33.1 = 4,187 \text{ panels}$$

Recalling that the actual throughput was **45.8 panels per hour**, actual cycle time was **816 hours**, and actual WIP level was **37,000 panels**, we can make some quick observations. First we make a quick Little's law check of the data:

$$\begin{aligned} TH \times CT &= 1,100 \text{ panels/day} \times 34 \text{ days} \\ &= 37,400 \text{ panels} \sim 37,000 \text{ panels} \end{aligned}$$

Since Little's law applies precisely only to long-term averages, it is not expected for it to hold exactly. However, this is certainly well within the precision of the data and hence suggests no problems.

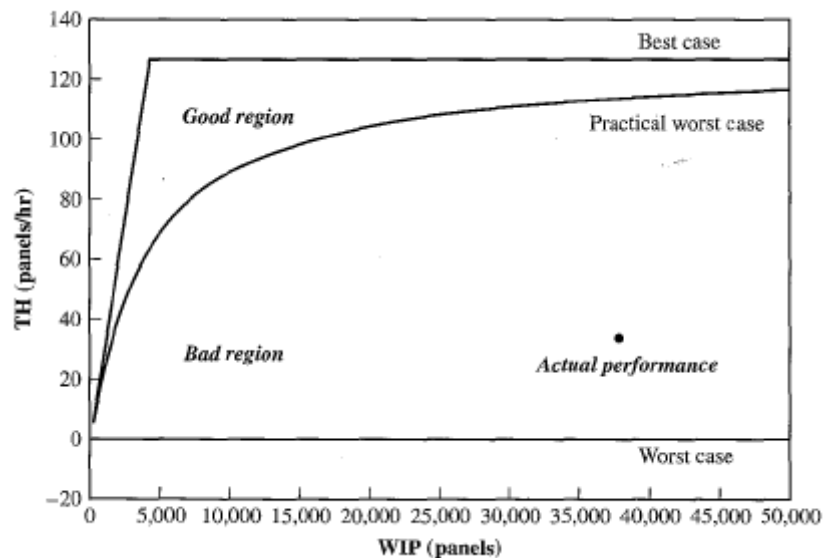


Figure 1: Throughput versus WIP in HAL example

1.2.6 LABOR CONSTRAINED:

Throughout this chapter, the focus has been on lines in which machines are the primary constraint. However, in some systems, workers perform multiple tasks or tend more than one workstation. These types of systems exhibit more complex behavior than the simple lines considered so far, since the flow of work is affected by the number and characteristics of both machines and operators. Although the subject of flexible labor is much too broad to treat comprehensively here, it is possible to make some observations about how labor-constrained lines relate to the simple lines presented earlier. Three situations will be considered

1.2.6.1 AMPLE CAPACITY CASE :

The first situation is the one in which labor is the only constraint on output. A realistic situation that approximates this behavior is the example of a prepress graphical production facility of catalogs and other marketing materials. This firm received content (text, photos, etc.) from its clients and converted these to electronic engraving data via a series of steps (e.g., scanning, color correction, page finishing), which it then sent to a printer to be made into paper products. Most of the prepress steps required a computer along with some peripheral equipment. Because computer equipment was inexpensive relative to the cost of delays, the firm installed enough duplicates of each station to ensure that technicians virtually never had to wait for equipment to perform the various tasks. The result was many more machines than people, which meant that labor was the key constraint in the system. A primary reason the graphics company installed ample capacity at its stations was to facilitate its flexible labor policy. Instead of having specialists for each operation, the company had cross-trained the workforce so that almost everyone could do almost every operation. This allowed the company to assign workers to jobs instead of stations. A

worker would follow a job through the system, performing each operation on the appropriate workstation.

In a system like this, capacity is defined by labor rather than equipment. To characterize capacity, T_o will represent the average time for one job to traverse the system, which is assumed independent of which worker is assigned to the job. Furthermore, also once a worker starts a job, he or she continues with it until it is done. Stopping work midway through a job cannot improve throughput and will only increase cycle time, so unless some customers have higher priority than others, there is no reason to do this. Under these assumptions, jobs are released into the system only when a worker becomes available, and since there is no blocking due to equipment, cycle time is always T_o . If there are n workers in the line, all working at the same rate, then each puts out a job every T_o time units, which means that throughput is n/T_o . Since the ample capacity case is an ideal situation, any changes to our assumptions can only decrease throughput. Examples of such changes include less-than-ample equipment so that blocking occurs, intermittent arrival of work that may cause starving, partial cross-training so that jobs may have to wait for a "specialist" at some stations, or any other change that prevents workers from being completely busy. Hence, the following factory physics law.

1.2.6.1.1 LAW (LABOR CAPACITY):

$$TH_{max} = n / T_o$$

This law provides a way to introduce labor into the capacity calculations. For instance, in a line that has more stations than workers, the bottleneck rate of the equipment rb may be a poor estimate of the capacity of the line. Where throughput is constrained by labor, n/T_o may be a more realistic and useful upper bound on capacity. This bound is applicable to a wide range of systems, including those with fully or partially cross-trained workers. One class of systems to which it does not apply, however, is that in which a worker can process more than one job simultaneously.

1.2.6.2 FULL FLEXIBILITY CASE:

The next case is the one in which workers are completely cross-trained. Furthermore it is assumed that workers are tied to jobs as in the ample capacity case. However, unlike in the ample capacity case, equipment is limited so workers may become blocked, . Once a worker finishes a job at the end of the line, he goes back to the beginning and starts a new one. If the workers have identical work rates, then this line is logically identical to the CONWIP lines considered previously, except that the WIP level is now the number of workers. Hence, the behavior of the line will lie somewhere between the best and worst cases, with the practical worst case defining the division between good and bad lines. Furthermore, all the improvement strategies listed earlier increasing capacity, reducing line balance, using parallel machine stations, and reducing variability-still apply to this case. The assumption of fully cross-trained workers who walk jobs all the way through the line may not be realistic in many situations. For instance, if the workstations require very different skills, it may make sense to have workers pass jobs from one to another. One mechanism is the

bucket brigade (see Bartholdi and Eisenstein 1996). In this system, whenever the 111 worker farthest downstream in the line completes a job, he or she moves up the line and takes the job from the next worker upstream. That worker in turn moves upstream and takes the job from the next worker. And so on, until the worker farthest upstream takes a new job. If all workers work at the same speed and there is no delay due to the handing off of the jobs, then there is no logical difference in this system from the one depicted in Figure 7.14. The line still operates as a CONWIP line with the WIP level set by the number of workers. Only the identities of the workers assigned to each job are changed. While the bucket brigade system may not differ logically from the system with workers tied to jobs, it does differ practically. Each worker will tend to operate machines in a zone. Indeed, in the case where all process times are perfectly deterministic (i.e., the best case), the line will settle into a repetitive cycle where each worker processes jobs through the same sequence of stations. The cross-training and job transfers allow the line to balance itself so that each worker spends the same amount of time with a job. Notice that blocking is still possible in the bucket brigades. Whenever an upstream worker catches up with the next worker downstream, he will be blocked unless the station has extra equipment. Hence, it makes sense to organize the workers so as to minimize the frequency with which this happens, by placing the fastest workers downstream and the slowest workers upstream. Bartholdi and Eisenstein (1996) show that this arrangement from slowest to fastest can significantly improve throughput and observed that this tends to be the practice in industry where such systems are used.

1.2.6.3 CONWIP LINES WITH FLEXIBLE LABOR

If workers stay tied to jobs (or hand off jobs directly to one another as in the bucket brigade system), then the number of jobs in the system always equals the number of workers and the system behaves logistically as a CONWIP line. But in many, if not most, systems, the number of jobs will typically exceed the number of workers. If workers can rove through the system and work at different stations, then the performance of the system will depend on how effectively labor is allocated to promote flow through the system. This can get complex, since there are countless ways that labor can be dynamically allocated in the system. One approach, which is a natural extension of the bucket brigade system to the case with more jobs than workers, is to have any worker who becomes free take the next job upstream, either from the upstream worker or from a buffer. Whenever a worker becomes blocked because a downstream station is busy, the worker drops the job in the buffer in front of the station and moves upstream to get another job. This continues as long as the total number of jobs in the system does not exceed some preset limit (without such a limit, a fast worker at the front of the line would flood the line with WIP).

If all stations consist of single machines, so that no passing is possible, then at any time worker n (the last worker in the line) will be working on the job farthest downstream. Worker $n - 1$ will be working on the next-farthest job downstream that is not blocked by worker n . And so on. If passing on multi-machines stations is possible, then the workers can get out of order. But the basic intent is still to keep workers working whenever possible on the jobs farthest downstream. Keeping workers busy tends to maximize throughput; working on downstream jobs tends to minimize cycle times. Hence, we would expect this policy to work reasonably well. Of course, other flexible labor policies are possible. Which is appropriate depends on a variety of

factors, including the degree of worker cross-training, the relative speed of the workers at the different stations, and the efficiency with which jobs can be passed from one worker to another. If there is no difference in the speed of workers, then the throughput of the system depends entirely on how often unblocked jobs are idle for lack of a worker. If this never happens, then the system will operate like a regular CONWIP line. If it happens so frequently that the workers might just as well be tied to one job each, then the system will operate as a CONWIP line with only as many jobs as workers. Hence, we can bound the throughput of a CONWIP line with flexible workers as in the following factory physics law.

1.2.6.3.1 LAW (CONWIP WITH FLEXIBLE LABOR):

In a CONWIP line with n identical workers and w jobs, where $w \geq n$, any policy that never idles workers when unblocked jobs are available will achieve a throughput level $TH(w)$ bounded by

$$TH_{cw}(n) \leq TH(w) \leq TH_{cw}(w)$$

Where $TH_{cw}(x)$ represents the throughput of a CONWIP line with all machines staffed by workers and x jobs in the system.

1.3 CONCLUSION:

In this chapter the fundamental behavior of a single production line was examined by studying the relationships among cycle time, WIP, throughput, and capacity.

A thread that has emerged from this analysis of basic factory dynamics is that a line can achieve the same throughput at a lower WIP level by either increasing capacity or improving the efficiency of the line. To be able to evaluate the relative effectiveness of capacity increases versus variability reduction, the science of factory physics must be further developed to describe the behavior of production systems involving randomness. That will be done in the next chapters.

2. VARIABILITY BASICS:

2.1 INTRODUCTION:

Little's law implies that it is possible to achieve the same throughput with long cycle time and large WIP or short cycle time and small WIP.

Penny Fab One from Chapter 7 achieves full throughput at a WIP level of

$W_o = 4$ jobs (the critical WIP) If it behaves like the best case. But if it behaves like the practical worst case, it requires a WIP level of **27 jobs** to achieve **90 percent of capacity**. If it behaves like the worst case, **90 percent of capacity** is not even feasible. The big difference is **Variability**.

Variability exists in all production systems and can have an enormous impact on performance. For this reason, the ability to measure, understand, and manage variability is critical to effective manufacturing management. **In** this chapter basic tools and intuition for characterizing variability in production systems will be developed, for precision, there are points at which the formal language of probability

must be used. **In** particular, the concept of a **random variable** and its characterization via its **mean** and **standard deviation** are essential.

2.1.1 VARIABILITY AND RANDOMNESS:

Variability is closely associated with (but not identical to) **randomness**. Therefore, to understand the causes and effects of variability, one must understand the concept of randomness and the related subject of **probability**.

The worst and the practical worst case are both a behavior of a system whose performance is degraded by variability. However to understand the difference one must distinguish between controllable variation and random variation.

Controllable variation occurs as a direct result of decisions. For instance, if several products are produced in a plant, there will be variability in the product descriptors (e.g., their physical dimensions, time to manufacture, etc.). Likewise, if material is moved in batches from one process to the next, the first part to finish will have to wait longer to move than the last part, and so waiting times will be more variable than if moved one at a time.

In contrast, **random variation** is a consequence of events beyond immediate control. For example, a machine failure is not known. Such downtime adds to the effective process time of a job, since the job must wait for the machine to be repaired before completing processing. Since such contingencies cannot be predicted or controlled (at least in the immediate term), machine outages increase the variability of effective process times in a random fashion. Although both types of variation can be disruptive to a plant, the effects of random variation are more subtle and require more sophisticated tools to describe. For this reason, the focus will be mainly on the random variation in this chapter.

2.2 PROCESS TIME VARIABILITY:

The random variable of primary interest in factory physics is the **effective process time** of a job at a workstation.

2.2.1 MEASURES AND CLASSES OF VARIABILITY :

To effectively analyze variability, it has to be quantified. It can be done by using standard measures from statistics to define a set of factory physics variability classes.

Variance, commonly denoted by σ^2 (sigma squared), is a measure of absolute variability, as is the standard deviation σ , defined as the square root of the variance.

Often, however, **absolute variability** is less important than **relative variability**. A reasonable relative measure of the variability of a random variable is the standard deviation divided by the mean, which is called the coefficient of variation (**CV**).

The mean will be denoted (**t**) and σ denotes the variance, the coefficient of variation **c** can be written

$$c = \sigma/t$$

In many cases, it turns out to be more convenient to use the squared coefficient of variation (**SCV**)

$$SCV = c^2$$

2.2.1.1 CLASSES OF VARIABILITY :

Variability class	Coefficient of variation	Typical situation
<ul style="list-style-type: none"> Low(LV) 	$cv \leq 0.75$	<ul style="list-style-type: none"> Process times without outages
<ul style="list-style-type: none"> Moderate(MV) 	$0.75 \leq cv \leq 1.33$	<ul style="list-style-type: none"> Process times with short adjustments
<ul style="list-style-type: none"> High(HV) 	$cv \geq 1.33$	<ul style="list-style-type: none"> Process times with long outages (e.g., failures)

Table 4: Classes of Variability

2.2.1.1.1 LOW AND MODERATE VARIABILITY

Process times tend to have probability distributions that look like the classic bell-shaped curve. Figure 1 shows the probability distribution for process times with a mean of 20 minutes and a standard deviation of 6.3 minutes. The CV for this case is around 0.32, so it is in the low variability (LV) range. It is a characteristic of most LV process times to have a bell-shaped probability density.

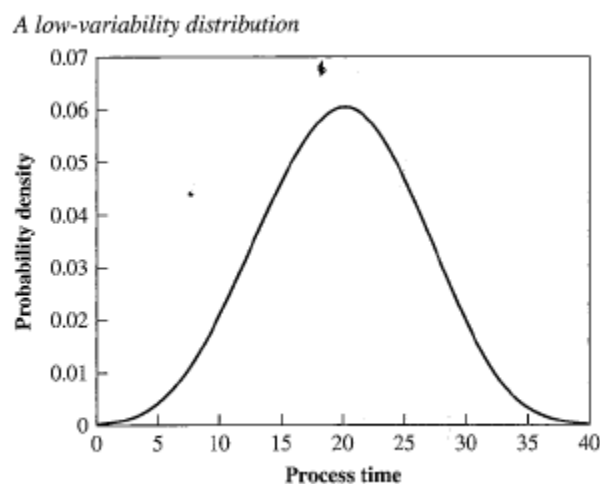


Figure 2 : A low-variability distribution

Now in a situation with a mean process time of 20 minutes but for which the CV is around 0.75, the beginning of the moderate-variability case. Figure 2 compares the two distributions. Notice that the LV case has most of its probability concentrated near the mean of 20. In the moderate-variability (MV) case, the most likely times are actually lower than the mean, around nine minutes. However, while the LV plot tails off around 40, the MV plot does not do so until around 80. Thus the means are the same, but the variances are much different. This difference is critical to the operational performance of a workstation.

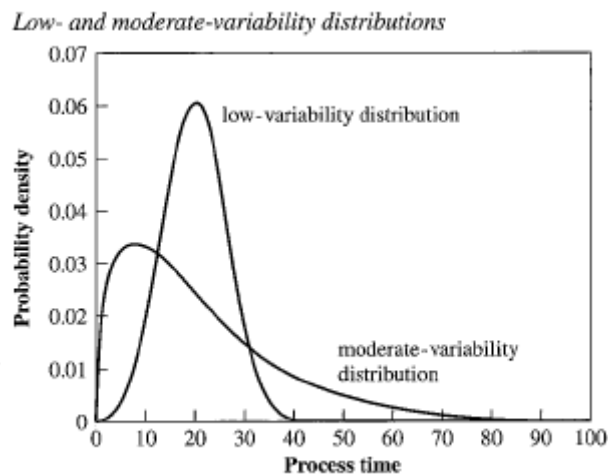


Figure 3 : Low- and moderate-variability Distribution

To get a sense of the operational effects of variability, suppose the **LV** process is feeding the **MV** process. For a while, the **MV** process will be able to keep up easily. However, once a long process time occurs, a queue of work begins to build in front of the second process. Offhand one might think that the long process times will be offset by the short process times, but this does not happen. A string of short process times at the second station might deplete the queue, causing the second station to become idle. When this occurs, capacity is lost and cannot be "saved up" for the next period of longer process times.

For now, we note that the greater the variability in effective process times, the greater the average queue. Given Little's law, this also implies that the greater the variability, the longer the cycle time.

2.2.1.1.2 HIGHLY VARIABLE PROCESS TIMES :

Suppose a machine has an average process time of 15 minutes with a CV of 0.225 when there are no outages. This would be less variable than the previous low-variability case. But now suppose the machine has outages that average 248 minutes and occur, on average, after 744 minutes of production. We can show (details are given later) that this results in an effective mean process time of 20 minutes (as before) and an effective CV of a whopping 2.5! Figure 3 compares this high-variability (HV) distribution with the previous LV distribution. Because the HV distribution is taller and thinner, it might appear less variable than the LV distribution. This is because one cannot see what is happening farther out in time. Once past 40 minutes or so, the picture changes. Figure 4 compares the distributions on a different scale for time greater than 40 minutes. Here it can be seen that the LV distribution immediately drops to almost no probability while the HV distribution appears almost uniform. It is going down very slowly indeed. This implies that there is a small probability that the process times will be extremely long. It is also the reason that the distribution for the highly variable process times appears to have a lower mean on the other plot. Most of the time, it takes around 15 minutes. However, about out of every

50 jobs takes around 17 times as long. This in ates the mean to around 20 and drives the CV up to 2.5.

The effect of this level of variability on the production line can be severe. For instance, suppose the throughput is one job every 22 minutes. There should be no problem from a capacity perspective since the average process time including outages is 20 minutes. However, an outage of 250 minutes will build up a queue of almost 12 jobs. When the machine Comes back up, the rate at which this queue is depleted is $1/16 - 1/22 = 1/47$. Thus, the time to clear the queue formed would be around 536 minutes, assuming no more outages occur! If an outage occurs during this time, it adds to the queue. Under conditions commonly found with complex equipment (i.e., times to failure that are exponentially distributed), the probability of such an outage is $1 - \exp(536/744) = 0.51$. This means that more than 50 percent of the time an outage occurs before the queue would be cleared. Thus the average queue will be greater than 12 jobs.

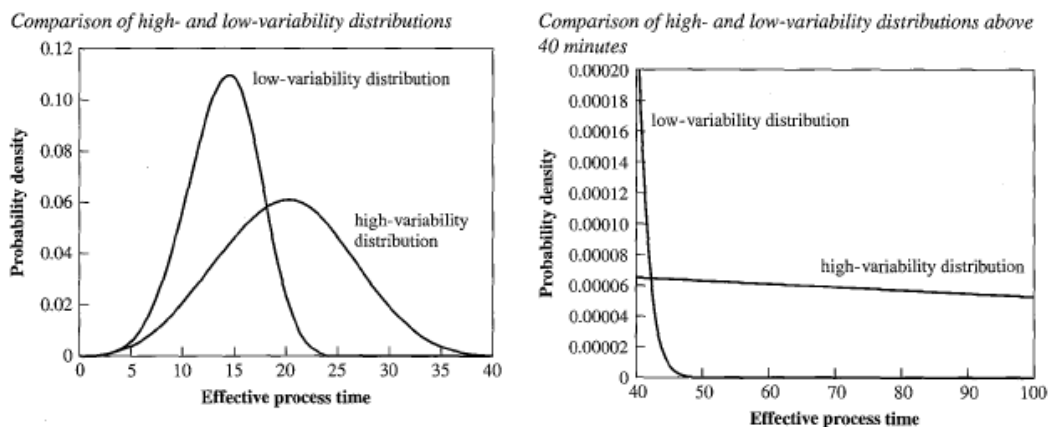


Figure 4 : Comparison of high- and low-variability distributions

2.2.2 CAUSES OF VARIABILITY :

To identify strategies for managing production systems in the face of variability, it is important to first understand the causes of variability. The most prevalent sources of variability in manufacturing environments are:

- "Natural" variability, which includes minor fluctuations in process time due to differences in operators, machines, and material.
- Random outages.
- Setups.
- Operator availability.
- Recycle.

2.2.2.1 NATURAL VARIABILITY:

Natural variability is the variability inherent in **natural process time**, which excludes random downtimes, setups, or any other external influences. In a sense, this is a catchall category, since it accounts for variability from sources that have not been explicitly called out. Because many of these unidentified sources of variability are operator-related, there is typically more natural variability in a manual process than in an automated one. But even in the most tightly controlled processes, there is always

some natural variability. For instance, in fully automated machining operations, the composition of the material might differ, causing processing speed to vary slightly. We let t_o and σ_o denote the mean and standard deviation, respectively, of natural process time. Thus, we can express the coefficient of variation of natural process time as:

$$co = \sigma_o / t_o$$

In most systems, natural process times are LV and so $Co < 0.75$.

Natural process times are only the starting point for evaluating effective process times. In any real production system, workstations are subject to various **detractors** that serve to inflate both the mean *and* the standard deviation of effective processing time.

2.2.2.2 VARIABILITY FROM PREEMPTIVE OUTAGES (BREAKDOWNS):

Breakdowns are referred to as **preemptive outages** because they occur whether wanted or not (e.g., they can occur right in the middle of a job). Power outages, operators being called away on emergencies, and running out of consumables are other possible sources of preemptive outages. Since these have similar effects on the behavior of production lines, it makes sense to combine them and treat them all as machine breakdowns in the fashion discussed (i.e., include outages due to these other sources, as well as true machine breakdowns, when computing MTTF and MTTR).

To see how machine outages cause variability, the Briar Patch Manufacturing example is considered:

Briar Patch Manufacturing has two very similar workstations as part of its plant. Both are composed of a single machine that runs at a rate of four jobs per hour (when it is not down). Both are subject to the same pattern of demand with an average work load and both are subject to periodic unpredictable outages.

Parameter/Machine	Tortoise 2000	Hare X19
t_o	15 min	15 min
σ_o	3.35 min	3.35 min
SCV	0.05	0.05
Availability	75%	75%
Outages	Short and frequent	Long and infrequent
mf	114 min	744 min
mr	38 min	248 min

Table 5 : Parameters of Tortoise 2000 & Hare X19

We suppose that repair times are variable and have $CV = 1.0$ (moderate variability) for both machines. Most capacity planning tools used in industry account for random outages when computing *average* capacity. This is done by computing the **availability**, which is given in terms of m_f and m_r by:

$$A = m_f / (m_f + m_r)$$

Adjusting the natural process time t_o to account for the fraction of time the machine is unavailable results in an **effective mean process time** t_e of

$$t_e = t_o / A$$

In both cases, $t_e = 20$ minutes.

$$r_e = m / t_e = A \cdot m / t_o = A \cdot r_o = 0.75(4 \text{ jobs/hour}) = 3 \text{ jobs/hour}$$

The effective capacity of the Hare X19 and the Tortoise 2000 is the same. Considering only the effects of breakdowns on availability and capacity, the two workstations would generally be regarded as equivalent.

However, when variability effects are included, the workstations are very different.

Consider how they will behave as part of a production line. If the Hare X19 fails for 12.4 hours (its average failure duration), it will need 12.4 hours of WIP to keep from starving. On the other hand, the Tortoise 2000 needs less than one-sixth as much WIP to be covered for an average-length failure. Since failures are, by their very nature, random, the WIP in the downstream buffer must be maintained at all times to provide protection against throughput loss. Clearly, a line with the Tortoise 2000 will be able to achieve the same level of protection, and hence the same level of throughput, with less **WIP**, than same line with the Hare X19.³ The net effect is that the line with the Hare X19 will be less efficient (i.e., will achieve lower throughput for a given **WIP** level or will have higher **WIP** and cycle time for the same throughput) than the line with the Tortoise 2000.

The CV for the Hare X19 was 2.5. Assuming that the times to failures are exponentially distributed (i.e., they are MV). However, no particular assumptions are made about the repair times other than that they are from some probability distribution. σ_r is defined to be the standard deviation of these repair times and ($cr = \sigma_r / m_r$) to be the **Cv**. In the example **Cr** is 1.0 (i.e., assuming that repair times have moderate variability). Under these assumptions the mean can be calculated, variance, and squared coefficient of variation (SCV) of the effective process time (t_e , σ_e^2 and ce^2 respectively) as

$$t_e = \frac{t_0}{A} \quad (8.4)$$

$$\sigma_e^2 = \left(\frac{\sigma_0}{A}\right)^2 + \frac{(m_r^2 + \sigma_r^2)(1 - A)t_0}{Am_r} \quad (8.5)$$

$$c_e^2 = \frac{\sigma_e^2}{t_e^2} = c_0^2 + (1 + c_r^2)A(1 - A)\frac{m_r}{t_0} \quad (8.6)$$

The CV of effective process time ce can be computed by taking the square root of ce^2 . Notice that the mean effective process time, given by Equation (8.4), depends only on the mean natural process time and the availability and is hence the same for both stations:

$$t_e = \frac{t_0}{A} = \frac{15}{0.75} = 20.0 \text{ minutes}$$

However, the SCV of effective process time in Equation (8.6) depends on more than the mean process time and availability. To understand the effects involved, we can rewrite (8.6) as

$$c_e^2 = c_0^2 + A(1 - A)\frac{m_r}{t_0} + c_r^2 A(1 - A)\frac{m_r}{t_0}$$

The first term is due to the natural (unaccounted for) variability in the process. The second term is due to the fact that there are random outages. Note that this term would be there even if the outages themselves (i.e., the repair times) were constant (i.e., even if $cr = 0$). For instance, a periodic adjustment that always takes the same time to complete would have

$cr^2 = 0$. Thus eliminating variability in repair time will do nothing to reduce this term. However, the last term is due explicitly to the variability of the repair times and would vanish if this variability were eliminated. Notice that both of the second two terms are increasing in mr for a fixed availability. Hence, all other things being equal, long repair times induce more variability than short ones. 3Actually, the line with the Hare X19 will require more than 12.4 hours ofWIP, and the line with Tortoise 2000 will require more than 4.133 hours of WIP, because these are only *average* downtimes. But the point remains the same: The line with the Hare X19 requires substantially more WIP to achieve the same throughput as the line with the Tortoise 2000. This is frequently a good assumption in practice, particularly for complex equipment since such machines tend to be combinations of old and new components. Thus, the memoryless property of the exponential tends to hold for the time between *any* outage, which could be caused by failure of an old component or a new one.

Substituting numbers into these equations yields

$$c_e^2 = 0.05 + (1 + 1)0.75(1 - 0.75)\frac{248}{15} = 6.25$$

or $ce = 2.5$, which shows that the Hare X19 is well up in the HV range. However, the Tortoise 2000 has

$$c_e^2 = 0.05 + (1 + 1)0.75(1 - 0.75)\frac{38}{15} = 1.0$$

And so $C_e = 1$, which shows that it is in the MV range. Hence a line with the Hare X19 will exhibit much more variability than one with the Tortoise 2000.

This analysis leads to the conclusion that a machine with frequent but short outages is preferable to one with infrequent but long outages, provided that the availabilities are the same. This is a potentially valuable insight, since in practice we maybe able to convert long, infrequent failures to shorter, more frequent ones (e.g., through preventive maintenance procedures).

2.2.2.3 VARIABILITY FROM NONPREEMPTIVE OUTAGES

Nonpreemptive outages represent downtimes that will inevitably occur but for which we have some control as to exactly when. In contrast, a preemptive outage, which might be caused by catastrophic failure of a machine or when the machine becomes radically out of adjustment, forces a stoppage whether or not the current job is completed. An example of a no preemptive outage occurs when a tool starts to become dull and needs to be replaced or when the mask used to expose a circuit board begins to wear out. In situations like these we can wait until the current piece or job is finished before stopping production.

Process changeovers (setups) can be regarded as no preemptive outages when they occur due to changes in the production process (such as changing a mask) as opposed to changes in the product. Changeovers due to changes in product (e.g., setting up for a new part) are more under our control. Other no preemptive outages include preventive maintenance, breaks, operator meetings, and shift changes. These typically occur between jobs, rather than during them. Nonpreemptive outages require somewhat different treatment than preemptive outages. Since the most common source of nonpreemptive outages is machine setups, we will frame our discussion in these terms. However, the approach is applicable to any form of nonpreemptive outage. As with preemptive outages, ordinary capacity calculations do not fully analyze the impacts of nonpreemptive setups. Average capacity analysis only tells that short setups are better than long ones. It cannot evaluate the differences between a slow machine with short setups and a fast one with long setups that have the same effective capacity.

For example, consider the decision of whether to replace a relatively fast machine requiring periodic setups with a slower flexible machine that does not require setups.

Machine 1, the fast one, can do an average of one part per hour, but requires a two-hour setup every four parts on average. Machine 2, the flexible one, requires no setups but is slower, requiring an average of 1.5 hours per part. The effective capacity re for machine 1 is:

$$re = 4 \text{ parts} / 6 \text{ hours} = 2/3 \text{ part/hour}$$

Since this is a single-machine workstation, the effective process time is simply the reciprocal of the effective capacity, so $te = 1.5 \text{ hours}$. Thus, machines 1 and 2 have the same effective capacity.

Traditional capacity analysis would consider the two machines equivalent and hence would offer no support for replacing machine 1 with machine 2. However, the previous factory physics treatment of machine breakdowns showed that considering variability can be important in evaluating machines with breakdowns. All other things being equal, machine 2 will have less variable effective process times than machine 1 (i.e., because every fourth job at machine 1 will have a long setup time included in its effective process time). Thus, replacing machine 1 with machine 2 will serve to reduce the process time CV and therefore will make the line more efficient. This variability reduction effect provides further support for the JIT preference for short setups and is a clear motivation for flexible manufacturing technology. However, the evaluation of the benefits of flexibility can be subtle. The above condition of "all other things being equal" requires that the natural variability of both machines 1 and 2 be the same (i.e., so that the setups for machine 1 will unambiguously increase the CV of effective process times). But what if the flexible machine also has more natural variability? In this case, we must compute and compare the CV of effective process times for both machines. To compute the CV of effective process times for a machine with setups, we first require data on the natural process times, namely, the mean t_0 and variance σ_0^2 . (Equivalently, we could use the mean t_0 and the CV c_0 , since $\sigma_0^2 = c_0^2 t_0^2$.) Next we must describe the setups, which we do by assuming that the machine processes an average of N_s parts (or jobs) between setups, where the setup times have a mean duration of t_s and a CV of c_s . We also assume that the probability of doing a setup after any part is equal. That is, if an average of 10 parts are processed between setups, there will be a 1-in-10 chance that a setup will be performed after the current part, regardless of how many have been done since the last setup. Under these assumptions, the equations for the mean, variance, and SCV of effective process time are, respectively,

$$t_e = t_0 + t_s/N_s$$

$$\sigma_e^2 = \sigma_0^2 + \sigma_s^2/N_s + (N_s - 1/N_s^2)t_s^2$$

$$c_e^2 = \sigma_e^2/t_e^2$$

To illustrate the usefulness of these equations, consider another example that compares two machines. Machine 1 is a flexible machine, with no setups, but has somewhat variable process times. Specifically, the natural process time has a mean of $t_0 = 1.2$ hours and a CV of

$c_0 = 0.5$. Machine 2 performs an average of $N_s = 10$ parts between setups and has natural process times with a mean of $t_0 = 1.0$ hours and a CV of $c_0 = 0.25$. The average setup time is $t_s = 2$ hours with a CV of $c_s = 0.25$. Which machine is better? First, consider the effective capacity. Machine 1 has:

$$r_e = 1/t_e = 1/1.2 = 0.833$$

While machine 2 has

$$r_e = 1/t_e = 1/(1 + 2/10) = 0.833$$

So the two machines are equivalent in this regard. Therefore, the question of which is better becomes, which machine has less variability?

$Ce^2 = 0.31$ for machine 2, as compared to $Ce^2 = Co^2 = 0.25$ for machine 1. Thus, machine 1, the more variable machine without setups, has less overall variability than machine 2, the less variable machine with setups. Of course, this conclusion was a consequence of the specific numbers in the example. Flexible machines do not always have less variability.

2.2.2.4 VARIABILITY FROM RECYCLE:

Another major source of variability in manufacturing systems is quality problems. The simplest quality case to analyze is that of rework on a single workstation. This happens when a workstation performs a task and then checks to see whether the task was done correctly. If it was not, the task is repeated. If we think of the additional processing time as an outage, it is easy to see that this situation is equivalent to the nonpreemptive outage case. Hence, rework has analogous effects to those of setups, namely, that it both robs capacity and contributes greatly to the variability of the effective process times. As with breakdowns and setups, the traditional reason for reducing rework is to prevent a loss of effective capacity (i.e., reduce waste). Of course, as with traditional analyses of breakdowns and setups, this perspective would regard two machines with the same effective capacity but different rework fractions as equivalent. However, an analysis like that done above for setups shows that the CV of effective process times increases as the fraction of rework increases. Hence, more rework implies more variability. More variability causes more congestion, WIP, and cycle time. Hence, these variability impacts, coupled with the loss of capacity, make rework a disruptive problem indeed.

2.3 FLOW VARIABILITY:

We mean by Flow Variability, the variability at one station that can affect the behavior of other stations in a line by means of another type of variability. Flows refer to the transfer of jobs or parts from one station to another and to analyze the effect of variability on the line, we must characterize the variability in flows

2.3.1 CHARACTERIZING VARIABILITY IN FLOWS

The main of studying flows is the variability of arrivals which is the description of the arrival of jobs to one workstation and determination on how this affects the variability of departures from that workstation (and hence arrivals to other workstations), therefore the flow variability can be characterized for the entire line.

- Arrivals rate with high and Low CVs:

In order for the workstation to be able to keep up with arrivals and to keep it from becoming overloading all realistic cases has proved that it is essential that capacity must exceed the arrival rate, that is,

$$r_e > r_a$$

The coefficient of variation of the inter-arrival times c_a is defined by,

$$c_a = \frac{\sigma_a}{t_a}$$

We refer to this as the arrival **CV**, to distinguish it from the process time CV. Intuitively, a low arrival CV indicates regular, or evenly spaced, arrivals, while a high arrival CV indicates uneven, or “bursty” arrivals and this difference is illustrated in Figure 5.

The arrival CV c_a , along with the mean inter-arrival time t_a , summarizes the essential aspects of the arrival process to a workstation.

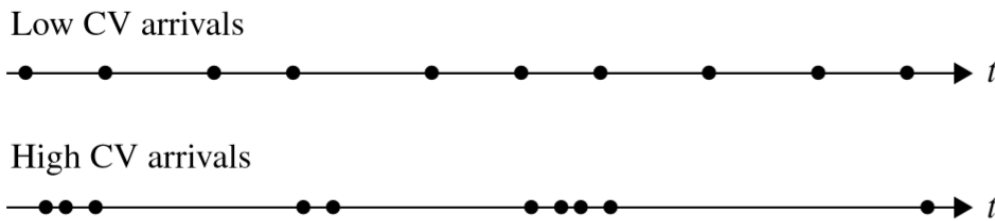


Figure 5: High and Low CV arrivals

The arrivals can be classified according to the arrival CV c_a as follows:

$$\text{Low variability (LV)} \quad c_a \leq 0.75$$

$$\text{Moderate variability (MV)} \quad 0.75 < c_a \leq 1.33$$

$$\text{High variability (HV)} \quad c_a > 1.33$$

The inter-arrival times will tend to be memoryless (i.e., exponential), and therefore c_a will be close to one. Even when the arrivals from any given source are quite regular (i.e., LV), the superposition of all the arrivals tends to look MV.

- **Departures rates:**

In a serial production line, where all the output from workstation i becomes input to workstation $i + 1$, the departure rate from i must equal the arrival rate to $i + 1$, so

$$t_a(i + 1) = t_d(i)$$

Indeed, in a serial production line without yield loss or rework, the arrival rate to every workstation is equal to the throughput TH. Also, in a serial line where departures from i become arrivals to $i + 1$, the departure CV of workstation i is the same as the arrival CV of workstation $i + 1$

$$c_a(i + 1) = c_d(i)$$

These equations are illustrated in Figure 6 below/

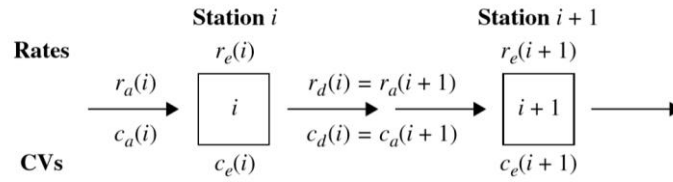


Figure 6: Cascade Equations

Variability in departures from a station are the result of both variability in arrivals to the station and variability in the process times. The relative contribution of these two factors depends on the utilization of the workstation which is defined by

$$u = \frac{r_a \times t_e}{m}$$

With $0 < u < 1$

An obvious upper limit on $u = 1$ (that is, 100%), which implies that the effective process times must satisfy $t_e < \frac{m}{r_a}$.

If $u \approx 1$, then the station is almost always busy. Therefore, under these conditions, the interdeparture times from the station will be essentially identical to the process times. Thus, we would expect the departure CV to be the same as the process time CV (that is, $c_d = c_e$).

At the other extreme, when $u \approx 0$ the station is very lightly loaded.

A good, simple method for interpolating between these two extremes is to use the square of the utilization as follows:

$$c_d^2 = u^2 c_e^2 + (1 - u^2) c_a^2$$

When there is $m > 1$:

$$c_d^2 = 1 + (1 - u^2)(c_a^2 - 1) + \frac{u^2}{\sqrt{m}}(c_e^2 - 1)$$

- The information on the variability in demand is often available:

We can approximate the following equation with precedent one:

$$c_a^2 = \frac{\sigma_a^2}{t_a^2} \rightarrow \frac{\sigma_n^2}{\mu_n}$$

This relation well if $\mu_n \geq 10$.

Note that the random variable is the number of demands in the period and not the total demand.

- One important cause of flow variability is batch arrivals. These happen whenever jobs are batched together for delivery to a station.

In general, if we have a batch size k , this analysis will yield $c_a^2 = k-1$

The reason is that batching confounds two different effects. The first effect is due to the batching itself. This is not really a randomness issue, but rather one of bad control. The second is the variability in the batch arrivals themselves.

2.4 VARIABILITY INTERACTIONS—QUEUEING:

A Queueing system which is the science of waiting combines the components that have been considered so far: an arrival process, a service (i.e., production) process, and a queue.

The queueing discipline can be first-come, first-served (**FCFS**); last-come, first-served (**LCFS**); earliest due date (**EDD**); shortest process time (**SPT**); or any of a host of priority schemes. The queue space can be unlimited or finite. The variety of queueing systems is almost endless, the job of queueing theory is to characterize performance measures in terms of descriptive parameters.

- **The M/M/1 Queue:**

The M/M/1 queue is tractable and offers valuable insight into more complex and realistic systems, it is also the simplest queue to analyze with the key of Memoryless property of the exponential distribution.

- The expression for expected WIP:

$$WIP(M/M/1) = \frac{u}{1-u}$$

- Average cycle time:

$$CT(M/M/1) = \frac{WIP(M/M/1)}{r_a} = \frac{t_e}{1-u}$$

- Average time in queue:

$$CTq(M/M/1) = CT(M/M/1) - t_e = \frac{u}{1-u} t_e$$

- The Queue yields:

$$WIPq(M/M/1) = r_a \times CTq(M/M/1) = \frac{u^2}{1-u}$$

- The G/G/1 Queue:

In the real world, unfortunately the M/M/1 queue cannot be satisfy the manufacturing systems because of the exponential property which cannot be applied on the inter-arrivals and process times, and that lead to another queuing model G/G/1, the memoryless property will be replaced by means of a “two-moment” approximation, which makes use of only the mean and standard deviation (or CV) of the inter-arrival and process time distributions. Although cases can be constructed for which this approximation works poorly, it is reasonably accurate in typical manufacturing systems. Because it works well, this approximation is the basis of several commercially available manufacturing queueing analysis packages.

- Average time in queue:

$$CTq(G/G/1) = \frac{c_a^2 + c_e^2}{2} \frac{u}{1-u} t_e = V * U * T$$

V: Dimensionless variability term.

U: Utilization term.

T: Time term.

And to have an effective result the utilization (u) need to have a specific percentage

$$0.1 < u < 0.95$$

- Parallel Machines M/M/m :

The VUT equation gives us a tool for analysing workstations consisting of single machines. However, in real-world systems, workstations often consist of multiple machines in parallel. The reason, of course, is that often more than a single machine is required to achieve the desired workstation capacity. To analyse and understand the behaviour of parallel machine stations, we need a more general model which is defined by the M/M/m queueing system. Although the steady-state probabilities for this queuing system can be computed perfectly ,they are messy and provide little additional intuition.

Sakasegawa (1977) proposed an approximative form for the waiting time in queue that offers intuition and is quite accurate.

$$CTq(M/M/m) = \frac{u^{\sqrt{2(m+1)}-1}}{m(1-u)} t_e$$

- Parallel machines and General times G/G/m:

A parallel machine station with general (nonexponential) process and interarrival times is represented by a G/G/m queue, and here is a closed-form expression for the waiting time;

$$CTq(G/G/m) = \left(\frac{c_a^2 + c_e^2}{2}\right) \left(\frac{u^{\sqrt{2(m+1)}-1}}{m(1-u)}\right) t_e$$

2.5 EFFECTS OF BLOCKING:

The science of Factory Physics deal with an important topic which is the behavior of systems with finite queueing space. In a real manufacturing systems queues never become infinite. They are bounded by limitations of space, time, or operating policy.

- The M/M/1/b Queue:

The M/M/1/b queue is where process and interarrival times are exponential, as they are in the M/M/1 queue, but where there is only enough space for b units in the system.

For the case where $u \neq 1$, the average WIP and throughput are

$$WIP(M/M/1/b) = \frac{u}{1-u} \frac{-(b+1)u^{b+1}}{1-u^{b+1}}$$

$$TH(M/M/1/b) = \frac{1-u^b}{1-u^{b+1}} r_a$$

For the case where $u = 1$, WIP and throughput simplify to

$$WIP(M/M/1/b) = \frac{b}{2}$$

$$TH(M/M/1/b) = \frac{b}{b+1} r_a$$

The M/M/1/b model can be interpreted as a system of two machines and the buffer between the two machines is finite and is equal to B, If both machines have exponential process times, the model for the behaviour of the second machine and the buffer is given by the M/M/1/b queue, where;

$$b = B + 2$$

We also noticed that;

$$WIP(M/M/1) > WIP(M/M/1/b)$$

And using this equation, we see that if $u \neq 1$;

$$TH = \frac{1 - u^b}{1 - u^{b+1}} ur_a < ur_e$$

And if $u = 1$;

$$TH = \frac{b}{b+1} r_a < r_e$$

Those last expressions shows that, the smaller the buffer size b , the greater the reduction in throughput.

- **The General blocking models:**

To analyse variability effects, we need to extend the M/M/1/b model to more general process and interarrival time distributions and for that we are going to use three cases.

1- Arrival Rate Less than Production Rate ($u < 1$):

First we compute WIP_{nb} by using Kingman's equation and Little's law.

$$WIP_{nb} = \left(\frac{c_a^2 + c_e^2}{2} \right) \left(\frac{u^2}{1 - u} \right) + u$$

$$TH \approx \frac{1 - u\rho^{b-1}}{1 - u^2\rho^{b-1}} r_a$$

With

$$\rho = \frac{WIP_{nb} - u}{WIP_{nb}}$$

By combining Kingman's equation (to compute ρ) with the M/M/1/b model, we incorporate the effects of both variability and blocking.

These equation make a result of this ;

$$WIP < \min\{WIP_{nb}, b\}$$

2- Arrival Rate Greater than Production Rate ($u > 1$):

To approximate the WIP in the case in which the arrival rate is greater than the production rate, we just have to put the machines in reverse order.

$$WIP_{nb} = \left(\frac{c_a^2 + c_e^2}{2} \right) \left(\frac{\frac{1}{u^2}}{1 - \frac{1}{u}} \right) + 1/u$$

3- Arrival Rate Equal to Production Rate ($u = 1$):

Here is a good approximation of TH;

$$TH \approx \frac{c_a^2 + c_e^2 + 2(b-1)}{2(c_a^2 + c_e^2 + b-1)} r_e$$

With this approximation of TH, we can use Little's law for bounds on WIP and CT.

2.6 VARIABILITY POOLING:

Variability pooling is a way to deal with congestion effects by combining multiple sources of variability, and it plays an important role in a number of manufacturing situations. Here we discuss how it affects batch processing, safety stock aggregation, and queue sharing.

- **Batch Processing:**

The CV of the time to process the batch is

$$c_0(\text{batch}) = c_0 / \sqrt{n}$$

With;

$$c_0 = \frac{\sigma_0}{t_0}$$

Thus, the CV of the time to process decreases by one over the square root of the batch size. We can conclude that process times of batches are less variable than process times of individual part.

2.7 CONCLUSION:

This chapter has traversed the complex and subtle topic of variability all the way from the fundamental nature of randomness to the propagation and effects of variability in a production line. Points that are fundamental from a Factory Physics perspective include the following:

- Variability is a fact of life.
- There are many sources of variability in manufacturing systems.
- The coefficient of variation is a key measure of item variability.
- Variability propagates.
- Waiting time is frequently the largest component of cycle time.
- Limiting buffers reduces cycle time at the cost of decreasing throughput.
- Variability pooling reduces the effects of variability.

3. THE CORRUPTING INFLUENCE OF VARIABILITY:

The Fundamental objective of a manufacturing supply chain is “Make money now and in the future in ways that are consistent with our core values”.

The supply chain is composed of two essential elements which are demand and transformation and the most common cause of the lack of alignment of these two element is variability.

In this chapter, we will use the tools for characterizing and evaluating variability in process times and flows to expand on our formal model and describe fundamental behavior of manufacturing systems involving variability.

3.1 GOOD AND BAD VARIABILITY:

Variability is typically equated with “**muda**”, the Japanese word for waste, which suggests that it should always be eliminated. Regardless of whether variability is good or bad in business strategy terms, it causes operating problems and therefore must be managed. The specific strategy for dealing with variability will depend on the structure of the system and the firm’s strategic goals.

3.2 VARIABILITY LAWS:

- **Influence of variability:**

Variability increases whenever there is a decrease in uniformity. There is many source that can affect variability which can be either randomness or control, here’s two fundamental law of Factory Physics that shows how variability degrades performance.

- **Law (*Variability*):** Increasing variability always degrades the performance of a production system.
- **Law (*Buffering Variability*):** Variability in a production system will be buffered by some combination of inventory, capacity and Time.

- **Variability Buffering:**

The buffering law could also be called the “law of pay me now or pay me later” because if you do not pay to reduce variability, you will pay in one or more of the following ways; lost throughput, wasted capacity, inflated cycle times, Larger inventory levels, long lead times and/or poor customer service.

If you cannot pay to reduce variability, you will pay in terms of high WIP, under-utilized capacity, or reduced customer service (i.e., lost sales, long lead times, and/or late deliveries).

- **Flexibility:**

A flexible buffer is one that can be used in more than one way, we can state the following corollary to the buffering law.

Corollary (Buffer Flexibility): Flexibility reduces the amount of variability buffering required in a production system.

3.3 FLOW LAWS:

- **Product Flow:** In a stable system, over the long run, the rate out of a system will equal the rate in, less any yield loss, plus any parts production within the system.
- **Capacity:** In steady state, all plants will release work at an average rate that is strictly less than the average capacity.
- **Utilisation:** If a station increases utilization without making any other changes, average WIP and cycle time will increase in a highly nonlinear fashion.
- **Variability and Flow:** In a line where releases are independent of completions, variability early in a routing increases cycle time more than equivalent variability later in the routing.

3.4 BATCHING LAWS:

Batching is an important determinant of performance it can affect flow variability and waiting inventory, there are two different kinds of batches:

Process Batch: many transfer batches processed together.

- Related to length of *setup*.
- The longer the setup the larger the lot size required for the same capacity.
- **Law (Process Batching):** In stations with batch operations or significant changeover times:
 - The minimum process batch size that yields a stable system may be greater than one.
 - As process batch size becomes large, cycle time grows proportionally with batch size.
 - Cycle time at the station will
- There are two types of process batches which are:
 - **Simultaneous (parallel) batch:** represents the number of parts produced simultaneously in a “true batch” workstation, such as a furnace or heat treatment operation.

Time to form batch: $w = \frac{k-1}{2} \frac{1}{r_a}$

Time to process batch: $te = t$

Arrival of batches: ra/k

Utilization: $u = (ra/k)(t)$

For stability: $u < 1$ requires $k > r_a * t$

Average wait-for-batch time:

$$WT = \frac{k-1}{2} \frac{1}{r_a}$$

Average queue plus process time at station:

$$CT = \frac{c_a^2/k + c_0^2}{2} \frac{u}{1-u} t + t$$

- **Sequential (serial) batches:** represent the number of transfer batches that are processed before the workstation is changed over to another part or family

Time to process batch: $t_e = kt + s$

Arrival of batches: ra/k

Utilization: $u = (ra/k)(kt + s) = ra(t + s/k)$

For stability: $u < 1$ requires

$$k > \frac{sr_a}{1-tr_a}$$

The average time in queue CT_q is given by the VUT equation :

$$CT_q = \frac{c_a^2 + c_e^2}{2} \frac{u}{1-u} t_e$$

Average cycle time depends on move batch size:

Move batch = process batch $CT_{non-split} = CT_q + t_e$

Move batch =1 $CT_{split} = CT_q + s + \frac{k+1}{2}$

Move (transfer) Batch: many parts moved at once

- The smaller the move batch, the shorter the cycle time.
- The smaller the move batch, the more material handling.
- **Law (Move Batching):** Cycle times over a segment of a routing are roughly proportional to the transfer batch sizes used over that segment, provided there is no waiting for the conveyance device.

3.5 THE CYCLE TIME:

- **Definition (Station Cycle Time):** The average cycle time at a station is made up of the following components:

Cycle Time = move time + queue time + setup time +
process time + wait-to-batch time + wait-in-batch time + wait-to-match time

- **Definition (Line Cycle Time):** The average cycle time in a line is equal to the sum of the cycle times at the individual stations less any time that overlaps two or more stations.
- **Law (Assembly Operations):** The performance of an assembly station is degraded by increasing any of the following: 1. Number of components being assembled. 2. Variability of component arrivals. 3. Lack of coordination between component arrivals.
- **Law (Lead Time):** The manufacturing lead time for a routing that yields a given service level is an increasing function of both the mean and standard deviation of the cycle time of the routing.

3.6 PERFORMANCES AND VARIABILITY:

- **Lean manufacturing:**

Definition: Production of goods or services is lean if it is accomplished with minimal buffering costs.

Insights:

- Lean is about more than waste elimination.
- Variability reduction is key to lean.
- Choosing the right mix of inventory, capacity and time buffering is also important.

3.7 DIAGNOSTIC AND IMPROVEMENTS:

- **Increasing throughputs :**

Throughput of a line is given by

$$TH = \text{bottleneck utilization} \times \text{bottleneck rate}$$

A basic checklist of policies for increasing throughput is as follows.

- Increase bottleneck rate by increasing the effective rate of the bottleneck. This can be done through equipment additions, staff additions or training, covering stations through breaks or lunches, use of flexible labor, quality improvements, product design changes to reduce time at the bottleneck, and so forth.
- Increase bottleneck utilization by reducing blocking and starving of the bottleneck. There are two basic ways to do this:
 - Buffer bottleneck with WIP
 - Buffer bottleneck with capacity.
- **Reducing cycle time:**

In most production systems, we have seen actual process and move times are a small fraction of total cycle time. The following is a brief checklist of generic policies for reducing each of these terms.

- **Queue time** is caused by utilization and variability. Hence, the two categories of improvement policies are as follows:
 - Reduce utilization by increasing the effective rate at the bottleneck.
 - Reduce variability in either process times or arrivals at any station.
 - **Process batch** time is driven by process batch size. The two basic means for reducing (sequential or simultaneous) process batch size are as follows:
 - Batching optimization to better balance batch time with queue time due to high utilization.
 - Setup reduction to allow smaller batch sizes without increasing utilization.
 - **Wait-to-match time** is caused by lack of synchronization of component arrivals to an assembly station. The main alternatives for improving synchronization are as follows:
 - Fabrication variability reduction to reduce the volatility of arrivals to the assembly.
 - Release synchronization by using the shop floor control and/or scheduling systems to coordinate releases in the line to completions at assembly..
 - **Station overlap time.** Unlike the other “times,” we would like to increase station overlap time because it is subtracted from the total cycle time. It can be increased by the use of lot splitting where feasible.
- **Improving Customer Service**

Elements of Customer Service:

- lead time

- fill rate (% of orders delivered on-time)
- quality

Law (Lead Time): The manufacturing lead time for a routing that yields a given service level is an increasing function of both the mean and standard deviation of the cycle time of the routing.

$$\text{Lead time} = \text{average cycle time} + z s \times \text{standard deviation of cycle time}$$

Lead time = Reduce CT Visible to Customer

Average cycle time = Reduce Average CT

standard deviation of cycle time = Reduce CT Variability

3.8 CONCLUSION:

The primary focus of this chapter is the effect of variability on the performance of production lines. The main points can be summarized as follows:

- ✓ Variability degrades performance.
- ✓ Variability buffering is a fact of manufacturing life
- ✓ Flexible buffers are more effective than fixed buffers. Material is conserved.
- ✓ Releases are always less than capacity in the long run.
- ✓ Variability early in a line is more disruptive than variability late in a line. Cycle time increases nonlinearly in utilization.
- ✓ Process batch sizes affect capacity.
- ✓ Cycle times increase proportionally with transfer batch size.
- ✓ Matching can be an important source of delay in assembly systems.
- ✓ Diagnosis is an important role for Factory Physics.

4. PUSH AND PULL PRODUCTION SYSTEMS:

Push-pull is also known as lean inventory strategy. It demands a more accurate forecast of sales and adjusts inventory levels based upon actual sale of goods.

4.1 PUSH VS PULL SYSTEMS:

Push System	Pull System
<ul style="list-style-type: none"> ▪ The pull system plan the production of labor according to the demand (real / expected) on the basis of information outside the system ▪ Establishes a pre-determined limit on WIP. 	<ul style="list-style-type: none"> ▪ The push system allows exits based on information from inside the system ▪ It does not set a limit on WIP

Table 6: Push System VS Pull System

The effectiveness of the attraction depends mainly on the type of endogenous information used to attract work into the system.

4.2 THE MAGIC OF PULL:

The success of several high-profile Japanese companies, including Toyota, in the 1980s was the result of using pull systems, which is shown in two levels:

- *Micro level:* This system had an efficient production control system that facilitates low cost manufacturing by promoting high throughput, low inventory and low number of rework
- *Macro level:* Communication of quality products to the market on time at a competitive cost and in a responsive mix

The key to all these desirable features that made the Toyota Production System such an attractive basis for a business strategy is that, there is a limit on the maximum amount of inventory in the system.

- **Reducing manufacturing cost:**

A WIP cap reduces manufacturing costs by reducing costs due to shipping and engineering modifications which will improve flexibility and promote better timing of released work.

- **Reducing variability:**

High level customer service allows low cycle time variability.

Low cycle time -> precise due date -> shorter customer deadline.

The pull system is less variable than the push system since the pull system avoids the explosion of WIP which also prevent Cycle time explosions and to have a high throughput rate and low WIP level, it is necessary to reduce the sources of disturbing variability.

- **Improving quality:**

The quality assurance benefits of pulling at each station can be attained via inspection transactions independently of the mechanism used for achieving the needed limit on WIP. A short queue allows for quick and safe inspection.

- **Maintaining flexibility:**

It Prevent the release of parts when the plant is too crowded

- **Facilitating work ahead**

4.3 CONWIP:

The protocol under which a new job is introduced to the line each time a job exits is called CONWIP (constant work in process).

- Basic Mechanics

The production line has only one route and the jobs are the same, so that WIP can be reasonably measured in unit.

- CONWIP: closed queued network
- Push / MRP: open queue network
- Kanban / Pull: network in queue closed but with blocking
- **Mean Value Analysis:**

For the case in which all stations consist of single machines, we can do this by using a technique known as mean-value analysis (MVA)

The MVA algorithm computes the cycle time, throughput, and station-by-station WIP levels as a function of the number of jobs in the CONWIP line in iterative fashion by using the following:

$$CT_j(w) = \frac{t_e^2(j)}{2} [c_e^2(j) - 1] TH(w - 1) + [WIP_j(w - 1) + 1] t_e(j)$$

$$CT(w) = \sum_{j=1}^n CT_j(w)$$

Note: it only for single / parallel machine.

It develops cycle time flow curves for PWC (particular worst case) and it's an iterative procedure which develops the measure of the line with the WIP level w as a function of $(w-1)$

4.4 CONCLUSION:

In this chapter, we have made the following basic points:

- ✓ Push systems schedule the release of work on the basis of demand information, while pull systems authorize the release of work on the basis of inventory status within the system.
- ✓ The “magic” of pull systems is that they establish a WIP cap, which prevents producing unnecessary WIP that does not significantly improve throughput.
- ✓ The simplest mechanism for establishing a WIP cap is CONWIP (constant work in process), in which the WIP level in a line is held constant by synchronizing releases to departures.
- ✓ CONWIP exhibits the following advantages over a pure push system:
 - The WIP level is directly observable, while the release rate in a push system must be set with respect to (unobservable) capacity.
 - It requires less WIP on average to attain the same throughput.
 - It is more robust to errors in control parameters.
 - It facilitates working ahead of a schedule when favourable circumstances permit it

Chapter

2

Application Modelling

In this chapter we presented our UML modelling for the application as well as a part of our source code to explain its functioning.

CHAPTER 2 : APPLICATION MODELLING

1. UML :

UML is a standard language for specifying, visualizing, constructing, and documenting the artefacts of software systems. It's not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG (Object Management Group) standard.

"A picture is worth a thousand words", there are a number of goals for developing UML but the most important is to define some general purpose modelling language, which all modellers can use and it also needs to be made simple to understand and use.

1.1 USER CASE DIAGRAM:

The use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction, and these internal and external agents are known as actors. Use case diagrams consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.



Figure 7: User Case

From the preliminary study we were able to identify the use cases in the figure below which manages:

- **Define the workshop** : this user case allows the admin to create and define the parameters of the workshop of the system that will be studied
- **Inject Data**: this user case allows the user to inject the data of the system that are needed for the study and it allows only to the admin to modify them.
- **Visualize Result**: this user case allow to both admin and user to visualize the result calculated.
- **Analyse Result** :this user case permit to the admin to analyse the result that be founded

The diagram is composed from two actors who are related:

- ✓ **Admin**: the actor that have access and the authority to modify the system.
- ✓ **User**: The actor that can only access to the system and view the result but without any modification allowed .

1.2 SEQUENCE DIAGRAM:

The sequence diagram captures the time sequence of the message flow from one object to another, the purpose of sequence diagram is to visualize the interactive behaviour of the system. Visualizing the interaction is a difficult task. Hence, the solution is to use different types of models to capture the different aspects of the interaction.

The purpose of sequence diagram is:

- To capture the dynamic behaviour of a system.
- To describe the message flow in the system.
- To describe the structural organization of the objects.

- To describe the interaction among objects.

From the preliminary study we were able to identify 4 alternative for the sequence diagram:

→ 1st Alternative:

It describes the message for the admin to define the type of the workshop as an example Flowshop or Jobshop so the appropriate data will be requested if no mistake happens.

→ 2nd Alternative:

It describes the message for the admin to determinate the stations data including all the information of it so the appropriate routing will be requested if no mistake happens.

→ 3rd Alternative:

It describes the message for the admin to define the routing for the line of the workshop all the information requested for all the types of products will be have to be determinates if no mistake happens.

→ 4th Alternative:

It describes the message to open an excel file with all of the functions calculated and visualize the result.

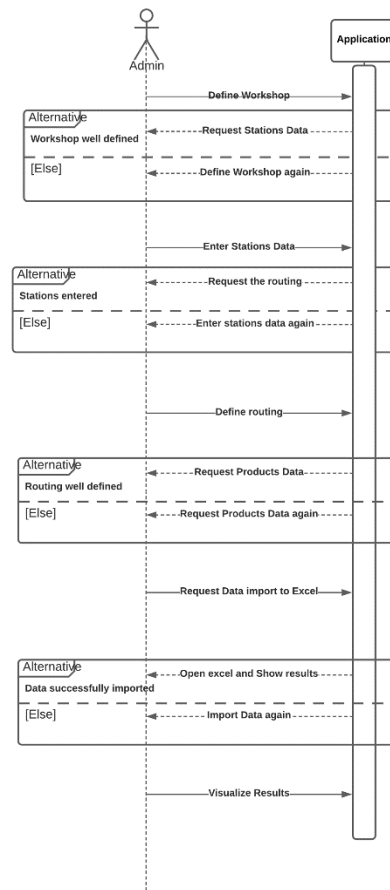


Figure 8: Sequence Diagram

1.3 CLASS DIAGRAM:

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application, it shows a collection of classes, interfaces, associations, collaborations, and constraints.

The purpose of the class diagram can be summarized as:

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

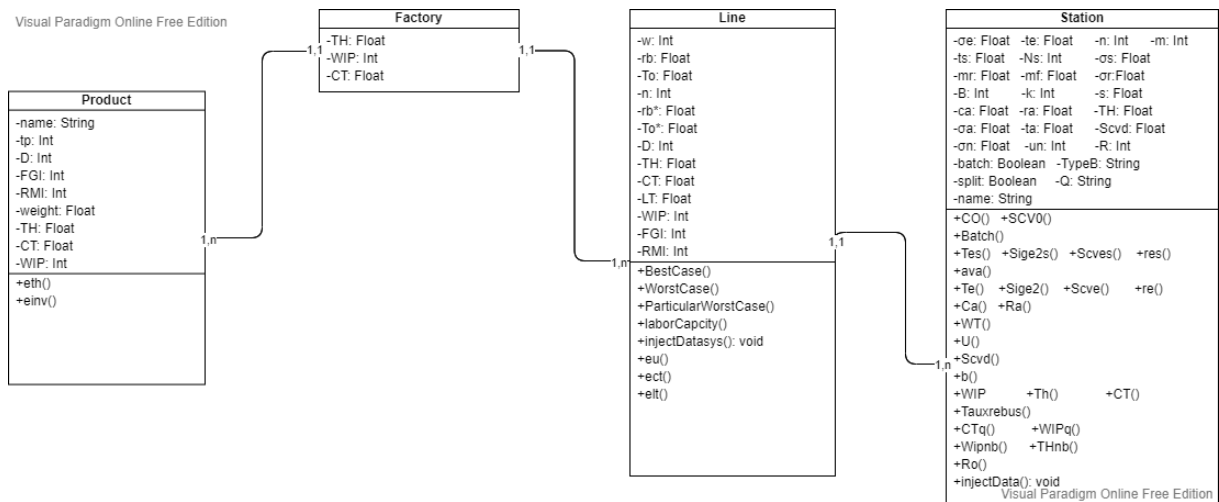


Figure 9: Class Diagram

From the preliminary study we were able to identify 4 class for the Class Diagram:

✓ **Class Station:**

This class is used to represent a workstation including multiple machine with different parameters as an example; process time or batch size, there is 4 types of attributes: integer, float, string and Boolean.

All the function return a float parameter except one «Inject Data» it's a void function

✓ **Class Line:**

This class regroups all the stations with different routings represented by the lines or the products itself, there is 2 types of attributes as integer and float.

All the function return a float parameter except one «Inject Data» it's a void function

✓ **Class Factory:**

This class regroups all the lines and represents the performance indicators for the whole factory

✓ **Class Product:**

This class represents the performance indicators dedicated to the products, there is 3 types of attributes as integer, float and string.

2. PROGRAMMATION OF THE APPLICATION:

2.1 PYTHON:

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.^[1]

In the late 1980s, Guido van Rossum began working on Python after working on its ancestor the ABC programming language, and the first version Python 0.9.0 was launched in 1991. In 2000 new features were introduced and Python 2.0 was released. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and most Python 2 code does not run unmodified on Python 3. The end of Python 2 was in 2020 after being discontinued with version 2.7.18.

2.1.1 WHY DID WE CHOOSE PYTHON?

The first reason that made us choose to use Python in developing our application because it is ranked as one of the most popular programming languages with a very large and active community that contributes to its evolution through helping to develop modules and working on open-source projects or simply by responding to other users' questions on different web forums and because our knowledge of Python's syntax was basic when we started, we ended up searching and finding a lot of guidance out of that.

The second reason is that The 19th annual KDnuggets Software Poll (over 2,300 voters) results showed that Python is the go-to programming language for data scientists and analysts because of its decent library availability, and since we'll be working with data – on a smaller scale and in a less complex context – , easy integration with Excel and the possibility of using already made modules and functions to manipulate data was essential to us in order to realize our objectives and leave a space for possible evolution.

2.2 VISUAL STUDIO CODE:

Visual Studio Code is a source-code editor based on the Electron framework, built by Microsoft for Windows, Linux and macOS. It was first announced on April 29 at the 2015 Build conference. A Preview build was released shortly thereafter.

It includes support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users have control over the theme, keyboard shortcuts, and can install extensions that add additional functionality. Users are also able to use extensions for additions to the editor and language support.

VS Code can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, C++ and Python.

2.2.1 WHY DID WE CHOOSE VISUAL STUDIO CODE?

Visual Studio code is more than just a code editor, it also includes very useful developer tooling, combining simplicity with utility makes it perfect for day-to-day use and saves the user a lot of time leaving him to focus on executing ideas.

Also in the Stack Overflow 2019 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 50.7% of 87,317 respondents reporting that they use it which-just like Python- makes it attractive to use thanks to the large users community and the non-stopping contribution on different websites and forums.

2.3 CODE:

2.3.1 THE GENERAL FUNCTIONING:

The main goal was to create a desktop application that's capable of calculating indicators of performance for lines, workshops and stations, using only Factory Physics® equations.

This application has to consider a large number of parameters of a station as well as variability and generalize to lines then to the whole workshop.

This code creates

- ✓ A list of station class objects - representing all the stations in a workshop - then it makes as many copies of that list as the number of lines/routings lines in the workshop.

The copied lists are then modified in order to:

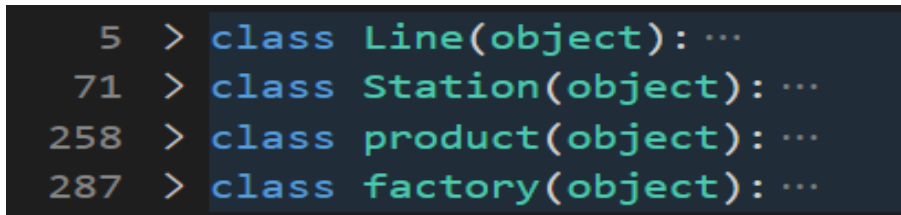
- Define lines/routings
- Deletes stations that don't belong to the line/routing.
- ✓ A list of line class objects is then created - representing all the lines in a workshop -
 - Lines objects use station objects lists to calculate parameters such as (throughput, cycle time, WIP).
- ✓ A workshop object is created.
 - It uses Line objects list to calculate parameters such as (throughput, cycle time, WIP) by using product demands as weights to generalize.
- ✓ A list of Product class objects is then created - representing all the Products in a workshop -
 - It uses Line objects list to calculate parameters such as (throughput, cycle time, WIP).
- ✓ All the results that the functions return, and all the attributes are finally exported to an excel sheet in order to be visually represented and analyzed.

2.3.2 THE DETAILED FUNCTIONING:

Classes:

Following our UML model, we defined 4 classes:

- Factory
- Line
- Station
- Product



```

5 > class Line(object): ...
71 > class Station(object): ...
258 > class product(object): ...
287 > class factory(object): ...

```

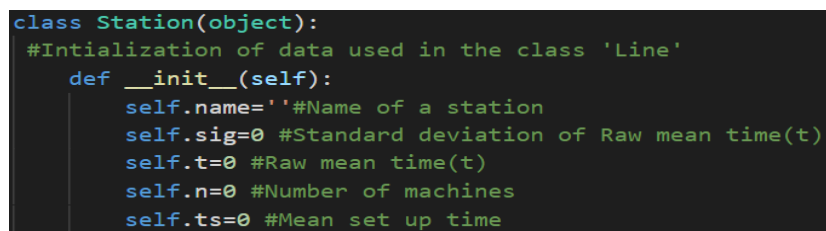
Figure 10: Screenshot of Application Program – Class

Factory, line, station and product classes have their own set of attributes and methods representing: times, queue, throughput, work in process...etc. as we tried to include every possible Data, parameter and equation we could find in the factory physics book and coded it in the form of class methods and attributes.

“Station” class has “injectData ()” function, it is called when a class object is created to give the user the ability to input all the values of the attributes needed while creating the object.

Example:

- A “Station” object has attributes that vary from the raw mean time, type of queue, routing...etc.



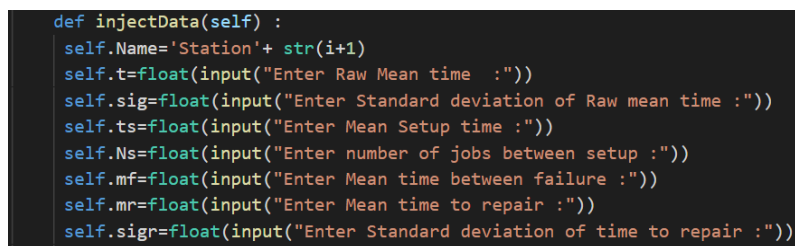
```

class Station(object):
    #Initialization of data used in the class 'Line'
    def __init__(self):
        self.name='#Name of a station
        self.sig=0 #Standard deviation of Raw mean time(t)
        self.t=0 #Raw mean time(t)
        self.n=0 #Number of machines
        self.ts=0 #Mean set up time

```

Figure 11: Screenshot of Application Program - Station Class

- It has “injectData ()” function for inputs:



```

def injectData(self) :
    self.Name='Station'+ str(i+1)
    self.t=float(input("Enter Raw Mean time :"))
    self.sig=float(input("Enter Standard deviation of Raw mean time :"))
    self.ts=float(input("Enter Mean Setup time :"))
    self.Ns=float(input("Enter number of jobs between setup :"))
    self.mf=float(input("Enter Mean time between failure :"))
    self.mr=float(input("Enter Mean time to repair :"))
    self.signr=float(input("Enter Standard deviation of time to repair :"))

```

Figure 12: Screenshot of Application Program - Function of Class Station

- It is called when a class object is created:

```
for i in range(Nbs):
    print("Station "+str(i+1))
    P=Station() #We create an object called P for the class station
    P.injectData() #we call the function to inject all the data needed
    list.append(P) # we add another object into the existing list.
```

Figure 13: Screenshot of Application Program -Creation of Station Object

When a “line” or a “factory” object is created, inputs are needed but we also need to perform operations on the “station” or the “line” objects already created that form the intended line or factory -by definition a line is a set of stations and a factory is a set of lines- and return the value to the instanced “factory” or “line” object attribute.

In order to simplify this process and cascading the stations (from a calculation perspective) following the routing, we used lists.

NB: Lists in Python used to store collections of data, their items are indexed, ordered, changeable, and allow duplicate values.

2.3.2.1 CASCADING THE STATIONS:

[Hopp&Spearman, 2000] “The starting point for studying flows is the arrival of jobs to a single workstation. The departures from this workstation will in turn be arrivals to other workstations. Therefore, once we have described the variability of arrivals to one workstation and determined how this affects the variability of departures from that workstation (and hence arrivals to other workstations), we will have characterized the flow variability for the entire line ”

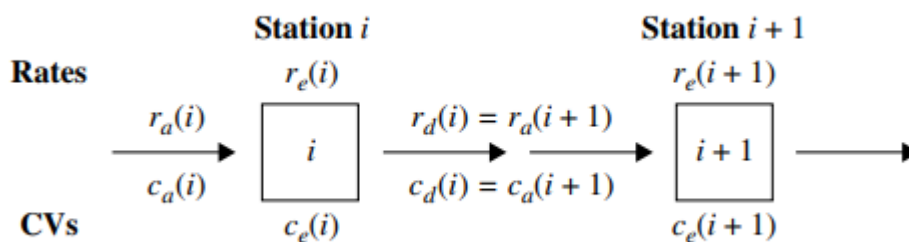


Figure 14: Cascade Equations

It is important to mention that most of **Factory Physics**[®] focus on individual workstations, but to perform a full analysis using these equations some form of cascading will be needed.

The authors have taken that into consideration but only variability wise, that’s why they did an interpolation:

Variability in departures from a station are the result of both variability in arrivals to the station and variability in the process times. The relative contribution of these two factors depends on the utilization of the workstation.

Two extreme cases can be noticed:

➤ **1st case:**

If utilization is close to one, then the station is almost always busy. Therefore, under these conditions, the interdeparture times from the station will be essentially identical to the process times:

$$td = te$$

➤ **2nd case:**

At the other extreme, when utilization is close to zero, the station is very lightly loaded. Virtually every time a job is finished, the station has to wait a long time for another arrival to work on. Because process time is a small fraction of the time between departures, interdeparture times will be almost identical to interarrival times:

$$td = ta$$

Based on the previous, the authors used a simple method to interpolate between the two extremes using the square of utilization in order to calculate the coefficient of variation of interdeparture time **Cd**.

But to put the individual stations equations in cascade, we must be able to calculate the mean time of interdeparture **td** of a station so that we can use it as an interarrival time for the following station.

$$ta(i + 1) = td(i)$$

Following the same logic that the authors used, the utilization can be used to interpolate between the two extremes in order to calculate **td**, we did that and we obtained this equation:

$$td = u \cdot te + (1 - u) \cdot ta$$

The influence of interarrival time or the processing time on the interdeparture time depends totally on the utilization, a higher utilization will make the processing time more dominant and a lower utilization will make the interarrival time more dominant:

$$\text{if } U = 0 \text{ then } td = ta$$

$$\text{if } U = 1 \text{ then } td = te$$

Which represents accurately the two extreme cases previously described and showed its coherence in the calculations.

we implemented this equation in the code by adding a method `td` to the station class so that each station object calculates its interdeparture time, next when the routing/lines lists are created, for every station its interarrival time would be the interdeparture time of the previous one, same is done for the interarrival and interdeparture coefficient of variation:

```
for i in range(1,nbss['nbs'+str(j)]):
    rout['listt'+str(j)][i].ta=rout['listt'+str(j)][i-1].td #w
    rout['listt'+str(j)][i].Ra()
    rout['listt'+str(j)][i].Td()
    rout['listt'+str(j)][i].ca=rout['listt'+str(j)][i-1].scvd
    rout['listt'+str(j)][i].Scvd() #We Calculate Scvd
```

Figure 15: Screenshot of Application Program – Cascade Equations

2.3.2.2 THE INDUCTION OF LINE AND FACTORY PARAMETERS:

We focused on three performance indicators in this application:

- Throughput (TH)
- Work-in-process (WIP)
- Cycle time (CT)

Line/routing:

To calculate the cycle time CT and the work in process WIP of a line/routing we just need to sum the parameter for every station that constitutes the line/routing:

$$WIP_{line} = \sum WIP(i)$$

$$CT_{line} = \sum CT_i$$

Then the Throughput can be easily deduced using little's equation:

$$(i) = WIP_{line} / CT_{line}$$

The equations to calculate WIP and CT for a station are established by the authors, mentioned in the 1st chapter of this thesis and implemented in our application:

```
def WIP(self): #this function calculates Work in process depending on the type of Queue of the system
def Th(self): #this function calculates Throughput depending on the type of Queue of the system...
def CT(self): #this function calculates Cycle Time depending on the type of Queue of the system and typ
```

Figure 16: Screenshot of Application Program - Functions

And the induction for the line/routing is also implemented in our code, the predefined function “sum” was used to sum the return value from objects methods of station type on line correspondent list:

```

routt[j].CT=sum(P.CT() for P in rout['listt'+str(j)]) #Sum up the Cycle time of all the product
routt[j].WIP=sum(P.WIP() for P in rout['listt'+str(j)]) #Sum up the WIP of all the product tha
routt[j].TH=routt[j].WIP/routt[j].CT #Throughput is being calculated from the law of little

```

Figure 17: Screenshot of Application Program - Functions Implementations

“**routt**” being the list of lines and “**rout**” being the dictionary that carries the names of the copied list stations and P the station objects already created.

NB:in python a dictionary is a collection which is ordered, changeable and does not allow duplicates, we used it to automate the creation of copied lists and simplify the use of loops on them.

Workshop:

An induction is also used to calculate indicators of performance for the whole workshop starting from lines, but in reality, most factories produce a variety of products and a workshop can have multiple lines that produce multiple products, so a simple summation wouldn't be a helpful generalization and that's what made us think of using line's weight.

For a company with a variety of products, certain products could be more important than others so a generalized indicator for the whole workshop wouldn't be meaningful unless it takes into account the products that it's making, their lines/routings and their importance.

A way to do that was to give weights to each line/routing depending on the importance of the demand of the product.

To implement this in our code, product class has a type and weight attributes:

```

class product(object):
    def __init__(self):
        self.tp=0 #Product Type

```

Figure 18: Screenshot of Application Program - Product Type Attribute

And the line class has a product type attribute:

```
class Line(object):
    #Intialization of data used in the class 'Line'
    def __init__(self):
        self.tp=0 #Type of product
        self.WIP=0 #Work in Process
```

Figure 19: Screenshot of Application Program - Product Type Attribute in Line Class

So, the performance indicators will be induced by products first, then multiplied but the product's given weight to be finally induced for the whole workshop.

```
prod[j].WIP=sum(S.WIP for S in routt if S.tp is j)
prod[j].CT=sum(S.CT for S in routt if S.tp is j)
prod[j].TH= prod[j].WIP/prod[j].CT
F.TH=sum( prod[j].TH* prod[j].weight for j in range( np))
F.CT=sum( prod[j].CT* prod[j].weight for j in range( np))
F.WIP=sum( prod[j].WIP* prod[j].weight for j in range( np))
```

Figure 20: Screenshot of Application Program - Performance Indicators

Finally, we decided to put all the result that our code calculate in an excel file that is composed from five sheet; each one is dedicated to a class so the results will be organize as wanted.

And to implement this we use the module Pandas that works with Data Frames so we created each data frame to a sheet so our result will be organized like we want

```
df1 = pd.DataFrame(stations, columns=['Station '
df2 = pd.DataFrame(lines, columns=['line ' + str
df3 = pd.DataFrame(products, columns=['Product '
df4 = pd.DataFrame(factories, columns=['Factory'
df5 = pd.DataFrame(datas, columns=['Station_Data
```

Figure 21: Screenshot of Application Program - Data Frames

Chapter

3

Interface Desktop Application

In this chapter we will present the interface of the desktop application and we will test it using data of real companies.

CHAPTER 3: INTERFACE DESKTOP APPLICATION

1. USER INTERFACE:

A graphical user interface GUI is necessary for almost any web or desktop application, not only for the aesthetics, but also to make it user friendly and increase its utility to be within the reach of the less technically savvy users, a GUI simply makes the application more accessible.

1.1 DEFINITION:

A graphical user interface GUI is a visual way of interacting with a computer using items such as windows, icons, and menus, used by most modern operating systems.

A way of arranging information on a computer screen that is easy to understand and use because it uses icons (= pictures), menus, and a mouse rather than only text.

[Definition of graphical user interface from the Cambridge Advanced Learner's Dictionary & Thesaurus © Cambridge University Press]

1.2 GUI PROGRAMMING IN PYTHON

Python has a large number of GUI toolkits available for it, from TkInter to a number of other cross-platform solutions. The major cross-platform technology upon which Python frameworks are based includes Qt.

PyQt is a free software implemented as a Python plug-in. It was developed by Riverbank Computing.it supports Microsoft Windows, MacOS and Linux.

it implements around 440 classes and 6,000 methods including:

- A set of GUI widgets
- Classes for accessing databases
- Scintilla-based rich text editor widget
- Data aware widgets
- XML parser
- SVG support
- Classes for embedding ActiveX controls on Microsoft Windows

1.3 THE DESKTOP APPLICATION GUI:

We used PyQt5 to develop a graphical user interface that matches our code, in this part we won't show the code or the development but we'll focus on the user experience and the overall functioning.

The interface is composed of one window that contains three tabs:

- Main
- Stations
- Lines
- Products

The interaction with each tab executes a specific part of our code.

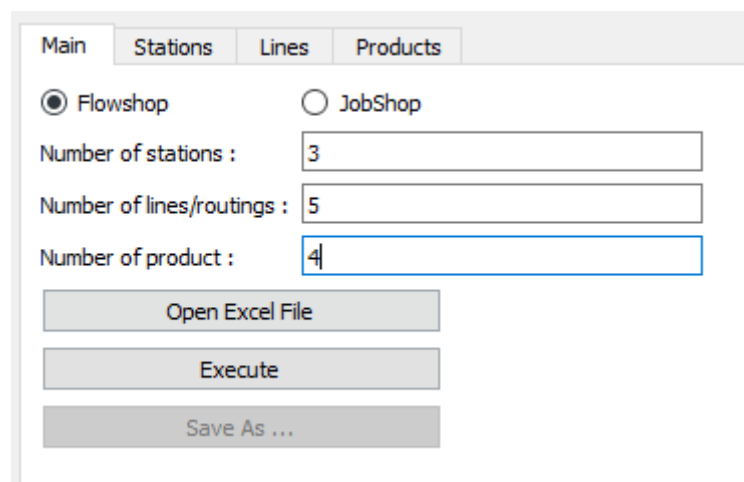


Figure 22: Screenshot of the Interface Application- Main Window

Main tab: in this tab, the user can start by choosing the type of layout corresponding to the manufacturing system, two layouts are considered in our source code: Flow Shop and Job Shop layout.

Then it gives entry zones for three variables: number of stations, lines/routings and product.

The number of stations here represent all the workstations regardless of their lines or layout.

When a job shop layout is selected, the entry zone for the number of products is blocked, because we assume that by definition every product in this type of layout has its own unique routing.

The open excel file button gives the possibility of entering all the data necessary through an automated excel file without passing by the interface. If the user clicks on this button an already made excel file will open so that the data can be entered there.

	t	sig	ts	ns	mf	mr	sig1	sig2	n	sig3	ts	sig4	ur	r	d	l	pk	tk	s	pbatch	tbatch	type1	split	
Nombre Station	4																							
Station 1	59,7	1,11	0	3	1	1	0	0	1	1,57	6,4	0	0	1	GG1	0	17	1	0	0	True	True	Sim	False
Station 2	46,4	2,1	0	3	1	1	0	0	1	1	1	0	0	1	GG1	0	1	1	0	0	True	True	Sim	False
Station 3	27,5	0,49	0	3	1	1	0	0	1	1	1	0	0	1	GG1	0	5	5	0	0	True	True	Sim	False
Station 4	34,5	9,34	0	3	1	1	0	0	1	1	1	0	0	1	GG1	0	5	50	0	0	True	True	Sim	False
Nombre Route	1																							
Route 1	0	2	1	2	3	4																		
Nombre Produits	1																							
Produit 1	0	4500	5000	45000	1	1234000																		
workshop	1																							

Figure 23: Screenshot of the Excel File with the Import Data

This method is more suited for users that already have collected the kind of data needed for the application, so the transfer can be easily automated between the files where the data and this input file. The results are the same whether the user chooses to work this method or keep entering data manually in the interface.

Station tab: in this tab, all the quantities of all the workstations are entered from mean processing time to standard deviation and meantime between failures as well as existing of batching and its types...etc.

We considered a large amount of data when it comes to workstations so that most of the factory physics equations can be implemented.

Station Name	Raw Mean Time	Standard Deviation	Mean Setup Time	Job's Number Between Setup	Mean Time Between Failure	Mean Time To Repair	Standard Deviation Of Repair Time	Standard Deviation Of Setup	Parallel Machine's Number	Standard Deviation Of Inter-Arrival Time	Mean Time Of Inter-Arrival Time	Standard Deviation Of Demands
1 S1												
2 S2												
3 S3												

Figure 24: Screenshot of the Interface Application -Station Window

Line tab: in this tab, the product type produced on this line/routing is added as well as number of operators, then the user must define the routing by simply entering the order of the workstation in the corresponding line/routings and entering 0 in the case of workstation that doesn't belong.

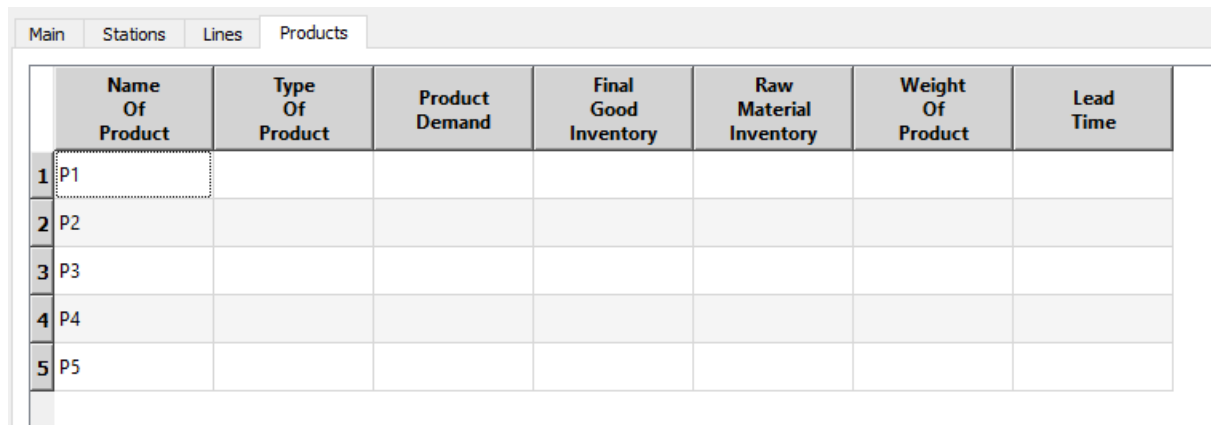
Name	Product Type	Number Of Operators	S1	S2	S3
1 L1					
2 L2					
3 L3					
4 L4					
5 L5					

Figure 25: Screenshot of the Interface Application- Lines Window

Product Tab: in this tab, the user enters information about the products such as names, types, inventory and also weight.

We added weight so that we'll be able to use it with the aim to generalize the performance indicators to the whole workshop.

This weight will be calculated by the importance of the product's demand.

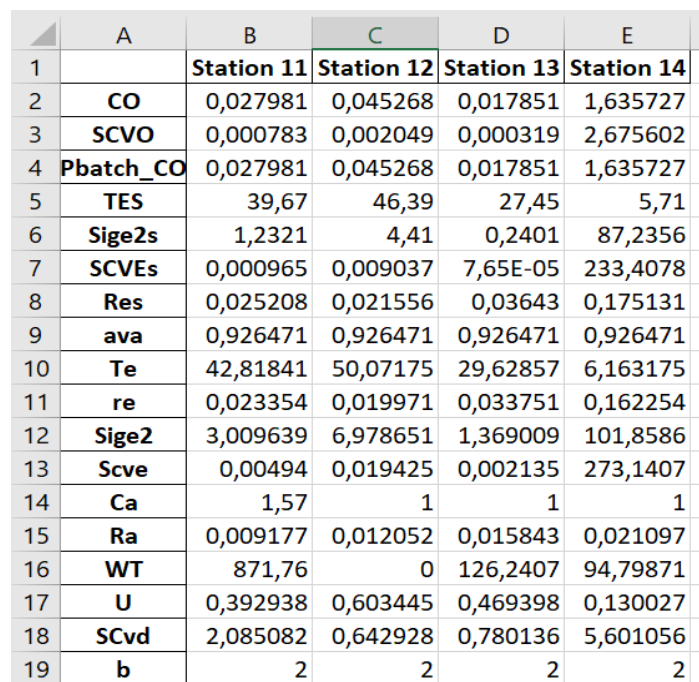


	Name Of Product	Type Of Product	Product Demand	Final Good Inventory	Raw Material Inventory	Weight Of Product	Lead Time
1	P1						
2	P2						
3	P3						
4	P4						
5	P5						

Figure 26: Screenshot of the Interface Application- Products Window

Finally when all the data is entered, the user can simply return to the main tab and click on the execute button.

An excel file will open, carrying all the calculations that our source code did for workstations, lines as well as the generalization for the whole workshop.



	A	B	C	D	E
1		Station 11	Station 12	Station 13	Station 14
2	CO	0,027981	0,045268	0,017851	1,635727
3	SCVO	0,000783	0,002049	0,000319	2,675602
4	Pbatch_CO	0,027981	0,045268	0,017851	1,635727
5	TES	39,67	46,39	27,45	5,71
6	Sige2s	1,2321	4,41	0,2401	87,2356
7	SCVEs	0,000965	0,009037	7,65E-05	233,4078
8	Res	0,025208	0,021556	0,03643	0,175131
9	ava	0,926471	0,926471	0,926471	0,926471
10	Te	42,81841	50,07175	29,62857	6,163175
11	re	0,023354	0,019971	0,033751	0,162254
12	Sige2	3,009639	6,978651	1,369009	101,8586
13	Scve	0,00494	0,019425	0,002135	273,1407
14	Ca	1,57	1	1	1
15	Ra	0,009177	0,012052	0,015843	0,021097
16	WT	871,76	0	126,2407	94,79871
17	U	0,392938	0,603445	0,469398	0,130027
18	SCvd	2,085082	0,642928	0,780136	5,601056
19	b	2	2	2	2

Figure 27: Screenshot of the result Excel File - Station Sheet

	A	B	C
1		line 1	
2	CT	298,1402	
3	WIP	3,203105	
4	TH	0,010744	

Figure 28: Screenshot of the result Excel File - Line Sheet

	A	B
1		Factory
2	CT	540,6283
3	TH	0,005925
4	WIP	3,203105

Figure 29: Screenshot of the result Excel File - Product Sheet

These results will then be presented in the form of tables and graphs to help visualize and simplify the evaluation and the analysis of all the possible parameters.

2. EXAMPLES OF APPLICATION:

2.1 CARDBOARD PAPERS:

Our first example is a company that produces different cardboard papers on different stations we considered only five products to understand how our application works.

The workshop is composed from five stations that produces different products; each station is assigned to one product, which means there is 5 lines that are composed from only one station.

In the following table we present the data that we have in our possession:

Station Parameters	Station 1	Station 2	Station 3	Station 4	Station 5
Te (Process Time)	19	17	19	19	28
Sige	0.7	0.8	0.9	0.9	0.85
MTBF	5578.615	18037	8633.1	10640.08	8017.615
MTTR	77.72	39.496	113.48	56.7	71.654
Yield loss	0.1	0.1	0.1	0.1	0.1

Table 7: Parameters of the workshop - Example1

After creating a model that works with factory physics equations, we started entering the data we collected into the application

The interaction with each tab executes a specific part of our code.

- **Main Tab:**

- We chose a Jobshop layout because it was the most appropriate layout for this example
- We entered 5 stations as mentioned previously
- We entered 5 lines as explained before
- As the layout is jobshop the number of products is equal to the numbers of lines

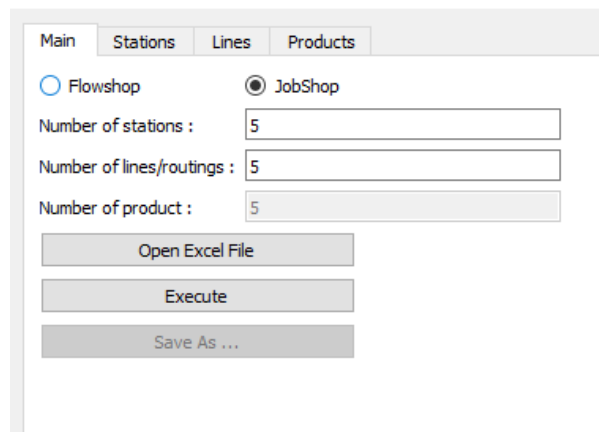


Figure 30: Screenshot of the Main Window –Example1

We could've also used an open excel file button that would give us the possibility of entering all the data necessary through an automated excel file without passing by the interface as we will do in the following example.

However, for this example we chose to stick on the interface.

- **Station Tab:**

In this tab, we entered the parameters of all the workstation from mean processing time to standard deviation and meantime between failures as well as the existence of batching and its types ...etc.

	Station Name	Raw Mean Time	Standard Deviation	Mean Setup Time	Job's Number Between Setup	Mean Time Between Failure	Mean Time To Repair	Standard Deviation Of Repair Time	Standard Deviation Of Setup	Parallel Machine's Number	Standard Deviation Of Inter-Arrival Time	Mean Time Of Inter-Arrival Time	Standard Deviation Of Demands
1	Station 1	19	0.7	0	3	5578.615	77.792	0	0	1	1	20.2	0
2	Station 2	17	0.8	0	3	2832.23	39.496	0	0	1	1	20.2	0
3	Station 3	19	0.9	0	3	8633.1	113.48	0	0	1	1	20.2	0
4	Station 4	19	0.9	0	3	6889.4	90.56	0	0	1	1	20.2	0
5	Station 5	28	0.85	0	3	5138.45	71.654	0	0	1	1	35	0

Figure 31: Screenshot of the Stations Window Part 1 –Example1

	Standard Deviation Of Demands	Number Of Demands	Number Of Machines	Type Of Queue	Buffer Size	Process Batch Size	Transfer Batch Size	Setup Time For Pbatch	Scrap Percentage	Process Batch Exist?	Transfer Batch Exist?	Type Of Pbatch	Pbatch Split Exist
1	0	0	2	GG1	0	1	1	0	0.1	1	1	Seq	1
2	0	0	3	GG1	0	1	1	0	0.1	1	1	Seq	1
3	0	0	2	GG1	0	1	1	0	0.1	1	1	Seq	1
4	0	0	2	GG1	0	1	1	0	0.1	1	1	Seq	1
5	0	0	4	GG1	0	1	1	0	0.1	1	1	Seq	1

Figure 32: Screenshot of the Station Window Part 2 –Example1

- **Line tab:**

In this tab, the product type was selected as well as the number of the operators (two operators), then we defined the routing by simply entering the order of the workstation, as explained before.

	Name	Product Type	Number Of Operators	Station 1	Station 2	Station 3	Station 4	Station 5
1	Line 1	0	2	1	0	0	0	0
2	Line 2	1	2	0	1	0	0	0
3	Line 3	2	2	0	0	1	0	0
4	Line 4	3	2	0	0	0	1	0
5	Line 5	4	2	0	0	0	0	1

Figure 33: Screenshot of the Lines Window –Example1

- **Product tab:**

In this tab the users enters information’s about the products such as names, types inventory and also weight

In this case, the weights are equally assigned to the products because of not having this kind of information.

	Name Of Product	Type Of Product	Product Demand	Final Good Inventory	Raw Material Inventory	Weight Of Product	Lead Time
1	Product 1	0	7500	12000	12000	0.2	432000
2	Product 2	1	2000	12000	12000	0.2	432000
3	Product 3	2	300	12000	12000	0.2	432000
4	Product 4	3	7500	12000	12000	0.2	432000
5	Product 5	4	2300	12000	12000	0.2	432000

Figure 34: Screenshot of the Product Window –Example1

Finally when all the data is entered, we return to the main tab and click on the execute button.

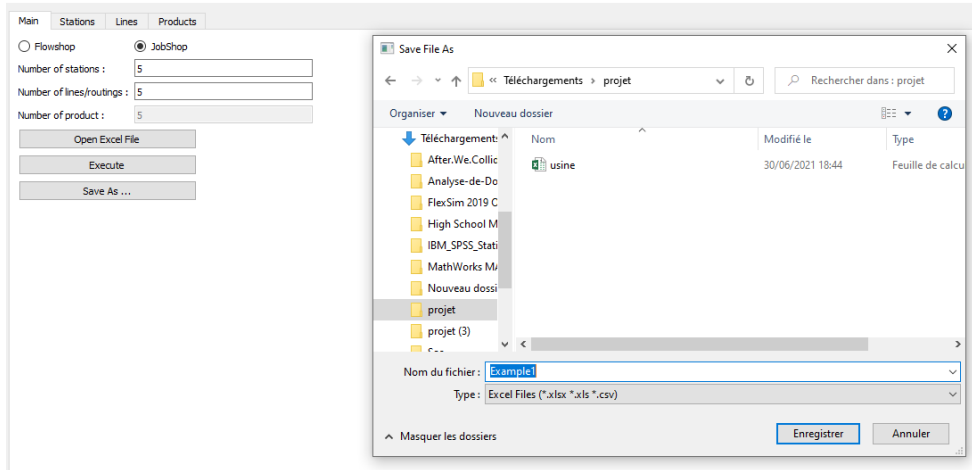


Figure 35: Screenshot of Open Excel File Window –Example1

Therefore, an excel file will be saved, carrying all the calculations that our source code did for workstation, line, product, factory and all the data we entered about the workstations.

	A	B	C	D	E	F	G	H	I	J
3	SCVO	0,001357	0,002215	0,002244	0,002244	0,000922				
4	Pbatch_CC	0,026051	0,027169	0,033495	0,033495	0,015179				
5	TES	19	17	19	19	28				
6	Sige2s	0,49	0,64	0,81	0,81	0,7225				
7	SCVEs	0,000665	0,001417	0,001817	0,001817	0,000666				
8	Res	0,052632	0,058824	0,052632	0,052632	0,035714				
9	ava	0,986247	0,986247	0,987026	0,987026	0,986247				
10	Te	19,26495	17,23707	19,24975	19,24975	28,39045				
11	re	0,051908	0,058015	0,051949	0,051949	0,035223				
12	Sige2	21,11467	10,02122	29,17311	23,44896	28,72015				
13	Scve	1,201248	0,337999	2,296758	1,483873	1,023361				
14	Ca	0,049505	0,049505	0,049505	0,049505	0,028571				
15	Ra	0,049505	0,049505	0,049505	0,049505	0,028571				
16	WT	0	0	0	0	0				
17	U	0,95371	0,85332	0,952958	0,952958	0,811156				
18	SCvd	0,002451	0,002451	0,002451	0,002451	0,000816				
19	b	2	2	2	2	2				
20	in Porcess	20,6031	5,817572	20,25757	20,2576	4,295369				
21	TH	0,07183	0,152617	0,038995	0,058993	0,042557				
22	CT	258,1496	34,30683	467,5416	309,0486	90,83837				
23	YieldLoss	0,007981	0,016957	0,004333	0,006555	0,004729				
24	out with yi	0,079811	0,169575	0,043328	0,065548	0,047286				
25	CTq	238,8847	17,06977	448,2919	289,7989	62,44792				
26	WIPs	11,00507	0,045000	22,10267	14,34240	1,704290				

Figure 36: Screenshot of the result Excel File - Station Sheet - Example1

For the others sheet that concerns Line, Product, Factory and data are presented in the Annex.

2.2 WHITE PAPER COMPANY :

Our second example is a company that produces different white papers on multiple stations we considered only one products to understand how our application works.

The workshop is composed from one line that is divided in 4 stations that produces multiple product;

In the following table, we present the data that we have in our possession:

Station	Station 1	Station 2	Station 3	Station 4
Parameters				
Te (Process Time)	59,67	12,8	36,3	4,03
Sige	1,11	2,1	0,49	1,6
Ta(Interarrivals times)	8	-	-	-
Siga	1	-	-	-
MTBF	22680	22680	22680	22680
MTTR	100	100	100	100
Process batch	17	1	5	5
Type of Process Batch	Simultaneous	-	Simultaneous	Simultaneous
Transfer Batch	1	5	5	50

Table 8: Parameters of the Workshop - Example 2

After creating a model that works with factory physics equations, we started entering the data we collected into the application

The interaction with each tab executes a specific part of our code.

Main tab:

- We chose a Flowshop layout
- We entered 4 stations as mentioned previously
- We entered one product that we'll be working on

Figure 37: Screenshot of the Main Window –Example 2

For this case we actually used the open excel file button that gave us the possibility of entering all the data necessary through an automated excel file without passing by the interface.

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1																										
2																										
3	Nombre Station		Station 1	59,7	1,11	0	3	22680	100	0	0	1	1	8	0	0	1	GG1	0	17	1	0	0	True	True	Sim
4	4		Station 2	12,8	2,1	0	3	22680	100	0	0	1	1	1	0	0	1	GG1	0	1	5	0	0	True	True	Sim
5			Station 3	36,3	0,49	0	3	22680	100	0	0	1	1	1	0	0	1	GG1	0	5	5	0	0	True	True	Sim
6			Station 4	4,03	1,16	0	3	22680	100	0	0	1	1	1	0	0	1	GG1	0	5	50	0	0	True	True	Sim
7	Nombre Route																									
8	1																									
9			Route 1	0	2	1	2	3	4																	
10	Nombre Produits																									
11	1																									
12			Produit 1	0	4500	5000	45000	1	1234000																	
13	workshop																									
14	1																									
15																										
16																										
17			Creer																							
18																										
19																										

Figure 38: Screenshot of the Excel File –Example 2

This method is more suited because we already had collected the kind of data needed for the application, so the transfer is easily automated between the files where the data and this input file.

So that the reason we chose to go by the excel file.

The Tabs:

The tabs that exist in our application will be automatically completed by the excel file and the representation are the same as seen before in Figures 30, 31, 32 and 33.

Therefore when all the data is entered, we return to the main tab and click on the execute button, an excel file will be saved, carrying all the calculations that our source code did for workstation, line, product, factory and all the data we entered about the workstations.

These results will then be presented in the form of tables to help visualize it and simplify the evaluation and the analysis of all the possible parameters.

	A	B	C	D	E	F	G
4	Pbatch_CO	0,018602	0,164063	0,013517	0,287841		
5	TES	59,67	12,8	36,25	4,03		
6	Sige2s	1,2321	4,41	0,2401	1,3456		
7	SCVEs	0,000426	0,118702	4,39E-05	0,111486		
8	Res	0,016759	0,078125	0,027586	0,248139		
9	ava	0,99561	0,99561	0,99561	0,99561		
10	Te	59,9331	12,85644	36,40983	4,047769		
11	re	0,016685	0,077782	0,027465	0,24705		
12	Sige2	27,55251	10,09271	16,22547	3,134388		
13	Scve	0,211344	0,616275	0,19859	0,599617		
14	Ca	0,125	0,125	0,158362	0,229592		
15	Ra	0,007353	0,009758	0,010961	0,014418		
16	WT	1088	0	182,4699	138,711		
17	U	0,440685	0,125455	0,399078	0,058363		
18	SCvd	0,015625	0,025079	0,052713	0,054575		
19	b	2	2	2	2		
20	in Porcess	0,7879	0,143452	0,664108	0,06198		
21	TH	0,012132	0,010674	0,017085	0,015028		
22	CT	64,94475	13,43914	38,87143	4,124308		
23	YieldLoss	0	0	0	0		
24	out with yi	0,012132	0,010674	0,017085	0,015028		
25	CTq	5,011657	0,5827	2,461599	0,076539		
26	WIPq	0,03685	0,005686	0,026981	0,001104		
27	WIP	0,03685	0,005686	0,026981	0,001104		
	Station	Line	Product	Factory	Data		

Figure 39: Screenshot of the result Excel File - Station Sheet - Example 2

For the others sheet that concerns Line, Product, Factory and data are presented in the Annex.

GENERAL CONCLUSION:

Factory physics is definitely a start or at least a successful attempt to build a manufacturing science, it is far from being just an analytical model.

The application developed during this project was an attempt to combine knowledge on factory physics with our knowledge in coding to create a tool that can carry out a profound analysis of manufacturing systems. The project was time limited so it wasn't possible for us to explore all the ideas we had, so from an evolution perspective many things can be added:

- Develop multiple methods to induce performance indicators for a workshop.
- Use statistical models, data analysis and artificial intelligence methods so that the application can operate directly on raw data.
- Consider scheduling.
- Provide the possibility to link the application directly with simulation models.
- Widen the layout choices (consider FMS)

Our work was an attempt to innovate by linking separate equations, using them in one unified tool, creating new notions and approximations to adjust the model and deal with real life manufacturing systems and finally simplifying the user experience.

BILBIOGRAPHICAL & WEBOGRAPHIC RESEARCH

Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises". Section 1.1. Archived from the original (PDF) on 23 June 2012.

*Rossum, Guido Van (20 January 2009). "The History of Python: A Brief Timeline of Python". *The History of Python*. Retrieved 5 March 2021.*

*Peterson, Benjamin (20 April 2020). "Python Insider: Python 2.7.18, the last release of Python 2". *Python Insider*. Retrieved 27 April 2020.*

[Hopp & Spearman, 2008] Wallace J. Hopp, Mark L. Spearman. *Factory Physics*. 3rd edition. Illinois: Waveland Press, 2008, 720 p

The Cambridge Advanced Learner's Dictionary & Thesaurus © Cambridge University Press

ANNEX:

ANNEX 1: SCREENSHOT OF THE RESULT EXCEL FILE - LINE SHEET - EXAMPLE 1

1	A	B	C	D	E	F
		line 1	line 2	line 3	line 4	line 5
2	CT	258,1496161	34,30683367	467,5416056	309,0486198	90,83836979
3	WIP	20,6030978	5,817571973	20,25757229	20,2576039	4,29536872
4	TH	0,079810685	0,169574728	0,043327849	0,065548275	0,047285841
5	BestCase(CT,TH)	(198.4588132768872, 0.103815484	(33.4259618489224, 0.174043517423893)	(194.97660489223793, 0.103897451	(194.9769235000275, 0.10389744353)	(30.486863598967453, 0.14089244393476327
6	WorstCase(CT,TH)	(396.91762655377437, 0.05190774	(100.27788554676718, 0.0580145058079)	(389.95320978447586, 0.051948725	(389.953847000055, 0.051948721769)	(121.94745439586981, 0.03522311098369082
7	PWC(CT,TH)	(208.0912877351533, 0.099009900	(44.91734072705023, 0.12951728395532)	(204.60148008191487, 0.099009900	(204.60179939633198, 0.0990099009)	(51.77970171162788, 0.08295468259524001)
8	THmax	0,103815484	0,116029012	0,103897451	0,103897444	0,070446222
9	Eu	0,001890253	0,004488743	0,001026186	0,001552459	0,000759951
10	Ect	0,073600729	0,495528097	0,040638095	0,061479	0,30823979
11	Elt	1	1	1	1	1
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						

ANNEX 2: SCREENSHOT OF THE RESULT EXCEL FILE - PRODUCT SHEET - EXAMPLE 1

1	A	B	C	D	E	F	G	H
		Product 1	Product 2	Product 3	Product 4	Product 5		
2	CT	258,1496	34,30683	467,5416	309,0486	90,83837		
3	WIP	20,6031	5,817572	20,25757	20,2576	4,295369		
4	TH	0,079811	0,169575	0,043328	0,065548	0,047286		
5	Eth	1,06E-05	8,48E-05	0,000144	8,74E-06	2,06E-05		
6	Einv	1,58E-07	3,74E-07	8,55E-08	1,29E-07	6,33E-08		
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								

ANNEX 3: SCREENSHOT OF THE RESULT EXCEL FILE - FACTORY SHEET - EXAMPLE 1

	A	B	C	D	E	F	G
1		Factory					
2	CT	231,977					
3	TH	0,081109					
4	WIP	14,24624					
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							

Navigation: Station | Line | Product | **Factory** | Data | +

ANNEX 4: SCREENSHOT OF THE RESULT EXCEL FILE - LINE SHEET - EXAMPLE 2

	A	B
1		line 1
2	CT	121,3796284
3	WIP	1,657439919
4	TH	0,013655009
5	BestCase(CT,TH)	(113.24713403880072, 0.014635601451792603)
6	WorstCase(CT,TH)	(187.70032070784586, 0.008830245537669678)
7	PWC(CT,TH)	(152.64954333850736, 0.010857811186989105)
8	THmax	0,017660491
9	Eu	0,002618782
10	Ect	0,928903816
11	Elt	1
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		

Navigation: Station | **Line** | Product | Factory | Data

ANNEX 5: SCREENSHOT OF THE RESULT EXCEL FILE - PRODUCT SHEET - EXAMPLE 2

	A	B	C	D	E	F
1		Product 1				
2	CT	121,3796				
3	WIP	1,65744				
4	TH	0,013655				
5	Eth	3,03E-06				
6	Env	1,05E-07				
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						

Navigation: Station | Line | **Product** | Factory | Data

ANNEX 6: SCREENSHOT OF THE RESULT EXCEL FILE - STATION SHEET - EXAMPLE 2

	A	B	C	D	E	F
1		Factory				
2	CT	121,3796				
3	TH	0,013655				
4	WIP	1,65744				
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						

Navigation: Station | Line | Product | **Factory** | Data

ANNEX 7: SCREENSHOT OF THE RESULT EXCEL FILE - DATA SHEET - EXAMPLE 2

A	B	C	D	E	F	
1	Station_Data 11	Station_Data 21	Station_Data 31	Station_Data 41	Station_Data 51	
2	t	19	17	19	19	28
3	sig	0,7	0,8	0,9	0,9	0,85
4	n	2	3	2	2	4
5	ts	0	0	0	0	0
6	Ns	3	3	3	3	3
7	sigs	0	0	0	0	0
8	m	1	1	1	1	1
9	mr	77,792	39,496	113,48	90,56	71,654
10	mf	5578,615	2832,23	8633,1	6889,4	5138,45
11	sigr	0	0	0	0	0
12	sig	1	1	1	1	1
13	ta	20,2	20,2	20,2	20,2	35
14	R	1	1	1	1	1
15	Q	GG1	GG1	GG1	GG1	GG1
16	B	0	0	0	0	0
17	k	1	1	1	1	1
18	s	0	0	0	0	0
19	Pbatch	VRAI	VRAI	VRAI	VRAI	VRAI
20	TypeB	Seq	Seq	Seq	Seq	Seq
21	split	VRAI	VRAI	VRAI	VRAI	VRAI
22	sign	0	0	0	0	0
23	un	0	0	0	0	0

ANNEX 9: SCREENSHOT OF THE RESULT EXCEL FILE - DATA SHEET - EXAMPLE 1

A	B	C	D	E	
1	Station_Data 11	Station_Data 12	Station_Data 13	Station_Data 14	
2	t	59,67	12,8	36,25	4,03
3	sig	1,11	2,1	0,49	1,16
4	n	1	1	1	1
5	ts	0	0	0	0
6	Ns	3	3	3	3
7	sigs	0	0	0	0
8	m	1	1	1	1
9	mr	100	100	100	100
10	mf	22680	22680	22680	22680
11	sigr	0	0	0	0
12	sig	1	1	1	1
13	ta	8	102,4784923	18,24699181	13,87109534
14	R	1	2	3	4
15	Q	GG1	GG1	GG1	GG1
16	B	0	0	0	0
17	k	17	1	5	5
18	s	0	0	0	0
19	Pbatch	VRAI	VRAI	VRAI	VRAI
20	TypeB	Sim	Sim	Sim	Sim
21	split	VRAI	VRAI	VRAI	VRAI
22	sign	0	0	0	0
23	un	0	0	0	0