UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

BACHELOR THESIS

Bachelor in Mechanical Engineering

DEVELOPMENT OF A MODEL FOR PERSISTENT STORAGE OF PROCESS DATA IN DOUBLE-SIDED GRINDING WITH PLANETARY KINEMATICS

Memoria & Annex

Author: Pablo Bartol Garrido
Supervisor: Michael List

Summon: June of 2018

Abstract

The terms Industry 4.0, Internet of Things and Big Data have been among the central themes in the industrial environment for several years. Through the use of modern information and communication technology, industrial production will connect more strongly in the so-called fourth industrial revolution. This requires that there be data available based on the optimization that can be carried out. These data include, for double face grinding, the characteristics of the process and the tool, but also the qualities of the work piece achieved in the process. So far, a lot of different parameters are stored before, during and after the processes. However, until now there is no central possibility of storing the data linked together persistently. In order to store the process data, in this project a relational database using MySQL has been created and a program to process the machine files and inserting data into the DB using Python language has been coded. With the help of this network, it should be possible to optimize the entire value chain. However due to time constrains it has not been possible to create a user-friendly environment and the project should be developed further by the next developer to come.

Keywords*:* Big Data, Database, MySQL, Python, Double Face Grinding.

Acknowledgements

I would like to thank my thesis advisor Michael List of the Department of Machine Tools and Factory Management at Technische Universitaet Berlin. He was always willing to help with simple administrative questions or complex questions about the software. He also provided me with all the papers I needed with information about double face grinding, as well as documentation about the programming language or database used in this work. He allowed this project to be done with a total freedom of choice but steered me in the right direction whenever I needed it.

I would also like to thank my family, friends and all the people I met and took care of me during the months spent in Berlin working on this project.

Glossary

DB – DataBase

IWF - Institut für Werkzeugmaschinen und Fabrikbetrieb

SQL – Structured Query Language

DBMS - Database Management System

RDBMS – Relational Database Management System

MySQL – Combination of the co-founders' daughter "My" and SQL

TOAD – Tool for Oracle Application Developers

MRR – Material Removal Rate

IDE - Integrated Development Environment

CamelCase –Practice of writing compound words or phrases such that each word or abbreviation in the middle of the phrase begins with a capital letter, with no intervening spaces or punctuation

GUI - Graphical User Interface

Index

1. Preface

1.1. Origin of the thesis and Motivation

Nowadays the manufacturing industry is experimenting a fast evolution (or revolution) towards what has been named as Industry4.0. In this next paradigm of the industry, the factories have been highly automatized and computerized, all process are linked and interact with each other and with external process. The terms of Internet of Things, Big Data, Deep Learning have been gaining importance in the current industrial environment.

The growing usage of technologies that employ Big Data and Deep Learning in order to optimize either the search of text, as keywords or phrases, or as in images for the search of objects. In this case, it can have a wide range of means, reducing the waste of materials from the workpiece, reducing the wear of the machine tool, improving the finishing quality of the workpiece, improving the process time and therefore making it more profitable.

The main recurrent problem when it comes to Big Data and Deep Learning is the lack of samples of data. Commonly the algorithms require a large number (tens of thousands) of samples to be able to be implemented.

Back to the point of interest, currently when a research or project is carried out in the present system only the data that the author considers relevant is stored and processed, and of that data, only a very small part gets published. Therefore, and taking into account the current state and the most likely future of the manufacturing industry it is necessary to create a persistent data storage system that will allow in a close future and once enough data has been stored to optimize the entire process not only including machine parameters, but also qualities of the workpiece achieved in the process.

1.2. Previous requirements

In order to fulfil the task of this project it was necessary to acquire a basic knowledge of manufacturing processes before writing this thesis in order to understand the most basic/fundamental characteristics about the machine process of interest. That basic knowledge, mostly consisted in the practical use of turning and milling machining and some theoretical knowledge of the polishing, grinding and sintering. In order to develop the software-related part of the project, some basic knowledge of Python and Database (DB) development was needed.

2. Introduction

2.1. Objectives of the thesis

Nowadays when a machine process is carried out in the present system, the data of said process is not being stored in a user-friendly environment and moreover it could be lost over time, therefore a persistent storage system will be created. Over time this persistent storage will be able to store enough data to implement self-learning algorithms that will eventually lead to the optimization of the process parameters.

On the other hand, the creation of a database (DB) will also help newcomers to the "Institut für Werkzeugmaschinen und Fabrikbetrieb" (IWF) who are unfamiliar with this process or who do not have a very deep knowledge of it to become familiar with it quickly.

2.2. Scope of the thesis

The scope of this project is very clear and consists of the following, creation of a DB for Double Face Grinding with Planetary Kinematics and importing the data of the machine process into the DB, either using software to process the machine files or asking the user about specific parameters that the machine could not be aware of. The creation of a visual interface to make more user-friendly the database either in .net, java, C++ or any other language is beyond the scope of this project and is left to be created at a later date or project.

In order to develop this project, first of all it is necessary to explain how does Double Face Grinding with Kinematic Planetary work and describe the main important characteristics. Secondly the different DB development software options available on the market are going to be discussed in order to choose the most adequate for this project. Thirdly the data parameters to store in the chosen software will be detailed and explained. Then is going to be explained and discussed which software is used in order to import data to the DB either asking the user for data and/or processing the machine files.

3. Double face grinding with planetary kinematics

3.1. Overview of the process

First of all, it is necessary to explain the process of interest. Double face grinding with planetary Kinematics is used to machining pieces in which there's a high demand on flatness. The workpieces are placed in a circular and outward teethed workpiece carrier, fixed to the machine by an interior and exterior pin ring, that are placed between an upper and a lower grinding tool and fixed be. Not only both of the grinding wheels are driven, also the workpiece carrier is driven by the rotating inner ping ring (1). Double face grinding with planetary kinematics main characteristics are: machining of two faces simultaneously, relative motion between the working partners, tension-free mounting and constant stress for the entire surface of the piece.

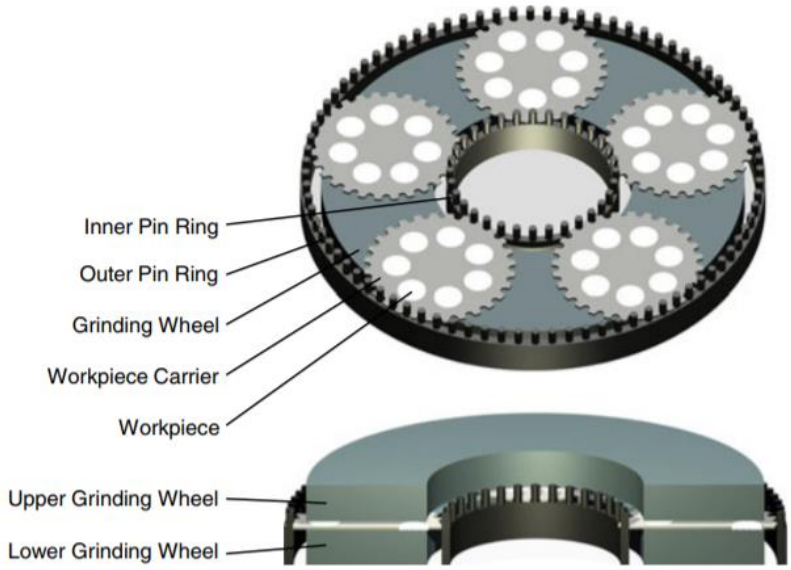

Fig. 3.1.1 Machine tool assembly (1)

At the IWF a machine from the manufacturer STÄHLI AG (Pieterlen/Biel) is available. This system is characterized by a highly rigid portal design, which allows tool revolutions up to 2,000 rpm and contact forces up to 4,000 daN. This is a fourfold increase compared to conventional machine systems. Thereby, cutting speeds up to 45 m/s can be achieved.

The process itself is divided into different steps by the machine as shown in Fig. 3.1.2 Those steps go from P0 to P5. The P0 is kind of a pseudo-step, the workpieces are fixed by the process force while the lubricant is switched on preventing them to float in the working space. In the P1 step the force is reduced to lower the friction while accelerating the wheels. In the P2 step the grinding wheels start to accelerate. The main step is P3 as it's the step where the grinding wheels reach their peak speed and the material removal takes place. In the P4 step the grinding wheels start to deaccelerate and finally in the step P5 the grinding wheel are stopped and ready to move away from the workpieces.

Process Name e.g. DFG223 Vorschliff							
	Process steps						
	P0	P1	P2	P3	P4	P5	(units)
Force	100	50	100	200	150	100	daN
	2	3	4	6	3	1	"
Stop at		0	0	15	0	0	µm
Work Time	3	5	5	0	0	3	"
Rate		0	0	0	0	0	µm/"
Upper Tool		0	-200	-746	-200	0	rpm
		0	4	4	3	0	" *Time to reach RPM (RAMP) in seconds*
Centre Inner pin ring		0	-100	-300	-100	0	rpm
		0	4	4	3	0	"
Lower Tool		0	200	600	200	0	rpm
		0	4	4	3	0	"

Fig. 3.1.2 Sample of the process input by the user at the machine

After this brief introduction to the process, in order to create a persistent storage of the process, it is necessary to identify the most important characteristics or critical parameters. The following parameters have been considered the most relevant or interesting:

3.2. Grinding wheel

Different grinding wheels will lead to different machine parameters thus its relevance. Composed of abrasive material particles bonded together. Grinding wheels are consumables, as the wheel cuts it releases individual grains of abrasive, in this wear process the fresh grains are exposed periodically beginning the next cycle.

Main important parameters of the grinding wheels are the outer and inner tool diameter, grain material, grain size and concentration, bond type, material of the body, type of layer and type of grooving.

3.3. Workpiece

The workpiece by itself is not a machine parameter but different materials of a workpiece will lead to different machine parameters. Most important specifications for the workpieces as an input into the process are: the material of the workpiece and the hardness of said material, key length of the workpiece as diameter for circular pieces or one side length for square workpieces, initial height, form of the workpiece. In addition to the previous parameters, the number of workpiece holders and the number of workpieces is important even though not related to the workpiece as they are not characteristics of it, but they are important for the process hence their relevance.

The main objective of the process is to produce quality workpieces. The main important parameters of the workpiece as an output are: The Surface Quality, including different types of measurements, and the FlatnessE0 of the piece.

3.4. Process kinematics

Given by the machine and generated by the different rotations speeds of the grinding wheels and the inner pin ring. Depending on the different speed ratio between the rotational parts the workpiece holder will describe different paths. A simplified description with the rotational speed ratio:

$$N_L = \frac{n_i}{n_l}$$

$$Where\ n_i = \ internal\ \ pin\ circle\ speed\ ;\ n_l = lower\ grinding\ wheel\ speed$$

Fig. 3.2.1 Rotational speed ratio (2)

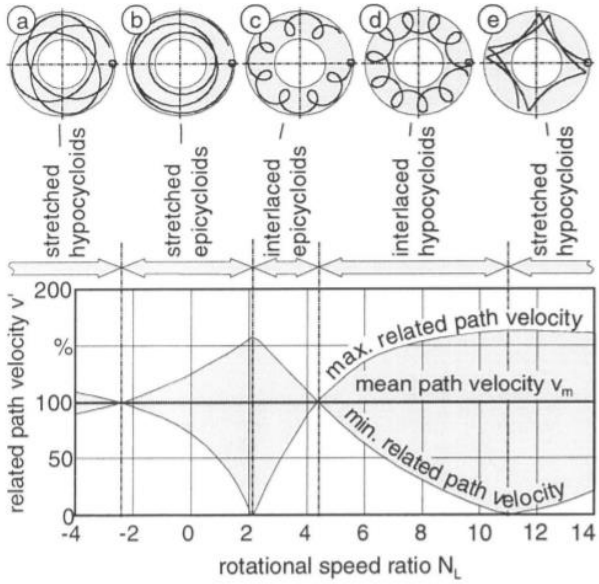

Fig. 3.4.1 Path types and velocities (2)

3.5. Rest of process parameters

In this category fall the machine parameters that don't have an impact on the kinematics. Those are the force which translates to the contact pressure. As a result of the force and the speed of the grinding wheels there are three different engines currents and the coolant parameters as the manufacturer, first use, flowrate, temperature of the coolant and type of coolant.

3.6. Tool wear

The wear of the grinding tools is not only important from the economical perspective, since there is relative motion between the different moving parts there will be an uneven wear of the tool. Even though tool wear is a result of the process, it will have an impact on the surface quality of the workpiece. Most important parameters that affect the tool wear are, grain size and grain concentration of the grinding wheels have a significant influence on wear behaviour (1) chip thickness, operation pressure, inconstant cutting speed of this process as velocity distribution is influenced by the radius, material removal rate. (3)

Regarding tool wear because of the uneven wear a distinguish between two different types of wear has to be made, the axial wear as the height of the amount of material lost and the profile wear as the general shape of the wear.

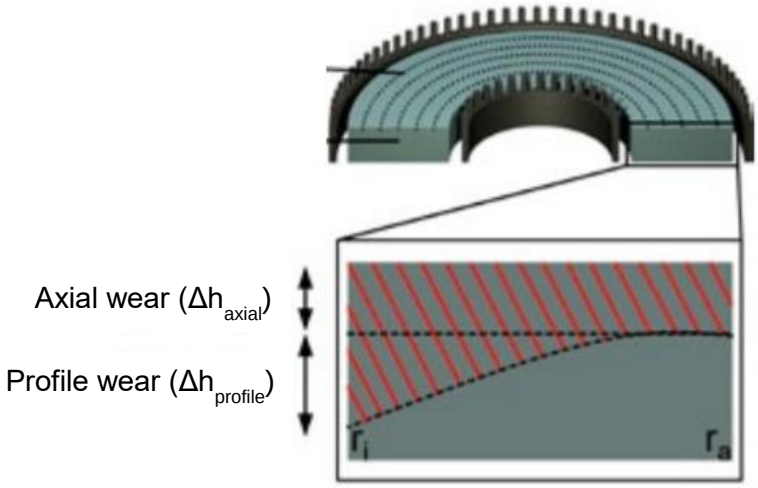

Fig. 3.6.1 Diagram of the different types of wear (4)

4. Data base design

Before identifying the important parameters of the process described in the previous chapter, it's needed to give a short explanation of the current options in the market for the DB design and management.

4.1. Software discussion

In order to create the database first is necessary to decide which type of data structure is going to be used. There are 3 main types of structures:

Structured data concerns all data which can be stored in database SQL in table with rows and columns. They have relational key and can be easily mapped into pre-designed fields. Structured data is highly organized information that uploads neatly into a relational database Structured data is relatively simple to enter, store, query, and analyse, but it must be strictly defined in terms of field name and type

Unstructured data may have its own internal structure but does not conform neatly into a spreadsheet or database. Most business interactions, in fact, are unstructured in nature. Today more than 80% of the data generated is unstructured. The fundamental challenge of unstructured data sources is that they are difficult for nontechnical business users and data analysts alike to unbox, understand, and prepare for analytic use.

Semi-structured data is information that doesn't reside in a relational database but that does have some organizational properties that make it easier to analyse. Examples of semi-structured: CSV but XML and JSON documents are semi structured documents, NoSQL databases are considered as semi structured. Semi-structured data is a form of structured data that does not conform with the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data.

Now that the data structures have been covered it is also necessary to choose which type of DB is going to be use between a relational DB and a non-relational DB. The relational DB, store of data in tables and rows, based on a branch of algebraic set theory known as relational algebra. The non-relational DB, presented in many forms (ex. Document-based, graph-based, object-based, etc.) may or may not be based on a single underpinning mathematical theory.

4.1.1. Database management systems

Considering the type of data to be processed, it has been considered the use of structured DB to be the most appropriate. Also, considering the background of this project it has been automatically discarded the use of any proprietary DB and are only considering the use of Public license DB as the following:

MySQL

MySQL (5) is a relational database management system developed by Oracle (Redwood Shores, Redwood City, California, United States, 1977) under a dual license: General Public License/Commercial License. MySQL is a widely used tool as it's fast, stable, there are developer interfaces and big amount of documentation, client-server architecture. Some companies that use MySQL are Facebook, Tesla, PayPal, GitHub to name a few.

SQLite

SQLite (6) is a relational database management system contained in a C programming library developed by Dwayne Richard Hipp. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others.

PostgreSQL

PostgreSQL (7) is an object-relational database management system with an emphasis on extensibility and standards compliance. As a database server, its primary functions are to store data securely and return that data in response to requests from other software applications. It can handle workloads ranging from small single-machine applications to large Internet-facing applications. Developed by the PostgreSQL Global Development Group, a diverse group of many companies and individual contributors.

MonetDB

MonetDB (8) is an open source column-oriented database management system developed at the Centrum Wiskunde & Informatica (Amsterdam, Netherlands, 1946). Designed to provide high performance on complex queries against large databases, such as combining tables with hundreds of columns and millions of rows. Applied in high-performance applications for online analytical processing, data mining, geographic information system, Resource Description Framework, text retrieval and sequence alignment processing.

Ingres

Ingres DB (9) is a commercially supported, open-source SQL relational database management system intended to support large commercial and government applications. Fully open source with a global community of contributors. However, Actian Corporation (Palo Alto, California, United States, 1980) controls the development of Ingres and makes certified binaries available for download, as well as providing worldwide support.

4.1.2. Database management tools

For DB management tools, as stated above, the only options considered are the Public licensed ones as the following:

TOAD

TOAD (10) is a DB management toolset from Quest Software, (Aliso Viejo, California, United States, 1987) use to manage both relational and non-relational databases using SQL. This software has allowed the reduction of the time to implement and manage the DB and also will allow future users to simple edit the DB.

SQuirreL SQL Client

SQuirreL SQL Client (11) is a database administration tool. It provides an editor that offers code completion and syntax highlighting for standard SQL. It also provides a plugin architecture that allows plugin writers to modify much of the application's behaviour to provide database-specific functionality or features that are database-independent. Free as open source software that is distributed under the GNU Lesser General Public License.

MySQL Workbench

MySQL Workbench (5), developed by Oracle, is a visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system.

DBeaver

DBeaver (12) is an SQL client and a database administration tool, provides an editor that supports code completion and syntax highlighting. It provides a plugin architecture (based on Eclipse plugins architecture) that allows to modify much of the application behaviour to provide database-specific functionality or features that are database-independent. Free as open source software that is distributed under the Apache License.

Given all the information above the use of MySQL and TOAD is believed to be the most appropriate for the following reasons:

MySQL, is easy to work with and can be installed very easily. Third-party tools, considering visual ones make it simple to get started with DB. Feature rich, supports a lot of the SQL functionality that is expected from a RDBMS. Secure, a lot of built-in security features. Scalable and powerful, can handle a lot of data and furthermore, it can be used at scale if needed. Speedy, giving up some standards allows to work very efficiently and cut corners providing speed gains. Multi-user, if needed can handle multiple clients access and use at the same time.

TOAD, syntax highlighting and autocomplete, execution plan analysis, debugging complex SQL queries, easy interaction with data, interactive tools to build out SQL, allows to make quick changes to DB and monitor process that are running on the DB.

4.2. Data Parameters

Once the process of Double Face Grinding with planetary Kinematics has been explained and it has been decided what software is going to be use, it's time to split the relevant data from the non-relevant data and to decide the divisions and categories of the data. The variables stored into the DB can have 4 types of sources, given by the machine, calculated, given by the user or stored as a file. The format to introduce the tables used is the following: Thesis Name (MySQL name)

Main process table (process)

All to be specified by the user. All tables are related to this table, making it the main table in the DB.

Name of the process

Author of the process

Date of the process

Cycle Counter, counter or number assigned by the machine for a process

Process parameters (parametersprocess)

The following parameters are given by the machine:

Rotating speed of the upper grinding wheel (N_u). Only storing the values of step 3 as it's the most relevant one.

Rotating speed of the internal pin circle (n_i)

Rotating speed of the lower grinding wheel (n_l)

Process time (tp)

The following parameters have to by specified by the user:

Distance between workpiece spot to the centre of the workpiece holder (eH) [mm]

Height difference of the workpiece (Δh) [mm], number of mm of material removed from the workpiece.

Number of workpiece holders

The following parameters are not going to be stored but rather are going to be calculated:

MRR ($\dot{h}_W$), can be calculated from the process time (tp) and the height variation (Δh)

$$\dot{h}_W = \frac{\Delta h}{tp}$$

Fig 4.2.1 MRR formula

Tool allocation (B) %

$$B = \frac{\sum A_W}{A_S} \cdot 100$$

Fig.4.2.1 Tool allocation formula (13)

A_W mm² Surface of a workpiece

A_S mm² Grinding wheel surface

Ratio of revolution inner pin/lower grinding wheel (N_L).

Ratio of revolution inner pin/upper grinding wheel (N_L').

Mechanical Model (mechanicalmodel)

Given by the machine:

Force (F) [daN]. Expressed on daN as it's the common unit for double face grinding,

Real tool wear (wear)

To be specified by the user:

Axial wear. Amount of height that the grinding wheel has lost. Fig. 3.6.1

Stored as a file:

Profile wear. Stored as X and Y values. Fig. 3.6.1

Rest of process parameters (parametersrest)

To be specified by the user:

Number of workpieces

Given from the machine:

Coolant flow rate [l/min]. Stored as Hot and Cold Flowrate. Storing the mean, min, max and standard deviation values for each.

Coolant temperature. Stored as mean, min, max and standard deviation.

Surface quality (surfacequality)

Following parameters to be measured by the user at the top and bottom of the workpiece, measuring at least 1 piece per workpiece holder

Roughness Average (Ra). Arithmetic average of the absolute values of the roughness profile. Provides a good general description of the height variations in the surface.

Average maximum height of the profile (Rz). Arithmetic mean value of the single roughness depths of consecutive sampling lengths.

Maximum height of the profile (Rt). Maximum vertical distance between the highest peak and the deepest valley.

Surface Roughness Average (Sa). The extension of Ra (arithmetical mean height of a line) to a surface. It expresses, as an absolute value, the difference in height of each point compared to the arithmetical mean of the surface.

Surface average maximum height (Sz). Arithmetic mean value of the single roughness depths of consecutive sampling lengths within a defined area.

Surface Maximum height(St). The sum of the largest peak height value and the largest pit depth value within the defined area.

Form quality (formquality)

Following parameters to be measured by the user at the top and bottom of the workpiece, measuring at least 1 piece per workpiece holder

Total thickness variation. Difference between the maximum and minimum values of the workpiece thickness.

Following parameters to be measured by the user, measuring at least 1 piece per workpiece holder

Flatness (E0). References to two parallel planes (parallel to the surface that it is called out on) that define a zone where the entire reference surface must lie

Tool Specifications (grindingwheel)

The following parameters have to by specified by the user:

Manufacturer of the grinding wheel.

Outer tool diameter (ro) [mm]

Inner tool diameter (ri) [mm]

Related to the tool specifications there are the following tables.

Type of Body (id_body)

Material of the body, aluminium or steel.

Type of grooving (id_grooving)

Types of grooving, radial, cylindrical

Type of layer (id_layer)

Type of the layer used in the grinding wheel, full layer or pallets

Information on the layer (Layer)

Grain material, material of which the grains of the grinding wheel are made of.

Grain size (dg) [μm], size of the grains.

Abrasive grain concentration [carat/mm3], concentration of grains.

Bond type, material used to hold on the abrasive grains.

Workpiece specifications (workpiece)

Following parameters to be measured by the user

Material of the workpiece

Vickers hardness, a measure of the hardness of a material, calculated from the size of an impression produced under load by a pyramid-shaped diamond indenter

Diameter (dw) [mm], using diameter as the critical measurement of the workpiece, in case of using other kind of piece use another dimension such as length of a side.

Initial height [mm] of the workpiece.

Surface [mm^2], area of the surface to be machined of one of the workpieces.

Description, user can write here a brief description of workpiece to be machined.

Coolant (coolant)

Following parameters to be specified by the user

Type of Coolant utilised

Manufacturer of the Coolant

First usage of the coolant

Machine input (process_steps)

Store the values of each step of the process as seen in the screenshot Fig. 3.1.2

Speeds are going to be stored by reading the machine file. Ramp time is to be stored by user input

Engines current (enginecurrent)

Store the value of electrical current (A) for the 3 engines. Storing the mean, min, max and standard deviation values for each.

Top Grinding Wheel

Bottom Grinding Wheel

Inner ring

4.3. Database scheme on TOAD

Visual scheme of the DB obtained from TOAD

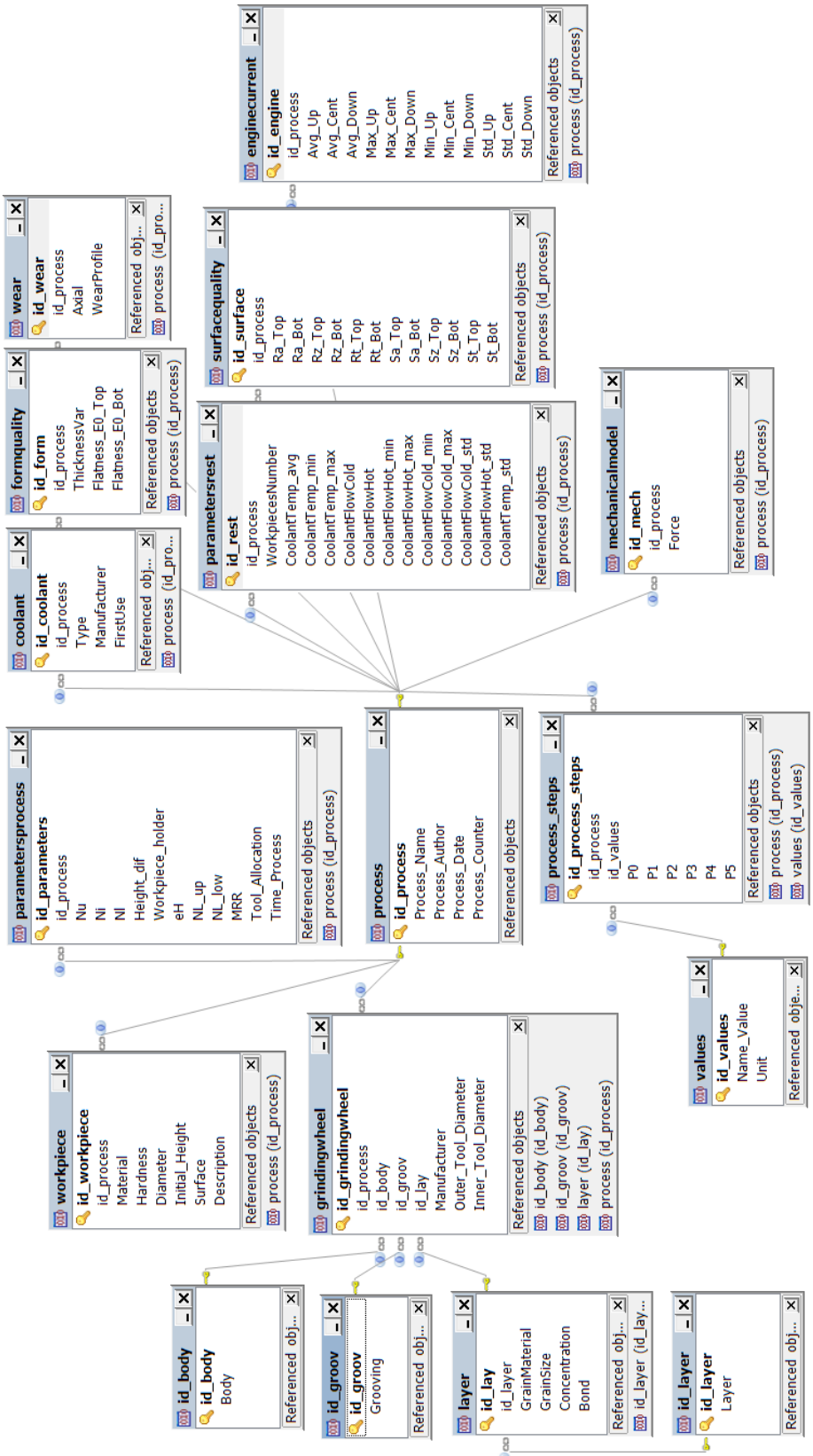

Fig. 4.3.1 Scheme of the DB

4.4. Database Discussion

First of all, regarding the parameters types if they're storing numbers they'll be generally of Float type, unless it has to be an integer number (ex. It is not possible to have 3.14 workpieces, either you have 3 or 4) if they have to store a string of text they'll be VarChar with different lengths in the range of 20 to 100 depending on the parameter, DateTime for storing dates and for storing a whole file a Blob data type is going to be used.

Having as reference the DB schema, allocated at the middle of Fig. 4.3.1, there is a key table named "process" in which we're storing the name of the process "Process_Name", the date of said process "Process_date", the author of the process "Process_Author" and the cycle counter as "Process_Counter". All are to be specified by the user, in case a date is not specified it is assumed that the process date is equal to the current date.

Regarding machine output information, those tables are "MechanicalModel", "ParametersProcess" and "ParametersRest". In "MechanicalModel" we're going to store the Force. In "Parameters" we're going to store the following data from the machine: N_u, n_i, n_l and the MRR. The distance between the workpiece "eH", the number of workpiece holder "workpiece_holder" and the height difference of the workpiece "Height_Dif" are to be specified by the user. The rest of parameters, Tool Allocation "Tool_Allocation" the revolution ratios "NL_up" and "NL_low" and process time "tp" are to be calculated from other parameters. "ParametersRest" is going to store the number of workpieces "WorkpiecesNumber", the temperature of the coolant at the entrance and exit of the machine "CoolantTempIn" and "CoolantTempOut" and the coolant flowrate "CoolantFlow". As those last three parameters are stored every 2 seconds by the machine, in the DB only will be stored the mean, maximum, minimum and standard deviation.

In the "GrindingWheel" table is to be stored the Manufacturer, the Outer Tool Diameter and the Inner Tool Diameter. Related to every "GrindingWheel" table there are three different tables. In the "id_body" tables is stored the material of the body making the user choose either aluminium or steel, in the "id_grooving" table is to be stored the type of grooving the grinding wheel has making the user choose between the different types, in the "layer" table is to be stored the data regarding the layer such as the grain material, grain concentration, grain size and type of bond and related to the "layer" table the "id_layer" containing the type of layer and making the user choose from different options.

In the "FormQuality" table is to be stored the thickness variation "ThicknessVar" of the workpiece, and the flatness measured at the top and bottom of the piece "Flatness_E0_top/bot".

In the "SurfaceQuality" table is going to be stored the different surfaces qualities, Ra, Rz, Rt, Sa, Sz and St, measured both at the top and at the bottom of the piece.

In the “Wear” table is going to be store the axial wear and the profile wear. The profile wear comes from a machine file that will be completely stored into the DB as a BLOB.

“WorkPiece” is to be stored the “Material”, “Hardness” as the vicker hardness, “Diameter” or the characteristic dimensional constrain of the workpiece, “Initial_Height” of the workpiece, “Surface” in mm^2 and a brief piece description as text made by the user “Description”.

“Coolant” is to be stored the type of coolant used “Type”, the “Manufacturer” and the first of use of the coolant as “FirstUse” in case the user doesn’t specify a date, the first use will be set to the current date.

In the “EngineCurrent” it’s going to be stored the electrical current (A) of the engines for the three different engines of the machine. As the current data comes from the machine files and is stored every 2 seconds it will only be stored as mean, maximum, minimum and standard deviation.

For every “process” there is a “process_steps”. In the “process_steps” there are the 6 steps the user inputs into the machine as shown in the Fig. 3.1.2. The “process_steps” table is related with the “values” table, fig 4.4.2, in that table it’s contained the “Name_Value” and “Unit”, which refer to the Fig.3.1.2 as “Name_Value” = Force and “Unit” = daN, and so on. Therefore, on the table “process_steps” it can be stored the process input data by the user as shown in Fig. 4.4.1.

id_process_steps *	id_proc...	id_values	P0	P1	P2	P3	P4	P5
239	500	1	100	50	90	300	90	50
240	500	6	0	0	-200	-746	-200	0
241	500	8	0	0	-100	-300	-100	0
242	500	10	0	0	200	600	200	0
247	500	2	2	3	4	6	3	1
248	500	4	3	5	5	0	0	3
249	500	7	0	0	4	4	3	0
250	500	9	0	0	4	4	3	0
251	500	11	0	0	4	4	3	0
252	500	5	0	0	0	0	0	0

Fig. 4.4.1. Data stored in “process_steps” table with id_process = 500

id_values *	Name_Value	Unit
1	Force	daN
2	Force	*
3	Stop at	um/*
4	Work time	*
5	Rate	um/*
6	Upper	rpm
7	Upper	*
8	Centre	rpm
9	Centre	*
10	Lower	rpm
11	Lower	*

Fig 4.4.2 “values” table

Then it is possible to retrieve the data as shown in Fig.4.4.3 using the following query:

```
SELECT `values`.Name_Value,
       process_steps.P0,
       process_steps.P1,
       process_steps.P2,
       process_steps.P3,
       process_steps.P4,
       process_steps.P5,
       `values`.Unit
FROM `double face grinding`.process_steps process_steps
     JOIN `double face grinding`.`values` `values`
        ON (process_steps.id_values = `values`.id_values)
WHERE process_steps.id_process = 500

ORDER BY `values`.id_values ASC
```

Fig. 4.4.2 Query to retrieve data from table “process_steps”

Name_Value	P0	P1	P2	P3	P4	P5	Unit
Force	100	50	90	300	90	50	daN
Force	2	3	4	6	3	1	*
Work time	3	5	5	0	0	3	*
Rate	0	0	0	0	0	0	um/*
Upper	0	0	-200	-746	-200	0	rpm
Upper	0	0	4	4	3	0	*
Centre	0	0	-100	-300	-100	0	rpm
Centre	0	0	4	4	3	0	*
Lower	0	0	200	600	200	0	rpm
Lower	0	0	4	4	3	0	*

Fig. 4.4.3 Data Obtained

For the “process_steps” table the force values in [daN] and the speeds in [rpm] are stored via reading the text file from the machine while the ramp times in seconds [*] are to be specified by the user.

5. Importing data into the Database

Now that the DB has been created and all the parameters have been defined it is time to import data into the DB. As mentioned in previous chapters there are four kinds of data regarding the source.

- Acquired from the machine via text file.
- Acquired from the user via console
- Acquired from calculations from other data parameters.
- Acquired via storing a file.

Python (Python Software Foundation, 2001) (14) in its version 3.4 is going to be used to create a program in order to acquire data from the text files, asking the user or calculation. Even though that version 3.4 is outdated this version has a MySQL connector, it is with said connector that is possible to send the data from Python to MySQL DB.

Python 3.4 does not provide with a graphic interface, in order to help with the development of the project an Integrated Development Environment (IDE) is going to be used. The chosen one is PyCharm (15) (Prague, Czech-Republic, 2000) developed by JetBrains, it presents a free community open-source version.

5.1. Software Discussion

There are different IDE options in the current market that could fit this project requirements.

Atom

Developed by the social-coding network/community GitHub. Atom has its own comprehensive package manager, and a huge community working on it, as well as built-in Git and GitHub integration, allowing to collaborate on coding projects in real-time via the Teletype package.

- Pros: Clean, smart interface, comprehensive package manager.
- Cons: Sizable memory footprint.

IDLE

Developed by Python (13) is fairly minimal compared to others but offers everything you need. Coded in Python language and uses a lightweight tkinter toolkit to draw its Graphical User Interface (GUI). IDLE lacks any project management facilities.

- Pros: Lightweight, powerful debugger.
- Cons: No project management capability

Thonny

Developed at the University of Tartu in Estonia and is itself written in Python language, features a powerful debugger great for learning the ins and outs of coding without worrying about how breakpoints work. Installation is non-user-friendly unless you are familiar with Python packages and in particular pip.

- Pros: Excellent debugger, ideal for novices, included as standard in new Raspbian versions.
- Cons: Installation

PyCharm

PyCharm (14) (Prague, Czech-Republic, 2000) developed by JetBrains, it presents a free community open-source version. Has a powerful real-time debugger, intelligent code editor that provides first-class support for Python, smart code navigation, fast and safe refactorings.

- Pros: Powerful debugger, ideal for novices.
- Cons: Not Lightweight compared to others but bearable size.

5.2. External packages or imports needed

In order to develop the python program some built-in modules have been imported, those modules are the following. CSV module in order to process and/or navigate through the text files, DATETIME module to insert dates in date format rather than a string, OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux, RE this module provides regular expression matching operations similar to those found in Perl. Both patterns and strings to be searched can be Unicode strings as well as 8-bit strings, STATISTICS module in order to calculate the mean and standard deviation, ITERTOOLS module used to skip the header on the text files, MATH module to import PI number.

Also, it has been needed to install the Mysql.Connector external package in order to connect the Python program with the DB.

5.3. Architecture of the program

The python program has been splitted into several small programs or routines. By doing two things are accomplished, it is easier to develop the project further and it is easier to navigate through the code as rather than one large file there are different smaller files.

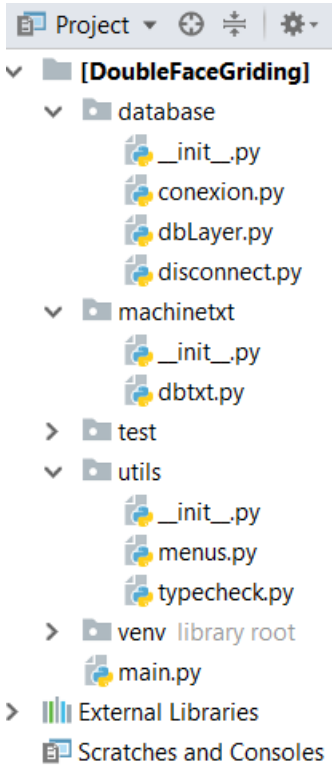

Fig. 5.3.1 Screenshot of the architecture of the python program

In the database module, the connexion.py file opens the connexion to the DB, the disconnect.py sends information to the DB and closes the connexion to it, the dbLayer.py contains the part of the program that requests information from the user and sends this information to the DB in a format that the MySQL DB can understand, in order to do so the dbLayer.py uses the conexion.py and disconnect.py functions. In the machinetxt module, the dbtxt.py contains the machine files processing part from their reading to the calculations of mean and standard deviation when needed, sends the information to the DB using the conexion.py and disconexion.py files. In the utils module, the menus.py contains the menus used in the program, the typecheck.py contains a function that prevents the user from inputting into an integer parameter and floating type or viceversa. The main.py file is the first file to be executed, shows the first menu that the user can interact with, allowing the user to start using the other modules.

5.4. Text files to process

As previously mentioned the Double Face Grinding machine outputs two text files. The files are names: Infocy.txt and InfoIstn.txt being n a number. The main difference between both files is that in InfoIst file information is stored every two seconds therefore the information found in those filetypes is the information that changes through time in the same process, such as the coolant flow or the engine current. In the InfoCy file the information found is the one that remains constant through time within the same process such as the speeds or loads at different steps. Processes can be related with the Number of Cycles (A_CYCL_COUNTER) and in case there is more than one process with the same Number of Cycles the date of the process can be used.

There is also another file that comes from the wear measurement machine. This type of file comes in .lvm extension and contains the wear profile of the grinding wheel. This file will be completely stored in the DB as a blob.

InfoIst.txt contains the following sets of data for each row stored:

Date; in format dd.mm.yyyy hh:mm:ss →11.01.2017 14:54:49

A_TIM_TOT_MIN; ?? → 0 ; Total time in min

A_TIM_TOT_SEC; ?? → 1 ; Total time in seconds

A_CYCL_COUNTER ?? → 128 ; Number of cycles

A_ACTIV_STEP; current step of the process → 0

A_CORRECTION ?? → 60580

A_MEASVALUE_AVERAGE ?? → 60580

A_MEAS_TAU_M1 → 0

A_LOAD_REAL_OP → 0

MESSBOLZEN_KRAFT_DAN → 26

A_SPEED_UP_IST_O → 0

A_SPEED_CENT_IST_O → 0

A_SPEED_DOWN_IST_O → 0

JOG_LOAD → 100

JOG_FB1 → 100

DURCHFLUSSMESSER_SPULUNG → 0 ; Flowrate uncooled /10 l/min

DURCHFLUSSMESSER_KUHLUNG →0 ; flowrate cooling /10 l/min

A_F_EXIT_AUTO → Falsch

TEMP_ZUFLUSS →207 → Coolant temp input in C · 10

TEMP_ABFLUSS → 201→ Coolant temp output in C · 10

Res → ;

A_COURENT_PLATE_UP → 9 → Engine current upper plate

A_COURENT_PLATE_CENTER → 1→ Engine current inner pin ring

A_COURENT_PLATE_DOWN → 2→ Engine current down plate

TRAVERSE_IST_POS_ANZ →6323

InfoCy.txt contains the following sets of data for each row stored:

Z_Text; → ;

NOM_ENREGIST; → schleudern → Name of the process

Date; → 11.01.2017 15:08:02

A_CYCL_COUNTER; → 128 → Number of cycles , process number

A_TIM_TOT_MIN; → 0

A_TIM_TOT_SEC; →10

Z_MEAS_VALUE; →60560

A_CORRECTION;→60560

A_MEAS_TAU_M1;→0

R_LOAD_0;→0 → Force at step 0

R_SPEED_UP_1;→400 → Speed at step 1 Upper grinding wheel

R_SPEE_CENT_1;→0 → Speed at step 1 inner pin ring

R_SPEE_DOWN_1;→ -400 → Speed at step 1 Lower grinding wheel

R_LOAD_1;→0 → Force at step 1

R_SPEED_UP_2;→ 0 → Speed at step 2 Upper grinding wheel

R_SPEE_CENT_2;→0→ Speed at step 2 inner pin ring

R_SPEE_DOWN_2;→0 Speed at step 1 Lower grinding wheel

R_LOAD_2;→0 → Force at step 2

R_SPEED_UP_3;→0 → Speed at step 3 Upper grinding wheel

R_SPEE_CENT_3;→0 → Speed at step 3 inner pin ring

R_SPEE_DOWN_3;→0 → Speed at step 3 Lower grinding wheel

R_LOAD_3; →0 → Force at step 3

R_SPEED_UP_4; →0 → Speed at step 4 Upper grinding wheel

R_SPEE_CENT_4; →0 → Speed at step 4 inner pin ring

R_SPEE_DOWN_4; →0 → Speed at step 4 Lower grinding wheel

R_LOAD_4; →0 → Force at speed 4

R_SPEED_UP_5; →0 → Speed at step 5 Upper grinding wheel

R_SPEE_CENT_5; →0 → Speed at step 5 inner pin ring

R_SPEE_DOWN_5; →0 → Speed at step 5 Lower grinding wheel

R_LOAD_5; →0 → Force at step 5

Z_FORCE_MAX;18

Z_meas_Raw_mini;→-42430

R_OPT_LOAD;→Falsch

R_P_FACTOR_UP;→0

R_P_FACTOR_DOWN;→0

R_DIR_CYC_CHANGE;→0

R_OPT_UPPER_CY;→Falsch

R_OPT_CENTER_CY; →Falsch

R_OPT_LOWER_CY; →Falsch

E_PART_THICKN;→500

E_PARTS_THICK_CTRL;→0

R_RATE_1; →0

R_RATE_2; →0

R_RATE_3; →0

R_RATE_4; →0

R_RATE_5; →0

A_LOAD_MAX_ABS; →0

R_TIMER_TAUX_M; →0

R_TIME_CTRL_TAUX→0

5.5. How does the program work?

When the main.py is opened a function named main() is called, that function shows a three-option menu. The options given to the user are, register a new process, modify an existent process or exit the program.

```
Select one of the following options:
    1 - New process
    2 - Update process
    E - Exit
Insert a number >>
```

Fig. 5.5.1 Screenshot of the main menu

If the user selects the register a new process option, a function contained in the dbLayer.py is called, the user has to input the Date, Process Name, Process Author and the Cycle Counter of that process. In case the Date is not inputted by the user it will be set to the current Date. After that a new menu is printed, in that menu the user can select from 11 different options which parameters to be registered.

If the user selects to modify and existent process it is mandatory to input the MySQL Id of the process to be modified, that way it is possible to retrieve the data of the process from the MySQL. After the user inputs an existent id of the process, the Counter, Name and Date of the process are printed, if user presses any the menu with 11 different options shows up, if user leaves blank it will go back to the main menu.

The 11 different options for the user are the following: 1 - Insert Height Difference, Number of Workpiece Holder and eH, 2 - Insert Number of Workpieces, 3 - Insert Grinding Wheel Parameters, 4 - Insert Form Quality Parameters, 5 - Insert Workpiece Parameters, 6 - Insert Axial Wear, 7 - Insert Surface Quality Parameters, 8 - Insert Coolant Parameters, 9 - Read and import data from a .txt, 10 - Read and import wear data from a .txt, 11 - Process input at the machine system.

```
Select one of the following options:
    1 - Insert HeightDif, Number of Workpiece Holder and eH
    2 - Insert Number of Workpieces
    3 - Insert Grinding Wheel Parameters
    4 - Insert Form Quality Parameters
    5 - Insert Workpiece Parameters
    6 - Insert Axial Wear
    7 - Insert Surface Quality Parameters
    8 - Insert Coolant Parameters
    9 - Read and import data from a .txt
    10 - Read and import wear data from a .txt
    11 - Process input at the machine system
    E - Exit
Insert a number >>
```

Fig. 5.5.2 Screenshot of the secondary menu

A complete review of the eleven different options is not necessary since from option 1 to 8 and option 11 are the same types of functions or said in another way they work in the same way. The user has to input the specified data on the console, that is for example in the option number 1- the Height Difference in [mm], the Number of Workpiece Holders and the eH in [mm]. When the user completes all the input information the function checks if there is already Data in the destination table for that process. If the destination table is empty then the information is sent to the DB using an INSERT function, in case there is already Data in the destination table an UPDATE function is used.

Option number 3 is slightly different to the other ones as it contains a menu inside it. Once the user does the initial information about the manufacturer and the grinding wheels dimensions, it will show another menu. This menu allows the user to input further data regarding the Layer, Body or Grooving.

```
Select one of the following options:
    1 - Insert Body>>
    2 - Insert Layer Information>>
    3 - Insert Grooving>>
    E - Exit>>
Select one of the following options: >>
```

Fig. 5.5.3 Screenshot of the Grinding Wheel menu

Option number 9 is set apart from the rest of functions as

it is stored in the dbtxt.py file. When this function is called it first retrieves the Cycle Counter the user has input, then asks the user to input the path as D:\Data\, in where the Infoist.txt and InfoCy.txt files are stored. First the InfoCy.txt is processed, from the text file the Name, Date and Counter are stored. The Date gets converted from a string format to a real Date format. Because there is some process that in the name uses the.txt column separator, in this case it is the semicolon, when converting the string date to datetime it gives error, so that when this happens, those processes are ignored by the program.

After the program has finished reading all the rows in the InfoCy.txt there are 3 scenarios. There are no process with the same counter the user has input, in this case the program prints that there are no processes with the counter and returns the user to the previous menu. If there is only on process with the same counter then the program keeps running. In case there are two or more processes with the same counter the user has to delete which ones not to process, in order to do so the program prints the name and dates of the processes to help the user.

Once the process is correctly located, it proceeds to process it. First, it opens again the InfoCy.txt and having located the process with the name with the cycle counter proceeds to save the rest of relevant parameters such as the forces and speeds in the different steps of the process for the upper grinding wheel, lower grinding wheel and the inner pin ring, as well as save the time of the process.

To process the InfoIst.txt it is necessary to open all the files named InfoIst because it is not possible to know a priori in which of the different files the process of interest will be stored. To find it, both the Cycle Counter and the date the process in which was performed are compared. Once the process is located, data is stored on the flowrate, lubricant temperature and motor current. But the data is stored every two seconds so for a process it may give a fairly big amount as in for a short process there are around 380 pairs of not so relevant data. In order to store the data in an efficient way and that it still has a physical meaning, only the maximum, minimum.

After all data from the machine files is stored into variables within the python program, the same process as before is used the function checks if there is already Data in the destination table. If the destination table is empty then the information is sent to the DB using an INSERT function, in case there is already Data in the destination table an UPDATE function is used.

Option number 10, read and import wear data from a .txt is a fairly simple function, given a path and file by the user as D:\Data\wear.txt, the function opens the file, stores the file as a variable and then sends the variable to the wear table in MySQL. If the Axial wear has been stored in the wear table, an UPDATE function is used to store the profile, otherwise a INSERT function is used.

Option number 11, process input at the machine system, is a function that aims to recreate Fig. 3.1.2. The user will be asked about the ramp times as the speeds and forces are stored in the txt file and are stored in their corresponding tables in the option number 9. If option number 9 has already been run the time ramps are stored using an UPDATE function otherwise an INSERT function is used.

5.6. Database operation

The current status of the database presented in this project is functional to some extent, but impractical due to the lack of a GUI. However, in this section several queries are presented in order to recover information from the database, so that even if there is no GUI the DB is in a minimally functional state. As all queries have more or less the same shape, that is a SELECT statement with the parameters a FROM statement with the database and the tables where the parameters are store and then a WHERE to only select one set of data.

As shown in Chaptor 4, with a given process id this query retrieves the process_steps with the process name:

```
SELECT process.Process_Name,
       `values`.Name_Value,
       process_steps.P0,
       process_steps.P1,
       process_steps.P2,
       process_steps.P3,
       process_steps.P4,
       process_steps.P5,
       `values`.Unit
FROM `double face grinding`.process_steps process_steps
     JOIN `double face grinding`.`values` `values`
        ON (process_steps.id_values = `values`.id_values)
     JOIN process
        ON process.id_process = process_steps.id_process
WHERE process_steps.id_process = 500
ORDER BY `values`.id_values ASC
```

Name_Value	P0	P1	P2	P3	P4	P5	Unit
Force	100	50	90	300	90	50	daN
Force	2	3	4	6	3	1	*
Work time	3	5	5	0	0	3	*
Rate	0	0	0	0	0	0	um/*
Upper	0	0	-200	-746	-200	0	rpm
Upper	0	0	4	4	3	0	*
Centre	0	0	-100	-300	-100	0	rpm
Centre	0	0	4	4	3	0	*
Lower	0	0	200	600	200	0	rpm
Lower	0	0	4	4	3	0	*

Fig. 5.6.1 Process Steps data retrieved

With a given process id this query retrieves all the grinding wheel related information, including the body, grooving and layer types.

```
SELECT id_grindingwheel
  , process.Process_Name
  , id_body.Body
  , id_groov.Grooving
  , id_layer.Layer
  , layer.GrainMaterial
  , layer.GrainSize
  , layer.Concentration
  , layer.Bond
  , Manufacturer
  , Outer_Tool_Diameter
  , Inner_Tool_Diameter
FROM grindingwheel
  JOIN process
    ON grindingwheel.id_process = process.id_process
  LEFT JOIN id_body
    ON id_body.id_body = grindingwheel.id_body
  LEFT JOIN id_groov
    ON id_groov.id_groov = grindingwheel.id_groov
  LEFT JOIN layer
    ON layer.id_lay = grindingwheel.id_lay
  LEFT JOIN id_layer
    ON id_layer.id_layer = layer.id_layer
WHERE grindingwheel.id_process = 293
```

Set 1

id_grindingwheel *	Process_Name	Body	Grooving	Layer	GrainMaterial
56	test	Aluminum	Full	Pellet	4

GrainSize	Concentration	Bond	Manufacturer	Outer_Tool_Diameter	Inner_Tool_Diameter
3	5	6	test	12	13

Fig. 5.6.2 Grinding wheel data retrieved

Query to retrieve all the processes, including the process name, author date and counter within two dates in format yyyyMMdd.

```
SELECT id_process, Process_Name, Process_Author, Process_Date,
Process_Counter
FROM process
WHERE Process_Date BETWEEN '20180501' AND '20180619';
```

6. Next steps

As mentioned in the previous chapter the database is currently in a minimally functional state. What does this mean? This means that it is possible to insert processes and edit the processes by giving the process id. Although it is to be expected that the insertion of data in the DB does not vary much as the project progresses, the editing or updating of the tables is to be expected to undergo drastic changes, currently to update a single parameter of a table it is necessary to re-insert the rest of the parameters of that table, this is quite annoying for the user in the case of having to update some large table, for example in the case of wanting to update the table of Surface Quality, although only one parameter is to be changed it will be necessary to insert the 12 contained in that table.

The next steps to the further development of the DB should be the creation of a functional update/editing system so that the user can easily change the parameters of the table and delete the processes as desired. Also, to make life easier for the user it is necessary to create a GUI, most popular libraries that implement GUI in python are Tkinter based on TCL/TK, WxPython based on WxWidgets a C/C++ multiplatform library, PyQT based on C++QT or PyGTK based on C GTK. Each has its strengths and drawbacks, so let's leave it to the next developer to choose.

Summary and conclusions

In this project the first steps have been taken towards the implementation of a functional data storage system for the double face grinding process with planetary kinematics. When this system is fully implemented it will be possible not only to conduct research based on the stored data but also to implement machine learning or deep learning tools.

The first steps of this project were to determine the parameters of the process that were interesting enough to be stored, based on published papers as well as previously acquired knowledge. Once it was determined what was to be stored in the DB, it was possible to proceed with the design of the DB, in which MySQL was chosen as it is a structured and relational DB model.

Once the database had been created, the files coming from the machine were examined, so that the parameters that have been considered relevant but are not found in those files will be requested from the user per console. Both to process the files and to request data from the user, it has been decided to use Python in version 3.4, although there are now later versions, this version does have a package to connect to the database, which the other versions lack.

At this time the program allows the user to read the machine files and insert the rest of the information manually by console, saving it in the completely new in-process database, as well as updating an existing process already in the database as long as the user knows the id of this process.

Although the database is in a functional state, it is far from user-friendly. The main drawbacks are the lack of a graphical interface for the user to interact comfortably, to be able to delete processes that have been inserted in the database and to improve the data update system, since currently to change a value in a table the rest of the values contained in that table must be reinserted.

Bibliography

1. **Uhlmann , E., Hoghé , T. and Kleinschnitker, M.** Grinding wheel wear prediction at double face grinding. *Int J Adv Manuf Technol.* 2013, Vol. 69, 2315-2321.

2. **Uhlmann, E. and Ardelt, Th.** Influence of kinematics. *Annals of the CIRP 48.* 1999, Vol. 1, 281-284.

3. **Uhlmann, E. and Hoghé, T.** Wear reduction at double face grinding with planetary kinematics. *Production Engineering.* 2012, Vol. 6, 237-242.

4. **Uhlmann, E., Kleinschnitker, M. and Hoghé, T.** Optimierungspotenziale beim Doppelseitenplanschleifen. *Jahrbuch Schleifen Honen Läppen Polieren.* 2016, Vol. 67, 67-77.

5. **MySQL.** [Online] Oracle. [Cited: 05 14, 2018.] https://www.mysql.com/.

6. **SQLite.** [Online] SQLite Consortium. [Cited: 05 14, 2018.] https://www.sqlite.org/index.html.

7. **PostgreSQL.** [Online] [Cited: 05 14, 2018.] https://www.postgresql.org/.

8. **MonetDB.** [Online] Centrum Wiskunde & Informatica. [Cited: 05 14, 2018.] https://www.monetdb.org/.

9. **Ingres DB.** [Online] Actian. [Cited: 05 14, 2018.] https://www.actian.com/data-management/ingres-sql-rdbms/.

10. **TOAD.** [Online] Quest. [Cited: 05 14, 2018.] https://www.quest.com.

11. **SQuirrel SQL.** [Online] [Cited: 05 14, 2018.] http://squirrel-sql.sourceforge.net/.

12. **DBeaver.** [Online] [Cited: 05 14, 2018.] https://dbeaver.io/.

13. **Ardelt, Thomas.** *Einfluss der Relativbewegung auf den Prozess und das Arbeitsergebnis beim Planschleifen mit Planetenkinematik.* Berlin : Berichte aus dem Produktionstechnischen Zemtrum Berlin, 2000.

14. **Python Software Foundation.** [Online] 2001. [Cited: 05 17, 2018.]

15. **JetBrain.** *https://www.jetbrains.com/pycharm/.* [Online] 2000. [Cited: 05 16, 2018.]

16. **Python Software Foundation.** *https://www.python.org/psf/.* [Online] 2001. [Cited: 05 14, 2018.]

Annex A: Queries to retrieve information from the DB

As seen in chapter 5.6 here there are some more queries to retrieve information.

With a given id process this query retrieves both the information of the workpiece and the information stored in the process parameters table.

```
SELECT process.Process_name
  ,parametersprocess.*
  ,workpiece.*
FROM process
  LEFT JOIN parametersprocess
    ON process.id_process = parametersprocess.id_parameters
  LEFT JOIN workpiece
    ON process.id_process = workpiece.id_process
WHERE process.id_process=492
```

With a given process id this query retrieves the process name and all the information in the parameters rest table.

```
SELECT process.Process_Name,
       parametersrest.WorkpiecesNumber,
       parametersrest.CoolantTemp_avg,
       parametersrest.CoolantTemp_min,
       parametersrest.CoolantTemp_max,
       parametersrest.CoolantTemp_std,
       parametersrest.CoolantFlowCold,
       parametersrest.CoolantFlowHot,
       parametersrest.CoolantFlowHot_min,
       parametersrest.CoolantFlowHot_max,
       parametersrest.CoolantFlowCold_min,
       parametersrest.CoolantFlowCold_max,
       parametersrest.CoolantFlowCold_std,
       parametersrest.CoolantFlowHot_std
FROM `double face grinding`.parametersrest parametersrest
     INNER JOIN `double face grinding`.process process
        ON (parametersrest.id_process = process.id_process)
WHERE process.id_process = 500
```

Annex B: Deleting data in the DB

```
DESCRIPCIÓN: SQL script to erase all data in 'double face grinding' DB.
Deletes data from all tables except for master tables or the following:
id_body, id_grooov, id_layer, values. Then we re-establish all
autoincrements to one.

-- Delete all tables that depend directly from process table
DELETE FROM coolant;
DELETE FROM enginecurrent;
DELETE FROM formquality;
DELETE FROM grindingwheel;
DELETE FROM mechanicalmodel;
DELETE FROM parametersprocess;
DELETE FROM parametersrest;
DELETE FROM process_steps;
DELETE FROM surfacequality;
DELETE FROM wear;
DELETE FROM workpiece;

-- Layer acts as a process but only for grindingwheel
DELETE FROM layer;

-- Delete process as it's the main table
DELETE FROM process;

-- Re-establish autoincrements to one
ALTER TABLE coolant              AUTO_INCREMENT=1;
ALTER TABLE enginecurrent        AUTO_INCREMENT=1;
ALTER TABLE formquality          AUTO_INCREMENT=1;
ALTER TABLE grindingwheel        AUTO_INCREMENT=1;
ALTER TABLE mechanicalmodel      AUTO_INCREMENT=1;
ALTER TABLE parametersprocess    AUTO_INCREMENT=1;
ALTER TABLE parametersrest       AUTO_INCREMENT=1;
ALTER TABLE process_steps        AUTO_INCREMENT=1;
ALTER TABLE surfacequality       AUTO_INCREMENT=1;
ALTER TABLE wear                 AUTO_INCREMENT=1;
ALTER TABLE workpiece            AUTO_INCREMENT=1;
ALTER TABLE layer                AUTO_INCREMENT=1;

ALTER TABLE process              AUTO_INCREMENT=1;
```

Annex C: Generating a DB with same scheme

```
DESCRIPCIÓN: We generate a new database with the name of schema that will
assing a `newDataBaseGrinding` (can change in the script) preconfigurate
tables with set values in them: id_body, id_, id_layer, values. Used the
workbench script as layer leaving the comments on it as
they have useful information in them

CREATE DATABASE  IF NOT EXISTS newDataBaseGrinding /*!40100 DEFAULT
CHARACTER SET utf8 */;
USE newDataBaseGrinding;


-- MySQL dump 10.13  Distrib 5.7.17, for Win64 (x86_64)
--
-- Host: localhost    Database: double face grinding
-- ------------------------------------------------------
-- Server version 5.7.21-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO'
*/;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `coolant`
--

DROP TABLE IF EXISTS `coolant`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `coolant` (
  `id_coolant` int(20) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `Type` varchar(80) CHARACTER SET latin1 DEFAULT NULL,
  `Manufacturer` varchar(80) CHARACTER SET latin1 DEFAULT NULL,
  `FirstUse` date DEFAULT NULL,
  PRIMARY KEY (`id_coolant`),
  KEY `FK_process_coolant` (`id_process`),
  CONSTRAINT `FK_process_coolant` FOREIGN KEY (`id_process`) REFERENCES
`process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `coolant`
--

LOCK TABLES `coolant` WRITE;
```

```
/*!40000 ALTER TABLE `coolant` DISABLE KEYS */;
/*!40000 ALTER TABLE `coolant` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `enginecurrent`
--

DROP TABLE IF EXISTS `enginecurrent`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `enginecurrent` (
  `id_engine` int(20) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `Avg_Up` float DEFAULT NULL,
  `Avg_Cent` float DEFAULT NULL,
  `Avg_Down` float DEFAULT NULL,
  `Max_Up` float DEFAULT NULL,
  `Max_Cent` float DEFAULT NULL,
  `Max_Down` float DEFAULT NULL,
  `Min_Up` float DEFAULT NULL,
  `Min_Cent` float DEFAULT NULL,
  `Min_Down` float DEFAULT NULL,
  `Std_Up` float DEFAULT NULL,
  `Std_Cent` float DEFAULT NULL,
  `Std_Down` float DEFAULT NULL,
  PRIMARY KEY (`id_engine`),
  KEY `FK_Engine_Process` (`id_process`),
  CONSTRAINT `FK_Engine_Process` FOREIGN KEY (`id_process`) REFERENCES
`process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `enginecurrent`
--

LOCK TABLES `enginecurrent` WRITE;
/*!40000 ALTER TABLE `enginecurrent` DISABLE KEYS */;
/*!40000 ALTER TABLE `enginecurrent` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `formquality`
--

DROP TABLE IF EXISTS `formquality`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `formquality` (
  `id_form` int(11) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `ThicknessVar` float DEFAULT NULL COMMENT ' max height - min height',
  `Flatness_E0_Top` float DEFAULT NULL,
  `Flatness_E0_Bot` float DEFAULT NULL COMMENT ' max height - min
height',
  PRIMARY KEY (`id_form`),
  KEY `FK_Form_Process` (`id_process`),
```

```
  CONSTRAINT `FK_Form_Process` FOREIGN KEY (`id_process`) REFERENCES
`process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `formquality`
--

LOCK TABLES `formquality` WRITE;
/*!40000 ALTER TABLE `formquality` DISABLE KEYS */;
/*!40000 ALTER TABLE `formquality` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `grindingwheel`
--

DROP TABLE IF EXISTS `grindingwheel`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `grindingwheel` (
  `id_grindingwheel` int(11) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `id_body` int(11) DEFAULT NULL,
  `id_groov` int(11) DEFAULT NULL,
  `id_lay` int(11) DEFAULT NULL,
  `Manufacturer` varchar(40) CHARACTER SET latin1 DEFAULT NULL,
  `Outer_Tool_Diameter` float DEFAULT NULL,
  `Inner_Tool_Diameter` float DEFAULT NULL,
  PRIMARY KEY (`id_grindingwheel`),
  KEY `FK_GrindingWheel_idbody` (`id_body`),
  KEY `FK_GrindingWheel_idgroov` (`id_groov`),
  KEY `FK_GrindingWheel_layer` (`id_lay`),
  KEY `FK_process_GrindingWheel` (`id_process`),
  CONSTRAINT `FK_GrindingWheel_idbody` FOREIGN KEY (`id_body`) REFERENCES
`id_body` (`id_body`),
  CONSTRAINT `FK_GrindingWheel_idgroov` FOREIGN KEY (`id_groov`)
REFERENCES `id_groov` (`id_groov`),
  CONSTRAINT `FK_GrindingWheel_layer` FOREIGN KEY (`id_lay`) REFERENCES
`layer` (`id_lay`),
  CONSTRAINT `FK_process_GrindingWheel` FOREIGN KEY (`id_process`)
REFERENCES `process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `grindingwheel`
--

LOCK TABLES `grindingwheel` WRITE;
/*!40000 ALTER TABLE `grindingwheel` DISABLE KEYS */;
/*!40000 ALTER TABLE `grindingwheel` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `id_body`
--
```

```
DROP TABLE IF EXISTS `id_body`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `id_body` (
  `id_body` int(20) NOT NULL AUTO_INCREMENT,
  `Body` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`id_body`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `id_body`
--

LOCK TABLES `id_body` WRITE;
/*!40000 ALTER TABLE `id_body` DISABLE KEYS */;
INSERT INTO `id_body` VALUES (1,'Aluminum'),(2,'Steel');
/*!40000 ALTER TABLE `id_body` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `id_groov`
--

DROP TABLE IF EXISTS `id_groov`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `id_groov` (
  `id_groov` int(11) NOT NULL AUTO_INCREMENT,
  `Grooving` varchar(40) DEFAULT NULL,
  PRIMARY KEY (`id_groov`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `id_groov`
--

LOCK TABLES `id_groov` WRITE;
/*!40000 ALTER TABLE `id_groov` DISABLE KEYS */;
INSERT INTO `id_groov` VALUES
(1,'Radial'),(2,'Pelletised'),(3,'Spiral'),(4,'Full');
/*!40000 ALTER TABLE `id_groov` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `id_layer`
--

DROP TABLE IF EXISTS `id_layer`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `id_layer` (
  `id_layer` int(11) NOT NULL AUTO_INCREMENT,
  `Layer` varchar(40) DEFAULT NULL,
  PRIMARY KEY (`id_layer`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `id_layer`
--

LOCK TABLES `id_layer` WRITE;
/*!40000 ALTER TABLE `id_layer` DISABLE KEYS */;
INSERT INTO `id_layer` VALUES (1,'Full Layer'),(2,'Pellet');
/*!40000 ALTER TABLE `id_layer` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `layer`
--

DROP TABLE IF EXISTS `layer`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `layer` (
  `id_lay` int(11) NOT NULL AUTO_INCREMENT,
  `id_layer` int(11) DEFAULT NULL,
  `GrainMaterial` varchar(40) DEFAULT NULL,
  `GrainSize` float DEFAULT NULL,
  `Concentration` float DEFAULT NULL,
  `Bond` varchar(40) DEFAULT NULL,
  PRIMARY KEY (`id_lay`),
  KEY `FK_layer_idlayer` (`id_layer`),
  CONSTRAINT `FK_layer_idlayer` FOREIGN KEY (`id_layer`) REFERENCES
`id_layer` (`id_layer`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `layer`
--

LOCK TABLES `layer` WRITE;
/*!40000 ALTER TABLE `layer` DISABLE KEYS */;
/*!40000 ALTER TABLE `layer` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `mechanicalmodel`
--

DROP TABLE IF EXISTS `mechanicalmodel`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `mechanicalmodel` (
  `id_mech` int(11) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `Force` float DEFAULT NULL,
  PRIMARY KEY (`id_mech`),
  KEY `FK_Mech_Process` (`id_process`),
  CONSTRAINT `FK_Mech_Process` FOREIGN KEY (`id_process`) REFERENCES
`process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `mechanicalmodel`
--

LOCK TABLES `mechanicalmodel` WRITE;
/*!40000 ALTER TABLE `mechanicalmodel` DISABLE KEYS */;
/*!40000 ALTER TABLE `mechanicalmodel` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `parametersprocess`
--

DROP TABLE IF EXISTS `parametersprocess`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `parametersprocess` (
  `id_parameters` int(11) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `Nu` float DEFAULT NULL COMMENT 'Rotating speed of the upper grinding
wheel',
  `Ni` float DEFAULT NULL COMMENT 'Rotating speed of the internal pin
circle',
  `Nl` float DEFAULT NULL,
  `Height_dif` float DEFAULT NULL COMMENT 'Height difference of the
workpiece',
  `Workpiece_holder` int(11) DEFAULT NULL,
  `eH` float DEFAULT NULL,
  `NL_up` float DEFAULT NULL,
  `NL_low` float DEFAULT NULL,
  `MRR` float DEFAULT NULL,
  `Tool_Allocation` float DEFAULT NULL,
  `Time_Process` float DEFAULT NULL,
  PRIMARY KEY (`id_parameters`),
  KEY `FK_Parameters_Process` (`id_process`),
  CONSTRAINT `FK_Parameters_Process` FOREIGN KEY (`id_process`)
REFERENCES `process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='Machine parameters';
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `parametersprocess`
--

LOCK TABLES `parametersprocess` WRITE;
/*!40000 ALTER TABLE `parametersprocess` DISABLE KEYS */;
/*!40000 ALTER TABLE `parametersprocess` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `parametersrest`
--

DROP TABLE IF EXISTS `parametersrest`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `parametersrest` (
  `id_rest` int(11) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `WorkpiecesNumber` int(11) DEFAULT NULL,
  `CoolantTemp_avg` float DEFAULT NULL,
  `CoolantTemp_min` float DEFAULT NULL,
  `CoolantTemp_max` float DEFAULT NULL,
  `CoolantFlowCold` float DEFAULT NULL,
  `CoolantFlowHot` float DEFAULT NULL,
  `CoolantFlowHot_min` float DEFAULT NULL,
  `CoolantFlowHot_max` float DEFAULT NULL,
  `CoolantFlowCold_min` float DEFAULT NULL,
  `CoolantFlowCold_max` float DEFAULT NULL,
  `CoolantFlowCold_std` float DEFAULT NULL,
  `CoolantFlowHot_std` float DEFAULT NULL,
  `CoolantTemp_std` float DEFAULT NULL,
  PRIMARY KEY (`id_rest`),
  KEY `FK_ParametersRest_Process` (`id_process`),
  CONSTRAINT `FK_ParametersRest_Process` FOREIGN KEY (`id_process`)
REFERENCES `process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `parametersrest`
--

LOCK TABLES `parametersrest` WRITE;
/*!40000 ALTER TABLE `parametersrest` DISABLE KEYS */;
/*!40000 ALTER TABLE `parametersrest` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `process`
--

DROP TABLE IF EXISTS `process`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `process` (
  `id_process` int(11) NOT NULL AUTO_INCREMENT,
  `Process_Name` varchar(80) DEFAULT NULL,
  `Process_Author` varchar(80) DEFAULT NULL,
  `Process_Date` datetime DEFAULT CURRENT_TIMESTAMP,
  `Process_Counter` int(11) DEFAULT NULL,
  PRIMARY KEY (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `process`
--

LOCK TABLES `process` WRITE;
/*!40000 ALTER TABLE `process` DISABLE KEYS */;
/*!40000 ALTER TABLE `process` ENABLE KEYS */;
UNLOCK TABLES;
```

```
--
-- Table structure for table `process_steps`
--

DROP TABLE IF EXISTS `process_steps`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `process_steps` (
  `id_process_steps` int(11) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `id_values` int(11) DEFAULT NULL,
  `P0` int(11) DEFAULT NULL,
  `P1` int(11) DEFAULT NULL,
  `P2` int(11) DEFAULT NULL,
  `P3` int(11) DEFAULT NULL,
  `P4` int(11) DEFAULT NULL,
  `P5` int(11) DEFAULT NULL,
  PRIMARY KEY (`id_process_steps`),
  KEY `FK_STEPS_PROCESS` (`id_process`),
  KEY `FK_STEPS_VALUES` (`id_values`),
  CONSTRAINT `FK_STEPS_PROCESS` FOREIGN KEY (`id_process`) REFERENCES
`process` (`id_process`),
  CONSTRAINT `FK_STEPS_VALUES` FOREIGN KEY (`id_values`) REFERENCES
`values` (`id_values`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `process_steps`
--

LOCK TABLES `process_steps` WRITE;
/*!40000 ALTER TABLE `process_steps` DISABLE KEYS */;
/*!40000 ALTER TABLE `process_steps` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `surfacequality`
--

DROP TABLE IF EXISTS `surfacequality`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `surfacequality` (
  `id_surface` int(11) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `Ra_Top` float DEFAULT NULL,
  `Ra_Bot` float DEFAULT NULL,
  `Rz_Top` float DEFAULT NULL,
  `Rz_Bot` float DEFAULT NULL,
  `Rt_Top` float DEFAULT NULL,
  `Rt_Bot` float DEFAULT NULL,
  `Sa_Top` float DEFAULT NULL,
  `Sa_Bot` float DEFAULT NULL,
  `Sz_Top` float DEFAULT NULL,
  `Sz_Bot` float DEFAULT NULL,
  `St_Top` float DEFAULT NULL,
  `St_Bot` float DEFAULT NULL,
```

```
  PRIMARY KEY (`id_surface`),
  KEY `FK_Process_Surface` (`id_process`),
  CONSTRAINT `FK_Process_Surface` FOREIGN KEY (`id_process`) REFERENCES
`process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `surfacequality`
--

LOCK TABLES `surfacequality` WRITE;
/*!40000 ALTER TABLE `surfacequality` DISABLE KEYS */;
/*!40000 ALTER TABLE `surfacequality` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `values`
--

DROP TABLE IF EXISTS `values`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `values` (
  `id_values` int(11) NOT NULL AUTO_INCREMENT,
  `Name_Value` varchar(11) DEFAULT NULL,
  `Unit` varchar(4) DEFAULT NULL,
  PRIMARY KEY (`id_values`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `values`
--

LOCK TABLES `values` WRITE;
/*!40000 ALTER TABLE `values` DISABLE KEYS */;
INSERT INTO `values` VALUES (1,'Force','daN'),(2,'Force','*'),(3,'Stop
at','um/*'),(4,'Work
time','*'),(5,'Rate','um/*'),(6,'Upper','rpm'),(7,'Upper','*'),(8,'Centre
','rpm'),(9,'Centre','*'),(10,'Lower','rpm'),(11,'Lower','*');
/*!40000 ALTER TABLE `values` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `wear`
--

DROP TABLE IF EXISTS `wear`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `wear` (
  `id_wear` int(11) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `Axial` float DEFAULT NULL,
  `WearProfile` longblob,
  PRIMARY KEY (`id_wear`),
  KEY `FK_proces_wear` (`id_process`),
```

```
  CONSTRAINT `FK_proces_wear` FOREIGN KEY (`id_process`) REFERENCES
`process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `wear`
--

LOCK TABLES `wear` WRITE;
/*!40000 ALTER TABLE `wear` DISABLE KEYS */;
/*!40000 ALTER TABLE `wear` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `workpiece`
--

DROP TABLE IF EXISTS `workpiece`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `workpiece` (
  `id_workpiece` int(11) NOT NULL AUTO_INCREMENT,
  `id_process` int(11) DEFAULT NULL,
  `Material` char(50) CHARACTER SET latin1 DEFAULT NULL,
  `Hardness` float DEFAULT NULL COMMENT 'Vickers',
  `Diameter` float DEFAULT NULL,
  `Initial_Height` float DEFAULT NULL,
  `Surface` float DEFAULT NULL,
  `Description` char(120) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id_workpiece`),
  KEY `FK_Process_Workpiece` (`id_process`),
  CONSTRAINT `FK_Process_Workpiece` FOREIGN KEY (`id_process`) REFERENCES
`process` (`id_process`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `workpiece`
--
LOCK TABLES `workpiece` WRITE;
/*!40000 ALTER TABLE `workpiece` DISABLE KEYS */;
/*!40000 ALTER TABLE `workpiece` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping events for database 'double face grinding'
-- Dumping routines for database 'double face grinding'
--
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
-- Dump completed on 2018-06-22 17:40:22
```