# Towards a data-driven method for classifying relative product complexity across the lifecycle

THOMAS HEDBERG, JR.
Industrial and Systems Engineering
Virginia Tech
thedberg@vt.edu

LOU ZHANG
Graduate Studies
Virginia Tech
lzhang02@vt.edu

Due: May 7, 2017

## Acronyms

| | |
|---|---|
| **2D** | two-dimensional. 8 |
| **3D** | three-dimensional. 3 |
| **AMT** | Association for Manufacturing Technology. 2 |
| **CAD** | computer-aided design. 3, 21 |
| **CAE** | computer-aided engineering. 4 |
| **CFD** | computational fluid dynamics. 4 |
| **CNC** | computer-numerically controlled. 3, 9 |
| **CPPD** | Collaborative Product and Process Development. 2 |
| **DFM** | design for manufacturing. 2 |
| **DFx** | design for X. 26 |
| **EDM** | electrical-discharge machining. 27 |
| **FEA** | finite-element analysis. 4 |
| **FIT** | Failure(s) in Time. 27, 28 |
| **HTTP** | Hypertext Transfer Protocol. 9 |
| **IPD** | Integrated Product Development. 2 |
| **NIST** | National Institute of Standards and Technology. 2, 3 |
| **PCI** | product complexity index. 21 |
| **SMS** | Smart Manufacturing Systems. 3 |
| **XML** | Extensible Markup Language. 9, 10 |

## 1 Introduction

The cost and time to bring a product to market remains a major hurdle for the United States manufacturing industries that must compete in a global economy. While concepts such as design for manufacturing (DFM), Collaborative Product and Process Development (CPPD), and Integrated Product Development (IPD) teams exist, the segregation of engineering and manufacturing organizations in a product lifecycle is still an ongoing problem due to limited resources and time. The segregation leads to horizontal fragmentation, which the business world refers to colloquially as *organizational silos* [1, 2, 3].

The organizational silos achieve complete vertical integration of processes and information within the silo, but collaborating and transitioning information between the various organizational silos is difficult. Industry knows about the problem, but has been unable to solve the problem due to inter-organizational conflict, limited availability of key resources, and the cost of implementing a solution. The increasing complexity of products being released to the marketplace is also not helping the situation because the engineering and manufacturing organization should be collaborating more to overcome the complexity challenges. But instead the organizations are becoming more segregated by trying to solve the problems within only their own domains. There is little holistic perception of the problem in industry.

The product definition is the foundation for a successful product lifecycle because design-engineering decisions become constraints on the remainder of the lifecycle [4]. The product definition is a collection of information that documents, records, and details all the requirements (e.g., dimensions, tolerances, materials, finishes) for a product to ensure the intended value is delivered to the customer. The ability to quantify the required manufacturing activity based on the product definition would enable collaborative engineering.

Manufacturing has long been data-driven, though the general public may not be aware of this. The issue of awareness may stem from differences in terminology. Instead of applying "data analysis" or "optimization," manufacturing instead employs methods in "six sigma" [5] or "lean processes" [6]. The former are essentially the methods applied in the latter.

Nevertheless, much of the manufacturing world has lagged in recent years in the domain of data analytics. While many of the analytics methods were developed in the 1970s to 1980s, manufacturing technology was not ready to support the advanced techniques. Now, technology has caught up and industry is keen to implement data-driven methods. But, a lack of widespread data-collection capabilities remains a gap [7].

When machines cannot communicate with each other or with a central computer, problems may arise. For starters, operators may become blind to the efficiencies that could be realized when using a data-driven approach. Additionally, the cost-savings associated with preventing problems with the machine could be lost. Further, machines should ideally be able to communicate across an enterprise, which requires a robust standard of communication, or a common machine language.

Two organizations have partnered together to help solve this problem and drive manufacturing forward as a whole. The Association for Manufacturing Technology (AMT), in partnership with National Institute of Standards and Technology (NIST), have developed a standard for machine communication, the MTConnect standard [8]. The standard is maturing, but also undergoing additional development and enhancements. We hope this project will help users better understand the power behind the MTConnect standard.

In this study, we dissected the key data attributes that may be collected with the MTConnect standard, and subsequently explored various predictive aspects of this data. We ultimately would like to leverage the knowledge gained from this study into improvements within the world of manufacturing and educate the community on various ways the MTConnect standard can help their businesses.

### 1.1 Problem Statement

We aimed to use machine-activity and product-definition data to classify a product based on relative complexity. We hypothesize that classifying a product by a relative-complexity score would help design and manufacturing engineers understand how their decisions would affect others. For example, the type of geometric features (e.g., holes, slots, bosses) contained within a product definition and the number of different feature definitions would affect the time it takes to plan and produce a part. We want to understand how the complexity score is affected by the different types of features in the product, the characteristics of those features, the types of manufacturing tools required, and the number of manufacturing setups required. Understanding how these elements affect the complexity score for products would help engineers better understand how a new product would relate to past experiences.

### 1.2 Data Description

Several data sources are available for our data-analytics exercise. The set of data is extracted from manufacturing machines (e.g., milling centers, turning centers) using an adapter based on the MTConnect standard [8]. This manufacturing data contains several attributes such as, but not limited to, a timestamp of observation, rotary velocity, path feedrate, tool position, computer-numerically controlled (CNC) code-block number, and the tool used.

More data sets are available as three-dimensional (3D) product models (e.g, computer-aided design (CAD) models) and CNC code (e.g., g-code). The 3D-product models include product-feature definitions and feature characteristics (e.g., tolerance requirements). The CNC code helps us understand the types of decisions (e.g., cutting speeds, tool feedrate) the manufacturing engineer made during the planning process. All of the data is acquired from the NIST Smart Manufacturing Systems (SMS) Test Bed [9]. The SMS Test Bed is a cyber-physical system that supports manufacturing research by enabling sensing and monitoring of manufacturing machines and processes [10].

### 1.3 Paper Organization

In this paper, we will describe and discuss our data-analytics study to identify a novel approach for classifying a product by a relative-complexity score. In Section 2, we will provide a set of background information to identify related previous work and to help the reader better understand the need for this novel approach. In Section 3 we will present the data preprocessing and exploratory data analysis that was conducted. In Section 4 we present our methodology for calculating a relative-complexity score. In Section 5, we provide details about our modeling decisions and the evaluation of our model. In Section 6, we will provide a brief discussion on our work presented in this paper. Lastly, in Section 7 will draw some conclusions on our work here, lessons learned, and recommendations for future steps.

## 2 Background

### 2.1 Defining the product lifecycle

Pugh [11] suggests a high-level activity model for the product lifecycle that starts with marketing, continues through design and manufacture, and ends with selling. While Pugh's [11] work may be over 25 years old, it still provides a nice high-level overview of the complete product lifecycle. Here, we also decomposed some phases of Pugh's lifecycle into several more phases of activities, functions, and/or roles to provide more

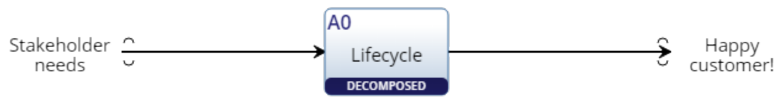


Figure 1: IDEF0: Top-level context diagram for the product lifecycle

context into the lifecycle for this paper. The IDEF0 diagrams[1] to describe the decomposition of Pugh's lifecycle are shown in Figures 1-4.

IDEF0 is a function-modeling methodology. Figure 5 displays the syntax for the box format used in IDEF0. The box represents a function. Arrows entering the box from the left represent inputs to the function. Arrows entering the box from the top represent controls on the function. Arrows leaving the box from the right represent outputs from the function. Lastly, arrows enter the box from the bottom represent a mechanism for the function.

The design phase is where the product is defined based on requirements derived from stakeholder (e.g., customer) needs. The design phase is decomposed (see Figure 3) into four sub-activities, which are *perform preliminary design*, *preliminary design review*, *perform detailed design*, and *critical design review*. Analysis activities are a part of the design phase, but would be sub-activities of the preliminary and detailed design activities. The analysis activities analyze a product using computer-aided engineering (CAE) tools (e.g., simulation-based finite-element analysis (FEA) and computational fluid dynamics (CFD)). The input to the design phase is the requirements outputted from the marketing phase. The requirements become a control on the preliminary-and-detailed-design activities. The output of the design phase is an approved design definition.

The manufacture phase is where a product is fabricated and inspected to ensure the product conforms to the approved design definition. The manufacture phase is decomposed (see Figure 4) into four sub-activities, which are *plan production*, *execute production*, *plan inspection*, and *execute inspection*. The input to the manufacture phase is the approved design from the design phase. The approved design become a control on the planning of the production and inspection activities. The output of the manufacture phase is a finished product when the product passes the inspection activity. When the product fails the inspection activity, the output of the manufacture phase returns to the either the plan production activity, the execute production activity, or both.

[1]The IDEF0 diagrams are to support a basic understanding of the product lifecycle to enable a discussion in the context of our work – the models are not considered complete for a full analysis.

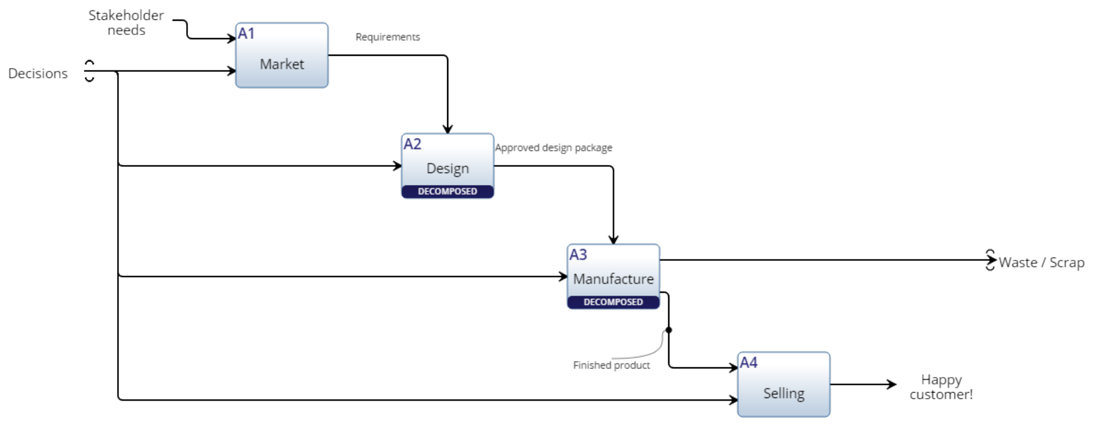


Figure 2: IDEF0: A0, Activity diagram for the product lifecycle

## 2.2 The Importance of Product Complexity

Inevitably, information will be loss as the data and context, that define the information, flows from one phase of the lifecycle to another. Businesses view time as money. Enabling design engineers to determine how product complexity affects the time and cost of a process has limited understanding. Quantifying, in a scientific manner, the time it takes to complete a process prior to starting work is difficult. The problem is compounded by products that are becoming increasingly more complex and much of the work is outsourced and/or distributed across multiple support organizations. Moreover, design engineers are responsible for all aspects of the design and are focused on the end user. But, manufacturing engineers carry dual responsibility.

First, the manufacturing engineer must provide the tooling, machines, operations, and technical understanding to manufacture the product in compliance with the product design. The manufacturing engineer also shares the responsibility of ensuring the design information meets manufacturing needs. While the former is well understood by most manufacturing and engineering organizations, the latter requirement is ambiguous from the engineering-organization's perspective. The review of manufacture-ability typically happens after the design is released and transferred to the manufacturing organization. Design engineers often fail to realize that their decisions become constraints on the remainder of the lifecycle [4]. Not engaging manufacturing earlier and striving to minimize the product complexity leads to the loss of understanding in the context and perception surrounding the decisions that were made. The end result is data and information that is incompatible with the data user's needs.

Data users, in this context, are the people who use the product-definition data in some workflow to produce the product. There is a need to provide tools and methods for analyzing the product and process complexity for design engineers because they are making decisions that influence the entire lifecycle, which must consider the preferences of the data and end users. When problems are found, the design must be sent back to the design engineers for repair. In addition, the increase in product complexity is also making the manufacturing-

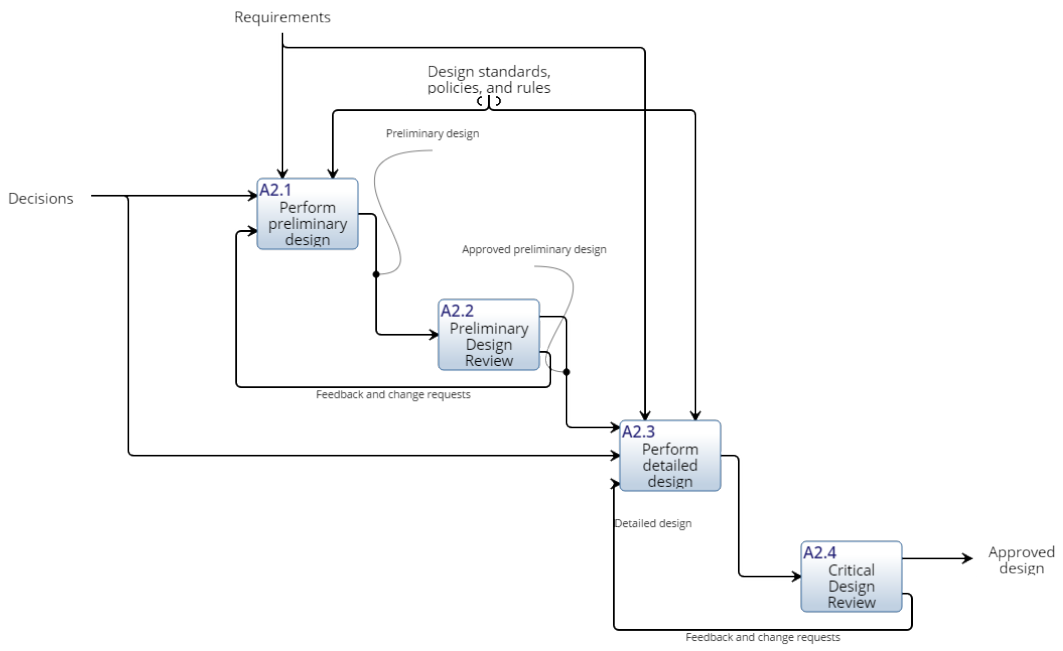

Requirements
Design standards, policies, and rules
Preliminary design
Decisions
A2.1 Perform preliminary design
Approved preliminary design
A2.2 Preliminary Design Review
Feedback and change requests
A2.3 Perform detailed design
Detailed design
A2.4 Critical Design Review
Approved design
Feedback and change requests

Figure 3: IDEF0: A2, Decomposed activity diagram for the design phase of the product lifecycle

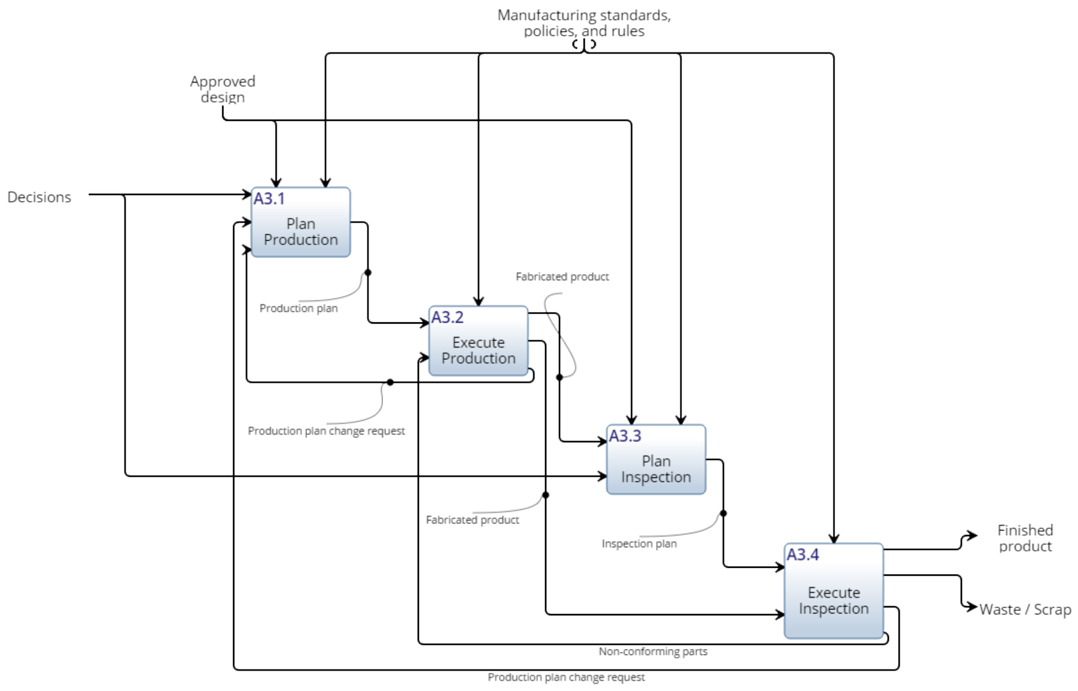


Figure 4: IDEF0: A3, Decomposed activity diagram for the manufacture phase of the product lifecycle

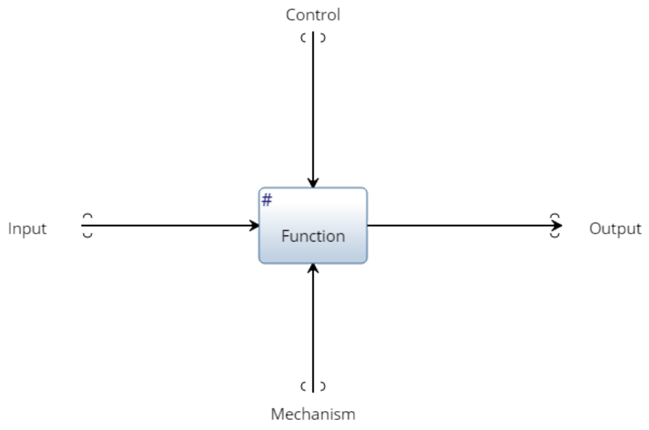


Figure 5: IDEF0 syntax for the integration definition of the box format

process planning and review time increase significantly. Design-engineering organizations must recognize this problem and reduce the overall product complexity to an optimal level for the given design while meeting the needs of all users in the lifecycle. Efficient decision-support tools are needed to alleviate the problem. Moreover, organizations must enable concurrent product-and process-design activities [11] to predict and mitigate risk to the value stream.

The decomposition of the lifecycle and process into activities, workflows, and tasks highlights the importance of understanding product-and-process complexity. At some point in the lifecycle, consideration of the complexity of the product and process must occur. However, the literature does not provide consensus on a scientific definition of complexity. Since Shannon [12] introduced the idea of information-based entropy in 1948, various research has developed many models on product-complexity entropy [13]. Recent research [14, 15, 16] has put forward additional methods and/or models for assessing product complexity. However, most of the developed models are difficult to understand in an efficient manner that is required by engineers trying to analyze the product complexity in real-time as the product designs, specifications, and/or processes are being realized [17]. Rezakhanlou and Villani [18] described Shannon's theory as the optimal rate of compression applied to a signal without losing any of the information being carried by the signal. In engineering, Rezakhanlou and Villani's description relates to the compression of the original product information to the interpretation of the design intent in a static two-dimensional (2D) drawing [19].

The number of components in products has been increasing significantly and the components are coming from a significant mix of mechanical, electrical, and software domains. Product definitions are reaching new levels of complexity that are difficult to interpret and manufacture using only traditional 2D definition drawings. Therefore, designs are beginning to reach the absolute limit of complexity. Shannon's theory only helps with half of the problem by trying to ensure no data is lost, but Shannon does not provide a solution for dealing with the additional interpretation challenges introduced with increased complexity [20].

De Toni et al. [13] conducted a review of 105 works published between 1948 and 2005 and concluded that static and dynamic complexity must be considered when considering complexity related to a workflow. Static and dynamic complexity are defined by Gaio et al. [21] as:

**Static Complexity** "is a characteristic associable to the systems – and so also to the production processes – that refers to the structure of the facilities or to the structure of the plant and considers the degree of difficulty for their management and control. Such type of complexity becomes important when the possible design of a facility or plant is studied."

**Dynamic Complexity** "refers to the analysis of the systems along the time, in other words it studies the trend of the real states that the process assumes within the considered time. [...] However from the point of view of the entropic measures we can consider ... the trend of the waiting queues (or the warehouses). In fact they absorb the variability of a system unit along the time."

Simply put, static complexity is concerned with models, parts, material, machines, etc., and dynamic complexity is concerned about the interactions of processes and labor with the product definition.

DeToni's review showed that much of the literature tries to quantify complexity to an absolute value and that is true for recent literature [22, 23, 24] too. But Frizelle et al. [25] proposed a different approach. Frizelle et al. [25], suggested the level of complexity of an operation is complementary to the maximum complexity an organization is capable of handling. That maximum-complexity capacity may equate to an organization's worst-case scenario for a given topic. Therefore, complexity may be quantified as a relative value using prior experience and/or knowledge gained. Expanding on that concept, a simple and effective way to calculate combined static and dynamic complexity could be based on domain knowledge.

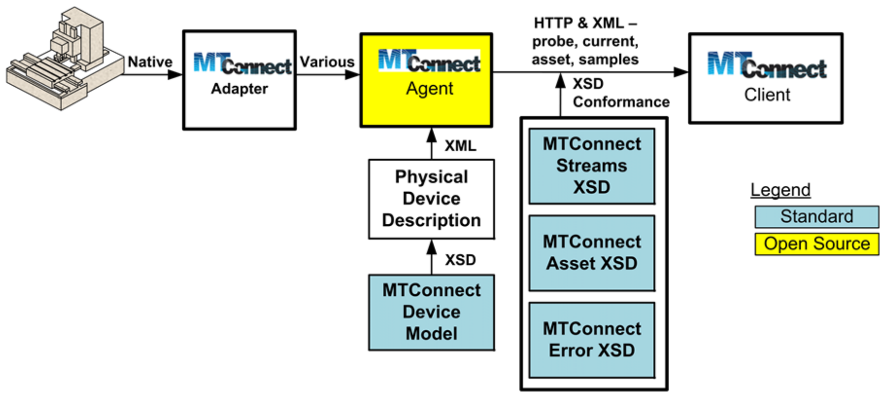


Figure 6: Schematic of MTConnect solution showing communication at all levels from manufacturing machine to MTConnect client. (Diagram is reproduced from [27].)

### 2.3 MTConnect

MTConnect is an open technical standard aiming to provide greater interoperability between controls, devices, and software in the manufacturing industry [8]. The standard is based on Extensible Markup Language (XML) and Hypertext Transfer Protocol (HTTP) and provides information models and communications protocols to enhance the data-acquisition capabilities of manufacturing equipment and applications, enabling a plug-and-play environment. While other communication protocols may exist for data transfer, the information models defined in MTConnect are the only common vocabulary and structure created for manufacturing-equipment data [26].

Within the context of this paper, the MTConnect standard was used to extract information from a CNC machine running a program. The result was data for an available set of manufacturing parameters. The MTConnect standard enabled conversion to a machine-readable format for further analytics to be carried out. The literature contains reports on applying the MTConnect standard to large-scale production facilities [27]. The MTConnect standard supported our work with a low-cost, but high-quality, solution for collecting data and enabling in-depth data analytics.

## 3 Data Preprocessing

### 3.1 Data Details

The data consists of raw output collected by an MTConnect adapter connected to a GF AgieCharmilles or 'GF Agie' machine, manufactured by GF Machining Solutions. The GF Agie is a 5-axis machining center, meaning it is capable of moving in the X,Y,Z directions as well as rotating on the A and B axes (yaw and pitch), resulting in the capability to machine a complex and diverse array of parts.

The output from the machine covers many aspects of the part-creation process and consists of over 100 data fields. The raw-data dump was originally organized into 57 text files, each representing a day's worth of machining activity. Over the 57 days of data we received, we found that 16 discrete parts were manufactured.

We first collated these 57 text documents into one table, resulting in a row count of about 3.1 million. The rationale behind this was because it is more meaningful for certain applications to analyze data by each manufactured part, rather than analyzing data by day. Thus we needed to join all the text files because some parts are machined over or across several days.

Each row in the joined table represents a single observation in time (i.e., events), and records such key features as X,Y,Z,A,B positions, various key happenings during that time period, and machine status. The next subsection (see Section 3.2) demonstrates the process of collecting and aggregating the data for further pre-processing.

## 3.2 Data Loading

After receiving the data in txt files, we began aggregating and loading the data into R using the code shown in Listing 1.

Listing 1: R code for aggregating and loading MTConnect data into R

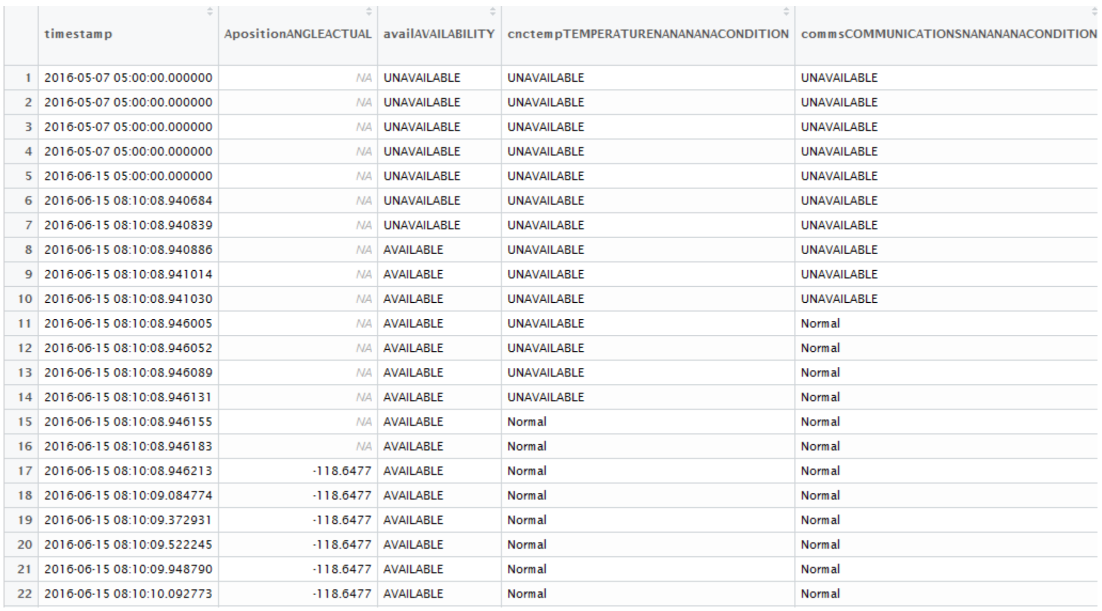
```
# Point R at the input files

file_list <- list.files("C:\\Users\\Viu52790\\Desktop\\GFAgie\\txts")
file_path_xml = "C:\\Users\\Viu52790\\Desktop\\GFAgie\\Devices.xml"
device_name = "GFAgie01"

# Read in the data and create the MTConnect Device in R

allPosition <- as.data.frame(matrix(0, nrow = 1, ncol = 7))

for(i in file_list){

name <- gsub("-",".",i)
name <- gsub(".csv","",name)
name <- gsub("GFAgie","Position_GF", name)

name2 <- gsub("-",".",i)
name2 <- gsub(".csv","",name)
name2 <- gsub("GFAgie","All_GF", name2)


file_path_dmtcd = i
temp <- assign(name, create_mtc_device_from_dmtcd(file_path_dmtcd, file_path_xml,
    device_name))
assign(name, merge(temp, "POSIT"))
assign(name2, merge(temp))

}
```

Looking at a snapshot of the initial data frame shown in Figure 7, it is clear we need to do some more investigation and maybe some data cleaning before we can begin any modeling. A data dictionary of available fields was retrieved using the “probe” capability of the MTConnect standard. The XML output of the probe is shown in

## 3.3 Exploratory Data Analysis

### 3.3.1 Time Series Exploration

We continue by exploring the various key features presented in the data and potential value it may have in modeling relative complexity. We decided to conduct a verification check, using the R code in Listing 2, to see if the data is usable, in addition to deriving some insights that can be used in modeling later. Figure 8 shows the results of our analysis. We determined that there is great value in looking at the distribution of



| | timestamp | ApositionANGLEACTUAL | availAVAILABILITY | cnctempTEMPERATURENANANANACONDITION | commsCOMMUNICATIONSNANANANACONDITION |
|---|---|---|---|---|---|
| 1 | 2016-05-07 05:00:00.000000 | NA | UNAVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 2 | 2016-05-07 05:00:00.000000 | NA | UNAVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 3 | 2016-05-07 05:00:00.000000 | NA | UNAVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 4 | 2016-05-07 05:00:00.000000 | NA | UNAVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 5 | 2016-06-15 05:00:00.000000 | NA | UNAVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 6 | 2016-06-15 08:10:08.940684 | NA | UNAVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 7 | 2016-06-15 08:10:08.940839 | NA | UNAVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 8 | 2016-06-15 08:10:08.940886 | NA | AVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 9 | 2016-06-15 08:10:08.941014 | NA | AVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 10 | 2016-06-15 08:10:08.941030 | NA | AVAILABLE | UNAVAILABLE | UNAVAILABLE |
| 11 | 2016-06-15 08:10:08.946005 | NA | AVAILABLE | UNAVAILABLE | Normal |
| 12 | 2016-06-15 08:10:08.946052 | NA | AVAILABLE | UNAVAILABLE | Normal |
| 13 | 2016-06-15 08:10:08.946089 | NA | AVAILABLE | UNAVAILABLE | Normal |
| 14 | 2016-06-15 08:10:08.946131 | NA | AVAILABLE | UNAVAILABLE | Normal |
| 15 | 2016-06-15 08:10:08.946155 | NA | AVAILABLE | Normal | Normal |
| 16 | 2016-06-15 08:10:08.946183 | NA | AVAILABLE | Normal | Normal |
| 17 | 2016-06-15 08:10:08.946213 | -118.6477 | AVAILABLE | Normal | Normal |
| 18 | 2016-06-15 08:10:09.084774 | -118.6477 | AVAILABLE | Normal | Normal |
| 19 | 2016-06-15 08:10:09.372931 | -118.6477 | AVAILABLE | Normal | Normal |
| 20 | 2016-06-15 08:10:09.522245 | -118.6477 | AVAILABLE | Normal | Normal |
| 21 | 2016-06-15 08:10:09.948790 | -118.6477 | AVAILABLE | Normal | Normal |
| 22 | 2016-06-15 08:10:10.092773 | -118.6477 | AVAILABLE | Normal | Normal |

Figure 7: Example Dataframe in R

timestamps for this, so we can get an idea of when data is recorded and if it is being recorded at a regular interval.

Listing 2: R code for conducting time series exploration

```r
#distribution of timestamp differences - demonstrate rapid recording of times

timediffs <- as.data.frame(1:nrow(allPosition))

for(i in 1:(nrow(allPosition)-1)){
timediffs[i,] <- allPosition$timestamp[i+1] - allPosition$timestamp[i]

}

timediffs[nrow(timediffs),] <- 0

colnames(timediffs) <- 'Difference (s)'

ggplot(timediffs, aes(`Difference (s)`)) + geom_histogram(binwidth = .2, col = '
    red', fill = 'green') +
ggtitle('Difference Between Observations (in seconds)') + xlim(-.01,5) + ylim(0,
    110000)
```

After removing outliers (< 1%), we found the lags between timestamps are largely under 1 second, with over 99% of observations having less than a 5 second lag from the immediately prior observation. From our domain knowledge of manufacturing, this is an acceptable tolerance and indicates the events are recorded on a semi-regular basis, with minimal inconsistency. It also indicates that there are no major gaps in data, with respect to time, in these 57 days of data. We conclude that the machine was continuously recording data for the entire 57 days and no serious omissions are present. Thus, the data was robust on the time level and no other processing needed to be done to regularize time periods.

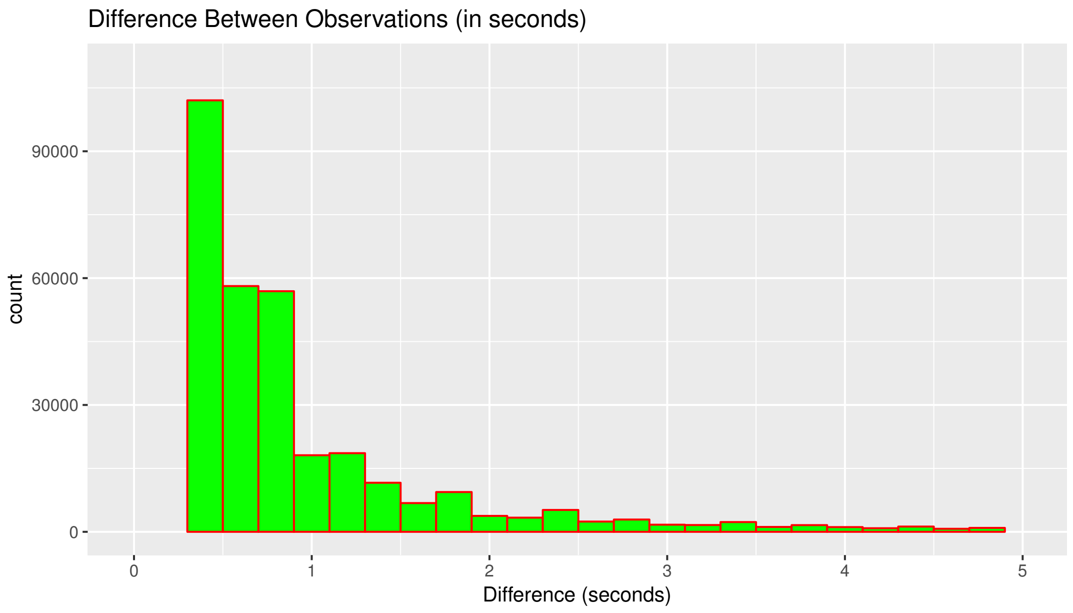


Figure 8: Histogram showing the difference between observations (in seconds)

### 3.3.2 Capacity Utilization

Next, using the R code in Listing 3, we observed the status of the machine over these 57 days to get an idea of the capacity utilization of the machine. The purpose of this is to determine if there is a realistic representation of machine usage during the data's time frame and ascertain that the observations are not suspect in the sense that they over or under-represent certain machine states. Because we found that the data was robust on the time level, this test's purpose is to further confirm the robustness of the data and give insight into how the machine was used. Figure 9 is the result of our analysis.

Listing 3: R code for plotting the machine status

```r
#plot of machine status

counts <- table(allPosition[9])
barplot(counts, main="Machine Status",
xlab="Status")
```

Based on our domain knowledge, we concluded that the result is a realistic representation, over 57 days, of machine status. ACTIVE indicates the machine is being utilized, INTERRUPTED means there is an error of some sort, READY means the machine is not in error but not being utilized, FAULT means there is an error in the machine, and UNAVAILABLE means the machine is down for some reason (e.g., powered off, maintenance). With an approximate 50% utilization rate, the data represents realistic levels and patterns of machine usage.

### 3.3.3 Automatic or Manual?

We next investigated another key attribute: controller mode. Controller mode indicates if the machine is automatically machining based off a previous program or if it's being controlled manually by an operator. Our R code for this analysis is shown in Listing 4 and the results are shown in Figure 10.

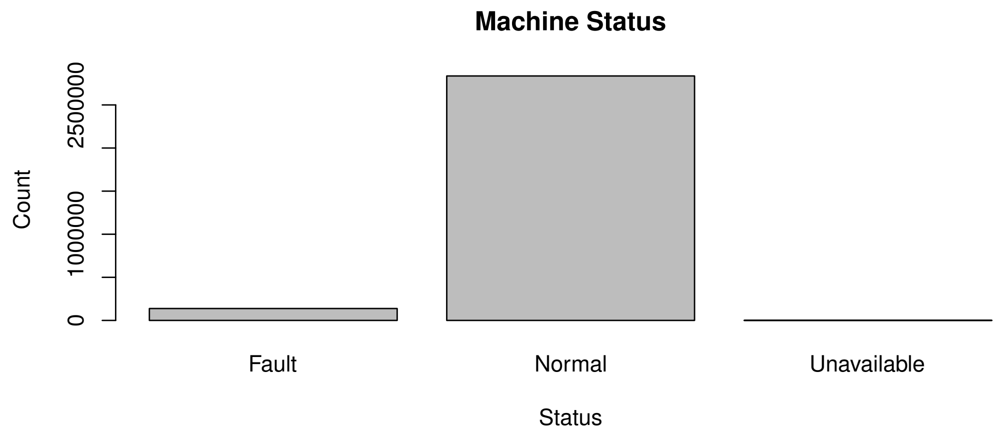


Figure 9: Histogram showing machine status as described by the collected data

Listing 4: R code for plotting the controller mode

```r
#plot controller mode

counts <- table(allPosition[14])
barplot(counts, main="Controller Mode",
xlab="Feedrate Override")
```

AUTOMATIC indicates the machine is running a pre-programmed automated routine, MANUAL indicates a manual takeover by a human, and MANUAL DATA INPUT means the program was altered by manual entered parameters. We saw that the majority of the time, the machine was running an automated program. This distribution is not only realistic, it appears to be favorable to our particular goal of determining relative complexity. This is because the data allows for a good mix between automated programs, which are often more complex, and manual input, which is often or necessarily less complex as it requires a high degree of attention and dexterity, both of which are constrained by the limits of human performance. This gives us a diverse set of data to analyze.

### 3.3.4 Machine Procedure

The most significant plot generated during the exploratory data analysis was a comparison of what parts are being machined. This allowed us to get an idea of part complexity based off the amount of time (approximated by observations) it took to machine the part, and allows us to see clearly the various routines that are run. The R code for this analysis is shown in Listing 5 with the results plotted in Figure 11.

Listing 5: R code for plotting the type of machine procedure

```r
#plot type of machine procedure

counts <- table(allPosition[21])

colnamesbarplot <- names(counts)
colnamesbarplot <- gsub(".H","",colnamesbarplot)

par(las=1, mar=c(4,15,2, 2))

```

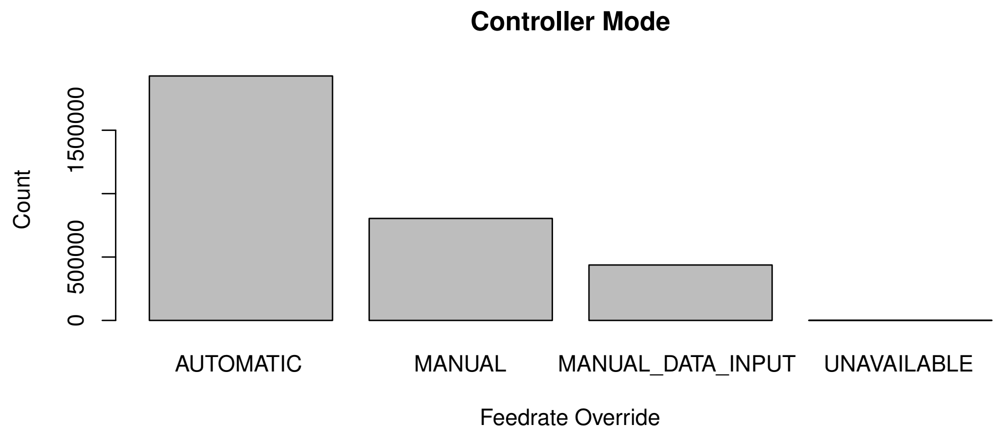


Figure 10: Histogram showing the controller mode as described by the collected data

```r
barplot(counts,space=c(0,5),legend.text=FALSE, beside=TRUE, horiz=TRUE,
density=NA,
col=c("red1","red4","green3"),
xlab="Time Occurences",
axes=TRUE, names.arg=colnamesbarplot, cex.names=.7, las=1, main = 'Programs Run on
    the Machine')
```

Based off our domain knowledge of manufacturing, the time it takes to machine a part may be a decent, although very broad approximation, of how complex is a part. We plan on supplementing this with additional information of course, including but not limited to X,Y,Z,A,B position and path feedrates.

### 3.3.5 Part Visualizations

Our next step was to visualize a couple of these parts to get an idea of what they look like, and consequently how intricate or complex are the parts. This will serve as guidance to come up with our final complexity quotient. We first visualize and discuss the Goodman Manifold Block using the R code in Listings 6 and 7 and Figures 12 and 13.

Listing 6: R code for plotting the Y position vs X position of the GOODMAN MANIFOLD BLOCK

```r
# Plotting Y position vs X position to get an idea of GOODMAN MANIFOLD BLOCK

ManifoldBlock <- allPosition[allPosition[21] == 'TNC:\\BRIAN\\GOODMAN MANIFOLD
    BLOCK.H',]

pos_data_mtc = select_(ManifoldBlock, colnames(allPosition[25]), colnames(
    allPosition[26]), colnames(allPosition[27]))

ggplot(pos_data_mtc) + geom_path(aes(x = XpositionPOSITIONACTUAL, y =
    YpositionPOSITIONACTUAL)) + ggtitle('Top Down View of Machining Process')
```

Listing 7: R code for plotting the X position vs Z position of the GOODMAN MANIFOLD BLOCK

```r
# Plotting X position vs Z position to get an idea of GOODMAN MANIFOLD BLOCK

```

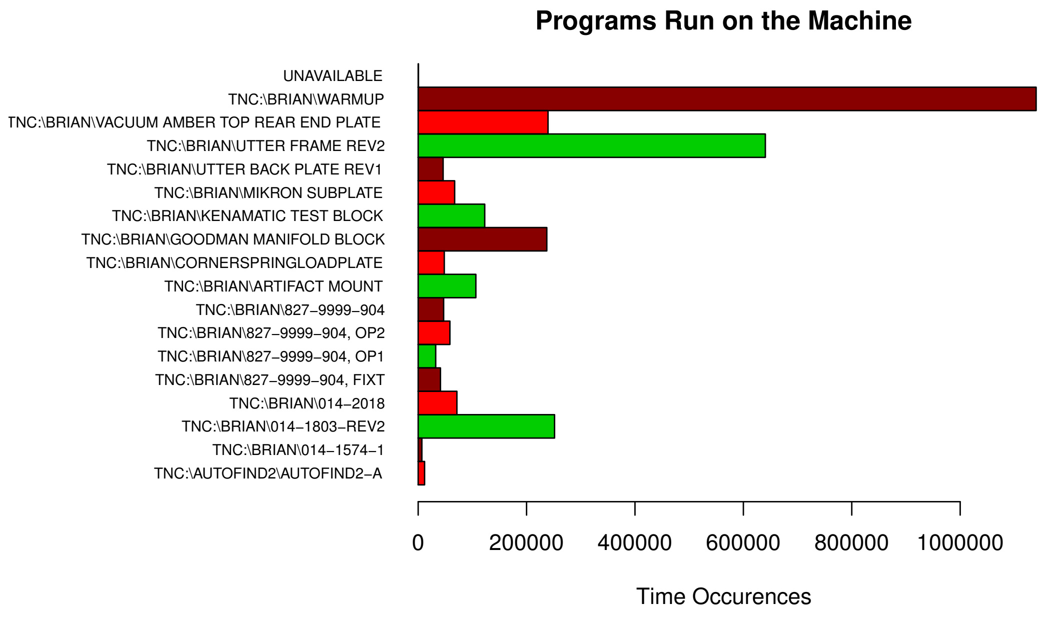


Figure 11: Histogram showing the active programs run on the machine as described by the collected data

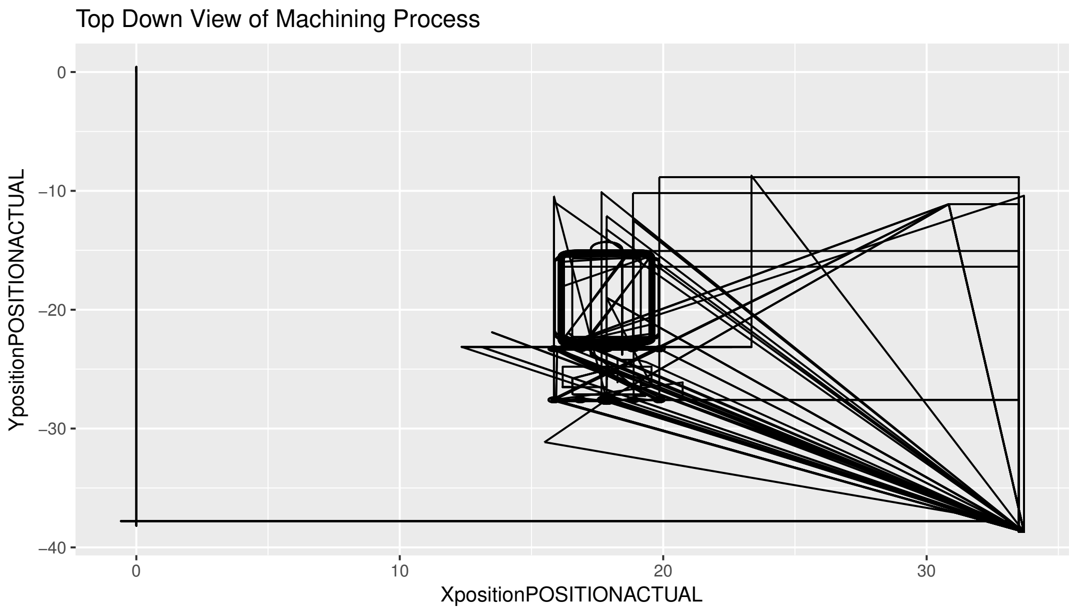


Figure 12: Plot showing the top-down view of the path position of the machine tool running a program for the GOODMAN MANIFOLD BLOCK part

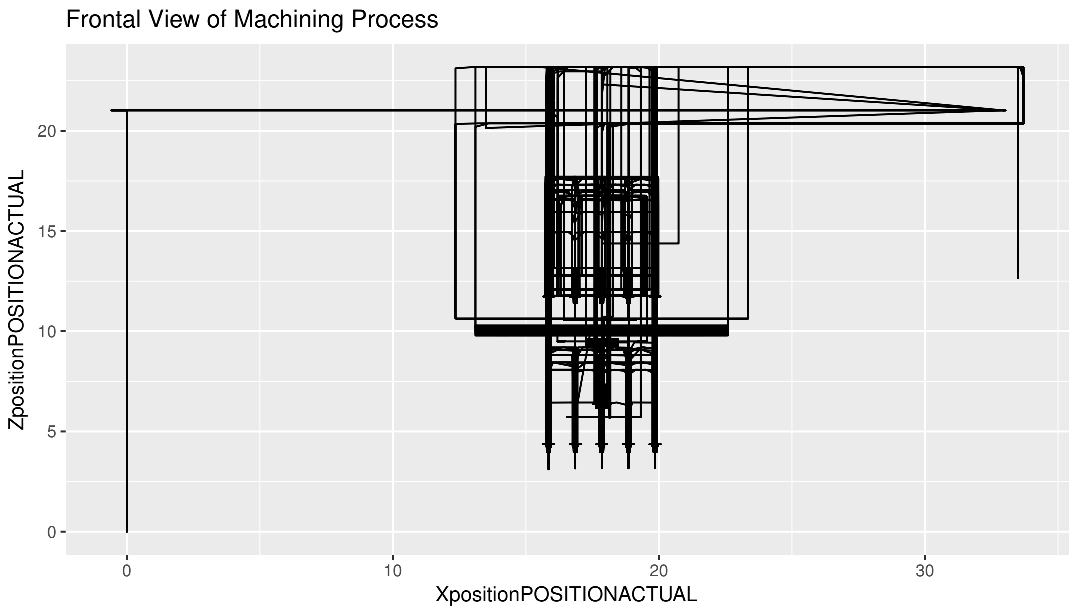


Figure 13: Plot showing the front view of the path position of the machine tool running a program for the GOODMAN MANIFOLD BLOCK part

```
ggplot(pos_data_mtc) + geom_path(aes(x = XpositionPOSITIONACTUAL, y =
    ZpositionPOSITIONACTUAL)) + ggtitle('Frontal View of Machining Process')
```

We compared this to an actual manifold block (see Figure 14, and could see the machining patterns leading to the final product, verifying the robustness of the data and this method of visualization.

We did another comparison using actual a object versus machining-position visualization to validate that the data is robust, this time taking the corner springload plate as an example. The R code for the plate is shown in Listings 8 and 9 and the results are shown in Figures 15 and 16.

Listing 8: R code for plotting the Y position vs X position of the CORNER SPRINGLOAD PLATE

```
# Plotting Y position vs X position to get an idea of CORNER SPRINGLOAD PLATE

SpringPlate <- allPosition[allPosition[21] == 'TNC:\\BRIAN\\CORNERSPRINGLOADPLATE.
    H',]

pos_data_mtc = select_(SpringPlate , colnames(allPosition[25]), colnames(
    allPosition[26]), colnames(allPosition[27]))

ggplot(pos_data_mtc) + geom_path(aes(x = XpositionPOSITIONACTUAL, y =
    YpositionPOSITIONACTUAL)) + ggtitle('Top Down View of Machining Process')
```

Listing 9: R code for plotting the X position vs Z position of the CORNER SPRINGLOAD PLATE

```
# Plotting X position vs Z position to get an idea of CORNER SPRINGLOAD PLATE

ggplot(pos_data_mtc) + geom_path(aes(x = XpositionPOSITIONACTUAL, y =
    ZpositionPOSITIONACTUAL)) + ggtitle('Frontal View of Machining Process')
```

We compared the output for the plate data to an actual corner springload plate (see Figure 17, only the

Figure 14: Examples of a manifold block

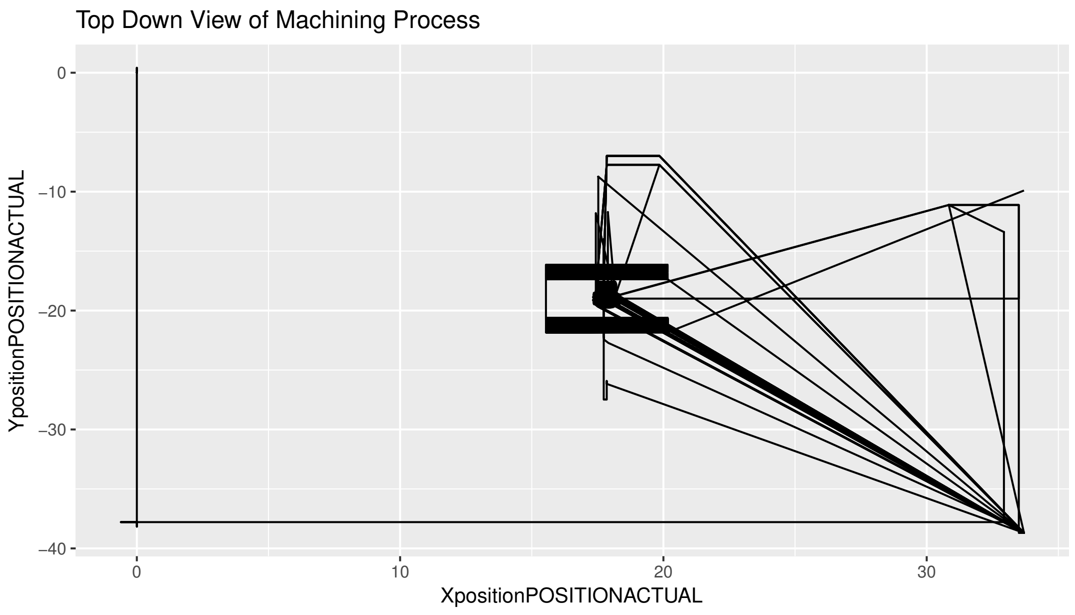


Figure 15: Plot showing the top-down view of the path position of the machine tool running a program for the CORNER SPRINGLOAD PLATE part

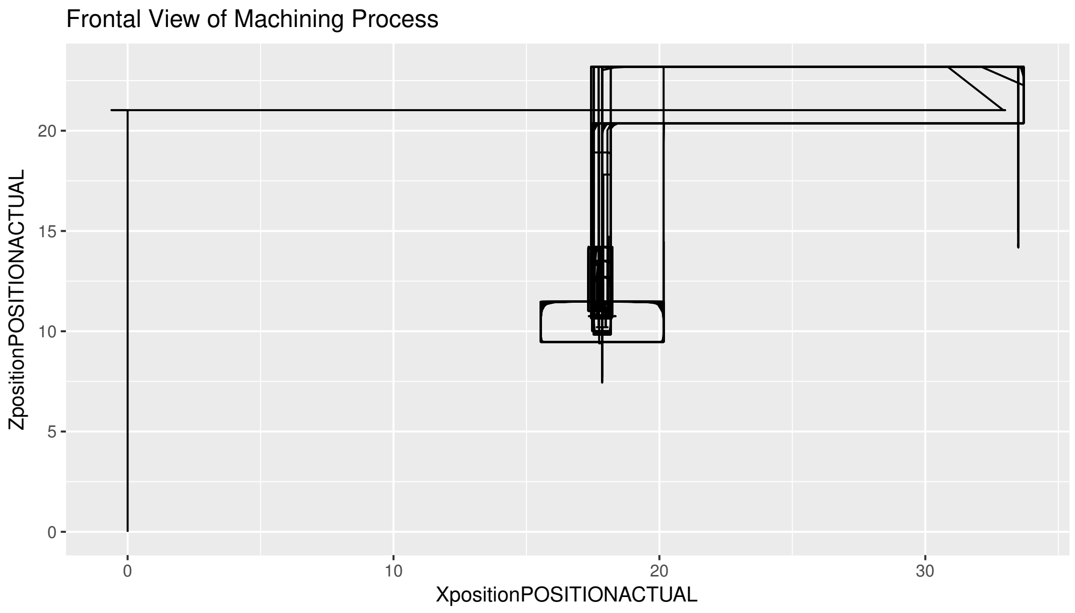


Figure 16: Plot showing the front view of the path position of the machine tool running a program for the CORNER SPRINGLOAD PLATE part

middle, tubular portion), and could see the machining patterns leading to the final product, further verifying the robustness of the data and this method of visualization.

## 4 Methodology

### 4.1 Baseline Complexity Computation

After verifying the robustness of the position data (A and B axes do not need to be verified visually; there is little value to plotting yaw/pitch), we developed an initial baseline to determine baseline complexity - a simple sum of the percent of unique values in each machining sequence on each axis, multiplied by the inverse of the ratio of the most frequently occurring observation to the second most frequently occurring observation (per axis). We then multiplied that sum by the log of the number of time observations for that part as a whole. Equation 1 and Table 1 is provided to demonstrate the baseline complexity product. The R code to calculate the baseline complexity is shown in Listing 10.

$$C_{baseline} = \sum_{1}^{n} \left(\frac{percentUnique}{freqRatio}\right) * log(nObs) \tag{1}$$

Where

- $C_{baseline}$ = baseline complexity
- n = the number of axes in the dataset
- percentUnique = percent of unique values in each axis vector

Figure 17: Example of a corner springload plate

Table 1: Sample R output from calculating the components of the baseline complexity of the manifold block

| freqRatio | percentUnique | zeroVar | nzv | ID | complexity |
|---|---|---|---|---|---|
| 1.655806621 | 0.5249498642 | FALSE | FALSE | A | 0.3170357321 |
| 1.539469367 | 0.2561553110 | FALSE | FALSE | C | 0.1663919507 |
| 2.061508787 | 12.6195756585 | FALSE | FALSE | X | 7.5767688972 |
| 1.130119658 | 14.3004600684 | FALSE | FALSE | Y | 12.6539344446 |
| 1.021576433 | 18.8788149446 | FALSE | FALSE | Z | 18.4800807189 |

- freqRatio = ratio of the count of the most occurring value to the count of the second most occurring value in each axis vector
- nObs = total number of observations in the entire dataset

Listing 10: R code for determining the baseline complexity

```r
#determine complexity

zerovarA <- nearZeroVar(ManifoldBlock$ApositionANGLEACTUAL, saveMetrics = TRUE,
    names = TRUE, allowParallel = TRUE)
zerovarC <- nearZeroVar(ManifoldBlock$CpositionANGLEACTUAL, saveMetrics = TRUE,
    names = TRUE, allowParallel = TRUE)
zerovarX <- nearZeroVar(ManifoldBlock$XpositionPOSITIONACTUAL, saveMetrics = TRUE,
    names = TRUE, allowParallel = TRUE)
zerovarY <- nearZeroVar(ManifoldBlock$YpositionPOSITIONACTUAL, saveMetrics = TRUE,
    names = TRUE, allowParallel = TRUE)
zerovarZ <- nearZeroVar(ManifoldBlock$ZpositionPOSITIONACTUAL, saveMetrics = TRUE,
    names = TRUE, allowParallel = TRUE)

zerovarManifoldBlock <- bind_rows(zerovarA, zerovarC, zerovarX, zerovarY, zerovarZ
    )

zerovarManifoldBlock$ID <- c('A','C','X','Y','Z')

zerovarManifoldBlock$complexity <- 1/zerovarManifoldBlock$freqRatio *
    zerovarManifoldBlock$percentUnique
```

By axis, *freqRatio* signifies the ratio of the count of the most common observation to the second most frequent one. *percentUnique* quantifies the percent of unique observations on each axis. *Zerovar* and *nzv* indicate if

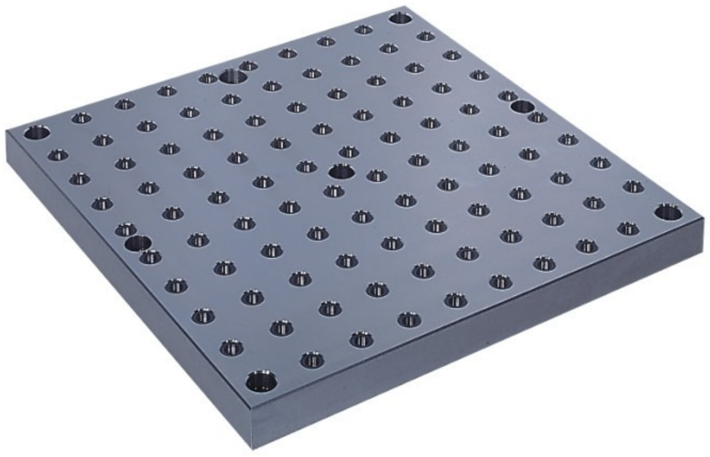


Figure 18: Example of a MIKRON SUBPLATE

the vector displays zero variance (all values in the vector are the same, indicating inactivity or something suspicious).

Our logic behind this calculation was, the more unique values a part has on each axis, the more detailed was the part. We needed to juxtapose that with the frequency ratio: if the most frequently occurring position observation was many times more than the next occurring observation, then it deflated the impact of unique values. Thus, we accounted for this and multiplied the inverse of this value by the unique value proportion. Next, we took the log of the number of observations and not the raw value because time contributes logarithmically to complexity, not linearly, as economies of scale can be realized with larger parts. The final product of all three represents a baseline complexity, which calculated using the R code shown in Listing 11. The resulting baseline complexity score for the values from Table 1 was 485.119.

Listing 11: R code for calculating the complexity of the GOODMAN MANIFOLD BLOCK

```
ManifoldBlockComplexity <- sum(zerovarManifoldBlock$complexity) * log(nrow(
    ManifoldBlock))
```

We compared the baseline score from the manifold block to a less complex part, the mikron subplate (see Figure 18, to verify our methodology on a macro-level. The result was 115.457.

The results of the comparison was expected because the subplate is significantly less complex than the manifold block. We used the calculated baseline complexities for all parts in the dataset to help us determine the worst-case scenario.

## 4.2 Relative-Complexity Score Formulation

After determining the baseline complexity score, we moved to determining the relative-complexity score for a set of parts. This score will assist in classifying the complexity of the parts. We made the following assumptions in the formulation of the relative-complexity score:

- Product complexity does not have an absolute value and is defined by relativity based on Frizelle et al. [25].

- The law of large numbers [28] applies to the convergence of time intervals for processing an information set.
- As product-entity diversity increases, product complexity increases [13].

Our goal in developing a relative-complexity score was to provide decision support to design-and-engineering organizations. Research [4] shows that every decision made early in the lifecycle becomes a constraint on the remainder of the lifecycle, because each subsequent decision further reduces the compliant solution space. Therefore, a method to estimate quantitatively the effects of a product-definition's complexity on the remainder of the product lifecycle must be able to process in near-real-time on the data available to the organizations. For the design functions, the most significant dataset is the CAD model.

The scoring formulation keeps processing requirements (e.g., manufacturing planning) in perspective. We guided the development of the scoring toward a method that analyzes a design-decision's effect on completing workflows using a count of product entities divided by the baseline complexity scores and a factor to account for error in interpretation and processing of the product design.

The economics domain has a product complexity index (PCI), which ranks products by the amount of capabilities or know-how necessary to manufacture the products [29]. The PCI is represented by Equation 2. Product complexity is determined by calculating the average diversity of countries that manufacture a specified product and the average ubiquity of all other products the countries manufacture [29].

$$M^{P}_{p,P'} \equiv \sum_{c} \frac{M_{cp} M_{cp'}}{k_{c,0} k_{p,0}}, \tag{2}$$

where $M_{cp}$ is a matrix that is 1 if a country produces a product, and 0 otherwise, $k_{c,0}$ is the measure of diversity, and $k_{p,0}$ is the measure of ubiquity [29].

The product complexity measure in the PCI from the economics domain is a simple, but effective measure of complexity based on a relative analysis for a specific product. We aim to generate a complexity measure for the engineering domain that enables efficient and effective scoring of a product's complexity. The measure we developed is represented in Equation 3.

The equation for calculating a complexity coefficient is given in Equation 3:

$$K_i = \frac{C_i}{C_{baseline,i}}, \tag{3}$$

where $K_i$ is the complexity coefficient of an entity $i$ and $C_i$ is the count of entity $i$.

An entity is any thing and/or action driven by a product design that the functions in the lifecycle consume. Our relative-complexity score is a vector consisting of complexity coefficients for a set of key entities:

**Characteristic** a dimension, tolerance

**Form feature** a feature that defines geometric form, fit, and/or function

**Machine tool** the required number of machine tools to manufacture the product in compliance with its definition

**Tool change** the required number of tool changes to manufacture the product in compliance with its definition

**Process change** the required number of process changes to manufacture the product in compliance with its definition

Table 2: Entity Counts and Complexity Class Label for a set of models from the NIST SMS Test Bed [9]

| | Entity Counts | | | | | |
|---|---|---|---|---|---|---|
| Model | Features | Characteristics | Machine Tool | Process Change | Tool Change | Complexity Class |
| 827-9999-904 Rev C | 9 | 99 | 1 | 2 | 12 | Medium |
| 827-9999-903 Rev C | 4 | 17 | 1 | 2 | 4 | Low |
| PDES NX-001 | 9 | 91 | 1 | 6 | 6 | Medium |
| PDES CREO-001 | 8 | 72 | 1 | 2 | 5 | Medium |
| PDES CATIA-001 | 12 | 70 | 2 | 4 | 7 | Medium |
| NIST-MTC-001 | 14 | 22 | 1 | 5 | 7 | Medium |
| NIST-MTC-002 | 11 | 21 | 1 | 2 | 7 | Low |
| NIST-MTC-003 | 13 | 14 | 1 | 2 | 8 | Medium |
| NIST-MTC-004 | 2 | 20 | 1 | 2 | 2 | Low |
| CTC 01 | 11 | 17 | 1 | 2 | 7 | Low |
| CTC 02 | 17 | 44 | 2 | 4 | 14 | High |
| CTC 03 | 7 | 28 | 2 | 3 | 2 | Medium |
| CTC 04 | 11 | 20 | 1 | 2 | 5 | Low |
| CTC 05 | 12 | 20 | 2 | 4 | 5 | Medium |
| FTC 06 | 13 | 61 | 1 | 2 | 9 | Medium |
| FTC 07 | 11 | 43 | 1 | 5 | 7 | Medium |
| FTC 08 | 12 | 48 | 1 | 2 | 8 | Medium |
| FTC 09 | 9 | 58 | 1 | 2 | 7 | Medium |
| FTC 10 | 15 | 70 | 1 | 3 | 12 | Medium |
| FTC 11 | 5 | 12 | 1 | 2 | 2 | Low |

Counts are generated for each of the five entities. Then, each entity count is divided by the standard baseline value for the entity. The standard baseline value is determine a posteriori by using the baseline complexity computation described in Section 4.1. The baseline complexity computation identifies the worst-case scenario from all past parts manufactured by an organization. The maximum baseline complexity score identifies the worst-case scenario and then the counts of each key entity are determined and set as the standard baseline value for the entity.

## 5 Model Evaluation and Results

For this section, we do not use the dataset referenced in the earlier "Data Preprocessing and Exploratory Data Analysis" section. The reason for this choice is because the dataset helps identify the worst-case scenario after a design was completed and fabricated. We wanted to enable better design decisions before the design is built. In addition, the value-add of prediction for that dataset is dubious. This is because domain knowledge suggests that predicting things such as machine status or operator status provides little meaning because they lie largely independent from the other features in that dataset. For example, machine unavailability would likely not be triggered by a specific X/Y/Z configuration, but rather because of power outages (planned or unplanned) or maintenance issues.

Thus, we imported a new dataset, which contains variables such as the number of characteristics, features, machine tools, process changes, and tool changes for each machined part. This was a dataset labeled by an expert, with complexity class (high, medium, or low) as the target. The data is shown in Table 2.

There was little training data, mostly stemming from the fact that there has not been much of an effort to collect and label part complexity data at this point in time. However, we believe that in the future, more samples will be collected and our methods can used on this new data.

We investigate the data for missing values or anomalies and cannot find any instances of this. This is because we have a small, curated data set that enjoys the luxuries of robustness and cleanliness that larger datasets do not necessarily enjoy.

## 5.1 Model Selection

Given the limited amount of data we had available to train our model, our options were restricted to models that could function based off of minimal training data. We also wanted to have an interpretable model, as the results could give insight to researchers on the critical components of determining complexity. Thus, in our consideration of modeling techniques, we excluded such models as neural networks and support vector machine, which may not be as interpretable and require more data to be effective. Due to the importance of interpret-ability, we narrowed our choices down to a random forest and a decision tree. We decided to go with a random-forest model because we could visualize the key components of models, have a more accurate prediction, and have some interpretable results (and most importantly, preserve the original variables and be able to see which ones were the most important).

Because a random-forest model works by constructing an aggregated bootstrap of variables on the original data set, the variance is much lower than that of a decision tree. The error of a model is calculated from the sum of its variance and bias, thus this property of random forests can greatly decrease the prediction errors. The choice of a random forest ultimately represents a compromise between interpret-ability, accuracy, and explain-ability.

### 5.1.1 Alternate Method Considerations

We considered doing some sort of bootstrapping to increase our sample size, but realized this was not feasible for several reasons. First, we only have one sample with a high complexity class, and we would need to oversample it to create a stratified sample. It would not be valid to simply duplicate this one observation several times. This would result in a situation in our dataset where there is no diversity in how a high complexity part is characterized, leading to a tree that would bias towards this one unique set of characteristics for high-complexity parts. Second, our goal behind this model is not necessarily to construct a predictive model right off the bat, but rather to determine if it is feasible to even make accurate classifications based off this small sample set. This iteration will set us up for more advanced analysis in the future, but the goal is more simplistic for the current model we constructed.

### 5.1.2 Validation Considerations

It is a necessary part of the data science process to split the data into testing and validation data. However, as mentioned in Section 5.1.1, we are not necessarily trying to construct a full model at this point, but merely prove that the features in our data set may have the potential to predict complexities in the future. The presence of only 20 data points meant that excluding even a few data points would significantly hamper the goal of this initial iteration; thus we decided to include all points in our training data.

## 5.2 Model Execution

We began by investigating the predictors for relative importance. We constructed a variable importance plot, shown in Figure 19, with the mean decrease in Gini index as the metric. The Gini index was calculated

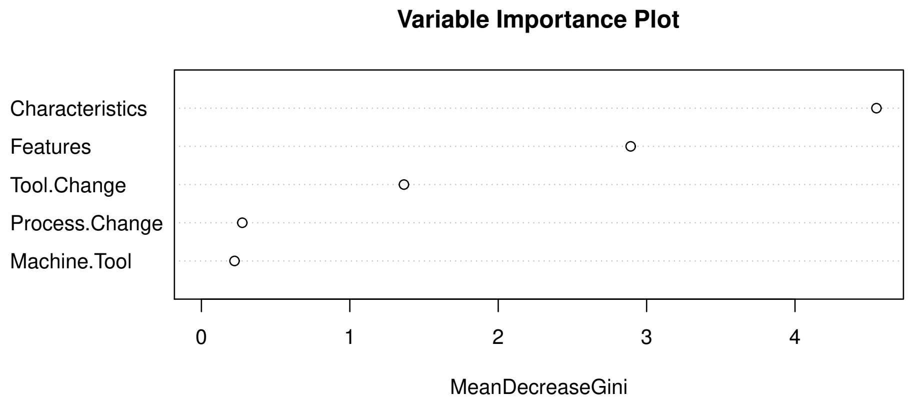


Figure 19: Variable importance plot with the mean decrease in Gini index as the metric

using Equation 4.

$$1 - \sum_j ([p(j|t)]^2 \tag{4}$$

The Gini essentially calculates the purity of the child nodes after a split. The higher the purities, the lower the index, and the better the splitting condition (a perfect case is one in that $p(j|t)$ is 1, or in our case if $p(high|characteristics > 50) = 1$, this is a great condition to split by). The variable with the highest mean decrease in Gini index is the best predictor, due to the fact that it allows for the most purity in the child nodes that are splitting with it.

Figure 19 shows that the count of part characteristics is the most important variable, followed by part features and the number of tool changes. We interpreted this to mean that across many splits, these variables respectively create the purest nodes by creating the lowest Gini children possible.

Next, we tuned the parameters of the forest to optimize our results and processing speed. We investigated the number of trees necessary to reach the minimum error. Each line in Figure 20 represents error for one of the features.

In Figure 20, we saw that the optimum number of trees is around 175, so we selected $ntree = 175$ as our parameter.

### 5.3 Model Results

After executing the random-forest algorithm, the yielded the confusion matrix (with actual classes on the left and predicted classes on the top) shown in Figure 21.

## 6 Discussion

With an out of box error rate of 20%, the model classifies well on the low and medium categories, and poorly on the high category. There is only one sample in the high category and it was not represented well when

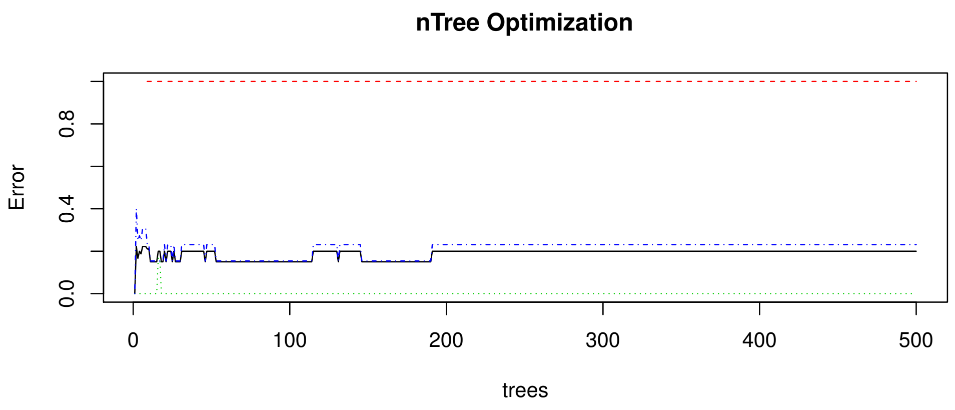
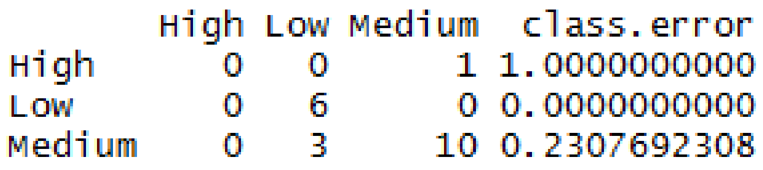


Figure 20: Variable importance plot with the mean decrease in Gini index as the metric



```
       High Low Medium  class.error
High      0   0      1 1.0000000000
Low       0   6      0 0.0000000000
Medium    0   3     10 0.2307692308
```

Figure 21: Confusion Matrix, with actual classes on the left and predicted classes on the top, for the results of the model using the random-forest algorithm

constructing the trees. We considered this misclassification more an artifact of our limited sample size than any failure of the model itself.

## 6.1 What Went Wrong?

We were unsuccessful in generating a complete model for classifying product definitions using relative complexity.So what went wrong? First off, the data too sparse. We had enough manufacturing data to determine a baseline complexity score for finding the worst-case part. However, the design dataset was too small to use the data for training, validation, and testing.

Secondly, we couldn't train the model based on Table 2. Could the complexity class label be wrong in Table 2? The labels are based on one expert's opinion. More experts and input must be sought if a subjective labeling is to remain. Otherwise, a more objective method for labeling the data should be investigated.

Lastly, is Equation 3 correct? We could not verify the validity of Equation 3 because we did not have enough design data to properly test the model. Additionally, the input of one expert for labeling the design dataset may be injecting too much uncertainty. Therefore, more investigation into the model, Equation 3, and expanding the design dataset is required.

## 6.2 Potential Benefits

While we were not able to validate our approach fully because our design dataset was too sparse, there are potential benefits to continuing the pursuit of our method. One potential benefit is simplify the measuring of product complexity to be practically usable by industry in near-real-time. Another benefit is the ability to account for unexpected error based on measurable product complexity.

### 6.2.1 The affects of product complexity

The definition of product complexity is ambiguous because much of the literature investigates product complexity as an absolute value or deals only with assemblies. However, this paper argues product complexity is relative to a baseline value, which provides a complexity ratio. Defining product complexity with an absolute value method presents a difficult challenge for the product lifecycle. The activity of product design is domain-specific within itself and the type of part being designed requires different domain-specific knowledge to understand the complexity of the product.

For example, a very complex sheet-metal bracket may never be more complex than a very basic turbo-fan engine blade. But within their respective domains, the very complex sheet-metal bracket may cost more to fabricate than the basic turbo-fan blade. This example shows that trying to create an algorithm to calculate the absolute complexity for all the different types of product being manufactured may not be useful or relevant to practical industrial use. Furthermore, Mattsson et al. [17] points to the need of a quick and efficient way to calculate complexity if engineers are expected to use the measure in daily work. All of this provides relevance to Frizelle et al.'s [25] concept – even two decades later. Therefore, our work uses the assumption that a relative complexity value allows domain experience to be factored into the determination and comparison of product complexity through the application of a simple method.

The standard baseline value in Equation 3 should be based on multiple factors such as, but not limited to:

- Domain experience (e.g., worst case scenario)
- Industry technology capabilities (e.g., size of tool crib in supplier machines)
- Industry recommended practices (e.g., the use of cosmetic form features – edge blends)

In addition, design for X (DFx) activities may utilize the complexity coefficient in real time. For example, determining the form feature, tooling change, and process change hi-base count values, $C_{hi-base}$ can determine an ideal number of varying corner and/or blend radii. A common cost and cycle time driver in manufacturing is required tooling changes due to different, seemingly arbitrary blend radii being used to finish internal radii and corners. While designs should strive to specify the largest radius possible, a common mistake is to not count the number of differing radius form features. By treating each radius change as a different form feature, the entity count value increases. As a result, the method calculates a higher complexity coefficient value for such a product, which signals a probable increase in product complexity for the engineer.

The complexity coefficient also identifies opportunities for product optimization. The complexity coefficient is a relative ratio value that is calculated by comparing the defined product to a baseline high-complexity product. Therefore, thresholds for complexity coefficients defined in design policies and/or recommended practices can give designers and engineers a goal to target in DFx activities. For example, a complexity coefficient of one says that the product complexity is equal to the baseline high-complexity product (i.e., a worst case configuration for a supplier), which signals significant cost because manufacturing the product may require special manufacturing processes. A complexity coefficient greater than one signals that product complexity is greater than the baseline. Conversely, the complexity coefficient that is less than one signals the product complexity is less than the baseline.

A company may determine that an entity complexity coefficient for a product, shown in Equation 3, should not be higher than 0.5 or 50% of its worst-case scenario. If the calculated entity complexity coefficient is above the threshold value, then the design engineer must, or at least should, investigate the individual entity complexity coefficient to determine where product optimization may enable a lower relative complexity calculation. The efficient calculation of complexity enables the design engineer to make effective decisions that could positively affect the product lifecycle. Further, the company could look at the vector complexity coefficients and compare the product complexity vector against a baseline vector.

If the desire is not optimization, then the complexity coefficient may signal when unconventional and/or an advanced-manufacturing process should be utilized. For example, the use of high-speed wire electrical-discharge machining (EDM) may be an option instead of milling for the creation of multiple small holes of varying size. Or the use of additive-manufacturing methods may be more economical than subtractive-manufacturing methods. Thus, the complexity coefficient provides the engineer with a quick way to understand how the product of focus compares against a known baseline (i.e., past experience) and when different engineering decisions may be required to define product more effectively.

### 6.2.2 Accounting for error

Unexplained anomalies present themselves, which makes error unavoidable in any process. However, error may be minimized to an acceptable level. It is useful to distinguish between three types of quantitative errors: mechanical errors, logic errors, and omission errors [30]. Mechanical errors are simple mistakes, such as typing a wrong number or pointing to a wrong cell in a spreadsheet. Logic errors involve utilizing the incorrect formula or process because of a mistake in reasoning. Logic error rates are higher than mechanical error rates because they are more difficult to detect and correct [31, 32]. Reason [33] suggests the most dangerous type of error is the omission error. The omission error is when something is left out or missing, which makes omission errors extremely difficult to detect [32, 34].

The work presented here is not intended to define different errors and how to detect them, but rather to understand the effect of errors on a workflow to support minimizing the risk of error. Equation 5 proposes that error could effect a workflow's time by some factor. Panko [35] suggests a human error rate of 2% to 5% in normal programming development given sufficient time constraints and the absence of stress. Whereas, Baumann [36] suggests advanced processors with large multi-megabit embedded memory have error rates of 50,000 Failure(s) in Time (FIT) per chip. Baumann uses the FIT as the unit of measure, where one FIT is one error in a billion device hours (opportunities), and translates 50,000 FIT to a computing system with a reliability that produces one failure every two years assuming 24 hours of continuous operation.

$$T_{E,t} = (1 + K_E) \sum_{i=1}^{n} t_{s,i}, \tag{5}$$

where $T_{E,t}$ is the total workflow process time adjusted for information reliability error, $t_{s,i}$ is the time of workflow process step $i$, and $K_E$ is the calculated coefficient-of-count error given by Equation 6.

$$K_E = 2C_E \prod_{i=m}^{n} K_i, \tag{6}$$

where $C_E$ is the count of errors given by Equation 7.

$$C_E = round \rightarrow integer \left( P(E) \sum_{i=1}^{n} C_i \right), \tag{7}$$

where $P(E)$ is the probability of a count error occurring or ${}^{1}/_{reliability}$.

Evident from the Panko [35] and Baumann [36] comparison, information reliability – the errors associated with the information – is the point where error may be controlled the most. The workflow creator should assess each workflow step to determine where risk of information error is present and how the risk may be mitigated. A mitigation plan could be to automate a workflow step that is processed manually by a human to reduce the probability of error. The process of determining the decision factors for which workflow steps should be automated and/or mitigated is out of scope of this paper, but it is recognized that there will be multiple factors (e.g., cost to automate) that eventually dictate which steps are investigated.

Equation 5 shows the error coefficient, defined in Equation 6, as a multiplier of the sub-total time of an entire workflow being analyzed. The two in Equation 6 accounts for the rework required, while the count of errors, shown by Equation 7, is calculated using the probability of error (e.g., Panko's 2% to 5% [35] or Baumann's FIT estimates [36].) Looking closer at Equation 6, the product of all complexity coefficients is also a factor in error. This is because the product complexity affects directly the time it takes to detect, process, and/or repair the error in the workflow. This suggests that an error in a low complexity product takes a small fraction of the creation time to repair. However, an error in a high complexity product close or equal to the complexity baseline takes almost the same amount of time to repair as the time it takes to create the data. Thus, the total complexity coefficient approaches a $2x$ factor in Equation 6 for error compared to creation time as product complexity approaches the baseline complexity values.

## 7 Conclusion

We conclude that based on the features available, it could be feasible to construct a predictive model using a random forest. This knowledge will act as a baseline for future iterations, when more parts are added to our database and we are able to construct a model with full cross-validation on a richer and more expansive data set. We expect that parameters may change and features may shift in importance, but that there could be at least some element of prediction extracted from these five features.

### 7.1 Lessons Learned

This project provided us an opportunity to gain insight from novel machine-activity-data uses. We learned more about the practical implications and predictive capabilities of using machine-activity data coupled with product-definition data. In future iterations, additional domain knowledge must be incorporated and investigated for contributions to predicting various items of importance for engineering and factory managers based on the relative-complexity measures explored in this project.

### 7.2 Recommended Next Steps

As we discussed in Section 6.1, there is additional work required to validate our approach. Several next steps are recommended. First, we must expand the design dataset by discovering more usable design models. Secondly, we must engage more domain experts to assist in the labeling of the design data. Lastly, the data analytics model should be reviewed and retested once more data is available and the uncertainty in the class labeling is reduced.

While we were not successful in our endeavor to classify a product based on relative complexity using machine-activity and product-definition data, we believe there we have shown sufficient motivation to continue our pursuit. The conclusions and lessons learned from our project will help inform our future work.

## References

[1] Vijay Govindarajan, Mark Sebel, and Jay Terwillinger. The first two steps toward breaking down silos in your organization. *Harvard Business Review*, 2011.

[2] Evan Rosen. Smashing silos. *Bloomberg Businessweek*, 2010.

[3] Brent Gleeson and Megan Rozo. The silo mentality: How to break down the barriers. *Forbes*, 2013. URL `http://www.forbes.com/sites/brentgleeson/2013/10/02/the-silo-mentality-how-to-break-down-the-barriers/`.

[4] Alejandro Salado Diez. *Measuring and influencing problem complexity and its impact on system affordability during requirements elicitation for complex engineered systems.* Ph.D. Dissertation, Stevens Institute of Technology, 2014.

[5] Peter S. Pande, Robert P. Neuman, and Roland R. Cavanagh. *THE SIX SIGMA WAY : How to Maximize the Impact of Your Change and Improvement Efforts.* McGraw-Hill Education, New York, second edition. edition, 2014. ISBN 9780071497329.

[6] Rachna Shah and Peter T. Ward. Lean manufacturing: context, practice bundles, and performance. *Journal of Operations Management*, 21(2):129–149, 2003. ISSN 0272-6963. doi: 10.1016/S0272-6963(02)00108-0.

[7] Thomas D. Hedberg Jr, Nathan W. Hartman, Phil Rosche, and Kevin Fischer. Identified research directions for using manufacturing knowledge earlier in the product life cycle. *International Journal of Production Research*, 55(3):819–827, 2017. ISSN 0020-7543. doi: 10.1080/00207543.2016.1213453.

[8] MTConnect Institute. MTConnect standard, 2014. URL `http://www.mtconnect.org/media/39542/mtc_part_1_overview_v1.3.pdf`.

[9] Thomas Hedberg Jr and Moneer Helu. NIST Smart Manufacturing Systems (SMS) Test Bed, 2016. URL `http://smstestbed.nist.gov`.

[10] Moneer Helu and Thomas Hedberg Jr. Enabling smart manufacturing research and development using a product lifecycle test bed. *Procedia Manufacturing*, 1:86–97, 2015. ISSN 2351-9789. doi: 10.1016/j.promfg.2015.09.066.

[11] Stuart Pugh. *Total design : integrated methods for successful product engineering.* Addison-Wesley Pub. Co., Wokingham, England ; Reading, Mass., 1991. ISBN 0201416395.

[12] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[13] Alberto F De Toni, Alessio Nardini, Fabio Nonino, and Gianluca Zanutto. Complexity measures in manufacturing systems. In *Proceedings of the European Conference on Complex Systems: "Towards a science of complex systems"*, pages 14–18, Paris (France), 2005.

[14] Joshua D. Summers and Jami J. Shah. Mechanical engineering design complexity metrics: Size, coupling, and solvability. *Journal of Mechanical Design*, 132(2):0210041–02100411, 2010. ISSN 10500472. doi: 10.1115/1.4000759.

[15] Josue R. Crespo-Varela, Gul E. Okudan Kremer, Conrad S. Tucker, and Lourdes A. Medina. An analysis of complexity measures for product design and development. In *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC/CIE 2012, August 12, 2012 - August 12, 2012*, volume 3 of *Proceedings of the ASME Design Engineering Technical Conference*, pages 523–532. American Society of Mechanical Engineers, 2012. doi: 10.1115/DETC2012-71309.

[16] Matthew Peterson, Gregory M. Mocko, and Joshua D. Summers. Evaluating and comparing functional and geometric complexity of products. In *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC/CIE 2012, August 12, 2012 - August 12, 2012*, volume 7 of *Proceedings of the ASME Design Engineering Technical Conference*, pages 551–558. American Society of Mechanical Engineers, 2012. doi: 10.1115/DETC2012-71408.

[17] Sandra Mattsson, Per Gullander, and Anna Davidsson. Method for measuring production complexity. In *28th International Manufacturing Conference*, 2011.

[18] Fraydoun Rezakhanlou and Cedric Villani. *Entropy methods for the Boltzmann equation (lectures from a special semester at the Centre Émile Borel, Institut H. Poincaré, Paris, 2001).* Springer Springer Springer, Germany, 2008.

[19] Thomas D. Hedberg Jr, Nathan W. Hartman, Phil Rosche, and Kevin Fischer. Identified research directions for using manufacturing knowledge earlier in the product life cycle. *International Journal of Production Research*, pages 1–9, 2016. ISSN 0020-7543. doi: 10.1080/00207543.2016.1213453.

[20] Thomas D. Hedberg Jr, Joshua Lubell, Lyle Fischer, Larry Maggiano, and Allison Barnard Feeney. Testing the digital thread in support of model-based manufacturing and inspection. *Journal of Computing and Information Science in Engineering*, 16(2):1–10, 2016. ISSN 1530-9827. doi: 10.1115/1.4032697.

[21] Loris Gaio, Francesca Gino, and Enrico Zaninotto. *Production systems: manual for the operational management of the company (Italian).* Carocci, Roma, 2003. ISBN 9788843024612.

[22] Kijung Park and Gul Kremer. The impact of complexity on manufacturing performance: A case study of the screwdriver product family. In *19th International Conference on Engineering Design (ICED13) Design For Harmonies*, volume 3. The Design Society, 2013.

[23] Lourdes A. Medina, Marija Jankovic, and Gül E. Okudan Kremer. Investigating the relationship between product design complexity and fda for medical device development, 2013.

[24] Waguih ElMaraghy, Hoda ElMaraghy, Tetsuo Tomiyama, and Laszlo Monostori. Complexity in engineering design and manufacturing. *CIRP Annals - Manufacturing Technology*, 61(2):793–814, 2012. doi: 10.1016/j.cirp.2012.05.001.

[25] G. Frizelle, G. Frizelle, and Business Intelligence. *The Management of Complexity in Manufacturing: A Strategic Route Map to Competitive Advantage Through the Control and Measurement of Complexity.* Business Intelligence Limited, 1998. ISBN 9781898085362.

[26] Thomas Hedberg Jr, Allison Barnard Feeney, Moneer Helu, and Jaime A. Camelio. Towards a lifecycle information framework and technology in manufacturing. *Journal of Computing and Information Science in Engineering*, 17(2):021010–021010–13, 2017. doi: 10.1115/1.4034132.

[27] Sid Venkatesh, Sidney Ly, Martin Manning, John Michaloski, and Fred Proctor. Automating asset knowledge with mtconnect. In *2016 International Manufacturing Science and Engineering Conference (MSEC)*, pages 1 – 10. ASME, 2016.

[28] Leonard E. Baum and Melvin Katz. Convergence rates in the law of large numbers. *Transactions of the American Mathematical Society*, 120(1):108–123, 1965. ISSN 00029947. doi: 10.2307/1994170. URL `http://www.jstor.org/stable/1994170`.

[29] Ricardo Hausmann. *The atlas of economic complexity : mapping paths to prosperity.* The MIT Press, Cambridge, MA, updated edition. edition, 2013. ISBN 9780262525428 (pbk. alk. paper).

[30] Raymond R Panko and Richard P Halverson Jr. Spreadsheets on trial: a survey of research on spreadsheet risks. In *System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on*, volume 2, pages 326–335. IEEE, 1996. ISBN 0818673249.

[31] Raymond R Panko and Salvatore Aurigemma. Revising the panko-halverson taxonomy of spreadsheet errors. *Decision Support Systems*, 49(2):235–244, 2010. ISSN 0167-9236.

[32] Carl Martin Allwood. Error detection processes in statistical problem solving. *Cognitive science*, 8(4): 413–437, 1984. ISSN 1551-6709.

[33] James Reason. *Human error*. Cambridge university press, 1990. ISBN 0521314194.

[34] David D Woods. Some results on operator performance in emergency events. *Institute of Chemical Engineers Symposium Series*, 90(3):21–31, 1984.

[35] Raymond R. Panko. Error rates in normal program development, 1997-2008. Retrieved from: `http://panko.shidler.hawaii.edu/HumanErr/Index.htm`.

[36] Robert Baumann. Soft errors in advanced computer systems. *IEEE Des. Test*, 22(3):258–266, 2005. ISSN 0740-7475. doi: 10.1109/mdt.2005.69.

## A MTConnect Probe Output for the GFAgie

Listing 12: MTConnect Probe Output from `https://smstestbed.nist.gov/vds/GFAgie01/probe`

```xml

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="/styles/Devices.xsl"?>
<MTConnectDevices xmlns:m="urn:mtconnect.org:MTConnectDevices:1.3" xmlns="
    urn:mtconnect.org:MTConnectDevices:1.3" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:schemaLocation="urn:mtconnect.org:MTConnectDevices:1.3
     /schemas/MTConnectDevices_1.3.xsd">
  <Header creationTime="2017-05-02T18:51:29Z" sender="c4591511bb74" instanceId="
      1493324232" version="1.3.0.17" assetBufferSize="1024" assetCount="0"
      bufferSize="1048576"/>
  <Devices>
    <Device id="GFAgie01" name="GFAgie01" uuid="mtc_adapter001">
      <Description manufacturer="Agie Charmilles" model="HPM600U">Agie Mikron
          HPM600U - GF Agie Charmilles HPM600U</Description>
      <DataItems>
        <DataItem category="EVENT" id="GFAgie01-dtop_1" name="avail" type="
            AVAILABILITY"/>
        <DataItem category="EVENT" id="GFAgie01-dtop_2" name="estop" type="
            EMERGENCY_STOP"/>
        <DataItem category="CONDITION" id="GFAgie01-dtop_3" name="system" type="
            SYSTEM"/>
        <DataItem category="EVENT" id="GFAgie01_asset_chg" type="ASSET_CHANGED"/>
        <DataItem category="EVENT" id="GFAgie01_asset_rem" type="ASSET_REMOVED"/>
      </DataItems>
      <Components>
        <Axes id="GFAgie01-axes_1" name="axes">
          <Components>
            <Linear id="GFAgie01-X_1" name="X">
              <DataItems>
                <DataItem category="SAMPLE" id="GFAgie01-X_2" name="Xposition"
                    nativeUnits="MILLIMETER" subType="ACTUAL" type="POSITION"
                    units="MILLIMETER"/>
              </DataItems>
            </Linear>
            <Linear id="GFAgie01-Y_1" name="Y">
              <DataItems>
                <DataItem category="SAMPLE" id="GFAgie01-Y_2" name="Yposition"
                    nativeUnits="MILLIMETER" subType="ACTUAL" type="POSITION"
                    units="MILLIMETER"/>
              </DataItems>
            </Linear>
            <Linear id="GFAgie01-Z_1" name="Z">
              <DataItems>
                <DataItem category="SAMPLE" id="GFAgie01-Z_2" name="Zposition"
                    nativeUnits="MILLIMETER" subType="ACTUAL" type="POSITION"
                    units="MILLIMETER"/>
              </DataItems>
            </Linear>
            <Rotary id="GFAgie01-C_1" name="C">
              <DataItems>
                <DataItem category="SAMPLE" id="GFAgie01-C_2" name="Cposition"
                    nativeUnits="DEGREE" subType="ACTUAL" type="ANGLE" units="
                    DEGREE"/>
              </DataItems>
            </Rotary>
            <Rotary id="GFAgie01-A_1" name="A">
              <DataItems>
                <DataItem category="SAMPLE" id="GFAgie01-A_2" name="Aposition"
                    nativeUnits="DEGREE" subType="ACTUAL" type="ANGLE" units="
                    DEGREE"/>
              </DataItems>
            </Rotary>
          </Components>
        </Axes>
        <Controller id="GFAgie01-controller_basic_1" name="controller_basic">
          <DataItems>
            <DataItem category="EVENT" id="GFAgie01-controller_basic_2" name="Fovr" nativeUnits="PERCENT" subType="OVERRIDE" type="PATH_FEEDRATE_OVERRIDE" units="PERCENT"/>
            <DataItem category="EVENT" id="GFAgie01-controller_basic_3" name="Sovr" nativeUnits="PERCENT" type="ROTARY_VELOCITY_OVERRIDE" units="PERCENT"/>
            <DataItem category="CONDITION" id="GFAgie01-controller_basic_4" name="servo" type="ACTUATOR"/>
            <DataItem category="CONDITION" id="GFAgie01-controller_basic_5" name="comms" type="COMMUNICATIONS"/>
            <DataItem category="CONDITION" id="GFAgie01-controller_basic_6" name="pneumatic" type="COMMUNICATIONS"/>
            <DataItem category="CONDITION" id="GFAgie01-controller_basic_7" name="hydraulic" type="COMMUNICATIONS"/>
            <DataItem category="CONDITION" id="GFAgie01-controller_basic_8" name="logic" type="LOGIC_PROGRAM"/>
            <DataItem category="CONDITION" id="GFAgie01-controller_basic_9" name="motion" type="MOTION_PROGRAM"/>
            <DataItem category="CONDITION" id="GFAgie01-controller_basic_10" name="cnc_temp" type="TEMPERATURE"/>
          </DataItems>
          <Components>
            <Path id="GFAgie01-path_basic_1" name="path_basic">
              <DataItems>
                <DataItem category="EVENT" id="GFAgie01-path_basic_2" name="execution" type="EXECUTION"/>
                <DataItem category="EVENT" id="GFAgie01-path_basic_3" name="mode" type="CONTROLLER_MODE"/>
                <DataItem category="EVENT" id="GFAgie01-path_basic_4" name="program" type="PROGRAM"/>
                <DataItem category="EVENT" id="GFAgie01-path_basic_5" name="CuttingTool" type="TOOL_ID"/>
                <DataItem category="EVENT" id="GFAgie01-path_basic_6" name="line" type="LINE"/>
                <DataItem category="EVENT" id="GFAgie01-path_basic_7" name="move" type="x:MOTION"/>
                <DataItem category="SAMPLE" id="GFAgie01-path_basic_8" name="path_pos" nativeUnits="MILLIMETER_3D" type="PATH_POSITION" units="MILLIMETER_3D"/>
              </DataItems>
            </Path>
          </Components>
        </Controller>
      </Components>
    </Device>
  </Devices>
</MTConnectDevices>
```