Simulating a Virtual Machining Model in an Agent-Based Model for Advanced Analytics

David Lechevalier[1], Seung-Jun Shin[2], Sudarsan Rachuri[3], Sebti Foufou[4], Y. Tina Lee[5], Abdelaziz Bouras[4]

[1]Le2i, Université de Bourgogne, Dijon, France, david_lechevalier@etu.u-bourgogne.fr

[2]Graduate School of Management of Technology, Pukyong National University, Busan, South Korea, sjshin@pknu.ac.kr

[3]Office of Energy Efficiency and Renewable Energy, Advanced Manufacturing Office, Department of Energy, Washington, DC, USA, sudarsan.rachuri@hq.doe.gov

[4]CSE Department, College of Engineering, Qatar University, Qatar, sfoufou@qu.edu.qa, abdelaziz.bouras@qu.edu.qa

[5]Systems Integration Division, National Institute of Standards and Technology, Gaithersburg, MD, USA, yung-tsun.lee@nist.gov

Abstract. Monitoring the performance of manufacturing equipment is critical to ensure the efficiency of manufacturing processes. Machine-monitoring data allows measuring manufacturing equipment efficiency. However, acquiring real and useful machine-monitoring data is expensive and time consuming. An alternative method of getting data is to generate machine-monitoring data using simulation. The simulation data mimic operations and operational failure. In addition, the data can also be used to fill in real data sets with missing values from real-time data collection. The mimicking of real manufacturing systems in computer-based systems is called "virtual manufacturing". The computer-based systems execute the manufacturing system models that represent real manufacturing systems.

In this paper, we introduce a virtual machining model of milling operations. We developed a prototype virtual machining model that represents 3-axis milling operations. This model is a digital mock-up of a real milling machine; it can generate machine-monitoring data from a process plan. The prototype model provides energy consumption data based on physics-based equations. The model uses the standard interfaces of Step-compliant data interface for Numeric Controls (STEP-NC) and MTConnect to represent process plan and machine-monitoring data, respectively. With machine-monitoring data for a given process plan, manufacturing engineers can anticipate the impact of a modification in their actual manufacturing systems.

This paper describes also how the virtual machining model is integrated into an agent-based model in a simulation environment. While facilitating the use of the virtual machining model, the agent-based model also contributes to the generation of more complex manufacturing system models, such as a virtual shop-floor model. The paper describes initial building steps towards a shop-floor

model. Aggregating the data generated during the execution of a virtual shop-floor model allows one to take advantage of data analytics techniques to predict performance at the shop-floor level.

Keywords: STEP-NC, MTConnect, milling, data generator, simulation, advanced analytics, manufacturing simulation

1 Introduction

Data analytics examines large amounts of data to extract insights to help make better decisions (LaValle et al. 2011). As an emerging topic, data analytics has being used with many applications in various domains (Sun and Reddy 2013; Guo et al. 2011; Ticknor 2013). These applications can typically be grouped into four categories: descriptive analyses, diagnostic analyses, predictive analyses, and prescriptive analyses. Descriptive analyses study what happened, while diagnostic analyses try to explain why it happened. Predictive analyses try to predict what will happen, while prescriptive analyses determine how to achieve a desired outcome.

The development of smart manufacturing platforms embraces big data analytics. The Smart Manufacturing Leadership Coalition recognizes the importance of big data in its leading effort on smart manufacturing including the development of an open platform for smart manufacturing and smart manufacturing system test beds (SMLC, 2016). Wang and Alexander (2015) identify big data issues and challenges in design and manufacturing and discuss the impact and opportunities that big data can have in manufacturing.

Noor (2013) examines the plethora of sensors integrated with modern products. He discusses the benefits of sophisticated and automated data analytics technologies in the filtering and processing of the information provided by these sensors. These benefits include reduced costs of defects and controls. The author provides a few examples from heavy industrial manufacturers such as Raytheon and Ford.

Manufacturing data are necessary to explore the use of the different techniques available with data analytics. The framework described by Lechevalier et al. (2014) highlights the importance and the necessity of data collection for developing data analytics in the manufacturing applications. Such applications may continuously generate large amounts of structured and unstructured data (Young and Pollard 2012). The data formats used in the proposed framework must be clearly specified for data to be analyzed. Many modern machines provide real-time data to monitor values of the operating parameters. These data can be specified in the MTConnect standard (MTConnect 2015), which facilitates communication between equipment and software applications.

Since acquiring actual data is expensive and time consuming, simulation approaches are explored. Brown and Sturrock (2009) show that simulation approaches allow manufacturers to lower costs and save time at the factory level by generating simulation data; the data are then analyzed to improve performance of the systems in the factory. In this simulation context, Marinov and Seetharamu (2004) defined virtual manufacturing as the concept of integrating different areas of manufacturing using

computer technology to create and execute of virtual models that mimic actual manufacturing systems. A virtual model represents a manufacturing system.

One virtual model can be comprised of several other virtual models that are considered sub-models. For example, a virtual shop-floor model is built using a set of virtual machining operation models. These models simulate the execution of operations at the machine-tool level. Individual virtual machining operation models can be developed and validated independently. Zhang et al. (2011) review several research efforts on the simulation of Numerical Control (NC) machining activities. They classify these simulation methods into four categories: solid-based, object space-based, image space-based, and web-based NC machining simulations. Bouhadja and Bey (2015) survey simulation methods for multi-axis machining. Hanwu and Yueming (2009) propose a Web-based NC machine tool operation virtual system, where various practical operations were executed virtually. The proposed machine operation system combines several technologies such as HTML (W3C 2014), Java (Gosling 2000), and Javascript (Flanagan 2006).

This paper makes two main contributions. First, we develop a virtual machining model for a 3-axis milling operation to generate machine-monitoring data from a process plan. We apply the approach described by (Shin et al. 2016) to a milling process. This model allows one to generate simulation data as if the data were produced by sensors from a real machine. The generated data help a manufacturing engineer to predict performance of the real machine and perhaps, make adjustment or modification of the machine or the process plan in order to improve the efficiency of the real machine. One advantage of this approach is that it is based on standards for process plan and monitoring data representation. This facilitates the comprehension and communication of the data by the manufacturing practitioners who are going to use this model. The standards are described in the next section.

Second, our integration of the virtual machining model into an agent-based model allows it to be incorporated into a broader simulation environment. We show the execution of the agent-based model and how it can contribute, in the future, to the generation of virtual shop-floor models.

This paper is organized as follows: Section 2 introduces the data models involved in the virtual machining model and related work. Section 3 presents the work to develop the milling-machine model and its three major components. Section 4 shows the integration of the virtual machining model into an agent-based model, the execution of the agent-based model in a simulation environment, and the validation of the simulation data. Section 5 describes the motivation, logic and challenges for building a virtual factory model. We conclude in Section 6.

2 Data definitions

In this section, we provide descriptions of the input and output of the virtual machining model. The input and output are in ISO 14649-1 Step-compliant data interface for Numeric Controls (STEP-NC) (ISO 2003) and MTConnect representations, respectively. STEP-NC supports standard representations for process planning while MTConnect enables standard representations of machine-monitoring

data. We also discuss related research that involves both STEP-NC and MTConnect standards.

2.1 Input data in ISO 14649 STEP-NC representation

ISO 14649-1 STEP-NC provides a data model for computer numerical control (CNC). NC programs allow manufacturers to automatically control machine tools. The use of NC machines and computers in manufacturing led to the development of computer-aided manufacturing (CAM) where computers interpret CAM files and send a set of instructions to the individual NC machines for production. STEP-NC provides capabilities to describe a machining operation by defining working steps and sequences for operation. ISO 14649-11 (ISO 2004), or Process Data in Milling, defines the required data model for describing a milling operation. Because this standard clearly defines the process sequence and the parameters involved in a milling operation, its use allows us to observe the impact of a given process plan on machine performance.

2.2 Output data in MTConnect representation

MTConnect is an XML-based (W3C 2008) communication standard that enables manufacturing equipment to provide data in structured XML to facilitate the organized retrieval of process information from NC machine tools. This standard is designed for the exchange of data between shop-floor equipment and software applications used for monitoring and data analysis. Vijayaraghavan et al. (2008) emphasize MTConnect capabilities to enhance data acquisition from devices and improve the integration of the data in software solutions. In MTConnect, a machine tool is treated as a device. Each device contains its own information model to communicate the available capabilities and components such as axes or spindle. This information model is called a probe document. In addition, devices generate MTConnect documents that capture the machine-monitoring data including the power consumption and the position of the tool. An agent, which is a computer program, collects data from the device and transmits the data to MTConnect-compatible applications.

Using the MTConnect standard for the output data, we ensure that the data have a well-formed structure that facilitates the integration with MTConnect-compliant downstream applications.

2.3 Related work to STEP-NC and MTConnect

STEP-NC and MTConnect are two different standards. Mapping the heterogeneous information contained in the two standards requires solutions that fill the gaps between their information models. Ridwan and Xu (2013) take advantage of these two standards for real-time machining operation optimization, as well as optimization that could be applied at the next run. They interpret STEP-NC programs and generate a tool path. They also collect real MTConnect data for further optimization. In contrast, our research strives for the generation of MTConnect data independent of any physical

machining operation. As mentioned in the Introduction, (Shin et al. 2016) proposed a virtual machining model for a 2-axis turning operation that generates machine-monitoring data from process-planning data. While they provide a specific interface for using their turning model, we want to facilitate the use of the model in a more complex scenario. We develop a virtual machining model for a 3-axis milling operation extending the approach followed for the turning model. We also integrate the virtual machining model into an agent-based model in a simulation environment. This integration is the first step towards a virtual shop-floor model.

3 A Virtual Machining Model for Milling Operation

In this section, we describe a virtual machining model for a milling operation. The logic flow of the virtual machining model; the model's three components (the STEP-NC module, the Milling Operation Modeler, and the MTConnect Module) and the benefits of the model are described.

3.1 Logical Flow of the Virtual Machining Model

The virtual machining model is composed of three components: the STEP-NC module, the Milling operation modeler (MillOp), and the MTConnect Module. The virtual machining model follows a logical flow. **Figure 1** illustrates the model's logical flow, that is, a step-by-step process from receiving input of STEP-NC part program to generating its final output of an MTConnect data file. Information about tools involved and outputs generated at each step is also presented in **Figure 1**.

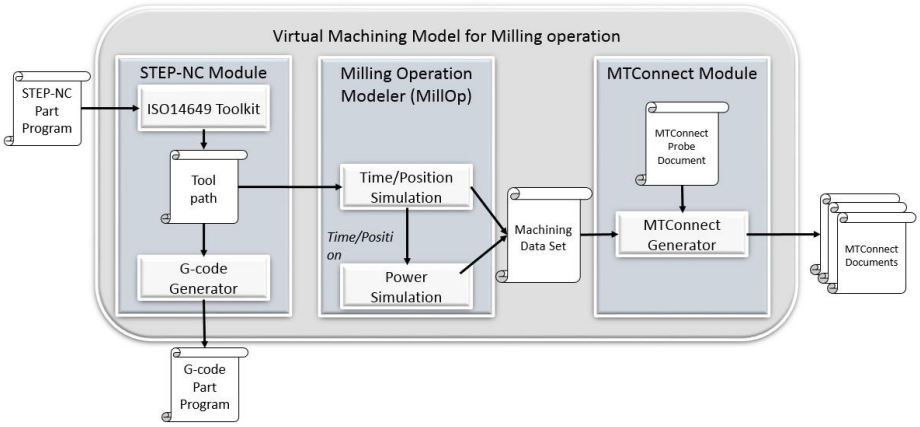


Figure 1. Logical flow of the Virtual Machining Model

The virtual machining model takes a STEP-NC part program as an input. The STEP-NC module interprets the part program and generates a tool path as well as a G-code part program, which is an NC program in the ISO 6983 format (ISO 1982). The STEP-NC module uses a toolkit referred to as ISO 14649 Toolkit in **Figure 1**. The tool path

output from the STEP-NC module is used as input to the MillOp. The MillOp computes kinematics and dynamics of the machine tool using physics-based equations and outputs a milling data set. The MTConnect module takes the milling data set as input and generates required data using the MTConnect format. We explain each of the model components in details in **Sections 3.2-3.4**.

3.2 STEP-NC Module

The STEP-NC module is composed of two components: 1) an ISO 14649 Toolkit, and 2) a G-code Generator. The ISO 14649 Toolkit (Kramer et al. 2006), developed at the National Institute of Standards and Technology (NIST), includes an ISO 14649 interpreter and an ISO 10303 (ISO 1994) interpreter. The STEP-NC module uses the ISO 10303 interpreter to analyze the characteristics of the ISO 10303 or STEP application-protocol file, and uses the ISO 14649 interpreter to parse and interpret STEP-NC files using ISO 14649's Parts 10, 11, and 111 (related to milling process data and tools). The Toolkit generates a tool path from the input STEP-NC file and its associated machine tool specification.

We developed a G-code generator that generates a G-code part program from a tool path. Using the G-code program, a G-code interpreter can generate a visualization of the tool path and we can check if the tool path matches the requirements described in the process plan.

Figure 2 shows the relationship among a STEP-NC part program, a tool path, and a G-code part program. The figure shows the communication of the required information

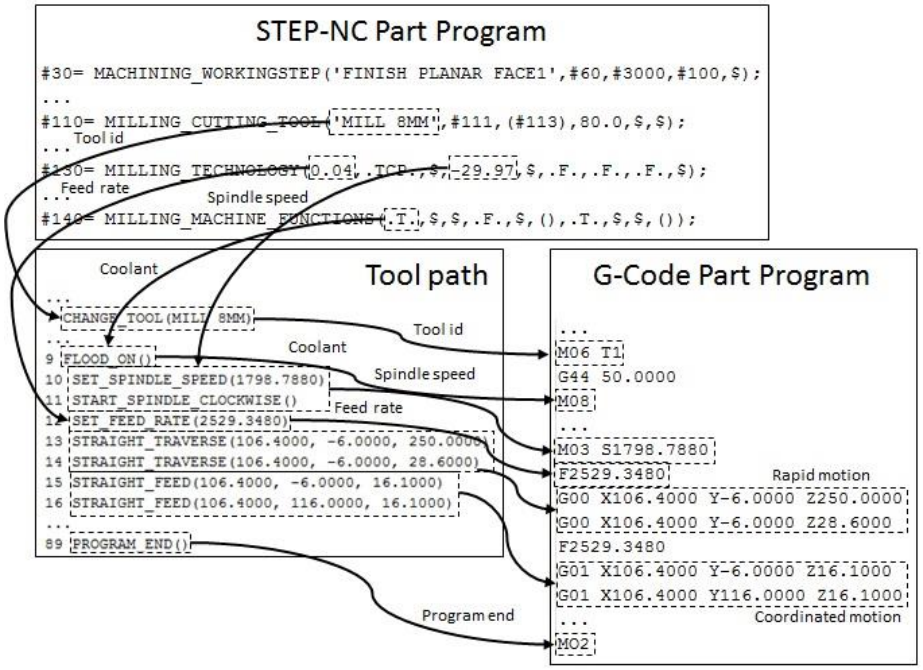


Figure 2. Mapping between STEP-NC, tool path and G-code

from a STEP-NC part program to a G-code part program via the tool path. In this example, the tool id, coolant command, feed rate, and spindle speed are included in the tool path based on the interpretation of the STEP-NC file. The tool path contains a set of motions for the tool to execute. The specifications of the motions are then included in the G-code part program. The G-code part program, in the **Figure 2** example, shows four types of commands: the command to change the tool based on the tool identification (M06 T1), the command to turn on the coolant (M08), and the commands to set up the spindle speed (M03 S1798.7880) and the feed rate (F2529.3480). In addition, a straight motion with no feed is identified as a rapid motion movement with the G00 instruction while a straight motion with feed is defined as a coordinated motion with the G01 instruction. It is important to note that different measurement units may be used in a STEP-NC part program and a G-code part program. For example, the spindle speed and feed rate values are different between the programs. In a G-code part program, setting up feed rate is necessary when a motion comes with a feed rate. This rule explains why the instruction F2529.3480 appears twice in the G-code part program. The first occurrence of the command sets up the feed rate, and at the same time it also appears in the tool path. The second occurrence appears in the G-code before a coordinated motion occurs.

3.3 The Milling Operation Modeler

The MillOp provides capabilities to compute dynamics and kinematics of machining operations. A STEP-NC program organizes a sequence of machining operations statically. From this program, a machine-executable G-code program is created. To simulate machine operation, it is necessary to compute the machine tool's events and movements, which are matched with sequential execution of the NC program. It is also necessary to compute the metrics based on kinematics (e.g., velocity and position) and dynamics (e.g., force and power) for the events and movements. This computation requires some properties of the machine tool, which are defined in the machine tool specification because machine tool properties often influence the simulation of a machine tool's kinematics and dynamics. In addition, the equations derived from physical model-based analysis for the metrics on a machine tool are needed for computation. In this paper, we compute the metric of power, that indicates the amount of energy consumed per unit-time by a milling machine tool.

The machine tool specification defines the capability and performance of a machine tool's main body and its constituent components. **Table 1** shows the properties of the machine tool specification used to describe the machine tool's capabilities.

Table 1. The property list for machine tool specification.

Component	Property	Symbol	Unit
Main body system	Basic power	P_b	W
Coolant system	Cooling power	P_c	W
Linear axis system	Acceleration/deceleration coefficient	a_L	m/s^2
	Rapid movement speed	v_i	m/s
	Pitch of feed screw	h_p	m/rad
	Friction coefficient in guide	μ_{gf}	-

	Table mass	m_t	kg
	Friction coefficient in bearing	μ_b	-
	Gear reduction ratio	r_g	-
	Acceleration of table	dw/dt	m/s^2
	Viscous damping coefficient	B	Nm/(rad/s)
	Pre-load force	F_p	N
	Servomotor efficiency	η_L	-
Rotary axis system	Rotor diameter	D_{rotor}	mm
	Rotor length	L_{rotor}	mm
	Friction coefficient	F_{r_coeff}	Ns/mm^2
	Gap between rotor and stator	gap	mm
	Application inertia	J_{app}	$kg*m^2$
	Rotor inertia	J_{rotor}	$kg*m^2$
	Acceleration/deceleration coefficient	α	rad/s^2
	Friction torque in the front bearing	T_{frfb}	Nm
	Friction torque in the rear bearing	T_{frrb}	Nm
	Spindle motor efficiency	η_S	-

We can use these properties to compute the metrics such as positions of the tool axial components and the corresponding energy use. For our model, we consider the metrics related to the linear and rotary axes as well as the coolant system. Each of these properties impacts the metrics that are computed in the MillOp. For instance, movement speed affects the time it takes the cutting tool to traverse to a new point. An increased coefficient of friction results in greater power consumption during cutting. To compute power values on a machine tool, the MillOp simulates times of machine components' motions, and then calculates linear-axial positions as a function of time. These positional data can be used to differentiate cutting and non-cutting motions by the cutting tool, and thus determine whether power consumption is at a cutting or idle rate.

First, we defined a position function by deriving theoretical equations presented by Avram and Xirouchakis (2011). This function is presented in Equation (1) assuming that linear velocity has a trapezoidal profile.

$$\begin{aligned} &\textit{if } 0 \le t \le t_a, \quad \textit{then } L_i(t) = 0.5 a_L t^2 \\ &\textit{else if } t_a \le t < t_a + t_s, \quad \textit{then } L_i(t) = 0.5 a_L t_a^2 + v_i (t - t_a) \\ &\textit{else if } t_a + t_s \le t < t_a + t_s + t_d, \quad \textit{then } L_i(t) = 0.5 a_L t_a^2 + v_i t_s + 0.5 T (2 v_i - a_L T), T = t - t_a - t_s \end{aligned} \quad , \quad (1)$$

Where, L: length from a previous point (mm), t: the current time (ms), t_a: acceleration time (ms), t_s: steady-state time (ms), t_d: deceleration time (ms), v_i: velocity on each axis (m/s).

Using this function, the MillOp computes the kinematics that include linear-axial positions as a function of time. These position data can be used to detect cutting or non-cutting motions that occur. The characterization of the motions contributes to the determination of power consumption.

Second, the MillOp computes the machine tool dynamics using the theoretical equations introduced by Altintas (2012). The power profile of a single NC code command for linear movement consists of acceleration, steady, and deceleration states. Power consumption during the steady state varies for cutting and non-cutting motions. During cutting motion, the power corresponds to the idle power plus the cutting power,

which results from cutting forces. We use a physics-based equation, as expressed in Equation (2), to calculate the cutting forces.

$$\begin{aligned} F_t &= K_{tc} bh + K_{te} b \\ F_f &= K_{fc} bh + K_{fe} b \end{aligned}, \tag{2}$$

where, F_t: tangential force (N), F_f: feed force (N), K_{tc}: tangential cutting coefficient, K_{fc}: feed cutting coefficient, K_{te}: tangential edge coefficient, K_{fe}: feed edge coefficient, b: cutting depth (mm), h: uncut chip thickness (mm).

Equations (3) and (4), respectively, present the linear-axial and rotary-axial power for a milling machine.

$$\begin{aligned} P_{L,a} &= \frac{T_a w}{\eta_L}, \quad T_a = J_e \frac{dw}{dt} + Bw + T_s \\ P_{L,s} &= \frac{T_s w}{\eta_L}, \quad T_s = (T_{gf} + \frac{\mu_b d_p (F_f + F_p)}{2} + T_f) / r_g \\ P_{L,d} &= \frac{T_d w}{\eta_L}, \quad T_d = -J_e \frac{dw}{dt} + Bw - T_s \end{aligned}, \tag{3}$$

where, $P_{L,a}$: acceleration power (W), $P_{L,s}$: steady-state power (W), $P_{L,d}$: deceleration power (W), w: angular velocity (rad/s), η_L: servomotor efficiency, T_a: acceleration torque (Nm), T_s: steady-state torque (Nm), T_d: deceleration torque (Nm), J_e: total inertia (kg/m2), B: viscous damping coefficient (Nm·s/rad), T_{gf}: torque friction in a guide way (Nm), μ_b: friction coefficient in bearing, d_p: feed-screw diameter (mm), F_f: feed force (N), F_p: pre-load force (N), T_f: torque by cutting force (Nm), r_g: gear reduction ratio.

$$\begin{aligned} P_{S,a} &= \frac{(T_{S,a} + T_{run})w}{\eta_S} \\ P_{S,s} &= \frac{T_{run} w}{\eta_S} \\ P_{S,c} &= P_{S,s} + \frac{2\pi F_c v_c}{\eta_S} \\ P_{S,d} &= \frac{(-T_{S,a} + T_{run})w}{\eta_S} \end{aligned}, \tag{4}$$

where: $P_{S,a}$: acceleration power (W), $P_{S,s}$: steady-state power during non-cutting (W), $P_{S,c}$: steady-state power during cutting (W), $P_{S,d}$: deceleration power (W), $T_{S,a}$: acceleration torque (Nm), T_{run}: steady torque (Nm), F_c: resultant cutting force (N), v_c: tangential cutting speed (mm·rad/min), w: angular velocity (rad/s).

The MillOp manipulates the information contained in the tool path and computes the different dynamics and kinematics using the equations described above. The MillOp includes a concept called *Movement.* For each motion in the tool path, the MillOp creates a new movement with all the current machine parameters (e.g., feed rate, spindle speed, and coolant) and the properties of the machine tool specification (e.g., rapid movement speed, and friction coefficient). An overview of a class diagram represented in UML (OMG 2015) that represents the *Movement* structure is shown in **Figure 3**. In

a UML class diagram, each concept is represented by a class. A line with a white diamond represents a containment relationship, with a numerical range at one end denoting the number of allowed instances. Triangular ends represent subclass relationships. In this diagram, we define a class, called *Movement*, which is extended by another class, called *StraightMovement* that is itself extended to by two classes called *TraverseStraightMovement* and *FeedStraightMovement*. These two classes define two movement types for the milling machine that represent different equations to compute the required metrics. The class diagram can be extended to represent additional movement types in the future.

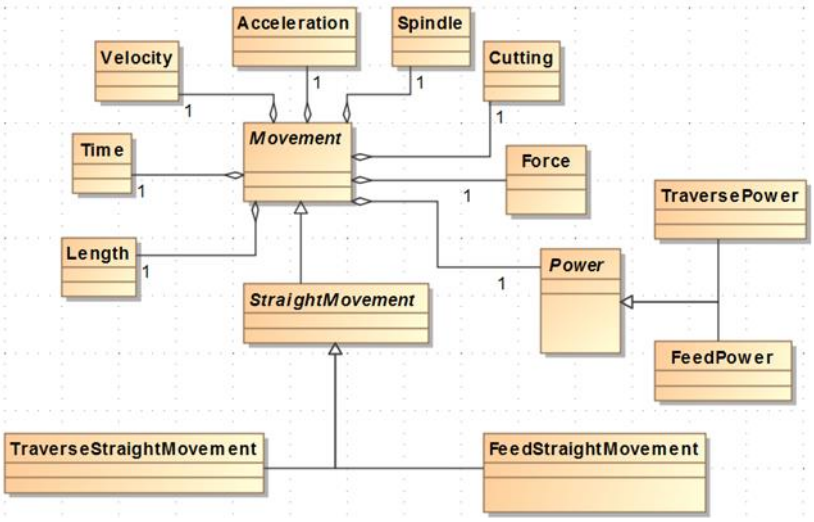


Figure 3. Class diagram representing the structure of a movement

The computed metrics are also represented as classes and are aggregated in the *Movement* class. The computed movement metrics are length, acceleration, velocity, time, cutting, spindle, force, and power. The class *Power* is extended by two classes: *TraversePower* and *FeedPower* that represent the power depending on the movement type. The *TraversePower* class contains the equations for computing power during a *TraverseStraightMovement*, while the *FeedPower* class contains equations to compute power during a *FeedStraightMovement*. When a new movement is created, the MillOp satisfies the equations to compute the metrics using the properties included in the movement. Depending on the type of the movement, some metrics are computed differently. For instance, *Cutting* in a *TraverseStraightMovement* contains attributes with no value since there is no cutting operation for this type of movement.

The MillOp generates a collection of *Movement* instances that represent the machine tool kinematics and dynamics during the processing of one workpiece, i.e., for one STEP-NC part program.

3.4 The MTConnect Module

Using the kinematics and dynamics generated by the MillOp, the MTConnect module creates an MTConnect document representing the position and power consumption of the tool. **Table 2** summarizes the data items available in an MTConnect document for the virtual machining model.

X-axis position and power consumption are available using *x_axis_position_actual_sample* and *x_axis_wattage_sample*. Data are available for y-axis and z-axis using the corresponding data items. The power consumed for running the spindle tool is available using *c_axis_wattage_sample*; the power consumed by the machine for the coolant flow is available using *coolant_wattage_sample*.; and the power consumed for running the machine is available using *electric_wattage_sample*.

For every machining data set generated by the MillOp, the MTConnect module generates an MTConnect document that includes time series data about the position and power consumption of the tool. The interval of time between two data values for the same data item is 100 milliseconds. We generate a corresponding MTConnect document for each STEP-NC file used as input in the virtual machining model. Thus, we observe the impact of the process plan on the machine power consumption even before executing the process plan.

Table 2. MTConnect data items available for the virtual machining model

Data items	Type	ID	Unit
X_AXIS_POSITION_ACTUAL_SAMPLE	POSITION	x_axis_position_sample	Millimeter
X_AXIS_WATTAGE_SAMPLE	WATTAGE	x_axis_wattage_sample	Watt
Y_AXIS_POSITION_ACTUAL_SAMPLE	POSITION	y_axis_position_sample	Millimeter
Y_AXIS_WATTAGE_SAMPLE	WATTAGE	y_axis_wattage_sample	Watt
Z_AXIS_POSITION_ACTUAL_SAMPLE	POSITION	z_axis_position_sample	Millimeter
Z_AXIS_WATTAGE_SAMPLE	WATTAGE	z_axis_wattage_sample	Watt
C_AXIS_WATTAGE_SAMPLE	WATTAGE	c_axis_wattage_sample	Watt
COOLANT_WATTAGE_SAMPLE	WATTAGE	coolant_wattage_sample	Watt
ELECTRIC_WATTAGE_SAMPLE	WATTAGE	electric_wattage_sample	Watt

3.5 Benefits

Having virtual machining models provides different benefits. Virtual machining models provide capabilities to create machine-monitoring data. These data can be used to observe the behavior of a machine for a given process plan. They can also play an important role to support the collection of real data. During data acquisition in a factory, errors might occur that lead to the generation of data sets with missing or inaccurate values. The problem of missing data for analysis has been studied for years (Lakshminarayan et al. 1999). Virtual machining models provide simulation data that can fill the real data set when there are data-related problems such as missing data.

The data generated from a virtual machining model can also contribute to the generation of a data set for benchmarking. Small and medium enterprises (SMEs) can use the data set for testing their data analytics solutions before making investments, such as investing in expensive sensors for collecting real data. The results can help guide further investment decisions.

Finally, virtual machining models allow users to change properties of the machine by changing the values in the associated XML document. The process plan can also be altered by changing the STEP-NC program. Kusiak (2017) highlighted that real-world experimentation is needed, using virtual reality among others to improve manufacturing systems. Changing the machine properties and the process plan, a manufacturing engineer can execute the virtual machining model to observe the impact of the parameter values on the power consumption or estimate time required to manufacture a part. Based on the impact analysis, the real machine parameters or the process plan are adjusted to optimize certain metrics of concern to the engineer.

4 Integration of the Virtual Machining Model into an Agent-based Model for Simulation

In this section, we describe the integration of the virtual machining model into an agent-based model for simulation. We first introduce the specification of an agent-based model that integrates with the virtual machining model. We also describe the execution of the agent-based model in a simulation environment and demonstrate the validation of the simulation data.

4.1 Machine State Chart

To build the agent-based model, we first develop a state chart diagram that describes the behavior of the milling machine we address. The state chart, shown in **Figure 4,** is implemented in the agent-based model for execution. It includes of six different states. The default machine state is the *idling* state. As soon as a batch arrives (represented by the transition called *batchReception* in the figure), the machine goes to the next state called *batchSetup*. The *batchSetup* state models the required machine setup for processing the batch. Once the batch is set up, the machine goes to the *partSetup* state where the machine sets up needed operations for a particular part. The next state, called *machining*, represents the milling process. Once the operations have been executed on the part, the machine goes to the *partEjection* state that models the unloading of the part. After the *partEjection* state, two alternative paths can be taken by the machine. When there are more parts to process in the batch, the machine goes back to *partSetup*, otherwise, the machine goes to the last state, *batchEjection*, where the batch is

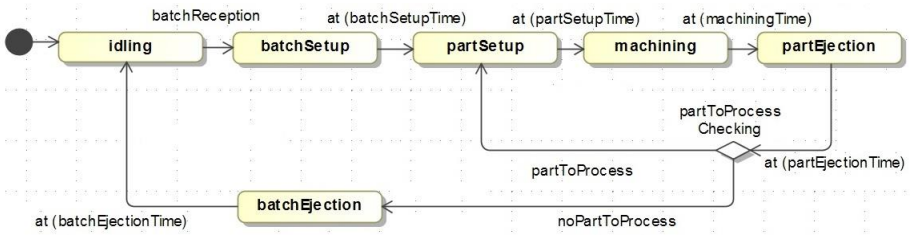


Figure 4. Machine state chart of the agent-based model

unloaded. After a batch has been ejected, the machine goes back to the *idling* state to wait for a new batch.

We define the *machining* state to enable the integration of the virtual machining model. The software environment called AnyLogic (Borshchev and Filippov 2004) allows us to customize the instructions for the execution of the states in an agent-based model using Java code. We generate a Java ARchive (JAR) file, which contains the virtual machining model functions, and include it in the AnyLogic project to use these functions at the *machining* state.

A duration and a power consumption measure are associated with each state. They can be collected during individual states except the *machining* state. **Figure 5** shows the time and power required for the states of the machine in an XML file. The XML file provides time and power for every state and all the property values required for the virtual machining model. For instance, we collect the time and the power required to set up the batch at the *batchSetup* state collecting the value of the attributes *batchSetupTime* and *batchSetupEnergy*.

```
<Machine id="TurningMc1"
    partSetupTime="30000" partEjectionTime="20000" batchSetupTime="60000"
    batchEjectionTime="30000" partSetupEnergy="1206" partEjectionEnergy="976"
    batchSetupEnergy="4400" batchEjectionEnergy="3400" basicRapidSpeed="1600"
    axisAccelation="6" pDefaultLoad="370" pDefaultCool="280"
    pitchFeedScrew="0.015" frictionCoefficientGuide="0.1" tableMass="30"
    workpieceMass="50" gravity="9.81" frictionCoefficientBearing="0.01"
    feedScrewDiameter="0.03" preloadForce="1800" gearReductionRatio="1"
    leadScrewMass="10" couplingInertia="0.0002875"
    viscousDampingCoefficient="0.005" axisServoMotorEfficent="0.930"
    angularAcceleration="10" rotorDiameter="5.7" rotorLength="0.3"
    frictionCoefficient="0.0000185" gap="2" frictionTorqueFront="0.2"
    frictionTorqueRear="0.2" applicationInertia="0.25" rotorInertia="0.25"
    spindleMotorEfficiency="0.95" randomProperty="0.05;0.1"
    randomType="Uniform"
/>
```

Figure 5. Machine specifications

All the values of time and power are subjected to a standard deviation to represent uncertainty associated with a physical machine. The distribution type and standard deviation of the uncertainty are defined by the attributes *randomProperty* and *randomType*. The attribute *randomProperty* contains the standard deviations for power during a traverse motion and a coordinated motion. The duration and power consumption for the machining state are computed using the virtual machining model. The value of the properties can be slightly different from one machine to another. Adjusting these values in the machine specifications allows one to use the properties values that correspond to the properties of the real machine and to obtain simulation values that are comparable to real values for the same operation.

Before transitioning to the next state, parameters and associated vales required for the virtual machining model are preset. For example, at the *batchSetup* state, we identify the process parameters – feed rate, spindle speed, and cutting depth – that control the tool path strategies and are necessary to make the given machined features. The values for these parameters are included in a STEP-NC program used by the virtual machining model. For the *machining* state, the duration and power consumption are

computed using the virtual machining model. During this state, we provide the STEP-NC program, complete with the parameter values, as an input to the virtual machining model. Using the appropriate functions, we can compute the machining time and the consumed power corresponding to the STEP-NC file given as input. In *partEjection* state and *batchEjection* state, we collect time and power consumed to achieve these ejection operations as we do for the setup states. Once a batch has been processed (after the *batchEjection* state), we generate Comma Separated Values (CSV) that aggregate the time data for the machine and MTConnect output files that provide time series data for the milling machine tool.

4.2 Execution of the Agent-based Model in a Manufacturing Use Case

In this section, we describe the execution of the agent-based model and the integrated virtual machining model using a manufacturing use case. We define a scenario to represent a milling machine in the simulation environment. In this scenario, a milling machine tool processes a steel part as shown in **Figure 6**. The steel part is processed to produce different pockets, slots, and holes. For this use case, we focus only on the tool path strategy for machining Pocket 1, a series of straight movements to create the square shape.

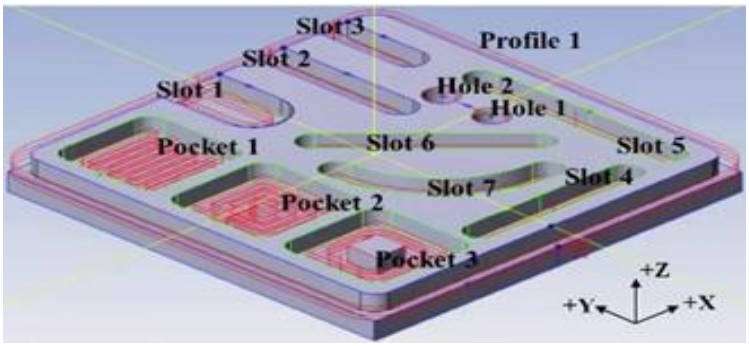


Figure 6. An example of a milling part

To implement this scenario, we create a process-flow model including our agent-based model that will simulate the flow of batches coming to the machine and the execution of the operation for Pocket 1 in the machine. This process flow model, as shown in **Figure 7**, represents a flow of batches coming to and leaving from the milling machine, the agent-based model we developed.

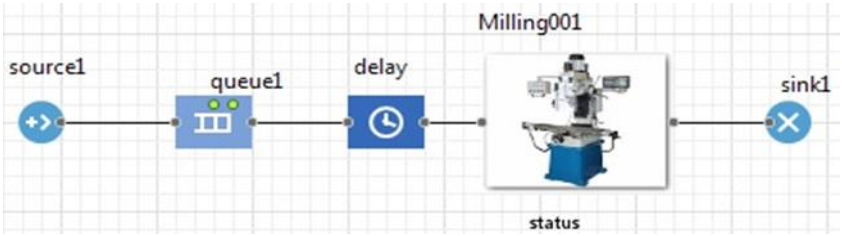


Figure 7. Process flow model

When we execute this model, the batch leaves *source1* to be held in *queue1*. When its turn comes, it moves on to *delay*. Once in *delay*, the batch waits until the milling machine is available, i.e., the machine is in *idling* state. Once all the parts of the batch have been processed, the batch goes to *sink1*. Each batch is composed of 10 steel parts that need to be machined.

Figure 8 shows the execution of the agent for this scenario. As soon as the execution of the process flow starts, the execution of the agent starts as well. The milling machine is in the *idling* state and goes to the *batchSetup* state as soon as a batch arrives at the machine. In the *batchSetup* state, the process parameters required for the virtual machining model are assigned randomly using a uniform distribution within the following ranges: feed rate [30, 90] mm/s, spindle speed [75.4, 226.2] rad/s, and cutting depth [2.5, 3.5] mm. The parameter values are included in a STEP-NC file that defines the sequence of processes to manufacture Pocket 1.

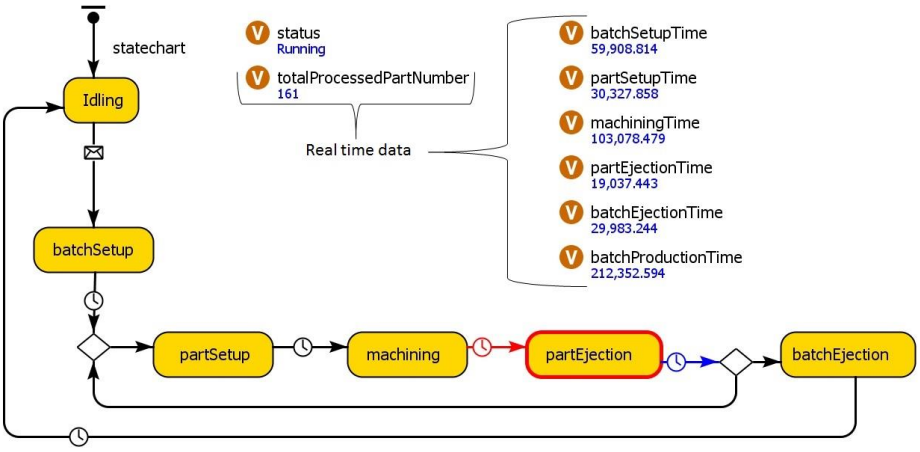


Figure 8. State chart execution in the agent

In the *machining* state, the STEP-NC file is used by the virtual machining model to simulate the execution of the milling machine. A tool path and a G-code program are generated. To validate these files, we use a G-code interpreter to draw the path of the tool. **Figure 9** shows the top view of the tool path in a G-code program interpreter.

The path shown in **Figure 9** corresponds to the expected path of the scenario. This confirms that the generated tool path from the STEP-NC program is correct and can be used by the MillOp of the virtual machining model. The MillOp satisfies the equations and generates an MTConnect document containing the time series data for each part processed. All along the execution, the agent provides real time data that are displayed as shown in **Figure 8**. These real data represent machine level data while the execution of the virtual machining model during the *machining* state represents process level data. Using an identical set of process parameters in the virtual machining model could result in different power values (±10 % uniform-random deviation during feed movement, and ±5 % uniform-random deviation during traverse movement). This represents the variation that may be encountered in real machine behavior. Values of these deviations can be adjusted in the machine specification to match with the variations of the actual machine.

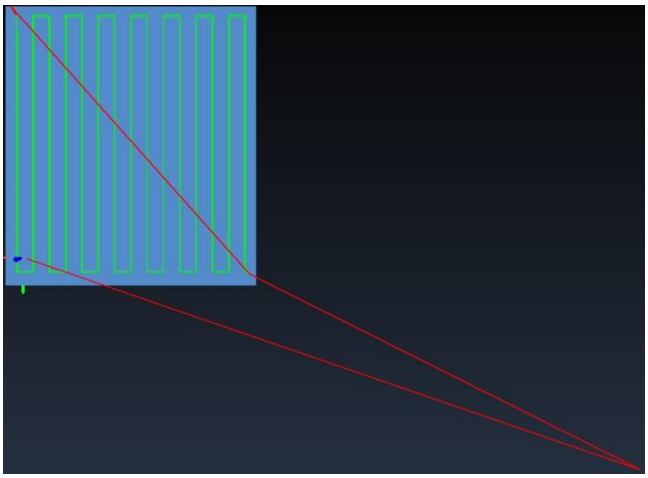


Figure 9. Top view of the tool path

The MTConnect documents are stored locally and named using the machine name and the identification of the related part. This allows the user to quickly find the MTConnect document related to a given part. **Figure 10** shows an example of an MTConnect document representing the tool position and the consumed power at a given time for the y-axis.

In addition to the information defined in the element *MTConnectStreams* (which

```
<?xml version="1.0" encoding="UTF-8"?>
<MTConnectStreams xmlns:m="urn:mtconnect.org:MTConnectStreams:1.2"
xmlns="urn:mtconnect.org:MTConnectStreams:1.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mtconnect.org:MTConnectStreams:1.2
http://www.mtconnect.org/schemas/MTConnectStreams_1.2.xsd">
    <Header creationTime="2016-04-10T09:52:00.000" sender="mtconnect"
    instanceId="100" version="1.2.0.10" bufferSize="131072"/>
    <Streams>
        <DeviceStream name="MILLING_MACHINE_1" uuid="MILLING_MACHINE_1">
            <ComponentStream componenetId="y_axis" component="Y_AXIS" name="Y_AXIS">
                <Samples>
                    <Position dataItemId="y_axis_path_position_actual_sample"
                    name="Y_AXIS_PATH_POSITION_ACTUAL_SAMPLE" sequence="559"
                    subType="ACTUAL" timestamp="2016-04-10T09:52:06.066Z">116.0</Position>
                    <Wattage dataItemId="y_axis_wattage_sample" name="Y_AXIS_WATTAGE_SAMPLE"
                    sequence="562" subType="ACTUAL"
                    timestamp="2016-04-10T09:52:06.066Z">0.0</Wattage>
                </Samples>
            </ComponentStream>
        </DeviceStream>
    </Streams>
</MTConnectStreams>
```

Figure 10. MTConnect sample

represent a collection of streams between a device and an agent), the MTConnect document defines a component, which is the source of the data in the element *ComponentStream* using the attribute *componentId*. A *ComponentStream* contains the information specific to the component within the Device. In this case, the component is the y-axis of the tool. Then, a *Sample* (concept defined in MTConnect to contain the data collected at a given time) is defined for this component that includes the *position*

and the *wattage* of the tool at the time provided in the attribute *timestamp*. The coordinate of the position on the y axis is 116 in this example. The wattage is the power consumed by the tool on the y axis at this time. In this example, there is no power consumption at this given time on this axis, which means that the tool is not now moving.

In real time, the agent-based model also aggregates data in a CSV document as shown in **Figure 11**. The aggregated data are identification of the part, material of the part, machine feed rate, machine spindle speed, cut depth of the machine, time to set up the machine, time to set up the part, time to machine the part (computed by the virtual machining model), time to eject the part, and time to eject the batch.

id	material	FR	SS	CD
0-0	steel	0.062695	-13.0768	2.838393
0-1	steel	0.058225	-27.9415	3.33384
0-2	steel	0.03982	-15.566	2.747615
0-3	steel	0.055288	-34.8973	3.182611
0-4	steel	0.061828	-34.8362	3.45596
0-5	steel	0.034072	-31.6083	3.013225
0-6	steel	0.032425	-15.3552	2.771119
0-7	steel	0.049753	-26.8478	2.824905
0-8	steel	0.033455	-24.3006	2.647896
0-9	steel	0.050983	-17.5584	3.425454

id	batchsetuptime	partsetuptime	machiningtime	partejectiontime	batchejectiontime
0-0	62628.9248	28766.58202	69063.26749	20038.03902	0
0-1	0	28573.81032	75848.96485	19731.22623	0
0-2	0	29519.13605	105958.9712	18766.82392	0
0-3	0	29553.60804	81247.20484	21829.35219	0
0-4	0	30412.31669	74285.72275	19275.90305	0
0-5	0	30420.64995	124555.7274	19750.98096	0
0-6	0	30957.2421	128437.9	19484.85633	0
0-7	0	28465.5163	88752.80297	20755.16166	0
0-8	0	30551.79807	125943.9701	19071.51644	0
0-9	0	29869.89036	84647.07421	19208.27402	29853.72316

Figure 11. Aggregated data

4.3 Data validation

The virtual machining model, presented in previous subsections, has been "partially" validated through the comparison of actual power data with the simulation power data of the milling machining. In general, data validation is performed under two scenarios: 1) The machine tool specification parameters are known, or 2) the machine tool specification parameters are impossible to acquire. Under the first scenario, data validation is performed by confirming simulation data are matched correctly with actual data. In the second scenario, data validation can only be done up to data calibration. The values of machine tool specification parameters are adjusted to make the simulation data be as correct as the actual data. In this experiment, we applied the second scenario due to the difficulty in acquiring machine specification parameters. Nevertheless, the

data calibration that we used would be an effective and practical way to validate the data generated by simulation models when all the machine tool specifications are not obtainable.

We used two different cases to validate the model as follows:

- Case 1:
  - feed rate: 124.46 mm/min
  - spindle speed: 183.3 rad/s (1750 RPM)
  - cutting depth: 1.5mm
- Case 2:
  - feed rate: 88.9 mm/min
  - spindle speed: 183.3 rad/s (1750 RPM)
  - cutting depth: 2mm

The number of cutting layers depends on cutting depth. In Case 1, there are four cutting layers, on the other hand, Case 2 has three cutting layers. **Table 3** presents the setup of experiments.

Table 3. Setup of Experiments

Property	Condition
Machine tool	Mori Seiki NVD 1500 DCG
Computerized Numerical Controller	Fanuc controller 0i
Workpiece	Cold finish mild steel 1018 (size: 10.16cm*10.16cm*1.27cm)
Cooling option	Wet
Cutting tool	Solid carbide
Tool geometry	Flat end mill (8mm diameter, 4 number of flutes)
Power measurement device	System insights high speed power meter (sampling interval: average 0.3 sec)

The power measurement device measures the summed power by all the machine components on average 0.3 sec. **Figure 12** shows the machined part and its machining feature (i.e., Slot 3 with 6mm thickness) that we used to compare actual data with the simulation data. This figure also includes relevant tool paths where we focus especially on feed and back movements.

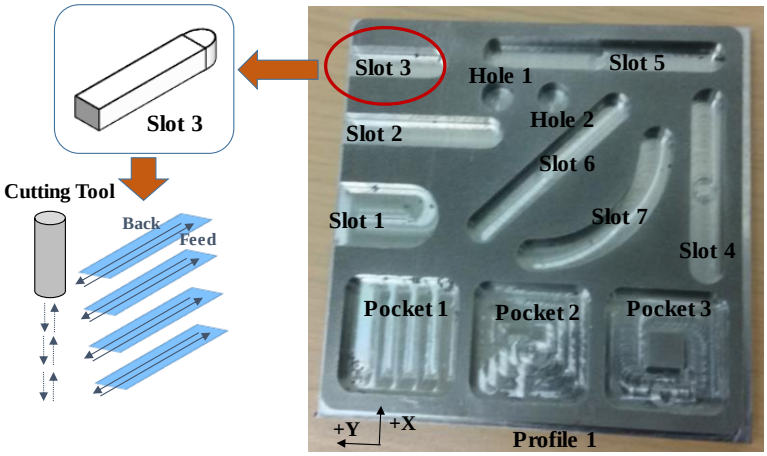


Figure 12. The machining feature and tool paths of the machined part used for data validation

Figure 13 presents time-series charts that plot the simulated and actual-measured powers over time for the two cases. **Figure 13** (a) corresponds to Case 1, and **Figure 13** (b) to Case 2. The trend of simulated power over time matches with actual power. Some differences take place due to heterogeneous nature of simulation with real phenomena. We marked these differences in **Figure 13** and analyzed them as follows:

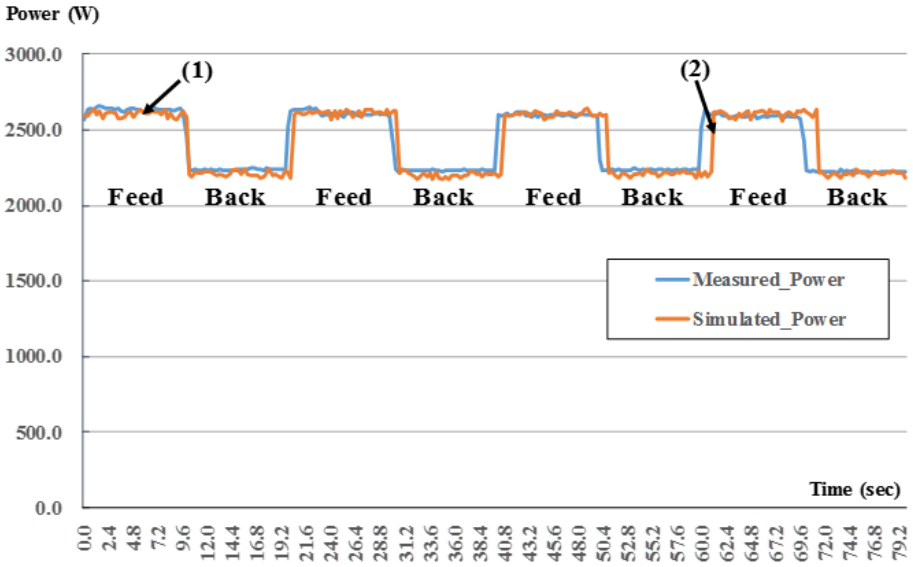
1) **Data fluctuation**: Actual power data typically contain fluctuation of data distributions during machining because of complex problems, including measurement errors, misalignment of a cutting tool, and vibration of a machine itself. This complex phenomenon unavoidably makes differences between our virtual machining model and real machining. To accommodate this problem, the virtual machining model is designed to control and generate random data distribution.

2) **Slight time difference**: Actual machining occasionally takes more time than the time designated by the input of feed rate. The reasons might be that the resistance of the cutting tool with the workpiece makes the tool movement slightly slow or the operator controls override mode in a CNC slows down the machining due to machine tool safety and tool breakage prevention.

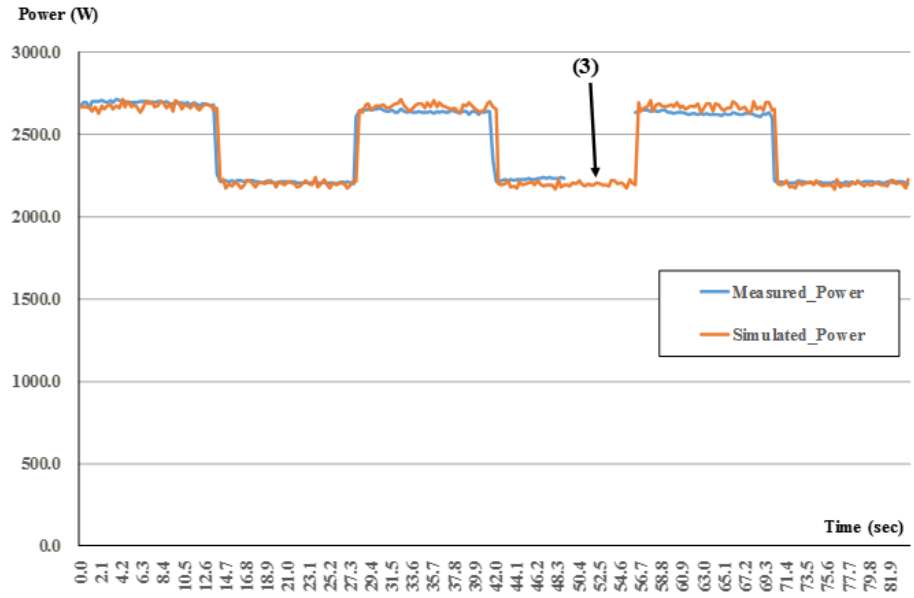
3) **Real data missing**: Missing or erroneous data are unavoidable in the actual data recorded by a physical measurement device. In this experiment, real data are missing and thus it makes the correctness of simulation result undecidable for a specific period of time. Nevertheless, we conjecture that the virtual machining model generates the simulation data as good as it appeared in the trends of the previous and next data in the same conditions.

Data validation in simulation is challenging due to the difficulties in collecting actual data or machine tool specification parameters and the limitation of simulation itself. In this experiment, we are able to compare the actual-measured and simulation data in one

simple machining feature: slotting. Machining a slot is not simple but it provides more explicit information than other machining features. Data validation on all machining features would be a future work for making our virtual machining model more practical.



(a) Case 1 (feed rate = 124.46 mm/min, spindle speed = 1750 RPM, cutting depth = 1.5mm)



(b) Case 2 (feed rate = 88.9 mm/min, spindle speed = 1750 RPM, cutting depth = 2mm)

Figure 13. Time-series charts for measured power and simulated power

In addition to the graphical validation, we numerically validated the power simulation data. We computed the Root Mean Square Error (RMSE) of the simulated power values compared to actual power values in the two described cases. In case 1, RMSE is 99.0 Watts (W). In case 2, RMSE is 67.5 W.

Energy is another important metric that can be calculated from power values and time. Generally, the energy metric is used for energy-efficient machining instead of power because energy-efficient machining results in minimizing energy consumption (electricity use), and not minimizing power. Using time and power, we calculated the measured energy and the simulated energy in Joules for each case. We then compared the values by calculating the relative error defines as follow:

Relative error = 100*(Measured energy – Simulated energy) / Measured energy

In case 1, relative error is 0.46%. In case 2, relative error is -0.07%. Thus, the virtual model provides accurate data in the two scenarios that have been studied.

5 Towards a virtual factory model

In this section, we present how to generate a virtual factory model from a collection of machining process models. We also discuss the motivations for and challenges involved in generating a virtual factory model.

5.1 Motivation and related work

The companies that possess virtual machining models have obvious technological and economical advantage. For instance, Procter & Gamble's simulation models helped improve the reliability of their complex production lines (Manyika 2012). Their simulations led to a 44 percent increase in plant productivity and savings of $1 billion in manufacturing costs globally.

A virtual factory model should represent the main features and operations of the system, i.e., the factory, in order to analyze the system. A virtual factory model's primary objective is to observe the behavior of the factory in response to the changes from its subsystems. Virtual factory models can be used to predict the system performance, compare alternative solutions for different change options, or generate simulation data as if they were real data. As an example, Terkaj, Tolio, and Urgo (2015) presented an ontology-based virtual factory for predictive analysis. The ontology can be continuously updated to reflect the events of the corresponding real factory. They use the ontology to assess the impact of production or maintenance decision on the factory.

A virtual factory model can be developed using a collection of machining models. Building a collection of machining models is similar to build a milling machine model, as described in Section 3. The approach we propose to build a virtual factory model uses the discrete event simulation (DES) technique. The virtual factory model would be composed of different, interconnected machining models (that are agent-based

models (ABMs) in this case) that represent the real factory. Such a model provides a multi-level simulation representing all machining processes, machines, and the layout of a shop floor. Schönemann et al. (2016) adopted this approach to propose a multi-level modeling and simulation of manufacturing systems for lightweight automotive components. They used a DES model to represent the process chain and ABMs in order to show how the machines execute the processes. The energy consumption of each process is computed using a MATLAB program executed by the corresponding ABM. Implementing this approach using AnyLogic, energy consumptions for different parts in their process chain are then compared. Jain et al. (2015) also discuss the benefits of combining different agent-based models in a unique DES model of a virtual factory.

A multi-level virtual factory model provides capabilities to observe the behavior of the factory as a whole or at different levels of granularity such as at the process, machine, or shop floor levels. A multi-level model also allows manufacturing engineers to observe the impact of system modifications. With a multi-level model, a variety of modifications can be easily tested. Examples of modifications include shop floor re-configuration and machine parameter or process parameter changes. With the milling machine model represented as an agent-based model, for example, it allows changing machining or process parameters, and also, generating simulation data at the process and machine levels.

A manufacturing engineer can use the data generated at the process and machine levels to fill in missing values in a real data set. The simulation data can also be compared to the actual data to detect any anomalies. The detection of an anomaly should trigger maintenance or inspection operations or an investigation of the issue.

The data analytics models provide alternatives to systematic executions of simulation models. The simulation data could be used to train data analytics models for predicting specific outputs such as throughput or energy consumption. Scoring a data analytics model (i.e., using a trained data analytics model with new input data to make prediction) is not as time consuming as running a complex simulation model; it helps manufacturing engineers save time and make decisions more quickly. Running the simulation model, however, could provide more detailed information for observing the behavior of the system at different levels with unique scenarios.

In practice, the complexities and high costs of data collection preclude generation of real data across an entire factory. Multi-level virtual factory models could address data collection gaps by generating simulation data. Such simulation data can also be used to train data analytics model, as simulation data are theoretically nearly identical to data collected from the real factory. Jain and Shao (2014) proposed a virtual factory, which is a high-fidelity simulation of the manufacturing system, to support data analytics. Feldkamp, Bergmann, and Strassburger (2015) identified that applying data analytics to simulation data enables the identification of causal relationships that are not revealed using simulation models alone.

5.2 Building a virtual factory model

A library of agent-based models representing machining processes supports the generation of virtual factory models. We mentioned that Shin et al. (2016) developed a virtual machining model for turning. That machining model could be represented in an

agent-based model by using the methodology presented in Section 4. Using the same methodology, a robust set of machining process models, which include both dynamics and kinematics machining processes, could be generated. Eventually, the model set can be used to produce a library that is large enough to support the development of basic factory models. This proposed library can be extended to include varied types of machining process models, enabling the creation of much more complex factory models.

Using the machining process models available in the proposed library, a manufacturing engineer could represent a factory in a simulation model by selecting the appropriate agents in the library and using them to represent processes involved in the actual factory. As described in Section 4, the agent parameters should be easily modified to match with the actual machine parameters.

Figure 14 shows an example virtual factory model that is composed of a milling agent-based model (described in Section 4) and a turning agent-based model (including the virtual model developed by Shin et al. (2016)).

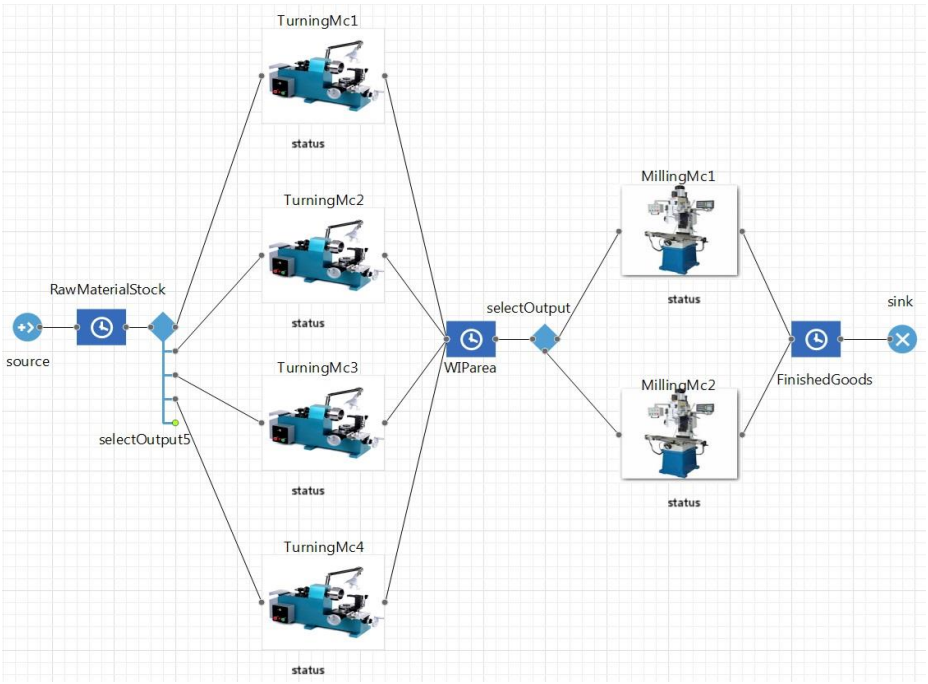


Figure 14. Virtual factory model representing an assembly line

The model of **Figure 14** demonstrates an assembly line scenario. Batches are held in the area called *RawMaterialStock* until one of the turning machines is available. As soon as a turning machine is available, a batch is sent to the machine and the parts of the batch are processed as described in Section **4.2**. Once the batch has been processed in the turning machine, it is sent to another area called *WIParea* until a milling machine is available. The milling machine will process the parts of the batch. Once the entire batch has been processed, it reaches the area of *FinishedGoods*.

A manufacturing engineer can simply build the example virtual model by using the agent-based models that are available in the proposed library and connecting them to represent the flow of batches as they occur in the real factory. In the given assembly line scenario, only turning and milling models are needed from the library. This example model represents a very basic assembly scenario but it is sufficient to show different levels of details.

At the factory level, the data collected from each machine can be aggregated to evaluate the assembly line's throughput or the energy consumption used to process all batches at the assembly line. The assembly line data, such as throughput and energy consumption, do not depend on the machine and process only. For instance, throughput varies with the availability of the machines involved in the factory. If machines are unavailable frequently, the batches stay longer in the waiting areas (e.g., RawMaterialStock or WIP areas). It, therefore, impacts throughput value.

Collecting actual factory data is often difficult due to the system's complexity. Simulation data, however, could be generated at the process level, at the machine level, or at the assembly line level by implementing the virtual machining model or the agent-based model, or aggregating the data resulted from several agent-based models, respectively.

In the example model, each agent, which represents a machine, could provide machine-monitoring data, in the MTConnect format, about energy consumption information and the location of the machine tool. A repository of MTConnect documents retains the monitoring data collected during the simulation. The agents also aggregate the data at the machine level to provide time and power consumed for processing a batch. Extending the agents with failure and maintenance states would enable the generation of data that better represent an actual machine behavior.

5.3 Challenges

Simulation has been identified as a key component towards Industry 4.0 (Hermann, Pentek, and Otto 2016). However, building a virtual factory model raises different challenges. First, building virtual models for each related machining process is time consuming and expensive. We anticipate proposed libraries, described in Section 5.2, could be created from 1) companies that have financial resources to and interests in building their own machining models, or 2) individuals or organizations who contribute their machining models through crowdsourcing.

Second, dynamics and kinematics are unknown or not easy to model for certain manufacturing processes. Since the number of processes that can be represented by their kinematics and dynamics is considered sufficient, we recommend focus on those processes to create real factory models. If dynamics and kinematics are not available, approximation models based on real data can be built in order to replace the physics-based models. The idea of approximation model is to use machine learning techniques to approximate complicated relationships between the inputs and the outputs of a process.

Third, an elaborate factory model with different levels of detail requires high computational capabilities. We believe this challenge can be overcome by taking advantage of a new breakthrough on cloud computing and parallel computing in

graphical process units that has unlocked new computational capabilities. A growing number of companies has already taken advantage of these capabilities to perform data analytics and deep learning towards artificial intelligence. Manufacturing companies should also leverage these capabilities to execute more complex simulation models efficiently and hence increase sustainability, productivity, flexibility, and competitive advantages.

6 Conclusion

This paper highlights two capabilities that enable manufacturers to apply data analytics for improving their operations: 1) the development of a virtual machining model that represents manufacturing processes of the machine, and 2) the integration of virtual machining models into agent-based models for simulating shop floor. We described an approach to develop virtual machining model, proposed an approach to include the virtual machining model in an agent-based model, and introduced how a virtual factory model be built based on a proposed library of agent-based models.

As an example, a virtual machining model for milling operation was developed. The virtual machining model supports the generation of machine-monitoring data in the MTConnect format from a process plan presented in a STEP-NC file. An agent-based model, i.e., a milling machine model, was created to include the virtual machining model to simulate the execution of the milling machine in a shop-floor scenario. In the same example, the milling machine model and another turning machine model were integrated in a simulation environment to demonstrate a simple virtual factory for simulating assembly line processes. A manufacturing engineer can simulate the execution of the machines and adjust the process plan before executing it in a real factory.

We described how to combine different agent-based models built in a similar approach, described in this paper, to create multi-level simulation models representing factories. With this proof-of-concept approach, a collection of agents can then be created to enable the generation of a virtual shop-floor model with finer levels of granularity. With the virtual shop-floor model, simulation data could be generated at the process, machine, and shop-floor levels. It offers new capabilities for the application of data analytics at the shop-floor level. The simulation data could be used to train data analytics models for predicting specific information at the factory level. Manufacturing companies could also anticipate potential returns before investing in new sensors for data collection or new data analytics software.

Future work lies in three directions. The first is the integration of maintenance and failure into our model. This integration can make data generation more realistic and enhance our simulation capabilities. The second is the development of additional virtual machining models and agent-based models for different operations with the goal for establishing a library for building virtual shop-floor models. The third is studying the application of data analytics techniques on data aggregated from virtual shop-floor models. These efforts will help manufacturers realize the benefits of multi-level modeling that we described in this paper.

ACKNOWLEDGEMENT

The research in this paper was conducted as part of the NIST program on the Design and Analysis of Smart Manufacturing Systems and supported in part by National Institute of Standards and Technology's Foreign Guest Researcher Program.

This research was partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1C1B1008820).

The authors would like to acknowledge KC Morris and Moneer Helu whose suggestions helped improve and clarify this manuscript.

DISCLAIMER

No approval or endorsement of any commercial product by NIST is intended or implied. Certain commercial software systems are identified in this paper to facilitate understanding. Such identification does not imply that these software systems are necessarily the best available for the purpose.

References

Altintas, Yusuf. 2012. *Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations, and CNC Design*. Cambridge university press.

Avram, Oliver Ioan, and Paul Xirouchakis. 2011. "Evaluating the Use Phase Energy Requirements of a Machine Tool System." *Journal of Cleaner Production* 19 (6): 699–711.

Borshchev, Andrei, and Alexei Filippov. 2004. "Anylogic—Multi-Paradigm Simulation for Business, Engineering and Research." In *The 6th IIE Annual Simulation Solutions Conference*, 150:45.

Bouhadja, Khadidja, and Mohamed Bey. 2015. "Survey on Simulation Methods in Multi-Axis Machining." In *Transactions on Engineering Technologies*, 367–382. Springer.

Brown, J. Ethan, and David Sturrock. 2009. "Identifying Cost Reduction and Performance Improvement Opportunities through Simulation." In *Winter Simulation Conference*, 2145–2153. Winter Simulation Conference. http://dl.acm.org/citation.cfm?id=1995750.

Feldkamp, Niclas, Sören Bergmann, and Steffen Strassburger. 2015. "Visual Analytics of Manufacturing Simulation Data." In *Proceedings of the 2015 Winter Simulation Conference*, 779–790. IEEE Press.

Flanagan, David. 2006. *JavaScript: The Definitive Guide*. O'Reilly Media, Inc.

Gosling, James. 2000. *The Java Language Specification*. Addison-Wesley Professional.

Guo, Hanqi, Zuchao Wang, Bowen Yu, Huijing Zhao, and Xiaoru Yuan. 2011. "TripVista: Triple Perspective Visual Trajectory Analytics and Its Application on Microscopic Traffic Data at a Road Intersection." In *Visualization Symposium (PacificVis), 2011 IEEE Pacific*, 163–170. IEEE.

Hanwu, He, and Wu Yueming. 2009. "Web-Based Virtual Operating of CNC Milling Machine Tools." *Computers in Industry* 60 (9): 686–697.

Hermann, Mario, Tobias Pentek, and Boris Otto. 2016. “Design Principles for Industrie 4.0 Scenarios.” In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 3928–3937. IEEE.

ISO. 2003. “ISO 14649-1: Industrial Automation Systems and Integration - Physical Device Control - Data Model for Computerized Numerical Controllers -- Part 1: Overview and Fundamental Principles.” http://www.iso.org/iso/catalogue_detail.htm?csnumber=34743.

ISO. 2004. “ISO 14649-11: Industrial Automation Systems and Integration -- Physical Device Control -- Data Model for Computerized Numerical Controllers -- Part 11: Process Data for Milling.” http://www.iso.org/iso/catalogue_detail.htm?csnumber=40896.

ISO. 2007. “ISO 10303-238: Industrial Automation Systems and Integration - Product Data Representation and Exchange -- Part 238: Application Protocol: Application Interpreted Model for Computerized Numerical Controllers.” https://www.iso.org/standard/38036.html.

ISO. 2009. “ISO 6983-1: Automation Systems and Integration -- Numerical Control of Machines -- Program Format and Definition of Address Words -- Part 1: Data Format for Positioning, Line Motion and Contouring Control Systems.” http://www.iso.org/iso/catalogue_detail.htm?csnumber=34608.

Jain, Sanjay, David Lechevalier, Jungyub Woo, and Seung-Jun Shin. 2015. “Towards a Virtual Factory Prototype.” In *2015 Winter Simulation Conference (WSC)*, 2207–2218. IEEE.

Jain, Sanjay, and Guodong Shao. 2014. “Virtual Factory Revisited for Manufacturing Data Analytics.” In *Proceedings of the 2014 Winter Simulation Conference*, 887–898. IEEE Press.

Kramer, Thomas R., F. Proctor, X. Xu, and J. L. Michaloski. 2006. “Run-Time Interpretation of STEP-NC: Implementation and Performance.” *International Journal of Computer Integrated Manufacturing* 19 (6): 495–507.

Kusiak, Andrew. 2017. “Smart Manufacturing Must Embrace Big Data.” *Nature* 544 (7648): 23.

Lakshminarayan, Kamakshi, Steven A. Harp, and Tariq Samad. 1999. “Imputation of Missing Data in Industrial Databases.” *Applied Intelligence* 11 (3): 259–275.

LaValle, Steve, Eric Lesser, Rebecca Shockley, Michael S. Hopkins, and Nina Kruschwitz. 2011. “Big Data, Analytics and the Path from Insights to Value.” *MIT Sloan Management Review* 52 (2): 20–32.

Lechevalier, David, Anantha Narayanan, and Sudarsan Rachuri. 2014. “Towards a Domain-Specific Framework for Predictive Analytics in Manufacturing.” In *Big Data (Big Data), 2014 IEEE International Conference on*, 987–995. IEEE.

Manyika, James. 2012. *Manufacturing the Future: The next Era of Global Growth and Innovation*. McKinsey Global Institute.

Marinov, Valery P., and Sreenath Seetharamu. 2004. “Virtual Machining Operation: A Concept and an Example.” In *Optics East*, 206–213. International Society for Optics and Photonics.

MTConnect Institute. 2015. “MTConnect.” http://www.mtconnect.org/standard-documents.

Noor, Ahmed. 2013. “Putting Big Data to Work.” *Mechanical Engineering, ASME* 135 (10): 32–37.

OMG. 2016. “Meta Object Facility.” http://www.omg.org/spec/MOF/2.5.1/.

Ridwan, Firman, and Xun Xu. 2013. “Advanced CNC System with in-Process Feed-Rate Optimisation.” *Robotics and Computer-Integrated Manufacturing* 29 (3): 12–20.

Schönemann, Malte, Christopher Schmidt, Christoph Herrmann, and Sebastian Thiede. 2016. “Multi-Level Modeling and Simulation of Manufacturing Systems for Lightweight Automotive Components.” *Procedia CIRP* 41: 1049–1054.

Shin, Seung-Jun, Jungyub Woo, Duck Bong Kim, Senthilkumaran Kumaraguru, and Sudarsan Rachuri. 2016. “Developing a Virtual Machining Model to Generate MTConnect Machine-Monitoring Data from STEP-NC.” *International Journal of Production Research* 54 (15): 4487–4505.

SMLC, Smart Manufacturing Leadership Coalition. 2016. “The Smart Manufacturing Project.” Accessed June 6. https://www.smartmanufacturingcoalition.org/.

Sun, Jimeng, and Chandan K. Reddy. 2013. “Big Data Analytics for Healthcare.” In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1525–1525. ACM.

Terkaj, Walter, Tullio Tolio, and Marcello Urgo. 2015. “A Virtual Factory Approach for in Situ Simulation to Support Production and Maintenance Planning.” *CIRP Annals-Manufacturing Technology* 64 (1): 451–454.

Ticknor, Jonathan L. 2013. “A Bayesian Regularized Artificial Neural Network for Stock Market Forecasting.” *Expert Systems with Applications* 40 (14): 5501–5506.

Vijayaraghavan, Athulan, Will Sobel, Armando Fox, David Dornfeld, and Paul Warndorf. 2008. “Improving Machine Tool Interoperability Using Standardized Interface Protocols: MT Connect.” *Laboratory for Manufacturing and Sustainability*.

W3C. 2008. “Extensible Markup Language (XML) 1.0 (Fifth Edition).” https://www.w3.org/TR/2008/REC-xml-20081126/.

W3C. 2014. “Hypertext Markup Language (HTML).”

Wang, Lidong, and Cheryl Ann Alexander. 2015. “Big Data in Design and Manufacturing Engineering.” *American Journal of Engineering and Applied Sciences* 8 (2): 223.

Young, M., and D. Pollard. 2012. “What Businesses Can Learn from Big Data and High Performance Analytics in the Manufacturing Industry.” *Big Data Insight Group*.

Zhang, Yu, Xun Xu, and Yongxian Liu. 2011. “Numerical Control Machining Simulation: A Comprehensive Survey.” *International Journal of Computer Integrated Manufacturing* 24 (7): 593–609.