A virtual milling machine model to generate machine-monitoring data for predictive analytics

David Lechevalier[1/2], Seung-Jun Shin[1], Jungyub Woo, Sudarsan Rachuri[1], Sebti Foufou[3]

[1]National Institute of Standards and Technology, Gaithersburg, USA
david.lechevalier@nist.gov
seungjun.shin@nist.gov
jungyub.woo@nist.gov
sudarsan.rachuri@nist.gov

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

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

Abstract. Real data from manufacturing processes are essential to create useful insights for decision-making. However, acquiring real manufacturing data can be expensive and time consuming. To address this issue, we implement a virtual milling machine model to generate machine monitoring data from process plans. MTConnect is used to report the monitoring data. This paper presents 1) the characteristics and specification of milling machine tools, 2) the architecture for implementing the virtual milling machine model, and 3) the integration with a simulation environment for extending to a virtual shop floor model. This paper also includes a case study to explain how to use the virtual milling machine model for predictive analytics modeling.

Keywords: STEP, MTConnect, milling, data generator, data analytics

1 Introduction

The application of data analytics in manufacturing is one of the most promising methods to help manufacturers improve the productivity of their systems by saving money and time or reducing process flaws. Collecting manufacturing data is critical to make use of the different techniques available for data analytics. In the framework described in [1], the authors described the importance and the necessity of data collection to run data analytics in the manufacturing area, which continuously generates large amounts of data [2]. Data can be in different formats that can be defined as structured or unstructured. The suggested framework needs to be able to understand these different data formats to analyze the data. In particular, modern machines are able to provide real-time data to monitor the values of operating parameters. This data can be specified in the MTConnect standard [3] in order to facilitate the communication between equipment and software applications.

However, since acquiring data is still expensive and time consuming, simulation approaches that can generate data at a lower cost need to be explored. Simulation approaches have already allowed manufacturers to reduce costs and time at the factory level [4] by generating simulated data that they can analyze to improve the performance of their systems. While creating virtual machine models can allow manufacturers to generate simulated process data, using these models together will lead to a virtual shop floor model at the production level.

Combining simulation and data analytics at the process level can lead to a process efficiency improvement at a lower cost. In [5], authors have compared Bayesian Networks and Artificial Neural Networks for running analytics on real and simulated data with efficient results to predict the output values.

This paper introduces a virtual milling machine model to generate machine monitoring data from a process plan. In addition to the model, we also present an agent-based model including a machine-state-chart diagram. We integrate our virtual machine model into the agent model to use it in a simulation environment. We show how the agent-based model and the virtual machine model can be embedded in a shop-floor-level simulation environment combining discrete event and agent-based models. Finally, we illustrate how data analytics can be applied.

This paper is organized as follows: Section 2 introduces the characteristics and specifications of the virtual milling machine model. Section 3 presents the virtual milling-machine model, and its combination with an agent-based model into a simulation environment. Section 4 shows how a manufacturer can leverage this combination and use the generated data to run analytics for system improvement.

2 Specifications of the Virtual Milling Machine Model

In this section, we present the specification of input and output data for our virtual model. We also introduce the equations needed to compute the power metrics related to the milling process and finally show the state chart that we define for representing the behavior of a machine and include our model in a simulation environment.

2.1 Input data and output data: from STEP-NC program file to MTConnect document

We identify an ISO 14649 STEP-NC [6] program file (henceforth referred to as STEP-NC file) and MTConnect document respectively as input and output data of our virtual model. In [7], authors underline that a STEP-NC-based approach is promising for digital manufacturing while authors emphasize MTConnect capabilities to improve the interoperability of machine tools in [8]. Numerical control (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 to send a set of instructions to the NC machines in order to achieve the production defined in the original CAM file.

MTConnect is an XML-based [9] standard to represent machine monitoring data. This standard aims to provide interoperability so that manufacturers can monitor various brands and models of Computer Numerical Control (CNC) machines through a common interface. By using this standard for the output data, we ensure that the data will have a well-known structure that facilitates the communication for later uses.

2.2 Machine tool specification for kinematics and dynamics

To virtually model a milling machine, we compute kinematics (e.g., velocity and position) and dynamics (e.g., force and power) corresponding to the events and movements of the machine tool. A STEP-NC program specifies a sequence of machining operations, and is used to create an NC program in the ISO 6983 (G-Code) format [10]. Meanwhile, an MTConnect document generates continuous snap shots of a machine tool's actions and events using time as reference. Thus, for every instruction of the NC program, we need to compute the corresponding metrics of the machine tool. Computing these metrics requires equations that are derived from physical model-based analysis of machine tool metrics. We make a calculation of power, which indicates the amount of energy consumed per unit-time.

First, we defined a position function by deriving theoretical equations presented in [11]. This function is presented in Equation (1) assuming that linear velocity has a trapezoidal profile.

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

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, our virtual machine model 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 between a work piece and a cutting tool. The characterization of the motions contributes to determine the power consumption.

Second, our virtual model computes the machine tool dynamics using theoretical equations introduced in [12]. The power profile of a single NC code command for linear movement consists of acceleration, steady and deceleration states. Power during the steady state varies for cutting and non-cutting motions. During the cutting motion, the power corresponds to the cutting power, which is caused from cutting forces, plus the idle power. We use a physics-based equation, as expressed in Equation (2), to calculate the cutting forces. Equations (3) and (4), respectively, present the linear-axial and rotary-axial power for a milling machine. Units can be obtained in the reference paper [12].

$F_t = K_{tc} bh + K_{te} b$ $F_f = K_{fc} bh + K_{fe} b$ (2)	$P_{L,a} = \frac{T_a w}{\eta_L}, \quad T_a = J_e \frac{dw}{dt} + Bw + T_s$ $P_{L,s} = \frac{T_s w}{\eta_L}, \quad T_s = (T_{gf} + \frac{\mu_b d_p (F_f + F_p)}{2} + T_f)/r_g$ $P_{L,d} = \frac{T_d w}{\eta_L}, \quad T_d = -J_e \frac{dw}{dt} + Bw - T_s$ (3)	$P_{S,a} = \frac{(T_{S,a} + T_{run})w}{\eta_S}$ $P_{S,s} = \frac{T_{run} w}{\eta_S}$ $P_{S,c} = P_{S,s} + \frac{2\pi F_c v_c}{\eta_S}$ $P_{S,d} = \frac{(-T_{S,a} + T_{run})w}{\eta_S}$ (4)

2.4 Machine states

To integrate our virtual milling machine model inside an agent-based model, we develop a state chart that represents the different states of the machine. The model is presented below in **Figure 1**.

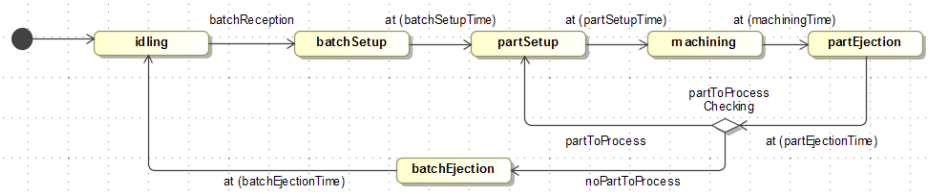

Figure 1. State chart diagram for a machine

The default machine state is the *idling* state. As soon as a batch arrives (represented by the transition *batchReception* in the figure), the machine goes to the next state called *batchSetup*. This state models the required machine setup for processing the batch. Once the batch is setup, the machine goes to the *partSetup* state where the machine sets up each part to execute the needed operations. 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 unloading of the part. After this state, two alternative paths can be taken by the machine in the state chart. If there are still parts to process in the batch, the machine goes back to *partSetup*. In the other case, the machine goes to the last state, which is the *batchEjection* state, when the batch unload step is modeled. After a batch has been ejected, the machine goes back to the *idling* state waiting for a new batch.

3 Description of the virtual machine model architecture and integration in an agent-based model

In the previous section, we have shown the specifications that define our virtual machine model. In this section, we introduce the architecture of the virtual machine model that makes it possible to generate machine monitoring data virtually. We then present the integration of the virtual model inside an agent-based model. We finally discuss the benefits of this integration.

3.1 Architecture of the virtual machine model

Figure 2 illustrates the process flow (including involved tools and generated outputs for each step) followed by the virtual milling machine model to generate MTConnect data. The virtual milling machine model takes, as an input, a STEP-NC file. The model parses and interprets this file using a toolkit for Parts 10, 11, and 111 (related to milling process data and tools) that is written in C++ and referred to as ISO 14649 Toolkit in the picture. This toolkit has been developed at the National Institute of Standards and Technology (NIST) for programing with ISO 14649, Parts 10 and 11, and is being applied to study different ISO 10303 [13] application-protocol file characteristics and their interpretation [14]. Using this toolkit, we can generate the sequence of G-Code instructions. We developed a physics-based modeler that we have integrated in our virtual model to transform the G-Code instructions into machine tool kinematics and dynamics.

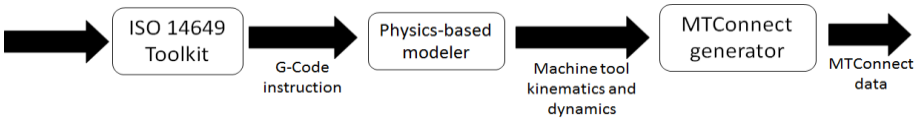

Figure 2. Step-by-step procedure of the virtual milling machine model

The computed movement metrics are length, acceleration, velocity, time, cutting, force and power. Computations are based on the equations introduced in the previous section. You can see below, in **Figure 3**, a class diagram representing the movement structure. For brevity, we show an overview of the class diagram. We define an abstract class called *Movement*. This class is extended by another abstract class called *StraightMovement* that is itself an extension by two classes called *TraverseStraightMovement* and *FeedStraightMovement*. These two last classes allow representation as two different movement types for the milling machine. The schema can be extended to represent additional movement types in the future. All the computed metrics are also represented as classes and are aggregated to the Movement class. The class Power is abstract and is extended by two classes: TraversePower and FeedPower that will represent the power depending on the movement type. The physics-based modeler instantiates this structure during the computations and generates a Movement collection that represents the machine tool kinematics and dynamics.

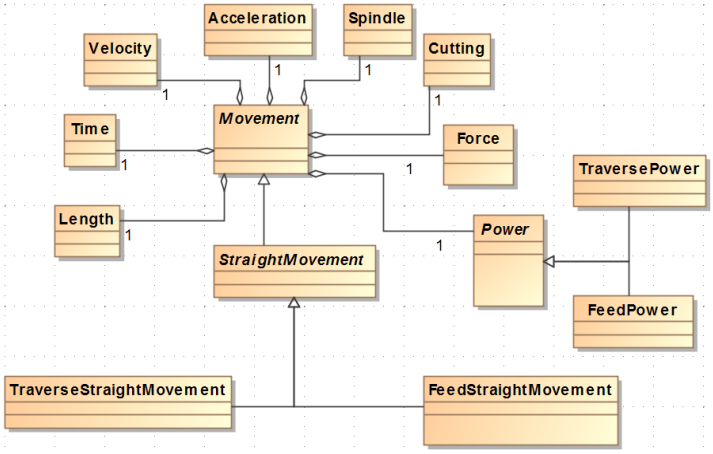

Figure 3. Class diagram representing the structure of a movement

To generate an MTConnect file corresponding to this STEP-NC file, we generate time series data representing the current position of the machine tool and the consumed power of the milling machine. Using the kinematics and dynamics that we generated in the previous step and an MTConnect generator that we developed as part of the virtual machine model, we generate MTConnect data representing the tool position and the consumed power every 100 milliseconds. We store these data in an MTConnect agent, which is a web service that collects the generated MTConnect samples. This MTConnect agent provides query functions that can be called to get specific sets of data previously stored.

3.2 Combination of the virtual machine model into an agent-based model using a simulated environment

To run our virtual machine model in a simulation environment, we integrate this virtual machine model in an agent–based model. The software environment called AnyLogic [15] allows us to extend the states and the transitions of a state chart using Java code. While implementing the state chart in an agent-based model, we can call virtual machine model functions by importing a Java ARchive (JAR) file that contains the needed functions. We first implement the state chart introduced in section **2.4** in the agent-based model. We extend this state chart by implementing additional Java code to initialize the parameters needed for the virtual machine model functions. During the batchSetup state, we get the time needed by the machine to set up the batch as well as the power consumed during this step by reading the machine specification described using XML. By following the same steps during the *partSetup* state, we generate the values of the machine parameters that depends on the properties of the material used for this batch. During the *machining* state, we include the parameters values inside a STEP-NC file given as an input to the virtual machine 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 by reading the machine specification as we do for the setup states. All the values of time and power are subjects to a standard deviation to represent the uncertainty at a real machine level. Once a batch has been processed (after the *batchEjection* state), we generate Comma Separated Values (CSV) and MTConnect output files that gives the time and the power consumed by the milling machine.

3.3 Benefits

This approach provides benefits for manufacturing simulation. The simulation applications reviewed in [16] illustrate the interest in simulation in the manufacturing area. While simulations for manufacturing operations, such as planning or scheduling or real-time control, seem to be the most important trend, generating machine-monitoring data can lead to a more accurate simulation. The agent-based model implementation allows manufacturers to use the milling model in a very easy way since the agent-based model can be used directly to represent one machine. Thus, the virtual milling model generates data during the simulation representing real machine behavior. Moreover, agent updates are regularly possible. Collecting real data punctually makes it possible to calibrate the virtual model. It also enables including realistic noise in the simulated data to give more accuracy to the virtual model. Finally, the agent-based model can be improved by integrating disturbances such as machine failure in the state chart.
Extending this approach, providing a library of agents can allow manufacturers to choose the machine model to represent the machine involved in their manufacturing systems. Updates on virtual machine models only require a library update. A manufacturer can use an agent from the library in the simulation without really knowing how the integrated virtual model is implemented. Finally, different agents representing the same machine can provide different capabilities depending on the studied problem such as power consumption, flow capabilities and material consumption by integrating a different virtual model.

4 Use Case

In this section, we will illustrate how to use the agent-based model to generate data. We will first present the specification of our use case, and then show the implementation in the simulation environment. The last part introduces the application of regression analysis to generate an analytical model.

4.1 Use case scenario

We define a scenario to represent a milling machine in the simulated environment. In this scenario, a milling machine tool manufactures a steel part, as shown in **Figure 4**. The process parameters – feed rate, spindle speed, and cutting depth – control the tool path strategies that are necessary to make the given machining features. We

assign the three process parameters randomly using a uniform distribution within the ranges given in **Table 1**. This process plan decision generates STEP-NC files. Each STEP-NC file is assigned to produce one part.

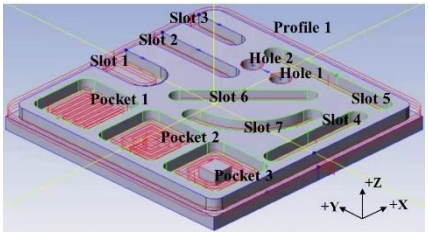

Figure 4. An example of a milling part

Table 1. Process plan data

Process parameter	Unit	Lower bound	Upper bound
Feedrate	mm/s	30	90
Spindle speed	rad/s	75.4	226.2
Cutting depth	mm	2.5	3.5

4.2 Implementation results

Given the process plan scenario in Section 4.1, we instantiate the agent-based model in a process flow model to collect MTConnect data. This process flow represents a flow of batch coming to the milling machine. Our machining model generates MTConnect documents for every part of the batch. To reproduce a real machine behavior, using an identical set of process parameters leads to different power values representing the variation that can occur in a real machine (±10 % uniform-random deviation during feed movement, and ±5 % uniform-random deviation during traverse movement). Using the agent-based model, we generate MTConnect time series data after every part ejection while running the simulation.

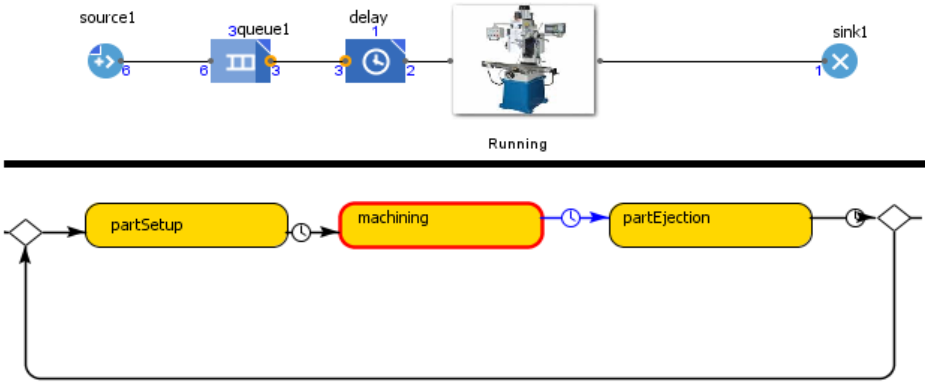

Figure 5. Example of simulation at the process flow and the agent levels.

Figure 5 shows the implementation of the scenario in Anylogic at the process and agent levels. The MTConnect document provides the following set of information: *x_axis_position*, *x_axis_wattage*, *y_axis_position*, *y_axis_wattage*, *z_axis_position*, *z_axis_wattage, c_axis_wattage*, *electric_wattage* and *coolant_wattage*.

4.3 Predictive modeling using generated data

Using the simulated data, we are able to run regression analysis to generate an analytical model by using machine learning techniques. This analytical model can then be used to predict values of the power consumption. After a normalization of the data, we train a neural network model with the first 500 samples of our simulated data. We give 300 new samples as inputs of the trained model and compare the outputs of the model and the simulated data generated by our virtual machine model using the same input parameters. **Figure 6** represents the comparison between the simulated total power (X-axis) and the predicted total power (Y-axis).

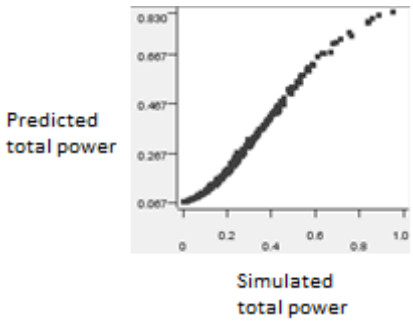

Figure 6. Scatter plot of the simulated and the predicted total power.

As you can see, the plot shows a slightly curved line showing that the predicted data are really close to the simulated data for the same input parameters. The coefficient of determination, representing how close the predicted data are to the simulated data, is 0.986. By applying this approach and after validation, a manufacturer can also use this model to compare the real outputs with the model outputs to establish diagnostic on a machine in a manufacturing system. Extending it to a full manufacturing system will allow a manufacturer to anticipate the behavior of the system in a simulation environment by taking advantage of the simulated data.

5 Conclusion

In this paper, we introduce a virtual milling machine model that allows us to generate machine-monitoring data in MTConnect format. We show that we can integrate this model in a simulated environment to take advantage of the generated data and generate a predictive model to finally improve or make a diagnostic on a milling process described in a STEP-NC file. In a future work, integration of

maintenance and failure in our model can make our generation of data more realistic and improve our simulation. Moreover, taking advantage of our model and other existing models [17], we will be able to develop a virtual shop floor model.

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

1. Lechevalier, D., Narayanan A., and Rachuri, S.: Towards a domain-specific framework for predictive analytics in manufacturing. In: 2014 IEEE Conference on Big Data, 2014.
2. Young, M., and Pollard, D.: What businesses can learn from big data and high performance analytics in the manufacturing industry. Big Data Insight Group, 2012.
3. MTConnect: Part 1-Overview and protocol, Version 1.2.0. MTConnect Institute, 2014.
4. Brown, E., and Sturrock, D.: Identifying cost reduction and performance improvement opportunities through simulation. In: Winter Simulation Conference, 2009.
5. Perzyk, M., Biernacki R., and Kochański, A.: Modeling of manufacturing processes by learning systems: The naïve Bayesian classifier versus artificial neural networks. In: Journal of Materials Processing Technology 164: 1430-1435, 2005.3
6. ISO, ISO 14649: 2003, Industrial automation systems and integration - Physical device control - Data model for computerized numerical controllers.
7. Yang, W., and X. Xu.: "Modelling machine tool data in support of STEP-NC based manufacturing." In: International Journal of Computer Integrated Manufacturing, 2008.
8. Vijayaraghavan, Athulan, Sobel, W., Fox, A., Dornfeld, D., & Warndorf, P.: "Improving machine tool interoperability using standardized interface protocols: MT Connect." In: International Symposium on Flexible Automation, 2008.
9. XML, XML Specification available at http://www.w3.org/TR/2008/REC-xml-20081126/.
10. ISO, ISO 6983-1: 1982, Numerical control of machines -- Program format and definition of address words -- Part 1: Data format for positioning, line motion and contouring control systems
11. Avram, O., and Xirouchakis P.: Evaluating the Use Phase Energy Requirements of a Machine Tool System. In: Journal of Cleaner Production 19: 699-711, 2011.
12. Altintas, Y.: Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations, and CNC Design. Cambridge University Press: Cambridge, 2012.
13. ISO, ISO 10303-1: 1994, Industrial Automation Systems and Integration - Product Data Representation and Exchange -- Part 1: Overview and fundamental principles.
14. Kramer, T. R., Proctor, F., Xu, X., & Michaloski, J. L.: Run-time interpretation of STEP-NC: implementation and performance. In: International Journal of Computer Integrated Manufacturing, Vol. 19, Iss. 6, 2006.
15. Grigoryev, I.. AnyLogic 7 in Three Days: A Quick Course in Simulation Modeling, 2015.
16. Negahban, A., and S. Smith J.: Simulation for manufacturing system design and operation: Literature review and analysis. In: Journal of Manufacturing Systems 33.2: 241-261, 2014.
17. Shao, G., Shin S., and Jain S.: Data analytics using simulation for smart manufacturing. In: Proceedings of the 2014 Winter Simulation Conference. IEEE Press, 2014.