An Analytical Framework for Smart Manufacturing

Amogh Kulkarni*, Daniel Balasubramanian*, Gabor Karsai*, Peter Denno†
*Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37212
Email: amogh.s.kulkarni, daniel.a.balasubramanian, gabor.karsai@.vanderbilt.edu
†National Institute of Standards and Technology, Gaithersburg, MD 20899
Email:peter.denno@nist.gov

***Abstract*—Smart manufacturing is an emerging paradigm for the next generation of manufacturing systems. One key to the success of smart manufacturing is the ability to use production data for defining predictive and descriptive models and their analyses. However, the development and refinement of such models is a labor- and knowledge-intensive activity that involves acquiring data, selecting and refining an analytical method and validating results. This paper presents an analytical framework that facilitates these activities by allowing ad-hoc analyses to be rapidly specified and performed. The proposed framework uses a domain-specific language to allow manufacturing experts to specify analysis models in familiar terms and includes code generators that automatically generate the lower-level artifacts needed for performing the analysis. We describe the use of our framework with an example problem.**

I. Introduction

Smart manufacturing systems (SMS) is an emerging paradigm for the next generation of manufacturing systems. Its goal is to enable data-driven decision making throughout manufacturing. SMS holds the potential to improve many aspects of manufacturing. For example, using real-time feedback about part quality, a SMS may adjust process parameters to optimize throughput.

Important to success in smart manufacturing is the ability to learn from data. Data informs control decisions in SMSs through two pathways yielding predictive models: where phenomena are not well-understood, data analytics (e.g. machine learning) can be applied; and where phenomena are understood, conventional analytical methods (e.g finite element analysis, operations research) are typically more effective.

Regardless of the pathway, the resultant computational model produced can be used either directly to infer control signals or "off-line" in trade studies and numerical optimizations. In both the machine learning and analytical pathways, the development and refinement of predictive models is both a labor- and knowledge-intensive activity involving the tasks of acquiring and conditioning data, selecting and refining an analytical method (e.g. experimental, physics-based etc.), and validating results. These tasks could be facilitated by a system that integrates access and visualization of production data with analytical capabilities. This paper describes the design and implementation of such a system.

There are several challenges to designing and building this type of framework. First, data may be produced in a variety of different formats, yet flexible tools to examine and organize the data for analysis are needed. Second, one must provide access to multiple analysis algorithms. Third, the framework should allow users to combine both process concerns and operations concerns in the analysis. Fourth, the framework should facilitate continual refinement of the analytical model. Finally, the framework must present the analysis results in a way that allows insights and actionable conclusions to be drawn.

This paper presents an analytical framework to support decision making in smart manufacturing settings. The paper focuses on the aspects of the framework that allow ad-hoc analyses to be specified and performed rapidly. The framework supports input data in the MTConnect standard[1], an industry standard for describing data produced by machine tools. It stores data using InfluxDB[2] [1], a time-series database built for real-time analytics applications. The framework also integrates various Python libraries like 1) OpenMDAO[3], which provides analysis algorithms, 2) SimPy[4] for developing discrete event simulation models of manufacturing processes, 3) scikit-learn[5] for developing data-driven models of manufacturing processes and 4) Bokeh[6] for visualizing analysis results. The Grafana[7] visualization engine is used to visualize the data logged in InfluxDB. Underpinning the framework is a *domain-specific modeling language* that allows users to express complex analyses using an intuitive, extensible, modeling language.

The rest of this paper is structured as follows. Section II gives brief background information on MTConnect and model-based software engineering. Section III describes the application of the proposed framework to a motivating example. Section IV gives the design and implementation of the framework. Related work is presented in Section V and Section VI concludes the discussion.

II. Background

Attempts to create the network of "things" in a manufacturing setting date back to the 1970s. These attempts, however, were largely unsuccessful because of 1) immature communication infrastructure, 2) lack of facilities for large data storage, and 3) limitations on computing power. Today, these impediments no longer exist and we are observing a fusion of manufacturing systems and Cyber-Physical Systems

[1]References to proprietary products are included in this paper solely to identify the tools actually used in the industrial applications. The identification does not imply any recommendation or endorsement by NIST as to the actual suitability of the product for the purpose.

(CPS) which has given rise to the field of Cyber Manufacturing [8]. "Industrial Internet-of-Things" (IIoT) have given rise to an explosion of data in manufacturing, providing new opportunities to improve process quality, make predictions, and perform fault-diagnosis and prognosis. We are in the midst of a major overhaul of the manufacturing landscape where more and more cyber-space is getting connected. In this new world, communication standards play a very important role.

MTConnect is one such communication standard. It is aimed at making data from Computer Numerical Control (CNC) machines machine-readable and more accessible. The development of standards like MTConnect have paved the way towards cross-platform analytical applications and sensor fusion. Although empirical techniques for data acquisition can be used for most of the analytics applications, the integration of Cyber-Physical Systems with modern manufacturing techniques provides more efficient means to acquire data [9], [10]. Communication standards like MTConnect and OPC-UA [11] enable near real-time streaming of factory data, which can be effectively used for the analysis of manufacturing processes.

Model-Based System Engineering (MBSE) is an effective approach for developing CPS applications as the transition from document-based to model-based methodologies continues [12]. This is especially the case as systems and processes grow more complex and require the integration and composition of multiple heterogeneous models [13]. Domain-specific modeling is a part of MBSE that deals with modeling environments to provide abstractions that cater to specific engineering disciplines. The framework described in this paper uses a Domain-Specific Modeling Language (DSML) to help small- and medium-scale manufacturers leverage the MBSE paradigm for performing manufacturing analysis.

III. Motivating example

The motivating example used to explain the framework is a machining process running on a MTConnect-enabled CNC workcenter. 20 units of a milled aluminum part were produced using end milling and face milling at various feeds and speeds. Each iteration of the process produced an MTConnect dataset. This aggregate data is used to produce a predictive model of the process using standard regression techniques. The regression model predicts the time required to produce a part for the aggregated value of spindle speed while producing a part.

The MTConnect data produced during the milling process under study contains the values of various process parameters such as spindle speed and line feeds during the production of the part. In addition to these process parameters, the MTConnect data contains information about the state of the CNC machine during production. Using these data streams, statistical analyses like linear regression can be performed to determine relationships such as that between values of process parameters and changes in the state of the machining tool. Such statistical models help to approximate the relationship between the process parameters and process performance metrics, where obtaining the accurate analytical relationship between the two is not possible.

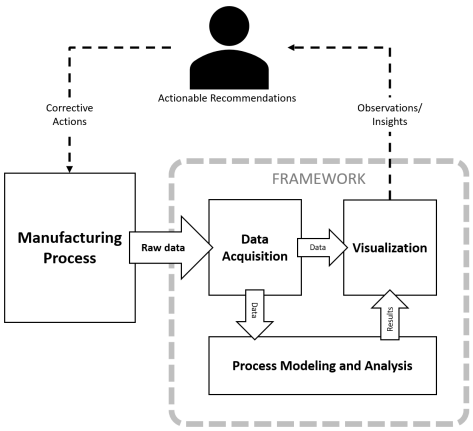


Fig. 1. Overview of the framework

This example describes how the framework can be used to develop a regression-based model. To simplify the explanation and demonstration of the framework, only one explanatory variable is used in the regression model, spindle speed. The model is trained over the acquired MTConnect data using linear regression, but other regression algorithms such as polynomial regression or Gaussian Process regression may also be used. The input feature for the learning algorithm is the integrated value of spindle speed over the process duration for a given part, and the output is the time required for producing the part.

Figure 1 gives a high-level view of the entire workflow. The manufacturing process under question can be considered as a "black-box" that produces the raw data. This data is saved in a data acquisition module, which is integrated with a domain-specific modeling environment. The modeling environment is used to create domain-specific models of the process. After the models have been developed, they can be used to run different analyses within the same environment. Visualization of both the input data as well as the analysis results provide the user with valuable insights about the nature of the process.

There is a vast body of literature on predictive, diagnostic, and descriptive analysis of manufacturing processes. In discrete-part manufacturing, surface finish and dimensional accuracy are key to part quality. Studies predicting surface roughness have used factorial design-of-experiment in turning processes [14], regression and neural-network based approaches [15], and empirical models developed from the data [16] and using genetic algorithm approaches with Response Surface Methodology (RSM) [17]. Predictive analysis of process efficiency include the prediction of energy consumption (i.e. sustainability analysis) by developing empirical models using data [18] and prediction of tool wear using neural networks [19]. Studies estimating process properties such as manufacturing cycle time have been done using data-mining approaches on stored data in the semiconductor manufacturing domain [20]. Predictive analysis of the energy consumption

of the process has been done by leveraging modern data-acquisition standards, where the energy consumed by the machining tool while producing a part is predicted using statistical techniques [21].

The typical workflow of the approaches listed above is to (1) gather the data from the process, (2) develop a data-driven model or an analytical model, and (3) use the model and acquired data to perform various analyses in order to gain insights about the system. The difficulty with this workflow is that it requires expertise in data acquisition techniques, efficient data-storage techniques, and advanced programming skills to implement and/or use the modeling platforms and data-visualization techniques. The work in this paper addresses this need and provides an implementation of integrated tools, stepping towards the goal of being able to rapidly prototype ad-hoc analyses.

The analytical (i.e. equation-based) models as well as data-driven models of a production process can be used for various analyses like optimization, Design-of-Experiment (DoE), Discrete Event Systems (DES) simulations, among others. The proposed framework uses Multi-Disciplinary Analysis and Optimization (MDAO) methodology for specifying optimization and DoE analyses and DES simulations to provide long-term predictions of the system under study. Also, the framework uses regression techniques to approximate process models. In the next section, various capabilities and individual components of the proposed framework are showcased.

IV. Framework

In this section, we discuss the analytical framework in detail. The framework is an integrated environment for (1) acquiring process data, (2) developing models of the process using the acquired data, and (3) using the developed models to perform analyses. This integrated environment is built using WebGME [22], a web-based modeling platform, which provides the means to integrate all the software components needed to accomplish the tasks mentioned above. The integrated environment can be divided logically into three pieces: (1) a domain-specific language, (2) an engine for code generation and execution, and (3) platforms for data acquisition and visualization. Figure 2 shows these software components in detail and the interfaces among them, which is an extension of our previous work [23]. Bold arrows in the figure signify the flow of data and artifacts between the individual components, and regular-weight arrows signify the dependencies.

A. The Domain-Specific Modeling Language

Domain-Specific Modeling Languages (DSMLs) are modeling languages tailored for a particular domain. In contrast, modeling languages like UML [24] are general-purpose. The main benefit in using a DSML is that it can provide concepts that are already familiar to domain experts, and can thus decrease the barrier to entry for the users who are knowledgeable in manufacturing but less experienced with data analysis. Additionally, a DSML provides the ability to compose models, to formally verify model compositions by enforcing constraints on the interfaces, and to improve the overall usability of the models beyond with the help of code-synthesis [25].

Developing a DSML with WebGME begins with the definition of a *meta-model*, which expresses the concepts in the language as well as the relationships between those concepts. Figure 3a shows a simplified version of the meta-model for our "MOdel Composition and Analysis" (MOCA) DSML. MOCA is primarily developed for modeling optimization and DoE scenarios. The figure roughly translates to following set of rules:

- A $Library$ can contain one or more $Component$(s).
- A $Component$ can have one or more $Port$(s), which act as the interfaces between multiple instances of $Component$s. A $Component$ can also contain instances of other $Component$s.
- A $DataConnection$ can be used to connect a $Port$ with another $Port$, $Objective$ or a $DesignVariable$.
- A $Port$ can either be an input port or output port depending upon the value of its "Direction" attribute.
- The $Component$ has a text attribute, "OutputFunction", that represents the explicit relation between the input and output $Port$s contained by it.
- The $Problem$ is the entity that represents an optimization problem or a DoE problem, depending upon the value of its "Type" attribute.
- The $Problem$ can contain $DesignVariable$s, an $Objective$ and one or multiple instances of $Component$s (which one intends to optimize or analyze).
- The $DesignVariable$ and $Objective$ can be connected to the contained $Component$(s) via the interface provided by $Port$s contained in them.
- The $DesignVariable$ has "Upper" and "Lower" attributes that signify the upper and lower bounds of the value that it can take in a $Problem$'s context.

In other words, the Library can contain one or more Components which can be reused, by creating Problem instances with different DesignVariables and Objectives. These Components can be composed to form assemblies of Components and the analysis such as Optimization or DoE can be specified over them using Problems.

Problems and Components provide a separation-of-concerns to the modeler between defining the model and using the model. Component specifies the business logic that determines how the model will behave.. Problem specifies how the underlying model, which is specified by an enclosed Component or an assembly of Components, will be used for the analysis. Problem captures information about the values to be assigned to the Ports of a Component (or an assembly of Components) during the optimization or DoE analysis.

1) Model definition: The behavior of a manufacturing process is captured by a Component. For example, a user can create a Component named "TurningProcess" with input Ports "SpindleSpeed" and "FeedRate" and an output Port "ProductionTime". This Component itself contains the explicit mathematical relationship between its inputs and outputs,

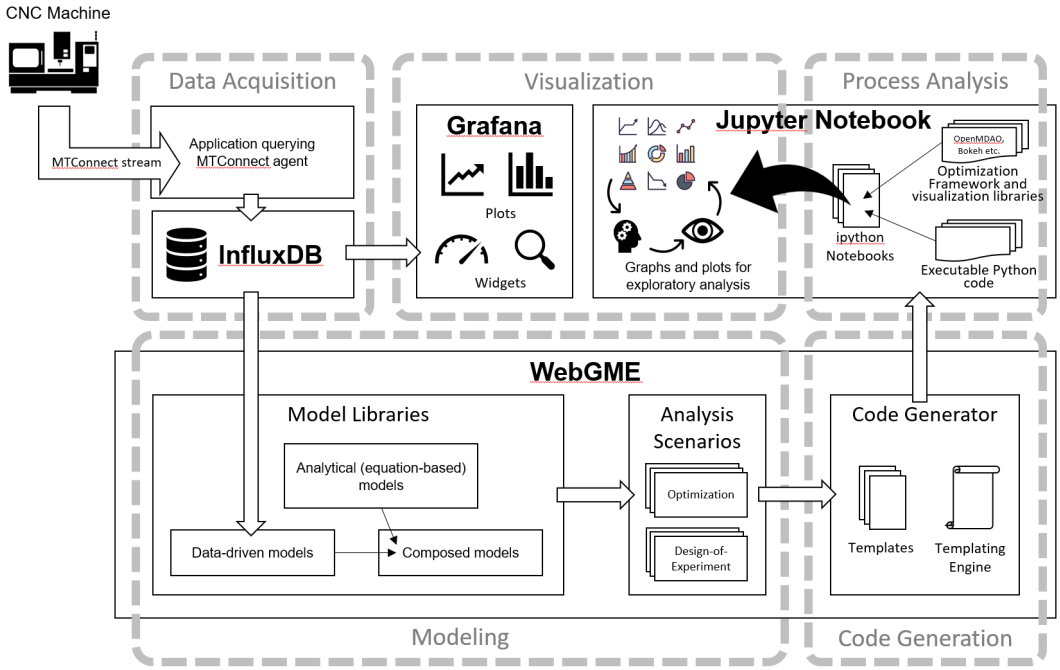


Fig. 2. Detailed architecture of the framework

saved as an attribute "OutputFunction". The mathematical relationship is stored in the form of a Python function. The function takes process parameters as inputs and returns the calculated production time. Since the Components can be composed, a complex, analytical relation can be divided into several Components.

In case where the user does not have such a mathematical relationship, a data-driven approach can be used to deduce it using regression analyses. For this, the modeling language contains a concept called $DataDrivenComponent$. It can have Ports similar to a Component, but it contains an additional modeling concept called $LearningAlgorithm$ which is used with data to learn the relationship between its inputs and outputs. The LearningAlgorithm specifies the type of regression analysis to be used, and also saves the learned models, so that they can be later used for analysis purposes, much like the aforementioned Components. The regression algorithms are provided by Scikit-Learn, a machine-learning library in Python.

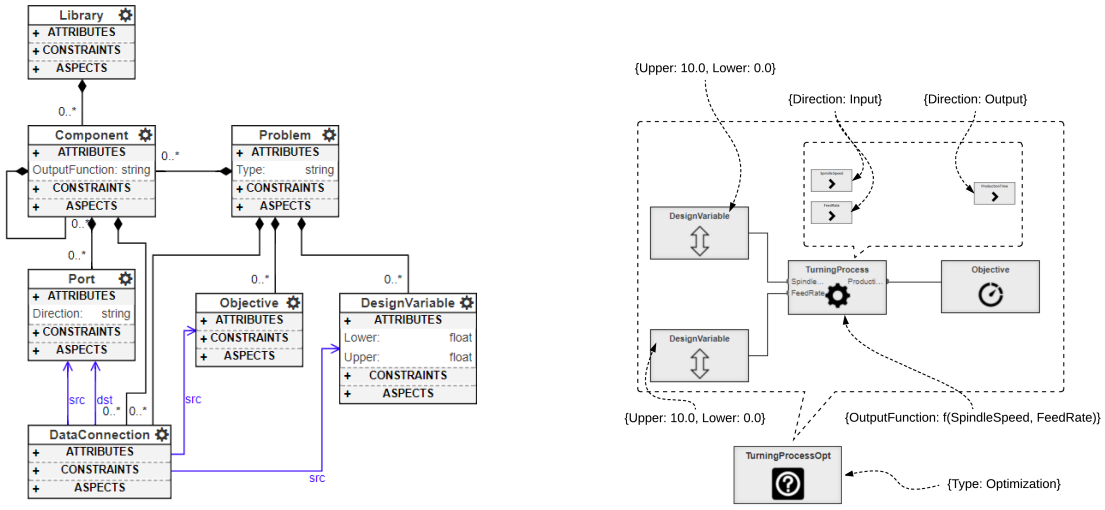
2) Analysis specification: Once the user has the model of the manufacturing process (derived either by data-driven techniques or defined explicitly), it can be used for various kinds of analyses by instantiating it within a Problem. A Problem can be set up in a way that it represents an Optimization analysis or a DoE analysis. In both the cases, A Problem specifies which Port(s) of the underlying enclosed Component are treated as design variables for that analysis scenario, by connecting them to DesignVariable(s). DesignVariables also provide the range in which the Ports are assigned values during the iterations of the optimizer or DoE driver routine. The optimization and DoE routines are part of OpenMDAO, which is an open-source, component-based library written in Python for MDAO analyses.

In the example discussed above, the user may use a Problem (named "TurningProcessOpt" in the diagram) to specify an optimization scenario of the "TurningProcess". Users can do this by associating "SpindleSpeed" and/or "FeedRate" inputs with DesignVariable(s) and "ProductionTime" output with Objective to be minimized. Figure 3b illustrates this by showing the hierarchy of these modeling entities along with their attributes.

MOCA modeling language also provides a modeling concepts for specifying Discrete Event Simulation models. These models can be used in predictive analyses to forecast the overall throughput of the process, and are parameterized over cycle time of the manufacturing process, mean time between failures (MTBF), among others. The modeling language provides the modeling concepts $Process$ and $Buffer$ that can be interconnected to model a production line. The code that is generated by these models uses SimPy, a Python library for modeling discrete-event simulation systems.

B. Code Generation and Execution

The actual analysis is performed by converting MOCA models into executable code. The web-based nature and client-server architecture of the WebGME tool entails that the generated code can 1) reside on the server-side where the WebGME tool is running as a service, or 2) on the client-side where the browser acts as a front-end using which the user interacts with the framework. Code generation is performed as follows - First, the code generator traverses the domain-specific model(s) and then populates the pre-defined code templates. The result is a fully executable Python codebase that uses the aforementioned libraries.



(a) Metamodel (b) Optimization of Turning Process Model

Fig. 3. A simplified version of MOCA metamodel and an example of a turning process model conforming to it

A Jupyter notebook is a web-based tool to publish readable and executable documents [26]. The generated Python code can be executed by using these Jupyter-notebook documents, which are also generated. As mentioned earlier, the user generates the Python code and can choose to either save it on the server or client side. Jupyter notebook being based on the client-server architecture, can leverage this to facilitate the user by providing the execution environment.

If the generated code is saved on the server, the user uses an instance of Jupyter notebook running on the WebGME server, and this notebook can be accessed through the WebGME client. In this case, the user does not need to have a Python environment on their side. On the other hand, if the user chooses to save the generated Python code on the client side, all the generated artifacts (executable code along with the Jupyter notebooks) can be downloaded and executed there. In this case, the user needs to have the Python environment and the necessary libraries installed locally.

Automated code-generation is a vital feature of the framework, since it makes the models computable. Also, the usefulness of the DSML is increased significantly if it not only represents the real-world entities in a domain-specific way, but also allows the user to interact with the representations without having to deal with their specific implementations. For example, the code generation abstracts the implementation details about how the optimization problems use the MDAO tools, as well as how the data is represented in communication standards like MTConnect. This makes the framework more approachable from an end-user point-of-view.

C. Data Acquisition and Visualization

1) Data Acquisition: One of the major strengths of this framework is its ability to integrate near real-time data-streams with the development and use of the process models. Data acquisition consists of two phases: (1) querying an MTConnect agent to acquire the data, and (2) storing that acquired data in a database. Once the data is in the database, the model-training and/or model-execution processes can query the database to obtain the data they need.

To specify how data is acquired, users create an instance of the *Database* modeling concepts, which captures the information about the URL of the MTConnect agent (which provides the stream of CNC data) and the path to the database. When an instance is created, a worker thread is spawned on the server, which queries the MTConnect agent and logs the MTConnect data in the database. The database used is InfluxDB, which is one of the most popular databases for storing high-volume, time-series data.

2) Visualization: Visualizations provide insights into underlying physical phenomena. The visualization techniques used in the framework can be applied to both results of the analyses and raw MTConnect data stored in the database. For visualizing the results of the DoE and DES analyses, the generated code for the models uses a plotting library named Bokeh which can render scatter plots and surface plots in the Jupyter notebook environment. The code for visualizing the analyses is generated along with the code for models, which removes the burden of interacting with the plotting libraries from the user. The user can also visualize the raw MTConnect data using Grafana, which is a web-based tool for visualizing the contents of an InfluxDB instance.

Interactive visualization provides additional exploratory capabilities. The Jupyter notebook provides means to interact with plots, which allows the user to selectively plot subsets of the results generated by the DoE analyses. Similarly, Grafana provides interfaces to execute SQL-like queries on the database, which fetches and plots the user-selected data. It also provides built-in mathematical operators like SUM and INTEGRAL, which can be used to perform data aggregation.

V. Related Work

The field of manufacturing process analysis, despite being well-studied, lacks a strong emphasis on generalizing analysis methods. There have been efforts to design and implement an

infrastructure that can facilitate both researchers and industry practitioners alike by allowing them to apply ad-hoc analyses to manufacturing systems. Tolio et al. describe such an integrated framework for the analysis and design of manufacturing systems [27]. Their virtual factory concept emphasizes the harmonization of heterogeneous information through a common data model, shared data storage and middleware to mediate the use of analytical tools. Relative to that work, we focus on efficiently assembling resources and tools to address unforeseen problems in processes and operations.

Data-driven models are the new frontiers of predictive modeling in manufacturing. Previous studies have shown that real-time predictive analysis using MTConnect can improve processes. Using MTConnect and statistical techniques, Bengtsson et. al. determine the parameters to populate discrete event simulation (DES) models, which are used to analyze the underlying manufacturing process [28]. Similar research has been done in sustainable manufacturing, where Shao et. al. use DES models [29] and Park & Law et. al use regression models [30], both of which are populated using MTConnect data. These examples show the benefits of MTConnect-like standards for data acquisition in the data-driven modeling paradigm. The work presented in this paper is complementary to the research mentioned above in analyzing manufacturing processes, by providing the tools to expedite the process of developing the predictive models.

VI. Conclusions

This paper presents an analytical framework for smart manufacturing which allows users to perform ad-hoc analyses. Users do this by creating *domain-specific models* that are used to specify input sources, analysis algorithms, and visualizations. Code generation transforms these domain-specific models into the appropriate lower-level artifacts, such as Python code. The generated code can then be used by regression algorithms, optimizers and a visualizer. The framework helps manufacturing experts concentrate on analyses rather than low-level implementation details and tool integration. We believe that frameworks such as ours are key enablers of the smart manufacturing vision.

References

[1] A. Vijayaraghavan, W. Sobel, A. Fox, D. Dornfeld, and P. Warndorf, "Improving machine tool interoperability using standardized interface protocols: MT connect," *Laboratory for Manufacturing and Sustainability*, 2008.

[2] "Influxdb, version 1.3." [Online], https://www.influxdata.com/time-series-platform/influxdb, 2017.

[3] J. Gray, K. T. Moore, and B. A. Naylor, "Openmdao: An open source framework for multidisciplinary analysis and optimization," in *AIAA/ISSMO Multidisciplinary Analysis Optimization Conference Proceedings*, vol. 5, 2010.

[4] N. Matloff, "Introduction to discrete-event simulation and the simpy language," *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August*, vol. 2, p. 2009, 2008.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] Bokeh Development Team, *Bokeh: Python library for interactive visualization*, 2014.

[7] *Grafana: the open platform for analytics and monitoring*, 2017.

[8] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert, "Industrial internet of things and cyber manufacturing systems," in *Industrial Internet of Things*, pp. 3–19, Springer, 2017.

[9] J. Lee, E. Lapira, B. Bagheri, and H.-a. Kao, "Recent advances and trends in predictive manufacturing systems in big data environment," *Manufacturing Letters*, vol. 1, no. 1, pp. 38–41, 2013.

[10] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.

[11] S.-H. Leitner and W. Mahnke, "Opc ua–service-oriented architecture for industrial applications," *ABB Corporate Research Center*, 2006.

[12] A. B. Feeney, S. Frechette, and V. Srinivasan, "Cyber-physical systems engineering for manufacturing," in *Industrial Internet of Things*, pp. 81–110, Springer, 2017.

[13] S. Friedenthal, R. Griego, and M. Sampson, "Incose model based systems engineering (mbse) initiative," in *INCOSE 2007 Symposium*, 2007.

[14] I. P. Arbizu and C. L. Perez, "Surface roughness prediction by factorial design of experiments in turning processes," *Journal of Materials Processing Technology*, vol. 143, pp. 390–396, 2003.

[15] T. Özel and Y. Karpat, "Predictive modeling of surface roughness and tool wear in hard turning using regression and neural networks," *International Journal of Machine Tools and Manufacture*, vol. 45, no. 4, pp. 467–479, 2005.

[16] X. Wang and C. Feng, "Development of empirical models for surface roughness prediction in finish turning," *The International Journal of Advanced Manufacturing Technology*, vol. 20, no. 5, pp. 348–356, 2002.

[17] P. Suresh, P. V. Rao, and S. Deshmukh, "A genetic algorithmic approach for optimization of surface roughness prediction model," *International Journal of Machine Tools and Manufacture*, vol. 42, no. 6, pp. 675–680, 2002.

[18] W. Li and S. Kara, "An empirical model for predicting energy consumption of manufacturing processes: a case of turning process," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 225, no. 9, pp. 1636–1646, 2011.

[19] N. Ghosh, Y. Ravi, A. Patra, S. Mukhopadhyay, S. Paul, A. Mohanty, and A. Chattopadhyay, "Estimation of tool wear during cnc milling using neural network-based sensor fusion," *Mechanical Systems and Signal Processing*, vol. 21, no. 1, pp. 466–479, 2007.

[20] P. Backus, M. Janakiram, S. Mowzoon, C. Runger, and A. Bhargava, "Factory cycle-time prediction with a data-mining approach," *IEEE Transactions on Semiconductor Manufacturing*, vol. 19, no. 2, pp. 252–258, 2006.

[21] R. Bhinge, N. Biswas, D. Dornfeld, J. Park, K. H. Law, M. Helu, and S. Rachuri, "An intelligent machine monitoring system for energy prediction using a gaussian process regression," in *Big Data (Big Data), 2014 IEEE International Conference on*, pp. 978–986, IEEE, 2014.

[22] M. Maróti, T. Kecskés, R. Kereskényi, B. Broll, P. Völgyesi, L. Jurácz, T. Levendovszky, and Á. Lédeczi, "Next generation (meta) modeling: Web-and cloud-based collaborative tool infrastructure.," *MPM@ MoDELS*, vol. 1237, pp. 41–60, 2014.

[23] A. Kulkarni, D. Balasubramanian, G. Karsai, P. O. Denno, and A. Narayanan, "A domain-specific language for model composition and verification of multidisciplinary models," in *2016 Conference on Systems Engineering Research*, 2016.

[24] J. Rumbaugh, I. Jacobson, and G. Booch, *Unified modeling language reference manual, the*. Pearson Higher Education, 2004.

[25] Á. Lédeczi, A. Bakay, M. Maroti, P. Volgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai, "Composing domain-specific design environments," *Computer*, vol. 34, no. 11, pp. 44–51, 2001.

[26] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, *et al.*, "Jupyter notebooks-a publishing format for reproducible computational workflows.," in *ELPUB*, pp. 87–90, 2016.

[27] T. Tolio, M. Sacco, W. Terkaj, and M. Urgo, "Virtual factory: An integrated framework for manufacturing systems design and analysis," *Procedia CIRP*, vol. 7, pp. 25–30, 2013.

[28] N. Bengtsson, J. Michaloski, F. Proctor, G. Shao, and S. Venkatesh, "Towards data driven sustainable machining combining mtconnect production data and discrete event simulation," in *Proceedings of the*

Proceedings of ASME 2010 International Manufacturing Science and Engineering Conference, 2010.

[29] G. Shao, S.-J. Shin, and S. Jain, "Data analytics using simulation for smart manufacturing," in *Proceedings of the 2014 Winter Simulation Conference*, pp. 2192–2203, IEEE Press, 2014.

[30] J. Park, K. H. Law, R. Bhinge, N. Biswas, A. Srinivasan, D. A. Dornfeld, M. Helu, and S. Rachuri, "A generalized data-driven energy prediction model with uncertainty for a milling machine tool using gaussian process," in *ASME 2015 International Manufacturing Science and Engineering Conference*, pp. V002T05A010–V002T05A010, American Society of Mechanical Engineers, 2015.