Networked Engineering Notebooks for Smart Manufacturing

Peter Denno,[a,1] Charles Dickerson[b], and Jennifer Harding[b]
[a] *National Institute of Standards and Technology, United States*
[b] *Loughborough University*

Abstract. A goal of the industrial internet is to make information about manufacturing processes and resources available wherever decision making may be required. Agile use of information is a cornerstone of data analytics, but analytical methods more generally, including model-based investigations of manufacturability and operations, do not so easily benefit from this data. Rather than relating anonymous patterns of data to outcomes, these latter analytical methods are distinguished as relying on conceptual or physics-based models of the real world. Such models require careful consideration of the fitness of the data to the purpose of the analysis. Verification of these analyses, then, is a significant bottleneck. A related problem, that of ascertaining reproducible results in scientific claims, is being addressed through executable notebook technology. This paper proposes to use notebook technologies to address that bottleneck. It describes how this notebook technology, linked to internet-addressable ontologies and analytical metamodels, can be used to make model-based analytical methods more verifiable, and thus more effective for manufacturers.

Keywords. Manufacturing analysis, process analysis, process optimization, analytical methods, empirical methods, industrial internet of things, metamodels

1. Introduction

Smart manufacturing [1] and similar initiatives [2], [3] are focused on improving manufacturing productivity through the use of inexpensive sensor networks, information systems, and software analytical tools. In manufacturing, a great variety of software analytical tools are being applied to an ever-expanding set of physical [4] and operational [5] problems including process optimization. However, in manufacturing, more so than engineering design, the preparation of analytical models and the interpretation of their results do not share a common methodology across the various usages. This limits the ability to develop systematic means to apply analytical techniques to decision making. The consequences of this limitation include missed opportunities to reuse knowledge, unreliable results, and high cost of analysis. A report from the NIST [6] suggests that it should be possible, near-term, to reduce the cost of system verification and validation ten-fold. But how will this reduction be achieved?

There is no silver bullet to making analytical techniques more accessible to more manufacturers, but opportunities have emerged with the development of technologies that support 1) the industrial internet of things, 2) manufacturing domain ontologies, 3) domain specific languages, 4) natural language processing, and 5) metamodeling. This paper describes how these technologies may be applied using the Python programming language and Jupyter Notebooks which are web applications for the creation and sharing of documents that embody analyses. Notebooks contain live code, equations,

[1] Corresponding Author: Peter Denno, email: peter.denno@nist.gov.

visualizations and explanatory text [7]. The paper describes a notebook-based methodology, tailored for use in smart manufacturing environments, to reduce the cost of developing and validating process optimizations.

Section 2 of the paper describes the problem space. Section 3 explains how "equations as objects," formal requirements specifications, notebook annotations, and a metamodel of optimization are used in our methodology to produce analyses that are verified and integrated into manufacturing operations. Ideas are illustrated using a case study implementing a turning operation optimization described by Abdelmaguid and El-Hossany [8]. Section 4 concludes the paper with a discussion of limitations of the described process and planned future work.

2. From Analytical Problems to Systematic Solutions

Information needed to formulate manufacturing process optimizations - for example, to improve machining operations - may originate from many viewpoints. A viewpoint is a set of related concerns relevant to a particular audience [9]. Decision making in manufacturing typically relies on a composition of such viewpoints. For example, deciding what jobs to initiate in a job-shop environment might involve a composition of viewpoints concerning product demand, process plans, machine capabilities, machine instances and their availability, and inventory. The problem of composing viewpoints for decision making (i.e. of formulating the relevant analysis) is, in part, the problem of knowing what can be inferred from the union of information sources. This fundamental problem resists simple solution [10].

This paper describes a method in which it is assumed that the most creative part of the process, the problem formulation, has already been performed (perhaps guided by analyses described in the literature [8]). The practical problem that remains is that of making the analysis correct and effective for the idiosyncrasies of the given manufacturing context and the supporting information technology.

Verifying the correctness of an analysis requires knowing 1) what the analysis is to achieve (i.e. its requirements), 2) that the method (algorithm) chosen is suitable to address these requirements, and 3) that inputs to the algorithm are appropriate to it. With this knowledge, one gains confidence in the validity of the analysis, but not an understanding of the bounds on certainty, the sensitivity of parameters, nor the risk that the implied recommendations entail. Of these four qualities (validity, certainty, sensitivity and risk), validity is the most basic requirement. Though the others are important, they are not discussed in this paper.

3. Verifying Process Optimizations in Smart Manufacturing Environments

The literature contains an abundance of mathematical and physics-based models of manufacturing processes. Some of these models are "simple", since they are neither computationally demanding, nor require nuanced understanding of model parameters. Simple models can be applied more directly in daily production operations. For example, such models can be used by manufacturing engineers to adjust operating parameters in reaction to variation in raw materials.

Other models are "complex" since they require extensive setup or a geometric model (such as is the case with finite element analysis, for example). Abstractions can sometimes be found over complex models so as to create a simple model more suitable for daily use. An example of such an abstraction is a reduced-order surrogate model based on computational experiments performed on a complex model [11]. This paper concerns simple models and the simple form of complex models.

The methodology requires three classes of information to verify the correctness of an analysis. First is information linking variables and relations to terms in an ontology defining the intended meaning, dimensionality, and sources for values (e.g. a schema providing dimensional inspection results). Second is information linking calls to specialized analytical tools (e.g. optimizers) to a conceptual model of that tool. Third is information containing formal statements of requirements describing what the analysis is to achieve. These three elements are discussed, in turn, in the following sections.

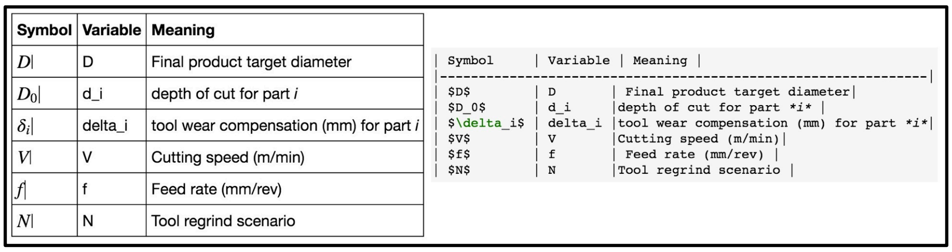
Verification is performed against analyses encoded in Jupyter notebooks using the Ipython language. Ipython is the interactive version of the Python language. Both Ipython and Python can use the many libraries developed for the language. Of particular value to manufacturing analysis are the scipy and numpy libraries for general mathematical tasks and optimization, the Pandas library for data structures and data analysis, and Statsmodels for statistics [12].

Jupyter represents user content (i.e. the analysis) as a Javascript Object Notation (JSON) data structure. The data structure (and its presentation in the browser) is divided into cells. A cell can be designated as containing (Python) code or markdown syntax annotations. Markdown cells can use LaTeX notation to represent mathematical formulas.

The design of the verification method is such that it does not interfere with the execution of the analysis. If the analysis provides markdown cells containing tables describing variables, these can be used in a separate process to annotate the analysis. If, on the other hand, such tables are not provided and sufficient information cannot be inferred through other means, verification will be incomplete.

3.1. Linking Variables to Ontology Terms and Sources

The JSON data structure can be parsed to identify variables defined in tables provided by the user; as is typical of Jupyter notebooks, these tables are in markdown syntax (See Figure 1). If the tables have the appropriate form, they will be interpreted as containing definitions of variables used in the analysis. Additionally, the process managing the validation can enable the user to associate OWL ontology terms with these variables. The ontology provides definitions and constraints on interpretation of elements important to the analysis such as symbols, variables, and their bindings to definitions. These definitions may concern manufacturing concepts.



Symbol	Variable	Meaning
D|	D	Final product target diameter
D_0|	d_i	depth of cut for part *i*
δ_i|	delta_i	tool wear compensation (mm) for part *i*
V|	V	Cutting speed (m/min)
f|	f	Feed rate (mm/rev)
N|	N	Tool regrind scenario

```
| Symbol      | Variable | Meaning |
|---------------------------------------------------------------|
| $D$         | D        | Final product target diameter|
| $D_0$       | d_i      |depth of cut for part *i* |
| $\delta_i$  | delta_i  |tool wear compensation (mm) for part *i*|
| $V$         | V        |Cutting speed (m/min)|
| $f$         | f        | Feed rate (mm/rev) |
| $N$         | N        |Tool regrind scenario |
```

Figure 1. A table as it appears in the Jupyter notebook (left), and the corresponding markdown syntax provide by the user (right).

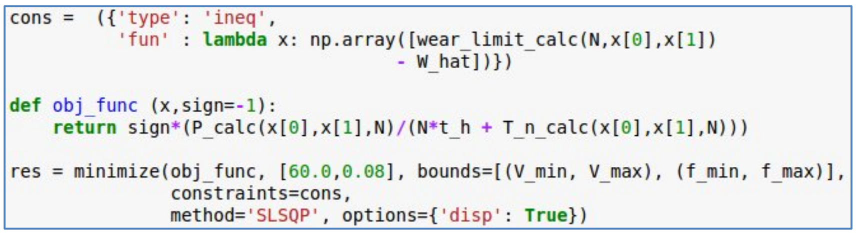
Similarly, the analysis can be annotated with knowledge of the sources of values (e.g. data conforming to XML schema or the schema translated to an OWL ontology).

Software has been written to parse the LaTeX-syntax mathematics in markdown cells to OWL statements. Because mathematical expressions can have complex syntax, a single mathematical expression can produce many interrelated OWL facts. This makes processing difficult, but not impossible. The software implements an "equations as objects" method of operation described in prior work [13]. By making mathematical relations visible outside of the tools in which they are initially developed, these relations can be reused and refined. For example, an empirically-defined, predictive model of a manufacturing process developed in one analysis (one notebook) can be referenced in an optimization or linked to additional operational data and refined in other notebooks.

3.2. Characterizing the Use of Optimizers with an Optimization Metamodel

A principal challenge in using analytical tools such as optimizers is that of knowing what pattern of tool use is appropriate to the problem at hand. There are two aspects to addressing this challenge: representing the domain problem and representing the capabilities and interface of the analytical tool. In the method, the domain problem is represented through a combination of the ontology links discussed in the previous section and the formal requirements characterization discussed in the next section. This section discusses how the user's specification of the objective function, constraints and call to the optimizer are characterized for subsequent validation.

The characteristics of optimization techniques can be modeled using a metamodel. A *metamodel,* as used here, is a model of a modeling language providing sufficient detail to serve as the storage form for instances of the model. Prior work [14] has produced a preliminary UML-based metamodel of optimization to represent optimization problems defined in the Optimization Programming Language (OPL). This metamodel is still under development and being adapted to represent the optimization techniques found in Python scipy. Depicted in Figure 2 is an optimization call such as used in the turning optimization. The vector *x* is the design vector; *x*[0] the cutting speed, and *x*[1] the feed rate. As shown, these are initialized to [60.0, 0.08] respectively. For brevity, only part of the constraint vector is shown.



```
cons =  ({'type': 'ineq',
          'fun' : lambda x: np.array([wear_limit_calc(N,x[0],x[1])
                                      - W_hat])})

def obj_func (x,sign=-1):
    return sign*(P_calc(x[0],x[1],N)/(N*t_h + T_n_calc(x[0],x[1],N)))

res = minimize(obj_func, [60.0,0.08], bounds=[(V_min, V_max), (f_min, f_max)],
               constraints=cons,
               method='SLSQP', options={'disp': True})
```

Figure 2. The call to the optimizer in the turning example.

The optimization metamodel describes a conceptualization of optimization problems. The use of the optimizer, such as in the code in Figure 2, represents a particular instance of the metamodel. This instance can be expressed as RDF and OWL triples. In creating this instance, it is essential to recognize how the information provided in the call to the optimization tool relates to the optimization metamodel. For example, in the case of the Scipy-based tool depicted in Figure 2, the parameter named *method* indicates the algorithm applied (sequential least squares programming); *bounds* provides constants bounding the feasible region; and, *constraints* provides a vector of functions that can be called with the design vector to evaluate inequality and equality constraints. Integrating this knowledge with variable and equation definitions acquired through the markdown cells requires parsing the call to *minimize* and related constraint definitions.

3.3. Stating Requirements about Analyses

Verification is performed against requirements that originate as informal statements about what the system (in our case an analysis) is to do. Prior work [15] provides a method to construct a controlled, natural-language parser for a domain-specific language. The method involves recognizing what the sentence intends by first classifying its verb by calculating the semantic distance to it from synonyms of all "proto-verbs" of the domain-specific language. The current implementation of the method does not provide such a parser; it only provides similar resulting statements based on a pilot set of proto-verbs: "optimize" "constrain" and "find." This set is likely to grow; the prior work required 19 proto-verbs. The resulting statements are encoded in a dialect of ISO Common Logic. An example statement in the dialect – one describing the requirement that the analysis should maximize profit under constraint of acceptable surface finish and dimension tolerance – is as follows:

```
(obligation-of
   analysis-x
   (that
      (and (optimize profit)
           (under-constraint surface-finish)
           (under-constraint dimensional-tolerance))))
```

3.4. Using Links, Metamodels, and Requirements to Verify Analyses

Inconsistencies in the analysis are identified using a Bayesian approach on a graph-based structure. The process is similar to that described by Herzig [16]. In the process, a graph of binary relations is produced from the sources described in the previous three sections (ontology-annotated variables and terms, the optimization metamodel instance, and requirement statements).

As an example of how inconsistencies are recognized, consider the segment of the turning analysis depicted in Figure 2. *W_hat* in the figure is defined through links to the ontology to be the maximum acceptable tool wear. The translation of the Python constraint through the optimization metamodel indicates that the expression of the constraint on tool wear, `wear_limit_calc(N, x[0], x[1], ) - W_hat`, has its sign reversed from what is correct. (Inequality constraints provided to the python minimization function should be stated such that their values are positive inside the feasible region.) The fact that tool wear is generally not a desirable quality of machining and yet the analysis seemingly seeks a value of tool wear *above W_hat* allows Bayesian inference to identify the inconsistency against the world model reflected by the ontology.

4. Conclusion

This paper discusses a methodology for performing and verifying optimization-based analyses of manufacturing processes and operations. The method applies a "soft system perspective" [2] to facilitate the use of analytical tools in smart manufacturing environments. Elements of the method have been implemented on the turning optimization described in [8] as well as an empirical model of an additive manufacturing process [13]. Much work remains in developing the analytical metamodel for optimization and integrating system components. An analysis for job shop scheduling is being developed. As production scheduling can involve very different viewpoints from those used in unit manufacturing processes, it will be necessary to develop the ontology towards these viewpoints.[3]

References

[1] J. Davis, T. Edgar, J. Porter, J. Bernaden, and M. Sarli, "Smart Manufacturing , Manufacturing Intelligence and Demand-Dynamic Performance," in *FOCAPO2012: Foundations of Computer-Aided Process Operations*, 2012.

[2] VDE-DKE, "The German Standardization Roadmap Industrie 4.0," *Vde Assoc. Electr. Electron. Inf. Technol.*, vol. 0, pp. 1–60, 2014.

[3] Government Office for Science, "The Future of Manufacturing," London, 2013.

[4] K. Allen, Dell, Alting, Leo, and H. Todd, Robert, *Fundamental Principles of Manufacturing Processes*. Industrial Press, 2005.

[5] J. Li and S. M. Meerkov, *Production System Engineering*. Springer Science+Business Media, 2009.

[6] National Institute of Standards and Technology, "Foundations for Innovation in Cyber-Physical Systems," Gaithersburg, 2013.

[7] Project Jupyter, "Jupyter Notebooks." [Online]. Available: http://jupyter.org/. [Accessed: 28-Apr-2016].

[8] T. F. Abdelmaguid and T. M. El-hossainy, "Optimal Cutting Parameters for Turning Operations with Costs of Quality and Tool Wear Compensation," *Proc. 2012 Int. Conf. Ind. Eng. Oper. Manag. Istanbul, Turkey, July 3 – 6*, pp. 924–932, 2012.

[9] ISO, *ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description*. 2011.

[10] N. F. Noy, "Semantic Integration: A Survey Of Ontology-Based Approaches," *Newsl. ACM SIGMOD Rec.*, vol. 33, no. 4, pp. 65–70, 2004.

[11] A. Jeang, H. C. Li, and Y. C. Wang, "A computational simulation approach for optimising process parameters in cutting operations," *Int. J. Comput. Integr. Manuf.*, vol. 23, no. 4, pp. 325–340, 2010.

[12] W. McKinney, *Python for data analysis*. O'Reilly, 2013.

[13] P. O. Denno and D. B. Kim, "Integrating views of properties in models of unit manufacturing processes," *Int. J. Comput. Integr. Manuf.*, no. March, pp. 1–17, 2015.

[14] I. Assouroko and P. O. Denno, "A metamodel for optimization problems," Gaithersburg, 2016.

[15] P. O. Denno and C. Chang, "Validating controlled English statements of requirements using functional models," 2016, pp. 1–10.

[16] S. J. I. Herzig, "A Bayesian Learning Approach To Inconsistency Identification in Model-Based Systems Engineering," Georgia Institute of Technology, 2015.

[17] P. Checkland, "Soft Systems Methodology: A Thirty Year Retrospective," *Syst. Res. Behav. Sci. Syst. Res*, vol. 17, pp. 11–58, 2000.

[2] Checkland [17] distinguishes "hard system perspectives' from "soft systems perspectives": in the former, the world is viewed as systemic; in the later, the world is viewed as unmanageably complex, but the process of inquiry is systemic.

[3] Certain commercial software products are identified in this paper. These products were used only for demonstration purposes. This use does not imply approval or endorsement by NIST, nor does it imply these products are necessarily the best available for the purpose.