Production as a Service: A Digital Manufacturing Framework for Optimizing Utilization

Efe C. Balta, *Student Member, IEEE*, Yikai Lin, Kira Barton, *Member, IEEE*, Dawn M. Tilbury, *Fellow, IEEE*, and Z. Morley Mao, *Member, IEEE*

***Abstract*— In current practice, product developers with customized small batch production needs come across the problem of finding capable and flexible manufacturers, whereas manufacturers face underutilization due to inconsistent demand. This paper presents a Production as a Service (PaaS) framework to connect users (consumers or product developers) who have customized small batch manufacturing needs with manufacturers who have existing underutilized resources. PaaS is a cloud-based, centralized framework based on a service-oriented architecture that abstracts the manufacturing steps of a product as individual (production) service requests. Using PaaS, the user is able to reach many capable manufacturers at once and receive quotations for the production request. On the other end, PaaS reduces the effort required to find new customers and enables the manufacturers to easily submit quotations to increase the utilization of their resources. The functionalities and concepts defined in this paper are illustrated through case studies.**

***Note to Practitioners*—In order to transition product fabrication from the design phase to manufacturing, there is a need for identifying capable manufacturers with available resources that could be paired with user requirements. We propose Production as a Service (PaaS) in this paper and present initial implementations of the front-end and back-end components with a customizable optimization algorithm. Proposed abstractions and data structures make PaaS a unique, efficient, and intellectual property preserving framework. The current implementation of the framework has been tested with new designs. The PaaS framework has the potential to scale over a large network of manufacturers to effectively coordinate geo-distributed manufacturers for custom manufacturing needs.**

***Index Terms*— Genetic algorithms, intellectual property, intelligent manufacturing systems, manufacturing automation, service-oriented systems engineering.**

Nomenclature

c^i	Component i, $c^i \in \mathcal{C}$.
$\mathcal{C}$	Set of all components in the product.
M_j	Process specification.
$\vec{y}$	List of manufacturing methods.
$\mathcal{T}_k$	Tolerance specification.
$(\mu_{\mathcal{T}}, \sigma_{\mathcal{T}})$	Process mean and standard deviation.
$\mathbb{D}$	Set of all quotes for optimization.
s	Abstraction of a production quote.
$\mathbb{P}_{\mathcal{C}}$	Set of all requested processes for components in $\mathcal{C}$.
$\mathcal{P}_{\mathbb{P}_{\mathcal{C}}}$	Set of precedence constraints for the set $\mathbb{P}_{\mathcal{C}}$.
$\vec{p}$	List of processes in a quote ($\vec{p} \subseteq \mathbb{P}_{\mathcal{C}}$).
S	Candidate solution.
X_{c^i}	Quotes in S associated with c^i.
$\mathcal{L}$	Location matrix.
κ_i, t_i, q_i	Cost, time, quality of quote s_i.
m	Minimum utilization rate.
β	Batch size.
b_{ij}	Percent of β to be sent from i to j.
$\alpha_0, \alpha_1, \alpha_2$	Cost, time, quality weights.
ζ_κ, ζ_t	Unit transportation cost and time.

Manuscript received October 24, 2017; revised April 18, 2018; accepted May 8, 2018. Date of publication June 27, 2018; date of current version October 4, 2018. This paper was recommended for publication by Associate Editor X. Hu and Editor M. P. Fanti upon evaluation of the reviewers' comments. This work was supported by NSF 1546036. *(Corresponding author: Efe C. Balta.)*

E. C. Balta, K. Barton, and D. M. Tilbury are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: baltaefe@umich.edu; bartonkl@umich.edu; tilbury@umich.edu).

Y. Lin and Z. M. Mao are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: yklin@umich.edu; zmao@umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2018.2842690

I. Introduction

With the increasing demand for product customization, manufacturing systems are becoming more flexible, customizable, and intelligent. Conventional manufacturing systems with static supply chains (SCs) and production plans for mass production lack flexibility to coordinate distributed manufacturers and their resources in an efficient manner. As a result, product developers with small batch custom manufacturing needs often struggle with the task of finding manufacturers with available resources and sufficient system flexibility to integrate new products into the production line. On the other hand, manufacturers with idle resources face underutilization due to inconsistent demand. Developing a method for automatically matching product developers with manufacturers will enable faster customized production cycles and more efficient use of the available resources in the manufacturing industry. Current efforts to address this challenge include companies that leverage a labor-intensive manual approach[1] and some automated Web-based solutions[2] which are not very flexible due to the lack of abstraction.

The use of digital manufacturing solutions to enable automated matching of resources to demands is of recent research interests in both academia and industry [1], [2]. Service-oriented architectures (SOAs) have been proposed in software engineering to address the need for collaboration and abstraction to accomplish the customized tasks [3], [4].

[1]seeedstudio.com.

[2]MFG.com and xometry.com.

The use of SOA in manufacturing automation is well documented in the literature [5], [6]. A hybrid service-oriented and agent-based approach to coordinate production in a flexible manufacturing system is proposed in [7]. Integration of different layers of the plant floor in a business-oriented encapsulation model is proposed without the integration of multiple manufacturers in [8]. Manufacturing-as-a-service framework and its implementation as an automated production system is given in [9]. However, the scalability of the approach is not discussed. These architectures also lack a mathematical framework to map the production requirements into a mathematical model for optimization of performance metrics.

The integrated e-SC is modeled as a network consisting of information and material interactions and optimized using an integer linear programming algorithm to give a mathematical foundation for the SOA in [10]. SOA-based manufacturing frameworks in the literature do not consider the abstractions of a customer's product requirements and manufacturer capabilities for efficient matching and intellectual property (IP) protection. A supply–demand matching hypernetwork model for various configurations of the SC as connectivity graphs is given in [11]. The need for a structured ontology for realizing service-based automation in manufacturing is studied by many, but the customer demand in terms of product requirements and manufacturing capabilities is not captured in the existing ontologies [12], [13].

Cloud manufacturing is a digital platform for effective communication of customer demands with the manufacturers that offer services [14]. The proposed framework in this paper borrows from cloud manufacturing to derive an extended framework that contains the necessary interfaces and mathematical models to realize cloud manufacturing concepts in implementation. In addition, data abstractions that preserve the user's IP distinguishes the proposed framework over previous work.

The proposed Production as a Service (PaaS) framework addresses a need for a unifying and scalable approach with the appropriate ontology to: 1) abstract the product data from the customer end; 2) translate the customer demand into manufacturer capabilities; and 3) optimize the requested services for each product for both the manufacturer and the customer. This paper is built on previous work in [15] and [16]. The primary contributions of this paper over the previous work are: 1) a formal description of the front-end and back-end functionalities of the PaaS framework; 2) the mathematical formulation of the matching process to utilize an optimization algorithm that considers batch splitting and precedence constraints; and 3) a formal description of the construction of the framework application programming interfaces (APIs) to integrate the front-end and the back-end components.

II. PaaS Framework Overview

The overview of the PaaS framework is given in Fig. 1. A product is decomposed into its components to identify the required manufacturing processes, which are abstracted as service requests. The information flow in the framework is shown in Fig. 1. First, a service request is instantiated using request for quotation (RFQ) input from the user. Second, the abstracted design feature data are parsed using the feature extraction module. Third, manufacturers with suitable manufacturing capabilities submit quotes using a predefined quote format. Solutions are formed using a specialized optimization algorithm that takes precedence constraints and customer-defined preferences on cost, time, and quality (CTQ) into consideration. Finally, the set of quotes that span a feasible production scheme (e.g., the solution set) are presented to the user. Once the user selects a specific solution, the manufacturers specified within the solution are linked with the user to refine details and initiate the production process.

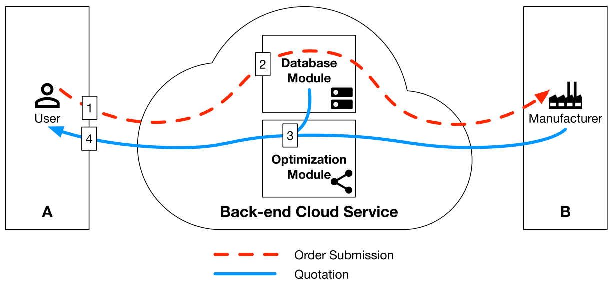


Fig. 1. High-level framework structure and the key steps in the manufacturing service request [15].

A. Definitions

1) *Product:* Final produced goods from the requested services.
2) *Component:* Discrete manufactured goods produced by a series of manufacturing processes.
3) *Process:* Single manufacturing step applied to a component or product.
4) *Feature:* Abstracted representation of the required data for manufacturing a subset of a component's geometry. Features are encoded in a predefined format associated with a process.
5) *Process Precedence Constraints:* Constraints on the manufacturing order of multiple processes. We denote a set of precedence constraints on an unordered set of z processes $A = \{\gamma_1, \ldots, \gamma_z\}$ by $\mathcal{P}_A = (\bar{I}_A, \bar{E}_A, \bar{M}_A)$.
   a) $\bar{I}_A$ set of admissible initial processes.
   b) $\bar{E}_A$ set of soft constraints. $\bar{e}^{\gamma_n}_{\gamma_j,\gamma_k}$ defines a soft ordered set $\{\gamma_j, \ldots, \gamma_n, \ldots, \gamma_k\}$.
   c) $\bar{M}_A$ set of hard constraints. $\bar{m}^{\gamma_n}_{\gamma_j,\gamma_k}$ defines a structured order $\{\gamma_j, \gamma_n, \gamma_k\}$.

B. Basic Architecture

The PaaS framework consists of three main components, as shown in Fig. 1. The customer interface (A) is where the users submit service requests and receive optimized sets of solutions. The manufacturer interface (B) is where the manufacturers review service requests and submit production quotes. The back-end cloud service contains the secured database, optimization algorithms, and APIs. Coordination between the users and manufacturers requires translating user's requests into manufacturer capabilities. Direct communication between users and manufacturers is not required in the quoting

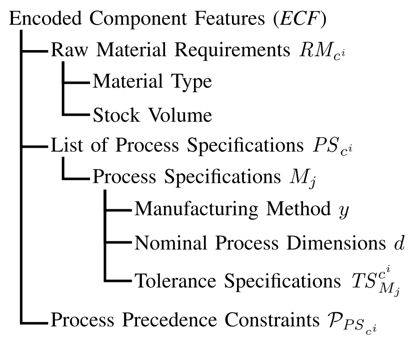

Encoded Component Features (ECF)
Raw Material Requirements RM_{c^i}
Material Type
Stock Volume
List of Process Specifications PS_{c^i}
Process Specifications M_j
Manufacturing Method y
Nominal Process Dimensions d
Tolerance Specifications $TS^{c^i}_{M_j}$
Process Precedence Constraints $\mathcal{P}_{PS_{c^i}}$

Fig. 2. Structure of the abstracted ECF format for component c^i.

process, which in turn increases the efficiency and security within the framework.

III. Customer Interface

The customer interface is where the user interacts with the framework to submit production requests and receives notifications once a list of possible production schemes are evaluated by the framework.[3] An RFQ interface for users to input requests and a feature extraction module to assist the user with request inputs are discussed in the following.

A. RFQ Interface

In the RFQ interface, the user submits manufacturing service requests. The RFQ interface captures data, such as the name of the project, deadline for quote submission, deadline for receiving orders, batch size, preferences on CTQ, and the encoded product features (EPFs).

In addition, the user can access the feature extraction module through the RFQ interface to abstract the design features and create the EPF.

For the current formulation, the submitted RFQs are assumed to be associated with small batch productions. As such, the user and manufacturer data are considered to be static during the matching process. Dynamic updates [18] will be addressed in the future work.

B. Encoded Product Features

EPF is an abstracted data format that can be used to describe critical aspects of the product to be manufactured while preserving IP and enabling efficient data sharing [15]. A feature is a subset of the model's geometry that is readily available as drawing objects (e.g., boss/extrudes and holes) in most commercial software. After the EPF has been created, the user's design is represented using this abstracted format so that sensitive data will not be shared with multiple manufacturers. Previous efforts to classify similar engineering designs for faster design cycles and sustainability can be found in the literature [19], [20]; however, the use of abstraction for IP protection is new in this framework. Importantly, EPF provides an effective data format for manufacturers to comprehend the requirements without a drawing.

$\text{EPF} = \{\text{ECF}_{c^i} | i = 1, \ldots, |\mathcal{C}|\}$ is made up of a list of features for each component c^i from the list of all components $\mathcal{C}$. Fig. 2 shows the hierarchical structure of the ECF data format. To effectively and efficiently assess the kind of data to be abstracted from a computer-aided design (CAD) file, we break down a product into individual components. Ameri and Dutta [21] propose the Manufacturing Service Description Language (MSDL) as an ontology for manufacturing service matching. MSDL uses the geometry and tolerances of a design for performing a matching operation. Here, we use similar design attributes to additionally protect the IP of the user and communicate the requested services to the manufacturers. $\text{ECF}_{c^i} = \{\text{RM}_{c^i}, \text{PS}_{c^i}, \mathcal{P}_{\text{PS}_{c^i}}\}$ represents the predefined attributes of component c^i in the design, where RM_{c^i}, PS_{c^i}, and $\mathcal{P}_{\text{PS}_{c^i}}$ are as follows.

1) Raw material RM_{c^i} denotes the material type and volume.
2) Process Specification $\text{PS}_{c^i} = \{M_1, \ldots, M_n\}$ is the list of required processes to manufacture the identified features.
   a) Process Specification $M_j = (y, d, \text{TS}^{c^i}_{M_j})$ defines the manufacturing method y, nominal process dimensions d, and tolerance specifications $\text{TS}^{c^i}_{M_j}$ for each feature.
   b) $\text{TS}^{c^i}_{M_j} = \{\mathcal{T}_0, \ldots, \mathcal{T}_l\}$ is the list of tolerance specifications for process M_j. A process tolerance is abstracted as $\mathcal{T}_k = (\mathcal{T}^{\text{ID}}, \mathcal{T}_{\text{type}}, \mathcal{T}_{\text{dim}}, \mathcal{T}_{\text{UD}}, \mathcal{T}_{\text{LD}})$, where $\mathcal{T}^{\text{ID}}$ denotes the unique ID of the tolerance, $\mathcal{T}_{\text{type}}$ denotes the type of tolerance (i.e., concentricity and fitting), $\mathcal{T}_{\text{dim}}$ defines a central limit (CL) for the tolerance, and $(\mathcal{T}_{\text{UD}}, \mathcal{T}_{\text{LD}})$ denote the acceptable upper (USL) and lower (LSL) deviations from the CL.

   For example, a through hole in component c^i denotes M_j as a drilling operation y, hole diameter d, and engineering tolerances $\text{TS}^{c^i}_{M_j}$ associated with the dimensions. With this abstracted data, a manufacturer can estimate the tooling cost and manufacturing time to provide a production quote.
3) Process Precedence Constraints $\mathcal{P}_{\text{PS}_{c_i}}$ denote the precedence relations among the set of required manufacturing processes PS_{c^i} for the component c^i.

We define the union of the requested processes (services) in PS_{c^i} for all components in $\mathcal{C}$ as the set $\mathbb{P}_{\mathcal{C}}$. Therefore, $\mathbb{P}_{\mathcal{C}} = \{p_j | j \in \text{PS}_{c^i}, \forall c^i \in \mathcal{C}\}$ is the union of all M_j from all the components in the EPF. Note that $p_j \in \mathbb{P}_{\mathcal{C}}$ is a unique abstraction of each M_j, and p_j is used in the mathematical formulation instead of M_j.

C. Feature Extraction Module

A feature extraction software is incorporated in the front-end, enabling the user to submit 2-D or 3-D drawings of the requested product to create the EPF. The extraction process has been automated by a feature extraction software to improve ease of use.

The feature extraction logic is given in Algorithm 1. Geometric features from a CAD files are extracted automatically, and the user is prompted to select/input required information. This method uses the existing feature recognition software to leverage the existent knowledge in computer graphics and to help the users with limited experience

[3]An implementation of the framework interfaces can be found in [17].

Algorithm 1 Feature Extraction Module

```
1: Get the components list $\mathcal{C}$ from the $CAD$
2: for each $c^i \in \mathcal{C}$ do
3:     Get raw material requirements $RM_{c^i}$
4:     Get the list of features from the $CAD$.
5:     Prompt the user to select features from the list.
6:     for each feature selected by the user do
7:         Get the nominal dimension $d$
8:         Request $y$ from a drop-down list.
9:         Get the list of dimensions in the feature
10:        Prompt the user to select dimensions from the list.
11:        for each dimension selected by the user do
12:            Get tolerance specification data $\mathcal{T}$ from $CAD$
13:            $TS_M^{c^i} \leftarrow \mathcal{T}$
14:        end for
15:        $M \leftarrow (y, d, TS_M^{c^i})$
16:        $PS_{c^i} \leftarrow PS_{c^i} \cup M$
17:    end for
18:    $ECF_{c^i} \leftarrow (RM_{c^i}, PS_{c^i}, \mathcal{P}_{PS_{c^i}})$
19:    $EPF \leftarrow EPF \cup ECF_{c^i}$
20: end for
```

in the field of CAD. The parsing of EPF data can be implemented with well-known CAD APIs, ensuring that the process is adaptable for different platforms and possible future extensions. For 2-D drawings, the user must identify the dimensions that constitute the geometry of a feature on line 4 in Algorithm 1, and then, the ECF is created from the 2-D drawing input.

In current practice, feature recognition software is often coupled with material cost and tooling cost to estimate production time and cost. There are well established standards and practices for encapsulating the geometry, feature recognition, product life-cycle data, and the manufacturing process requirements in CAD exchange [22]–[24]. Computer-aided process planning (CAPP) using CAD feature recognition data and production data is documented in the literature [25]–[27]. There are recent efforts for data exchange formats supporting geometric tolerances in 3-D models, such as the AP242 data format [28]. This paper proposes a new method combining existing CAD frameworks with feature recognition software to efficiently abstract the EPF data structure directly from 2-D/3-D models. It is assumed that an experienced engineer assigns the precedence constraints PS_{c^i} for the EPF. The feature extraction method is implemented using the API framework of Solidworks.

IV. Manufacturer Interface

The manufacturer interface allows manufacturers to interact with the PaaS framework to receive notifications of available jobs and submit production quotes with the required data. Interactions derive from the following two main categories.

1) *Notification of User Request:* Once a service request is processed, the manufacturer surveys (MFSs) are created by the production quote API using the RFQ data and forwarded to respective manufacturers with adequate capabilities.
2) *Submission of Manufacturing Quotes:* When a manufacturer submits a quote for one or multiple processes associated with a request, a response to the MFS is submitted. The responses are received and formatted as raw quotes (RQ) and quality surveys (QS) by the manufacturer interface.

Raw quotes, $\text{RQ} = (\vec{p}, \vec{y}, \kappa, t, m, \text{mfr}^{\text{ID}})$, capture the essential quote data, where $\vec{p} \subseteq \mathbb{P}_{\mathcal{C}}$ is the list of processes that may include a single process or multiple processes associated with the quote, denoted as $\vec{p} = \{p_1, \ldots, p_n | p_j \in \mathbb{P}_{\mathcal{C}}\}$, mfr^{ID} is the unique manufacturer ID, and remaining variables are defined in Nomenclature.

In the MFS, manufacturers are asked to specify a minimum batch size for the submitted quote. The minimum utilization rate m is calculated as the specified minimum batch size over the user-defined total batch size. To estimate the quality of the manufacturing service, manufacturers are asked to specify the process capabilities, which are then captured in the QS data structure. The QS data structure includes process capabilities for each $\mathcal{T}$ specified for the processes associated with the submitted quote. Thus, we define $\text{QS} = \{(\mu_{\mathcal{T}}, \sigma_{\mathcal{T}}) | \forall \mathcal{T} \in \vec{p}\}$, where $(\mu_{\mathcal{T}}, \sigma_{\mathcal{T}})$ are the process mean and variability in terms of standard deviation.

C_{pk}, the process capability index, is used in statistical process control to estimate the capability of a process to stay within the specified upper and lower bounds, including the cases when the CL is not centered between the bounds. We use $(\mu_{\mathcal{T}}, \sigma_{\mathcal{T}})$ specified by the manufacturer and $(\mathcal{T}_{\text{UD}}, \mathcal{T}_{\text{LD}}) \in \mathcal{T}$ specified by the design to compute C_{pk}

$$C_{\text{pk}}(\mathcal{T}) = \min\left[\frac{\mathcal{T}_{\text{UD}} - \mu_{\mathcal{T}}}{3\sigma_{\mathcal{T}}}, \frac{\mu_{\mathcal{T}} - \mathcal{T}_{\text{LD}}}{3\sigma_{\mathcal{T}}}\right]. \tag{1}$$

The value of C_{pk} index correlates to a yield percentage according to six sigma regulations [29]. The resulting yield is assigned to the manufacturing quote RQ as the quality index q. We use the target value $\mathcal{T}_{\text{dim}}$ in the quoting process for the manufacturers to estimate their process capabilities in terms of $(\mu_{\mathcal{T}_j}, \sigma_{\mathcal{T}_j})$ around the given target and specified (USL and LSL). Tolerance guidelines associated with the manufacturing methods in $\vec{y}$ are used when a manufacturer does not have sufficient statistical process data available to derive the process index.

V. Optimization Module: Genetic Algorithm Decision Maker

A genetic algorithm (GA) is proposed for evaluating the optimal production schemes using the quotes submitted to the framework. The GA method of [15] is extended by adding a batch splitting algorithm to enable multiple manufacturers to work in parallel for some or all of the services.

A. Problem Formulation

Abstraction of a quote RQ is defined as $s = (\vec{p}, \kappa, t, q, m, \text{mfr}^{\text{ID}}, s^{\text{ID}})$, where q is the quality index and s^{ID} is the unique quote ID.

Let $S = [s_1, \ldots, s_n]$ denote a feasible list of n quotes, where $s_i \in \mathbb{D}$ denotes a manufacturing quote and $\mathbb{D}$ is the set of all quotes. S is said to be feasible if and only if: 1) the precedence of the quotes in S satisfies the process precedence constraints $\mathcal{P}_{\mathbb{P}_{\mathcal{C}}}$ defined on set $\mathbb{P}_{\mathcal{C}}$ and 2) the quotes in S include all the requested processes in the set $\mathbb{P}_{\mathcal{C}}$, where $\mathcal{P}_{\mathbb{P}_{\mathcal{C}}} = \{\cup \mathcal{P}_{\mathrm{PS}^{c^i}} | \forall c^i \in \mathcal{C}\}$ denotes the union of all the precedence constraints on individual components. Therefore, $\mathbb{P}_{\mathcal{C}} = \mathrm{span}\{\vec{p}_1, \ldots, \vec{p}_{|S|} | \vec{p}_i \in s_i, \forall s_i \in S\}$ must hold by 1) for each feasible S. We can partition $\mathbb{P}_{\mathcal{C}}$ for each $c^i \in \mathcal{C}$ to define $X_{c^i} = [s_j, \ldots, s_k]$, which denotes the production quotes associated only with c^i. Then

$$S = \{X_{c^1} \cup \ldots \cup X_{c^z}\}$$

is an alternate representation that is used in evaluating the componentwise processing time. The feasibility condition must hold for all feasible solution candidates. From this point onward, any mention of the term *solution* refers to only feasible solutions unless stated otherwise. The type of solutions in this paper can be categorized into the following two groups.

1) *Single Manufacturer per Process (SMPP):* This type of solution allows only one manufacturer to be assigned to a requested manufacturing service. Thus, the batch size for each of the quotes is the total batch size.
2) *Multiple Manufacturer per Process (MMPP):* This type of solution allows the assignment of multiple manufacturers per requested manufacturing service. This solution type allows potentially faster production cycles with multiple manufacturers working in parallel for the bottleneck processes in the SMPP scenario.

Let $\mathcal{L} \in \mathbb{R}^{(n+1)\times(n+1)}$ be a distance matrix between the n manufacturers that contributed (single or multiple quotes per manufacturer) to $\mathbb{D}$ and the user. Then, $\mathcal{L}(i, j)$ denotes the physical distance between the locations of manufacturers i and j. We define $J_C(S)$ as the cost function

$$J_C(S) = \sum_{i=1}^{n} b_i \left[\kappa_i + \zeta_\kappa \mathcal{L}\left(\mathrm{mfr}_i^{\mathrm{ID}}, \mathrm{mfr}_j^{\mathrm{ID}}\right)\right]$$

where b_i is the optimal batch size percentage (given in Section V-B), ζ_κ denotes the unit transportation cost, and $\mathrm{mfr}_j^{\mathrm{ID}}$ denotes the next manufacturer to ship the processed goods, which is defined according to the precedence constraints. For the last process, $\mathrm{mfr}_j^{\mathrm{ID}}$ is the user's location.

$J_T(S)$ is the time function representing a generalized time model for a production scheme with possible assemblies

$$J_T(S) = \mathrm{argmax}\left\{U_{c^1}(S), \ldots, U_{c^k}(S) | c^i \in \mathcal{C}\right\} + t_{\mathrm{asm}}^*$$
$$U_{c^k}(S) = \sum_{i=1}^{|X_{c^k}|} (\tau_i) + \zeta_t \mathcal{L}\left(\mathrm{mfr}_{|X_{c^k}|}^{\mathrm{ID}}, \mathrm{mfr}_{\mathrm{asm}}^{c^k}\right)$$

where τ_i is the time for completing the list of services $\vec{p}_i \in s_i$, where $s_i \in X_{c^i}$, $\mathrm{mfr}_{\mathrm{asm}}^{c^i}$ is the assembly location for c^i, ζ_t is the unit transportation time, and t_{asm}^* is the time for the total assembly. Thus, $U_{c^k}(S)$ is the time for manufacturing and shipping c^k to its assembly location.

In this paper, we relax the conditional assignment of SMPP in [15] and allow manufacturers to work in parallel. Therefore, τ_i is piecewise-defined for the two cases when: 1) $\vec{p}_i \in S$ share services (MMPP) and 2) the quotes are disjoint in terms of processes (SMPP)

$$\begin{aligned} &(1) \ \cap p_i \neq \emptyset, \quad \tau_i = f_t(\bar{b}) : \quad \bar{b} = [b_1, \ldots, b_x] \\ &(2) \ \cap p_i = \emptyset, \quad \tau_i = t_i \end{aligned}$$

where $f_t(\bar{b})$ is the optimal time for x quotes in parallel and $\bar{b}$ is the vector of the batch sizes for the x quotes. Moreover, for the SMPP and the MMPP solutions, it is useful to introduce

$$\begin{aligned} T_{\mathrm{SMPP}} &= \inf\{J_T(S_i)\} : \quad \forall S_i \in \mathit{SMPP} \\ J_T(S)_{\mathrm{MMPP}} &\leq T_{\mathrm{SMPP}} : \quad \forall S \in \mathrm{MMPP} \end{aligned} \tag{2}$$

where T_{SMPP} denotes the least attainable processing time for the SMPP type solutions formed using the quotes in set $\mathbb{D}$, and (2) states that any solution of type MMPP cannot have a longer processing time than T_{SMPP}. The quality function $J_Q(S)$ is defined as follows:

$$J_Q(S) = \prod_{i=1}^{n} (q_i).$$

Since the previously assigned quality index q_i denotes a yield, the product of all the quality indices is the quality measure of the solution. Note that if there are additional quality inspections requested for the assembly of components, the quality function might be modified to better represent the expected quality instead of just the estimated index q_i.

The high-level optimization problem is given as

$$\min_S \ J(S) = \alpha_0 \frac{J_C(S)}{\mu_{\mathrm{cost}}} + \alpha_1 \frac{J_T(S)}{\mu_{\mathrm{time}}} + \alpha_2 (1 - J_Q(S)) \tag{3}$$
$$\text{s.t. } \mathcal{P}(S) \subset \mathcal{P}_{\mathbb{P}_{\mathcal{C}}} \tag{4}$$
$$b_i \geq m_i \beta \quad \forall b_i \in S \tag{5}$$

where $J(S)$ is the fitness function, α_0, α_1, and α_2 are the weights (implicit CTQ coefficients), β is the size of the total batch, and μ_{cost} and μ_{time} are normalizing means. $(1 - J_Q(S))$ is used since the quality should be maximized for an optimal solution. Equation (3) is the objective function to be minimized. In (4), we use $\mathcal{P}(\cdot)$ to denote the precedence scheme of the candidate solution S. Equation (5) specifies that the batch size percentages assigned to the quotes in parallel cannot be less than the minimum utilization rate specified by the manufacturer. Before moving to the GA formulation, we need to define the solution to the batch splitting problem.

B. Batch Splitting Problem

We pose a linear program (LP) to determine the optimal time $f_t(\bar{b})$ and batch size b_i for the quotes that are utilized in parallel. Without the loss of generality, we split the batch of size β between N manufacturers working in parallel for a single process p_i with l destinations to ship the processed goods. Then, we partition each $b_i \in \bar{b}$ as

$$b_{ij}, \quad \text{where} \begin{cases} i = 1, \ldots, N : & \text{Source} \\ j = 1, \ldots, l : & \text{Destination} \end{cases}$$

where $b_i = \sum_{j=1}^{l} b_{ij}$ is the batch size to be manufactured for quote s_i^{ID} and b_{ij} denotes the size of the batch to be sent from manufacturer i to destination j. Moreover, we define $B \in \mathbb{R}^{N\times l}$, where $b_{ij} = B(i,j)$. Let $L \in \mathbb{R}^{N\times l}$ be a matrix denoting the distances $\mathcal{L}(i,j)$ and $D \in \mathbb{R}^{N\times l}$ be a matrix with identical l columns as costs of the quotes, such that $D(\cdot,t) = [\kappa_1,\ldots,\kappa_N]^T\ :\ \forall t$. Then, the objective to be minimized for minimum cost can be written as

$$\sum_{i=1}^{N}\sum_{j=1}^{l} b_{ij}[\zeta_\kappa \mathcal{L}(i,j) + \kappa_i]$$

which is equivalently

$$tr(B(L+D)^T) = [\mathrm{vec}(L+D)]^T \mathrm{vec}(B)$$

in matrix form. Operator vec$(\cdot)$ is a vectorization operation. Superscript T denotes a transpose operation. Formulation to minimize the batch sizes for minimum cost is given by

$$\min_{b_{ij}}\ [\mathrm{vec}(L+D)]^T \mathrm{vec}(B) \tag{6}$$

$$\text{s.t.} \sum_{j=1}^{l} b_{ij} \geq m_i\beta, \quad i = 1,\ldots,N \tag{7}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{l} b_{ij} = \beta \tag{8}$$

$$\sum_{i=1}^{N} b_{ij} = \bar{d}_j, \quad j = 1,\ldots,l \tag{9}$$

where $\bar{d}_j$ denotes the required batch size of the destination j. Equation (7) is the minimum utilization rate constraint, (8) ensures that the total of the batches add up to β, and (9) ensures that the total of the delivered batch size to a destination j is the required amount $\bar{d}_j$ at the destination. Similarly, a formulation for computing $f_t(\bar{b})$ is given by

$$\min_{b_{ij}}\ f_t(\bar{b}) = \mathrm{argmax}\{b_{ij}[t_i + \zeta_t \mathcal{L}(i,j)]\}$$
$$\text{s.t. (7)–(9)} \tag{10}$$

which computes the optimal batch sizes b_i for the minimum process time, subject to the previously introduced constraints. Note that the proposed optimization models optimize the batch sizes to be manufactured by each manufacturer and the size of the batch to be sent to the consecutive manufacturers. Transportation and manufacturing times and costs in split batches are assumed to be linearly related to the original transportation time and cost (i.e., 70% of the batch is 70% of the cost). These changes could be incorporated in other ways, such as nonlinear relationships, rule-based, and others.

C. Constrained Genetic Algorithm

The optimization problem given by (3) requires computation of numerous candidate quotes to find an optimal solution. Furthermore, a feasible candidate solution S is a list of quotes, and since process precedence constraints do not pose a strict sequence of processes on the solution, the optimization needs to compute the optimal sequence as well, which creates a combinatorial problem.

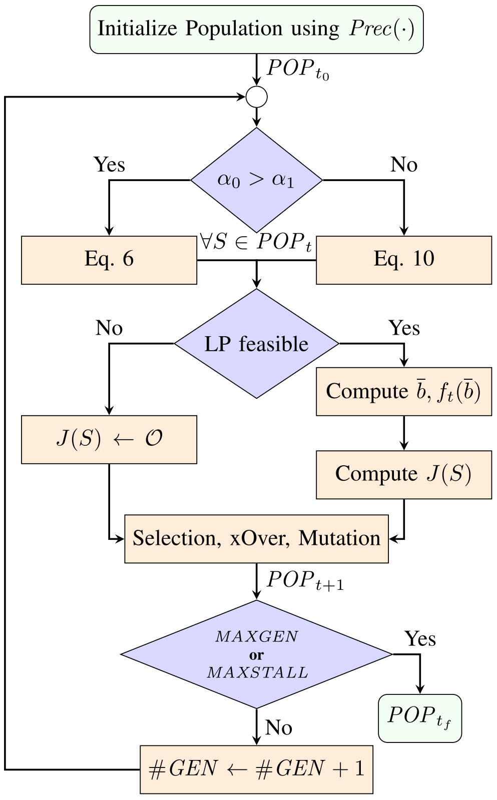

Initialize Population using $Prec(\cdot)$
POP_{t_0}
$\alpha_0 > \alpha_1$
Yes
No
Eq. 6
$\forall S \in POP_t$
Eq. 10
LP feasible
No
Yes
Compute $\bar{b}, f_t(\bar{b})$
$J(S) \leftarrow \mathcal{O}$
Compute $J(S)$
Selection, xOver, Mutation
POP_{t+1}
MAXGEN or MAXSTALL
Yes
POP_{t_f}
No
$\#GEN \leftarrow \#GEN + 1$

Fig. 3. GA flowchart for the MMPP type solution with batch splitting. $\mathcal{O}$ is a very large number, assigned to infeasible LPs.

Evolutionary algorithms (EA) are widely utilized in manufacturing studies [30]. GA, which is a type of EA, is well suited for problems where the solution space is large and brute-force techniques for optimization are not feasible [31]. Scheduling, resource allocation, and resource-constrained project scheduling problems have been solved with GA before [32]–[34].

The choice of GA in this paper is based on the proven success of evolutionary algorithms in manufacturing research. Different EA or swarm-based algorithms, such as particle swarm optimization and ant colony optimization, could also be incorporated in the decision maker instead of the GA. The batch splitting problem would remain the same; however, the precedence assignment procedure would have to be adapted to the chosen optimization method.

The proposed GA has a custom genotype (encoding of individuals) representation $G(S) = [s_1, s_2, \ldots, s_n]$, where $s_i \in S$ denotes the ith gene of the individual, which is equivalently the ith quote in the candidate solution S. Therefore, previously defined precedence operator $\mathcal{P}(S)$ and fitness function (3) $J(S)$ are also the operators for the individuals of the GA. The proposed GA is given in Algorithm 3. Following are some remarks on the proposed GA.

1) Function $Prec$ creates individuals of type SMPP. For the MMPP, additional genes s_a with $\vec{p}_a$, including the bottleneck process in S_i, are added to the individual. The condition in (2) determines when new genes are added.
2) α_0, α_1, and α_2 are the translations of the user input on the CTQ preference. This allows the evaluation of solutions tailored for user preferences of CTQ.

Algorithm 2 Precedence Constraints Assignment

```
1: function PREC($\mathcal{P}_{\mathbb{P}_\mathcal{C}}, \mathbb{D}$)
2:     Input: $\mathcal{P}_{\mathbb{P}_\mathcal{C}} = (\bar{I}_{\mathbb{P}_\mathcal{C}}, \bar{E}_{\mathbb{P}_\mathcal{C}}, \bar{M}_{\mathbb{P}_\mathcal{C}}), \mathbb{D}$
3:     Individual $G(S) = \{\}$    ▷ Initialize the individual
4:     $R = [0]^{1\times n}, n = |\mathbb{P}_\mathcal{C}|$    ▷ Keep track of the processes
5:     Pick random index $i^{\gamma_x} \in \bar{I}_{\mathbb{P}_\mathcal{C}}$
6:     Pick $s_p \in \mathbb{D}$, such that $\gamma_x \in \vec{p}_i$
7:     $G(S) \leftarrow G(S) \cup s_p$    ▷ Append the first gene
8:     $R(x) \leftarrow 1$    ▷ Mark used index
9:     while $R \neq [1]^{1\times n}$ do
10:        $K \leftarrow R$
11:        for all $(j,p,k) \in \bar{E}_{\mathbb{P}_\mathcal{C}} = \{\bar{e}^{\gamma_p}_{\gamma_j,\gamma_k}\}$ do
12:            if $R(j) = 1$ then
13:                $K(k) \leftarrow 1$    ▷ need $p, j$ first, lock $k$
14:            else if $R(j,p,k) = 0$ then
15:                $(K(p), K(k)) \leftarrow 1$    ▷ lock $p, k$
16:            end if
17:        end for
18:        for all $(a,b,c) \in \bar{M}_{\mathbb{P}_\mathcal{C}} = \{\bar{m}^{\gamma_b}_{\gamma_a,\gamma_c}\}$ do
19:            if $R(a) = 1$ then
20:                Choose $x$ as $b$, go to 28    ▷ Break
21:            else if $R(b) = 1$ then
22:                Choose $x$ as $c$, go to 28    ▷ Break
23:            else if $R(a,b,c) = 0$ then
24:                $(K(b), K(c)) \leftarrow 1$    ▷ lock $b, c$
25:            end if
26:        end for
27:        Pick random index $x$ from zeros of $K$
28:        Pick $s_p \in \mathbb{D}$ such that $x \in \vec{p}_i$
29:        $G(S) \leftarrow G(S) \cup s_p$    ▷ Append the next gene
30:        $R(x) \leftarrow 1$    ▷ Mark used index
31:    end while
32:    Output: $G(S)$
33: end function
```

3) A stochastic uniform selection function is used in the GA to select the parents. A two-point crossover and single-gene mutation are utilized. The genetic operators are induced on parents until enough feasible children, and mutated solutions are produced for the next generation.

The design variables for the GA, mutation rate, crossover rate, elite count, and the selection function are chosen at the design of the algorithm based on guidelines from the literature. The best individual [least fitness score $J(S^*)$] of the final generation is the optimal solution S^* for the optimization problem given in (3). Subsequently, $\mathcal{P}(S^*)$ is the optimal precedence scheme, functions $C(S^*), T(S^*), and Q(S^*)$ are the associated CTQ, and the batch sizes b_i are the optimal batch sizes if $S^* \in$ MMPP. Fig. 3 shows the flowchart of the GA for MMPP type solutions.

The GA decision maker with Algorithms 2 and 3 is implemented in the back end of the framework. The worst-case complexity of the proposed algorithm for the MMPP case can be approximated by the calls to the Simplex algorithm that solves the LP. Although Simplex has $O(2^n)$ (exponential time) worst-case complexity, the practical complexity is known to be much better than exponential and polynomial in some cases [35]. Experimentation with the implementation also concluded that the complexity of the optimization is approximately the complexity of the LP calls.

Algorithm 3 Proposed GA

```
1: Input: $\mathcal{P}_{\mathbb{P}_\mathcal{C}}, \mathbb{D}, \alpha_0, \alpha_1, \alpha_2, \mathcal{L}$
2: $GEN\# = 1$    ▷ Initialize population of size $POPSIZE$
3: while $i \leq POPSIZE$ do
4:     $G(S_i) = Prec(\mathcal{P}_{\mathbb{P}_\mathcal{C}}, \mathbb{D})$
5:     $POP_{t_0} \leftarrow POP_{t_0} \cup G(S_i)$    ▷ Append genotype
6:     $J(S_i)$    ▷ Assign fitness (Eq.3)
7:     $i \leftarrow i + 1$
8: end while
9: while $GEN\# \leq$ ($MAXGEN$ or $MAXSTALL$) do
10:    Select parents $Pa(S)$ for the next generation
11:    $EL \leftarrow Pa(S)^{\text{elite}}$    ▷ Preserve best elite individuals
12:    while $size(CH) \leq CHSIZE$ do
13:        $S_c^{\text{xOver}} \leftarrow S_j \times S_k$    ▷ Crossover parents in $Pa(S)$
14:        if $\mathcal{P}(S_c^{\text{xOver}}) \subset \mathcal{P}_{\mathbb{P}_\mathcal{C}}$ then
15:            $CH \leftarrow CH \cup S_c^{\text{xOver}}$    ▷ If the child is feasible
16:        end if
17:    end while
18:    $MU \leftarrow S_i^{\text{mut}}$    ▷ Induce mutation
19:    $POP_{t+1} \leftarrow EL \bigcup CH \bigcup MU$
20:    $J(S_i) \ : \forall S_i \in POP_{t+1}$
21:    $GEN\# \leftarrow GEN\# + 1$    ▷ Next generation
22: end while
23: Output: Last generation $POP_{t_f}$
```

VI. Framework APIs

The integration of the back-end, front-end, and database is shown in Fig. 4. The given information flow illustrates how the needs of the user are interpreted and translated into a quantitative abstracted format for use in the PaaS framework.

A. Production Quote API

Creates the EPF data format and MFS using data from the feature extraction module and RFQ. Production quote application programming interface (PQAPI) manages most of the data communication with the database module, manufacturer interface, and the Analysis API. PQAPI uses $\mathbb{P}_\mathcal{C}$ to create a list of manufacturing methods that are proposed for the manufacturing of the requested product. Manufacturers with matching capabilities are then queried from the database, and the MFSs are forwarded to those manufacturers. The user's initial preference on CTQ is also shared with the manufacturers.

B. Analysis API

The Analysis API uses the RQ, QS, and EPF data to perform the quality index (q) assignment in Section IV. Following this, each quote in the RQ format is abstracted into the quote format s introduced in the problem formulation (Section V-A) to create the set of optimization quotes $\mathbb{D}$ for the decision maker.

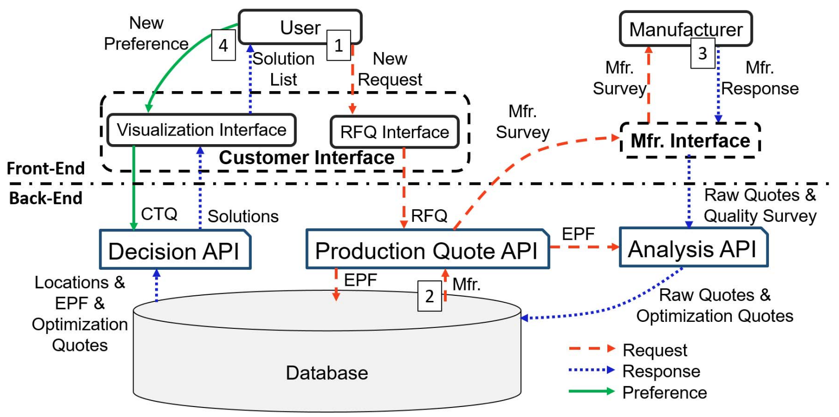


Fig. 4. 1—user submits a new production request to the framework through the customer interface (RFQ interface). 2—manufacturers with adequate capabilities are notified with the MFS. 3—manufacturer responds to the MFS through the mfr interface. 4—user changes the preference on the weight of CTQ of the evaluated solution until deciding on using a specific solution.

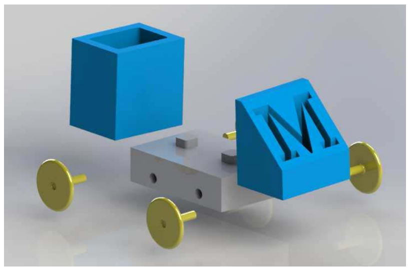


Fig. 5. Exploded view of the model truck design used in the case study.

C. Decision API

Using the distance matrix $\mathcal{L}$, set of quotes $\mathbb{D}$, process precedence constraints $\mathcal{P}_{\mathbb{P}_\mathcal{C}}$ captured by the EPF data structure, and user preferences on CTQ, the Decision API runs the proposed GA to obtain optimal solutions from the feasible solution set. A list of 10 solutions for every combination of $\{(\alpha_0, \alpha_1, \alpha_2) | \alpha_0 + \alpha_1 + \alpha_2 = 1; \alpha_0, \alpha_1, \alpha_2 \in \ 0.1i,\ i \in \mathbb{Z}_+\}$ is precomputed by the Decision API so that the visualization interface can call different sets of solutions for various CTQ.

VII. Case Study

This section presents a simulated case study to investigate the implementation of the framework. The user design is given as a model truck (see Fig. 5) containing seven parts and four different components; four wheels, a cabin, a trunk, and a chassis. The required processes to manufacture the truck are as follows [15]; 1) machining (face milling and drilling) of the chassis; 2) 3-D printing of the trunk; 3) 3-D printing of the cabin; 4) 3-D printing of the four wheels; and 5) assembly of the components.

A. Setup

The quotes in this case study are collected through online channels[4] and the 3-D Laboratory at the University of Michigan. Quotes are requested for the production of a single truck ($\beta = 1$) in order to eliminate the effect of nonstandardized discounts applied by some of the online channels for increasing batch sizes. A total of 37 quotes were collected for the 3-D printed parts, and three quotes were collected from machining suppliers. Additional quotes were generated using a Gaussian distribution of cost and time around standard process guidelines for machining. A total of $|\mathbb{D}| = 129$ quotes from 110 manufacturers, including the actual manufacturers, are used in the simulated case study with a batch size of $\beta = 100$. Layer heights of 3-D printed parts are used to estimate the quality of the associated quotes (less height denotes higher quality). The average quality is used in the combined solution since layer height does not denote yield. Since the model truck has a trivial assembly process, all the manufacturers are assumed to be capable of the required assembly. A partial ECF for the chassis is given in Table I.

TABLE I
PARTIAL ECF FOR CHASSIS. ALL DIMENSIONS ARE IN mm

ECF_{c^1}	RM_{c^1}	Material				Aluminum
		Nominal Volume				76.2×50.8×25.0
	PS_{c^1}	Process M_1	Manufacturing Method y			Drilling
			Process Dimensions d_1			$\Phi 6.0 \times 50.8$
			Tolerance Specifications TS_{M_1}	$\mathcal{T}_1$	*ID*	1
					$\mathcal{T}_{type}$	Fitting
					$\mathcal{T}_{dim}$	6.0
					$\mathcal{T}_{UD}$	+0.2
					$\mathcal{T}_{LD}$	-0.2
	$\mathcal{P}_{PS_{c^1}}$	$\bar{I}_{c^1}$				i^{p_1}
		$\bar{E}_{c^1}$				-
		$\bar{M}_{c^1}$				-

The set of components is defined as $\mathcal{C} = \{c^1, c^2, c^3, c^4\}$, and the set of requested processes $\mathbb{P}_\mathcal{C} = \{p_1, p_2, p_3, p_4\}$ denotes the machining of the chassis (p_1) and 3-D printing of the remaining components (p_2, p_3, p_4). We use the relaxed precedence scheme given in Algorithm 2 with $\mathcal{P}^{\text{rlx}}_{\mathbb{P}_\mathcal{C}} = (\{i^{p_1}, i^{p_2}\}, \{\bar{e}^{p_3}_{p_2,p_4}\}, \{\})$. The use of soft constraints enables

[4]3Dhubs.com, shapeways.com, makexyz.com, MFG.com, and xometry.com

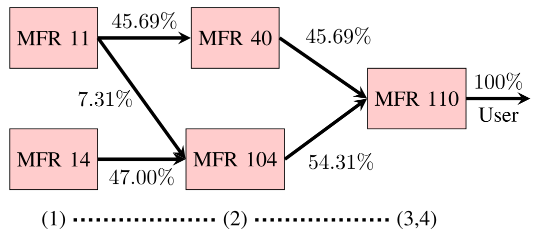


Fig. 6. The hybrid (parallel/serial) manufacturing system layout for the MMPP solution. Numbers on the bottom are the manufacturing processes conducted by each manufacturer. Percentages on the arrows denote b_{ij} for each manufacturer, i.e., $b_{(11,40)} = 45.69\%$, $b_{(11,104)} = 7.31\%$.

the combination of different precedence schemes, which in turn allows the optimization to evaluate a wider variety of solutions compared to the strict ordered precedence [15].

To illustrate the proposed functionalities of PaaS, we present the following three example solutions in this case study.

S₁: SMPP solution with random quotes that the user might choose without using the PaaS platform.

S₂: SMPP computed by the decision maker for balanced CTQ $(\alpha_0, \alpha_1, \alpha_2) = (0.32, 0.34, 0.33)$.

S₃: MMPP computed by the decision maker for minimum time $(\alpha_0, \alpha_1, \alpha_2) = (0, 1, 0)$.

Constants used for the case study are $\beta = 100$, $\zeta_\kappa = 0.75$ ($/km), and $\zeta_t = 0.0029$ (day/km), and $m = 0.4$ is arbitrarily chosen for all the quotes.

B. Results and Discussion

The random solution we created by picking reasonable (not the worst possible solution, balanced importance on CTQ) quotes for a feasible solution yields $J_C(S_1) = \$190.30$, $J_T(S_1) = 11.72$ days, and $J_Q(S_1) = 45.27\%$. S_1 denotes a baseline solution for our case study. S_2 is a balanced CTQ solution to present a comparable result to the random solution S_1. Following is $S_2 = [s_1, s_2, s_3]$.

$$s_1 : [(2,3), 28.26, 6.0, 0.4, 52.5, 104, 124]$$
$$s_2 : [(1), 32.42, 3.62, 0.4, 64.3, 14, 19]$$
$$s_3 : [(4), 25.00, 3.00, 0.4, 95.0, 110, 122]$$

and this solution yields $J_C(S_2) = \$114.29$, $J_T(S_2) = 9.73$ days, and $J_Q(S_1) = 70.6\%$. Even without inducing the batch splitting, we have a 25.2% improvement in the solution using the GA. Note that this rate of improvement is a rough lower bound for the improvement since we selected reasonable quotes from a large list of quotes, which is not the case for a user reaching a few manufacturers and collecting quotes. Moreover, by experimenting with the GA, we computed $T_{\text{SMPP}} = 6.68$ days.

Next, we show the effect of batch splitting with solution S_3. Fig. 6 illustrates the output of the solution in terms of batch splitting. Note that the time span of S_3 is less than T_{SMPP} since it is MMPP type. Fig. 7 shows a generation plot of the proposed GA for the MMPP type solutions. We realize a 65% best case improvement for the optimal solutions of GA over the initial random population (generation 0).

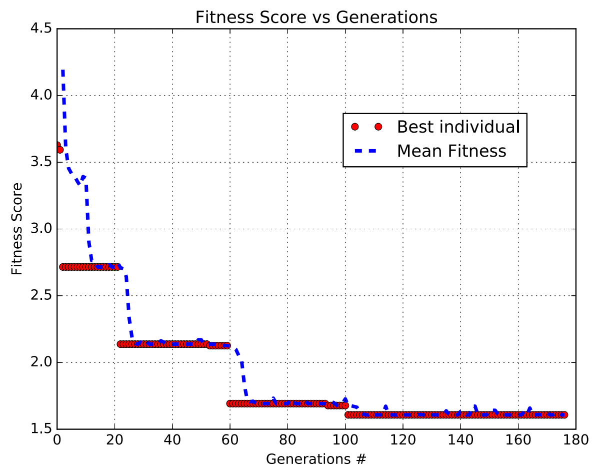


Fig. 7. Individual scores versus GA generations. Red solid dots: the scores of the best individuals. Blue dashed lines: the mean fitness scores of the generations. Number of quotes in the simulation is $|\mathbb{D}| = 129$. Constants used in the optimization: $\zeta_\kappa = 0.75$ ($/km) and $\zeta_t = 0.0029$ (day/km).

Two case study trucks were physically produced using distributed manufacturers for evaluation with the case study. Several important issues identified from these sample products include nonstandardized explanation of manufacturing capabilities, interfacing issues with the CAD data, and the lack of comparable data formats between different quotes.

VIII. Conclusion

In this paper, we presented a PaaS framework and introduced the key structures and functionalities for its realization. During the course of this research, we collaborated with an industrial partner to gain insight about the industrial point of view on small batch manufacturing. During these interactions, we identified a need for an automated framework that could abstract product data into a secure format for distribution to manufacturers as well as a structured method for matching manufacturers with users to optimize specified user requirements. To address this need, this paper introduces a PaaS framework.

The PaaS framework enables effective utilization of manufacturing resources for rapid realization of custom manufacturing needs of product developers. This paper can handle precedence and batch splitting constraints to efficiently evaluate optimal manufacturing solutions. Introducing CAPP capabilities for the feature extraction module to automatically assign precedence relationships and extract the most significant manufacturing features of a design is an important future direction. The proposed quality assignment model is a general approach that has limited flexibility for certain manufacturing methods (e.g., AM); future work will incorporate customized quality definitions. Future research will consider supply–demand dynamics in the service matching algorithm to incorporate real-time updates for enhanced user–manufacturer matching.

Acknowledgment

The authors would like to thank their industrial partners for their valuable input in supplying industrial data for this research.

References

[1] A. P. Kalogeras, J. V. Gialelis, C. E. Alexakos, M. J. Georgoudakis, and S. A. Koubias, "Vertical integration of enterprise industrial systems utilizing Web services," *IEEE Trans. Ind. Informat.*, vol. 2, no. 2, pp. 120–128, May 2006.

[2] P. Helo, M. Suorsa, Y. Hao, and P. Anussornnitisarn, "Toward a cloud-based manufacturing execution system for distributed manufacturing," *Comput. Ind.*, vol. 65, no. 4, pp. 646–656, 2014.

[3] L.-J. Zhang, J. Zhang, and H. Cai, "Service-oriented architecture," in *Services Computing*. Berlin, Germany: Springer, 2007, pp. 90–113.

[4] M. P. Papazoglou and W.-J. van den Heuvel, "Service oriented architectures: Approaches, technologies and research issues," *VLDB J.*, vol. 16, no. 3, pp. 389–415, 2007.

[5] T. Cucinotta *et al.*, "A real-time service-oriented architecture for industrial automation," *IEEE Trans. Ind. Informat.*, vol. 5, no. 3, pp. 267–277, Aug. 2009.

[6] G. Cândido, A. W. Colombo, J. Barata, and F. Jammes, "Service-oriented infrastructure to support the deployment of evolvable production systems," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 759–767, Nov. 2011.

[7] A. García-Domínguez, M. Marcos-Bárcena, I. Medina-Bulo, and L. Prades-Martell, "Towards an integrated SOA-based architecture for interoperable and responsive manufacturing systems," *Procedia Eng.*, vol. 63, pp. 123–132, 2013.

[8] T. Borangiu *et al.*, "A service oriented architecture for total manufacturing enterprise integration," in *Proc. Int. Conf. Exploring Services Sci.*, Cham, Switzerland: Springer, 2015.

[9] L. van Moergestel, E. Puik, D. Telgen, and J.-J. C. Meyer, "Implementing manufacturing as a service: A pull-driven agent-based manufacturing grid," in *Proc. 11th Int. Conf. ICT Educ., Res. Ind. Appl., Integr., Harmonization Knowl. Transf.*, vol. 1356, Lviv, Ukraine, May 2015, pp. 172–187.

[10] M. Dotoli, M. P. Fanti, C. Meloni, and M. Zhou, "Design and optimization of integrated E-supply chain for agile and environmentally conscious manufacturing," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 36, no. 1, pp. 62–75, Jan. 2006.

[11] Y. Cheng, F. Tao, D. Zhao, and L. Zhang, "Modeling of manufacturing service supply–demand matching hypernetwork in service-oriented manufacturing systems," *Robot. Comput.-Integr. Manuf.*, vol. 45, pp. 59–72, Jun. 2017.

[12] J. L. M. Lastra and I. M. Delamer, "Semantic Web services in factory automation: Fundamental insights and research roadmap," *IEEE Trans. Ind. Informat.*, vol. 2, no. 1, pp. 1–11, Feb. 2006.

[13] C. Legat, C. Seitz, S. Lamparter, and S. Feldmann, "Semantics to the shop floor: Towards ontology modularization and reuse in the automation domain," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3444–3449, 2014.

[14] F. Tao, L. Zhang, V. C. Venkatesh, Y. Luo, and Y. Cheng, "Cloud manufacturing: A computing and service-oriented manufacturing model," *Proc. Inst. Mech. Eng., B, J. Eng. Manuf.*, vol. 225, no. 10, pp. 1969–1976, 2011.

[15] E. C. Balta, K. Jain, Y. Lin, D. Tilbury, K. Barton, and Z. M. Mao, "Production as a service: A centralized framework for small batch manufacturing," in *Proc. 13th IEEE Conf. Autom. Sci. Eng. (CASE)*, Aug. 2017, pp. 382–389.

[16] M. Porter, V. Raghavan, Y. Lin, Z. M. Mao, K. Barton, and D. Tilbury, "Production as a service: Optimizing utilization in manufacturing systems," in *Proc. ASME Dyn. Syst. Control Conf.*, 2016, p. V002T21A012.

[17] Barton Research Group, *Production as a Service*. Accessed: May 24, 2018. [Online]. Available: https://brg.engin.umich.edu/research/smart-manufacturing/paas/

[18] F. Tao, J. Cheng, Y. Cheng, S. Gu, T. Zheng, and H. Yang, "SDMSim: A manufacturing service supply–demand matching simulator under cloud environment," *Robot. Comput.-Integr. Manuf.*, vol. 45, pp. 34–46, Jun. 2017.

[19] L. Zehtaban and D. Roller, "Automated rule-based system for opitz feature recognition and code generation from STEP," *Comput.-Aided Des. Appl.*, vol. 13, no. 3, pp. 309–319, 2016.

[20] T. AlGeddawy and H. ElMaraghy, "Manufacturing systems synthesis using knowledge discovery," *CIRP Ann.*, vol. 60, no. 1, pp. 437–440, 2011.

[21] F. Ameri and D. Dutta, "A matchmaking methodology for supply chain deployment in distributed manufacturing environments," *J. Comput. Inf. Sci. Eng.*, vol. 8, no. 1, p. 011002, 2008.

[22] N. W. Bowland, J. X. Gao, and R. Sharma, "A PDM- and CAD-integrated assembly modelling environment for manufacturing planning," *J. Mater. Process. Technol.*, vol. 138, nos. 1–3, pp. 82–88, 2003.

[23] Y. Oh, S.-H. Han, and H. Suh, "Mapping product structures between CAD and PDM systems using UML," *Comput.-Aided Des.*, vol. 33, no. 7, pp. 521–529, 2001.

[24] V. Allada and S. Anand, "Feature-based modelling approaches for integrated manufacturing: State-of-the-art survey and future research directions," *Int. J. Comput. Integr. Manuf.*, vol. 8, no. 6, pp. 411–440, 1995.

[25] X. Zhang, A. Nassehi, and S. T. Newman, "Feature recognition from CNC part programs for milling operations," *Int. J. Adv. Manuf. Technol.*, vol. 70, nos. 1–4, pp. 397–412, 2014.

[26] X. Zhang, B. Afsharizand, W. Essink, S. T. Newman, and A. Nassehi, "A STEP-compliant method for manufacturing knowledge capture," *Procedia CIRP*, vol. 20, pp. 103–108, 2014.

[27] W. Xiao, L. Zheng, J. Huan, and P. Lei, "A complete CAD/CAM/CNC solution for STEP-compliant manufacturing," *Robot. Comput.-Integr. Manuf.*, vol. 31, pp. 1–10, Feb. 2015.

[28] A. B. Feeney, S. P. Frechette, and V. Srinivasan, "A portrait of an ISO STEP tolerancing standard as an enabler of smart manufacturing systems," *J. Comput. Inf. Sci. Eng.*, vol. 15, no. 2, p. 021001, 2015.

[29] T. Pyzdek and P. A. Keller, *The Six Sigma Handbook*. New York, NY, USA: McGraw-Hill, 2014.

[30] C. Dimopoulos and A. M. S. Zalzala, "Recent developments in evolutionary computation for manufacturing optimization: Problems, solutions, and comparisons," *IEEE Trans. Evol. Comput.*, vol. 4, no. 2, pp. 93–113, Jul. 2000.

[31] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM J. Comput.*, vol. 2, no. 2, pp. 88–105, 1973.

[32] A. R. Contreras, C. V. Valero, and J. M. A. Pinninghoff, "Applying genetic algorithms for production scheduling and resource allocation. special case: A small size manufacturing company," in *Proc. Int. Conf. Ind., Eng. Other Appl. Appl. Intell. Syst.* Berlin, Germany: Springer-Verlag, 2005, pp. 547–550.

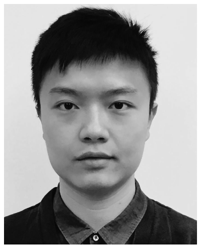
[33] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 566–579, Dec. 2002.

[34] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling," *Naval Res. Logistics*, vol. 45, no. 7, pp. 733–750, 1998.

[35] D. Goldfarb, "On the complexity of the simplex method," in *Advances in Optimization and Numerical Analysis*. Dordrecht, The Netherlands: Springer, 1994, pp. 25–38.

Efe C. Balta (S'17) received the B.S. degree in manufacturing engineering from the Mechanical Engineering Faculty of Istanbul Technical University, Istanbul, Turkey, in 2016, and the M.S. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2018, where he is currently pursuing the Ph.D. degree in mechanical engineering.

His current research interests include manufacturing science, automation, optimization, controls, and systems engineering.



Yikai Lin received the B.Eng. degree in communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA.

He has been actively studying the application of software-defined networking to several areas, including cellular networks and manufacturing. His current research interests include networking and computer systems, especially in software-defined networking and network function virtualization.

Kira Barton (M'11) received the B.Sc. degree in mechanical engineering from the University of Colorado at Boulder, Boulder, CO, USA, and the M.Sc. and Ph.D. degrees from the University of Illinois at Urbana–Champaign, Champaign, IL, USA.

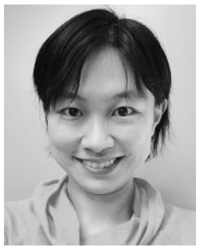
In 2011, she joined the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA, where she is currently an Associate Professor and a Miller Faculty Scholar. Her current research interests include modeling, sensing, and control for applications in advanced manufacturing and robotics, with a specialization in iterative learning control and microadditive manufacturing.

Dr. Barton received the NSF CAREER Award, the 2015 SME Outstanding Young Manufacturing Engineer Award, the Outstanding Young Alumni Award from the Department of Mechanical Science and Engineering, University of Illinois, in 2015, the Achievement Award from the Department of Mechanical Engineering Department, University of Michigan, in 2016, and the 2017 ASME Dynamic Systems and Control Young Investigator Award.

Dawn M. Tilbury (F'08) received the B.S. degree *(summa cum laude)* in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 1989, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 1992 and 1994, respectively.

In 1995, she joined the faculty of the University of Michigan, Ann Arbor, MI, USA, where she is currently a Professor of Mechanical Engineering with a joint appointment in Electrical Engineering and Computer Science. She has published over 150 articles in refereed journals and conference proceedings. Her current research interests include the area of control systems, including applications to robotics and manufacturing systems.

Prof. Tilbury was elected as a Fellow of the American Society of Mechanical Engineers in 2012 and a Life Member of the Society of Women Engineers.



Z. Morley Mao (M'07) received the B.S., M.S., and Ph.D. degrees from the University of California at Berkeley, Berkeley, CA, USA.

She is currently a Professor with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA.

Dr. Mao was a recipient of the NSF CAREER Award, the Sloan Fellowship, and the IBM Faculty Partnership Award. She has been named the Morris Wellman Faculty Development Professor.