2017 13th IEEE Conference on Automation Science and Engineering (CASE)
Xi'an, China, August 20-23, 2017

Production as a Service: A Centralized Framework for Small Batch Manufacturing

Efe C. Balta, Kshitij Jain, Yikai Lin, Dawn Tilbury, Kira Barton, Z. Morley Mao

***Abstract*— We present a *Production as a Service* (PaaS) framework that connects users (product developers) who have small batch production needs, with existing manufacturing facilities that have underutilized resources. PaaS is a web-based framework based on the service oriented architecture (SOA) design that breaks down the manufacturing of a product into several steps, or *services*, to incorporate various manufacturers with available capability in fulfilling the production request. Preservation of intellectual property (IP) is a major concern with SOA based manufacturing systems. Presented work aims to address this issue by using abstracted features to describe a product design, which enables manufacturers to provide quotations for the manufacturing steps without the need for the detailed technical data of the product. The paper also presents the optimization approaches employed to find feasible solutions based on the collected quotations for a production request by relaxing the precedence constraints on the manufacturing processes or the components. A case study is provided to illustrate the functionalities of the proposed framework. A detailed description of the PaaS architecture and the required APIs for its implementation are also provided.**

I. Introduction

Product customization is an increasing demand in most industries. Manufacturing paradigms are shifting towards mass customization and product personalization [1]. In order to cater to the ever-rising demand of customization, the manufacturing industry has witnessed a marked shift towards digital manufacturing paradigms in recent years. By combining advances in information technology (IT) with manufacturing, Service Oriented Architecture (SOA) based systems have been developed to address this specific need [2], [3]. Moghaddam et al. reviews and analyzes the emerging association between collaborative e-manufacturing and SOA towards the conception of new manufacturing paradigms [4]. Wu et al. propose a cloud based design and manufacturing model in which customers are enabled to configure, select and utilize customized product realization resources like Computer Aided Engineering (CAE) services with reconfigurable manufacturing systems [5]. Currently, most realizations of SOA based frameworks are limited by the fact that they offer only additive manufacturing (AM) services to their users. Developers with prototyping or small batch production needs often face the problem of finding the right manufacturers with adequate processing capabilities and production volumes. On the other hand, manufacturers with small facilities often have underutilized resources due to the inconsistency in the production orders. A *Production as a Service* (PaaS) framework is proposed in this paper to address this gap and facilitate the production of new products, while increasing the utilization of existing manufacturers.

An SOA based distributed manufacturing platform that leverages underutilized manufacturing facilities was proposed as a solution to the issue of facilitating multiple suppliers in production [6], but the challenges remain [7], [8]. Unlike conventional manufacturing systems, all the processed products do not arrive at one location for further inspection or assembly, which makes it difficult to schedule operations and trace back errors that might have occurred in earlier stages of production. Building a web based framework that would enable customers to connect and communicate with potential manufacturers without divulging sensitive information is another associated challenge. To analyze this problem, Helo et al. presents the needs and challenges for management of distributed manufacturing systems in a multi-company supply chain from an IT viewpoint [9].

This work builds upon the foundations of the PaaS framework [10], and extends its functionalities including the optimization algorithm. The issue of preserving intellectual property is also addressed. Three main contributions of this work are: (1) the design of the APIs; how they are used to encapsulate the abstracted information, and a detailed description of their functionalities, (2) the use of abstracted features to encode technical product details for IP preservation, and (3) the use of precedence relaxation as an updated version of the previously presented optimization [10].

The rest of the paper is structured as follows. Section II introduces the PaaS framework, explains key definitions and the architecture of the framework. Section III describes the use of feature abstraction and the means to do it in the context of the PaaS framework. In Section III, the design of the APIs, and their use for the framework functionalities are pointed out. Section IV introduces the updated optimization algorithm and the precedence constraints implemented in the optimization process, and Section V presents a case study that exemplifies the stages and functionalities of the PaaS framework. Section VI concludes the paper with a discussion of its contributions, and identifies future work.

II. PaaS Framework

PaaS aims to increase the utilization of the existing geo-distributed manufacturing facilities by connecting them with the product developers that have small batch custom

Efe C. Balta, Kshitij Jain, Kira Barton and Dawn Tilbury are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA {baltaefe,kshjain,bartonkl,tilbury}@umich.edu

Yikai Lin and Z. Morley Mao are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA {yklin,zmao}@umich.edu

978-1-5090-6781-7/17/$31.00 ©2017 IEEE

manufacturing needs. The user is asked to input predefined manufacturing information regarding the production request. The request is sent to the manufacturers with adequate capabilities, and queries quotes for the production. Through the framework, the user has the opportunity to reach multiple suppliers and get production quotes instead of contacting individual manufacturers one by one. Manufacturers are allowed to submit quotes according to their capacity and capability. Using PaaS, manufacturers have the opportunity to reach a community of developers, and choose the right manufacturing jobs to increase the utilization of their resources. As the quotes are collected from manufacturers, the combinations of quotes that describe feasible production schemes for the requested product are presented to the user. After the user decides on using one of the many possible schemes presented, respective manufacturers are notified to commence the production.

A. Definitions

- Component: A discrete manufactured good that is a result of consecutive manufacturing processes applied to a single initial raw material. Each component is denoted by $c^i \in \mathcal{C}$ where $\mathcal{C}$ is the set of components of a product.
- Product: The final produced good that will be sent to the user is the product in the context of this work. It represents the total of the manufactured goods rather than a single component.
- Process: It is a single manufacturing step that is applied to a component during the course of manufacturing. For instance, rough face milling using a single tool and fine machining the same surface using grinding are defined as two different processes in the context of this work. Assembly operations are also classified as processes.
- Feature: The encoded representation of information essential for the manufacturing of the requested product. A feature is an abstraction used to preserve IP and encode an engineering design in a predefined structure.
- Precedence Constraints: Precedence constraints impose restrictions on the order in which a component can undergo manufacturing processes, or a set of components can undergo assembly processes. We denote a set of precedence constraints by $\mathcal{P}_A = (\bar{I}_A, \bar{E}_A, \bar{M}_A)$, where $A = \{\alpha_1, \alpha_2, \ldots, \alpha_z\}$ is the set of processes. Sets $\bar{I}_A, \bar{E}_A$ and $\bar{M}_A$ are defined as follows:
  - $\bar{I}_A$ refers to the set of initial constraints, and contains the list of permissible initial processes.
  - $\bar{E}_A$ refers to the set of loose constraints. Each process in $\bar{E}_A$ is defined as $\bar{e}^{\alpha_n}_{\alpha_j,\alpha_k}$ which implies that process α_n has to occur after α_j, and before α_k, in set S.
  - $\bar{M}_A$ refers to the set of immediate constraints. Each process in $\bar{M}_A$ is defined as $\bar{m}^{\alpha_n}_{\alpha_j,\alpha_k}$ which implies that process α_n has to occur immediately after α_j, and immediately before α_k, in set S.
- Solution: The set of quotes that represent a feasible production scheme is referred to as a solution. The set of quotes in a solution make up an ordered set of processes that is required for the production of the product.

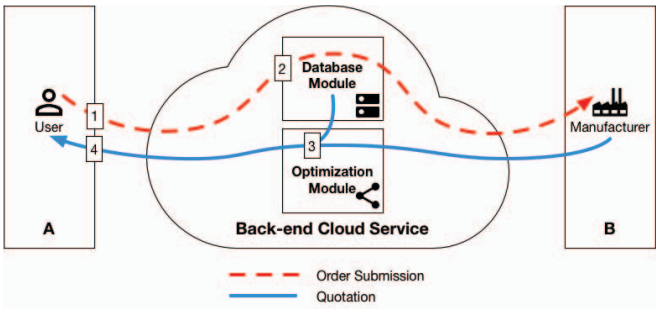

Fig. 1. Framework Structure and Key Steps

B. Basic Architecture

As shown in Fig. 1, the proposed framework consists of three main components: the user interface (A), the manufacturer interface (B), and the back-end cloud service. The user interface collects necessary information for the query from the users, and presents the user with feasible solutions computed by the cloud based on their preference; the manufacturer interface presents the abstracted product information to the manufacturers, and collects production quotes back from the manufacturers.

The cloud service works like a coordinator between users and manufacturers. Its two main functions are database and optimization. **Database module:** Both manufacturer and user profiles are stored in the database. A manufacturer profile contains manufacturing capabilities, factory locations and past quotations; a user profile contains location and past requests. **Optimization module:** The optimization module takes in manufacturer production quotes and computes production solutions based on user preferences.

When a user submits a request via the customer interface, the information will be sent to the cloud service program. The program then performs a query in the manufacturer profile database to find those with matching capabilities. These manufacturers will be notified of the request, and have the option to submit production quotes based on their availability. Fig. 1 shows the flow of information in the cloud. Some key steps shown in the figure are:

1) A user makes a new request via the web interface. Information is encoded in a standard format and sent to the cloud. This initial stage is defined as the *request for quotation* (RFQ). Submission of a detailed technical drawing of the requested product is not required at this stage.
2) The cloud service program queries the database and matches the request to capable manufacturers using the Production Quote API. The encoded user information is used to prepare manufacturer surveys to be shared with the matched manufacturers. As manufacturers reply to these surveys, raw production quotes are fed to the Analysis API. Quotes are analyzed by the Analysis API, and then fed to the database.
3) The analyzed quotes are fed into the Decision API. Using this information, along with the location information stored in the database, a list of solutions is computed and presented to the user.

4) The user has the option of modifying the optimization parameters to evaluate solutions with different characteristics and explore various feasible solutions available. As the user reaches a final decision on which solution to use, the framework stores this request and notifies the respective manufacturers.

C. Database design

In order to efficiently store, query and update manufacturer information and the associated production quotes, the database needs to be carefully designed. A non-relational database [11] is used as the database module to allow for flexible updates, such as the addition of new manufacturing capabilities. Due to the flexible architecture of this database, manufacturer and user profiles can both be stored in the database and linked with unique indexes. Each manufacturer is represented as a JSON object with different fields such as list of capabilities, list of production quotes, and facility locations. The use of JSON encoding enables the web interfaces to communicate with the back-end service.

III. Feature Abstraction and APIs

A detailed representation of the framework is illustrated in Fig. 2. The front-end of the framework houses the RFQ interface, the Mfr. interface and the Visualization interface. The RFQ interface allows users to submit production requests to the framework, while the Mfr. interface is used by manufacturers to submit quotes. The Visualization interface is responsible for conveying optimization weights specified by the user to the Decision API, as well as for presenting feasible production schemes to users. These interfaces act as a conduit between users and manufacturers, via the back-end of the framework. The back-end of the framework includes the Production Quote API (PQAPI), the Analysis API, the Decision API, and the Database. While the database is used to store user and manufacturer profiles, the APIs are responsible for various tasks ranging from quote analysis to finding an optimized solution. A detailed discussion of API functions is presented in the following sections.

A. Production Quote API (PQAPI)

On receiving a production request from the RFQ interface, this API converts it into an abstracted set of features is called Encoded Product Features (EPF). EPF contains information pertaining to manufacturing of the requested product, which is sufficient for manufacturers to provide quotations. The structure of an EPF is depicted in Fig. 3. The structure is broken down as follows:

- An $EPF = \{ECF_{c^1}, , \ldots, ECF_{c^z}\}$ is comprised of a set of *Encoded Component Features (ECF)*, where each c^i refers to either a component or an assembly.
- $ECF_{c^i} = \{RM_{c^i}, PS_{c^i}, \mathcal{P}_{PS_{c^i}}\}$ is the ECF for component c^i, where RM_{c^i} denotes the raw material requirements (material name and volume), $PS_{c^i} = \{M_1, \ldots, M_n\}$ denotes the list of process specifications M_j for c^i, and $\mathcal{P}_{PS_{c^i}}$ denotes precedence constraints. For assembly ECFs, RM_{c^i} is not required.
- The 3-tuple $M_j = (m_j, d_j, TS^{c^i}_{M_j})$ is the process specification, where m_j denotes the manufacturing process and material (i.e. machining - aluminum, AM - ABS), d_j denotes the nominal dimensions associated with m_j, and $TS^{c^i}_{M_j}$ denotes the tolerance specifications set for M_j. Information abstracted in M_j facilitates the manufacturers to estimate the required resources for this process.
- The set $TS^{c^i}_{M_j} = \{\mathcal{T}_0, \ldots, \mathcal{T}_l\}$ is the list of specified tolerances for M_j. A tolerance is defined as $\mathcal{T}_k = \{(\mathcal{T}^{\text{ID}}, \omega^{\text{ID}}, \mathcal{T}_{type}, \mathcal{T}_{dim}, \mathcal{T}_{UD}, \mathcal{T}_{LD}) | \omega^{\text{ID}} \in \{1, 2, 3\}\}$, where T^{ID} denotes the assigned unique ID of the tolerance, ω^{ID} denotes the user defined weight of the tolerance[1], $\mathcal{T}_{type}$ denotes the tolerance type (i.e. concentricity, planarity), $\mathcal{T}_{dim}$ denotes the nominal dimensions associated with the tolerance, and $(\mathcal{T}_{UD}, \mathcal{T}_{LD})$ denote the acceptable upper and lower deviation from the nominal dimension.

After the EPF is prepared, the PQAPI identifies the manufacturers with at least one of the requested manufacturing processes $m_j \in \mathcal{Z}_{ECF}$. The EPF and the list of capable manufacturers are used in preparing *manufacturer surveys* (MFS) for each identified manufacturer.

- MFS is defined as $MFS = \{M_{m_1}, M_{m_2}, \ldots, M_{m_p}\}$, where each M_{m_j} denotes a match between a requested M_j, and a manufacturer's capability. As a manufacturer decides on quoting for $\tilde{M} \subset MFS$, a *quality survey* $QS_{\tilde{M}}$ is generated.
- $QS_{\tilde{M}}$ is shared with the manufacturers through the manufacturer interface. $TS^{c^i}_{M_j}$ for each respective process $M_{m_j} \in \tilde{M}$ is shared with the manufacturers through the $QS_{\tilde{M}}$ and the manufacturers are asked to specify if each of the tolerances $\mathcal{T}_k \in TS^{c^i}_{M_j}$ could be met (1) or not (0).

PQAPI is also tasked with storing the EPF in the user profile, which in turn is stored in the database.

B. Analysis API

This API is responsible for extracting necessary information from the EPF and the raw quotes RQ, and converting them into Optimization Quotes (OQ), a format primed to be used by the optimization algorithm in the Decision API.

- A manufacturer's response to a MFS and $QS_{\tilde{M}}$ is defined as a *raw quote*, denoted by $RQ = (\tilde{m}, C, T, RQS_{\tilde{M}}, MFR^{\text{ID}})$, where $\tilde{m}$ is the set of processes for the quote, C and T are the cost and time for the quote, $RQS_{\tilde{M}}$ denotes the response of the supplier to the respective $QS_{\tilde{M}}$, and MFR^{ID} is the unique ID of the manufacturer. RQ is stored in the database of the respective manufacturers' profiles.
- The set $RQS_{\tilde{M}} = \{(\mathcal{T}^{\text{ID}}, \tau) | \forall \mathcal{T}^{\text{ID}} \in TS^{c^i}_{M_j}; \tau \in \{0, 1\}\}$ is defined as a list of tuples describing whether each required tolerance measure $\mathcal{T}^{\text{ID}}$ could met (1) or not (0) by the respective manufacturer.
- Analysis API uses these RQs and the EPF to filter and generate the unique OQs defined as $OQ^{\text{ID}} =$

[1]We propose a three-tier weight assignment scheme: a weight of *1,2* or *3* is assigned to the tolerance, depending on its necessity to the component.

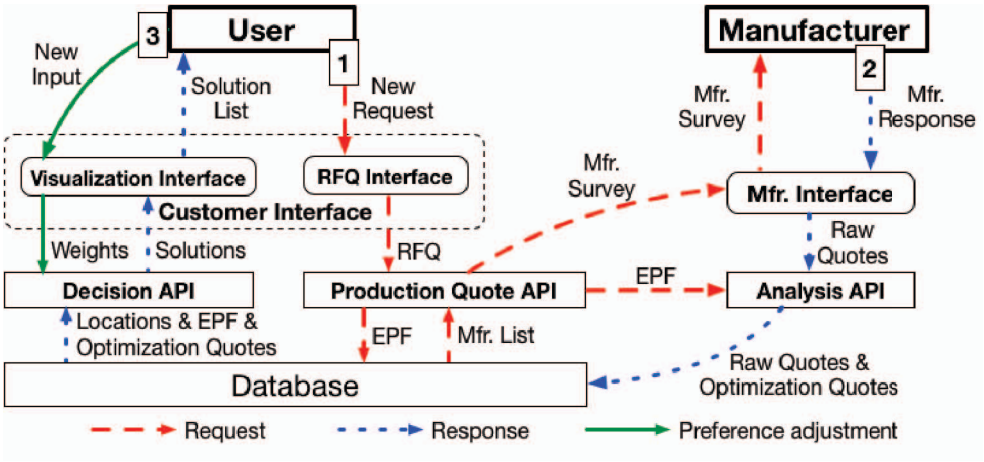

Fig. 2. APIs of the framework. (1) User submits a new production request to the framework through the Customer Interface (RFQ Interface). (2) Manufacturer responds to the manufacturer survey through the Mfr. Interface. (3) User changes the preference on the weight of cost, time, and quality of the evaluated solution, until deciding on using a specific solution.

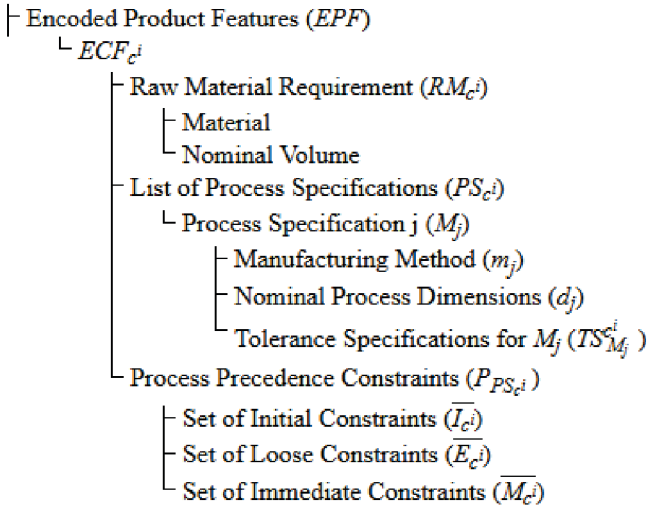

Fig. 3. Class attribute hierarchy for EPF

$[\tilde{m}, C, T, RQS_{\tilde{M}}, MFR^{\text{ID}}]$, where a unique ID identifies each OQ, and the rest of the data are defined previously.

A solution can be comprised of multiple quotations from different manufacturers, hence, assigning a quality measure to each quotation is not necessary. Instead, the set $RQS_{\tilde{M}}$ is denoted in each of the quotations so that the quality measure for the whole solution can be evaluated by the optimization algorithm. However, the Analysis API filters out quotations that do not satisfy all corresponding critical tolerances, thereby reducing the search space for the optimization algorithm. The Analysis API also maps each $RQS_{\tilde{M}}$ to its respective weights vector ($\Omega^{\text{RQS}_{\tilde{M}}}$). In addition, the Analysis API stores the submitted RQ in the database, in respective manufacturer profile.

C. Decision API

The decision API is responsible for running the optimization algorithm, and sending the optimal solution and the visualization information to the visualization interface. OQ that are evaluated by the Analysis API are stored in the database. The optimization quotes, EPF, and the location of the manufacturers and the user are needed for running the optimization algorithm. The API fetches the data from the database and runs the optimization for a number of predefined weights of the fitness function given in Eq. 1.

Visualization interface displays the evaluated solutions to the user. The user may choose to see the solutions for the predefined weights or choose a new weight to run the optimization. As the new weight is chosen by the user (denoted by the *new input* in Fig. 2), Decision API fetches the new values from the visualization interface to run the optimization with the new weights. The user iterates the cycle until settling on a desired solution. After the user decides on a particular solution, the decision is saved on the user's profile in the database and the respective manufacturers are notified to initiate the production. Full disclosure with sharing of sensitive data is expected at this final stage.

IV. Finding Feasible Solutions

A. Exhaustive Search

The optimization problem to be solved in the proposed framework is evaluating feasible solutions using the optimization quotes that the Analysis API generates In previous work [10], a Dijkstra's graph search [12] is proposed. This exhaustive search method is feasible for relatively small sets, but it is not scalable for larger sets. In this work, a genetic algorithm (GA) approach is utilized to tackle the problem with the added dimension of precedence. The GA approach is adopted to enable scalability with the increasing size and complexity of the PaaS framework.

B. Proposed GA Formulation

As the number of manufacturers in the framework increases, the computational cost of finding a feasible solution also increases. Use of evolutionary algorithms (EA) in manufacturing engineering is well documented in literature [13]. GA, which is a form of EA, is useful particularly in optimization applications where the solution set is large enough that computation of the optimal solution with brute force techniques is not feasible [12], [14], [15]. Contreras et

al. proposed a GA to solve the scheduling and resource allocation problem in a small manufacturing line [16]. The GA approach for optimizing network graphs proposed by Ahn gives an efficient genetic representation to solve graph search problems [17]. While these approaches are useful for solving such problems, precedence constraints on the solution are not extensively adopted in the literature. However robust precedence analysis techniques using precedence graphs and EA have been considered before [18], [19].

Algorithm 1 Proposed GA

Input: $\mathcal{P}_{\mathbb{D}} = (\bar{I}_{\mathbb{D}}, \bar{E}_{\mathbb{D}}, \bar{M}_{\mathbb{D}}), \mathbb{D}, \alpha_1, \alpha_2, L$
Generate individuals with $S_i = Prec(\mathcal{P}_{\mathbb{D}}, \mathbb{D})$ to create random population $POP(S) \leftarrow S_i$; $\#GEN = 1$.
Calculate fitness $J(S_i)$ (Eq.1) for each individual S_i ▷ Inputs α_1, α_2 and L, are used in $J(S_i)$.
while $\#GEN \leq MAXGEN$ **and** $mean(J(S_i)) \geq TOLER$ **do**
    Select m fittest S_i as parents $Pa(S)$ for next gen.
    $EL \leftarrow Pa(S)^{\text{elite}}$ ▷ Preserve best *elite* individuals
    while $size(CH) \leq CHSIZE$ **do**
        Crossover $S_j \times S_k = S_c$, where $(S_j, S_k) \in Pa(S)$
        if S_c obeys the precedence set $\mathcal{P}_{\mathbb{D}}$ **then** ▷ The child is feasible
            $CH \leftarrow S_c$
        end if
    end while
    Mutate the non-elite & non-parent individuals $S_i \rightarrow S_i^{\text{mut}}$. Change random $s_i \in S_i$ with random gene (quote) with same $s_{i_{proc}}$.
    $MU \leftarrow S_i^{\text{mut}}$
    $POP(S) \leftarrow EL \bigcup CH \bigcup MU$
    $\#GEN = \#GEN + 1$ ▷ Next generation
end while
Output: Last generation *POP(S)*

Let $L \in \mathbb{R}^{(n+1)\times(n+1)}$ be a distance matrix denoting the distance between each of n manufacturers who submitted quotes for production, and the user; and $L(I, J)$ denote the distance between manufacturers I and J. As mentioned in Section III, manufacturers are allowed to submit production quotes for a single process or several processes. A solution is said to be feasible when the precedence constraints are satisfied and all the production processes are covered by the quotes in the solution. Let S denote an individual solution in the genetic algorithm and s_i denote i^{th} gene (quote) of the individual. $S = \{s_1, s_2, \ldots, s_n | s_i \in \mathbb{D}\}$ denotes a feasible solution where $\mathbb{D}$ is the set of all available quotes (genes). The proposed GA is given in Algorithm 1. Each quote $s_i = [s_{i_{proc}} s_{i_{cost}} s_{i_{ttime}} s_{i_{quality}} s_{i_{no}}]$ has information slots for processes covered by the quote ($s_{i_{proc}}$), cost ($s_{i_{cost}}$), time ($s_{i_{ttime}}$), quality information ($s_{i_{quality}}$), and manufacturer ID ($s_{i_{no}}$). Note that each s_i is unique and mapped to an OQ^{ID}. Fitness function assigned to each individual to be minimized is as follows [10]:

$$\min_S \; J(S) = \frac{C(S)}{\mu_{cost}} + 2^{\alpha_1}\frac{T(S)}{\mu_{time}} + 2^{\alpha_2}Q(S) + \gamma P(S) \quad (1)$$

where α_1, α_2 are the linearized weights, assigned to the optimization variables, γ is an adjustable coefficient to adjust the effect of proximity-to-user constraint. μ_{cost}, and μ_{time} are the mean cost and mean time of all the quotes in set $\mathbb{D}$ respectively. These means are used in normalizing the results of the cost and time functions. $C(S)$, the cost function of an individual to be minimized is defined as:

$$C(S) = \sum_{i=1}^{N} [s_{i_{cost}} + \beta_1 L(s_{i_{no}}, s_{i+1_{no}})]$$

where N is the number of quotes in the solution S, β_1 denotes the unit transportation cost, and $s_{i+1_{no}}$ denotes the next manufacturer specified by the precedence constraints on the processes in $s_{i_{proc}}$. $T(S)$, time function for an individual S to be minimized is then defined as:

$$T(S) = argmax\,[Y_1(S), \ldots, Y_t(S)] + s^*_{time}$$
$$Y_j(S) = U_{c^k}(S) + \beta_2 L(s^{c^k}_{last_{no}}, s^{c^k}_{i+1_{no}})$$
$$U_{c^k}(S) = \sum_{i=1}^{|X_{c^k}|} \left[s^{c^k}_{i_{ttime}} + \beta_2 L(s^{c^k}_{i_{no}}, s^{c^k}_{next_{no}}) \right]$$

where set X_{c^k} is the set of quotes associated with component c^k, $s^{c^k}_{next_{no}}$ is the manufacturer who works on c^k after $s^{c^k}_{i_{no}}$ in the solution, β_2 denotes the unit time for transportation, $U_{c^k}(S)$ denotes the time required to completely manufacture c^k, $Y_j(S)$ denotes the time taken for transporting the finished component to its respective assembly location $s^{c^k}_{i+1_{no}}$ specified using precedence, s^*_{time} denotes the time taken for assembly and $T(S)$ is the time taken to manufacture the complete product.

$$Q(S) = \frac{-W^T B}{W^T \overrightarrow{\mathbb{1}}}$$

where $W = [\omega^{\text{ID=1}} \omega^{\text{ID=2}} \ldots \omega^{\text{ID=last}}]^T$ is a vector of tolerance weights ordered according to the tolerance IDs, $B = [\tau^{\text{ID=1}} \tau^{\text{ID=2}} \ldots \tau^{\text{ID=last}}]^T$ is a binary vector wherein each element indicates whether the tolerance with the specified ID has been met or not in the quotations $s_i \in S$. $\overrightarrow{\mathbb{1}} = [11\ldots1]^T_{\text{ID=last}\times 1}$ is a vector of ones with as many elements as the number of tolerances.

The proximity to user constraint is added to the fitness function (Eq. 1) as $P(S)$ defined as:

$$P(S) = \left[\frac{\beta_1 L(s_n, user)}{\mu_{cost}} + 2^{\alpha_1} \frac{\beta_2 L(s_n, user)}{\mu_{time}} \right]$$

where s_n indicates the last process in production. Note that a single run of GA outputs a final generation of solutions with the optimal and sub-optimal results that could be presented to the user through the visualization interface.

C. Precedence

As discussed in Section III-A, each ECF has its own $\mathcal{P}_{PS^{c^i}}$ defining precedence constraints on the processes. The precedence analysis for a precedence set $\mathcal{P}_{\mathbb{D}} = \{\mathcal{P}_{PS^{c^1}} \cup \ldots \cup \mathcal{P}_{PS^{c^n}}\}$ denoting the union of all the precedence constraint sets, is given in Algorithm 2. Note that assembly precedences are included in set $\mathcal{P}_{\mathbb{D}}$ as well.

Algorithm 2 Precedence constraints assignment.

function PREC($\mathcal{P}_{\mathbb{D}}, \mathbb{D}$)
  Input: $\mathcal{P}_{\mathbb{D}} = (\bar{I}_{\mathbb{D}}, \bar{E}_{\mathbb{D}}, \bar{M}_{\mathbb{D}}), \mathbb{D}$
  Individual S for n processes with vector $R = [0]^{1\times n}$
  Pick random index $i^{\alpha_x} \in \bar{I}_{\mathbb{D}}$
  Pick $s_0 \in \mathbb{D}$, where $\alpha_x \in s_{0_{proc}}$; $R_{s_{0_{proc}}} \leftarrow 1$
  while $R \neq [1]^{1\times n}$ **do**
    $K \leftarrow R$
    for all $(j,p,k) \in \bar{E}_{\mathbb{D}} = \{\bar{e}^{\alpha_p}_{\alpha_j,\alpha_k}\}$ **do**
      if $R_j = 1$ **then**
        $K_k \leftarrow 1$ ▷ *lock* k
      else if $R_j = R_p = R_k = 0$ **then**
        $(K_p, K_k) \leftarrow 1$ ▷ need j first
      end if
    end for
    for all $(a,b,c) \in \bar{M}_{\mathbb{D}} = \{\bar{m}^{\alpha_b}_{\alpha_a,\alpha_c}\}$ **do**
      if $R_a = 1$ **then**
        Choose x as b, **go to** 25
      else if $R_b = 1$ **then**
        Choose x as c, **go to** 25
      else if $R_a = R_b = R_c = 0$ **then**
        $(K_b, K_c) \leftarrow 1$ ▷ need a first, lock b, c
      end if
    end for
    Pick random index x from zeros of K
    Pick $s_i \in \mathbb{D}$ with $x \in s_{i_{proc}}$; $R_{s_{i_{proc}}} \leftarrow 1$
  end while
  Output: S
end function

Another part in the optimization that accounts for the precedence constraints is the crossover genetic operator in line 8 of Algorithm 1. After crossover is applied to selected individuals, a check function is applied to the children. Children resulting from crossover are forced to be feasible and the crossover operation is repeated on the generation until required number of feasible children are produced. This second checking ensures the feasibility of the offspring and the resulting solution. Using the given precedence analysis, the output of the optimization becomes a solution in terms of the given fitness function and the precedence constraints.

V. CASE STUDY

The case study is prepared for an assembly of a model truck made of 7 parts (Fig. 4); four wheels, a cabin, a trunk, and a chassis. The required processes are as follows:

1) Machining (face milling and drilling) of the chassis.
2) 3D printing of the trunk.
3) 3D printing of the cabin.
4) 3D printing of the four wheels.
5) Assembly of the components

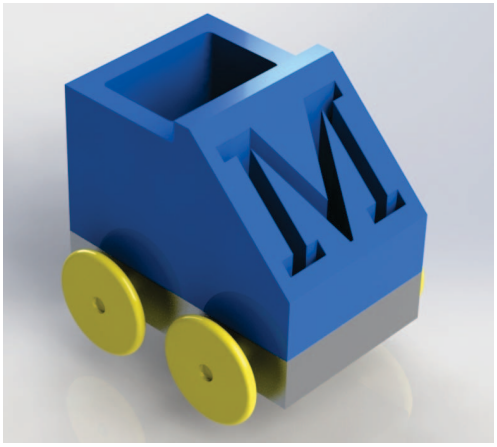

Fig. 4. The case study model truck. 4 Wheels (yellow), cabin (front, blue), trunk (back, blue), chassis (bottom, grey).

For each 3D printed part in the model truck, 37 quotes were collected using on-line 3D printing service channels, and the UM3D Lab at the University of Michigan. Majority of the quotes had Fused Deposition Modeling (FDM) as the printing process, while a few of them offered Stereolithography (SLA). Though the quality of products made by SLA is much higher than that of FDM due to finer layer height, the associated time and cost is higher with SLA in comparison to FDM. Although FDM is widely used by many, process tolerances and quality requirements are not unified across different manufacturers. While most manufacturers gave specific details about the layer height and the material for the printing, machine type, precision, and accuracy were not specified by all. Also, some manufacturers prefer to give quotes including the shipping cost and time, while others prefer to specify the cost and time for the printing job only. Due to the non-uniform nature of the quote data, collected quotes are processed to specify the cost and time for the production, and the quality index for the quotes are assigned according to the layer height (less is better) of each process since it denotes a norm for all the printing processes. Collected data is used to create a larger simulation set with 110 manufacturers including the actual manufacturers. The machining process quotes are generated using a normal distribution around central limits estimated for the required machining job. Since the product used in the case study requires an elementary assembly, all the manufacturers are assumed to be capable of the assembly process.

A. Encoded Product Features (EPF)

To describe the product without divulging any sensitive data, ECFs were prepared for each component. ECFs for the chassis and trunk are illustrated in Table I.

It is straightforward to realize from Table I that the chassis requires two processes, namely face milling and drilling. The nominal volume and material mentioned in RM_{c^1} indicate the amount of raw material required for the

chassis. The nominal dimension associated with each process (d_i) indicates the amount of material to be removed in that particular process. The tolerance information stated in $TS^{c^1}_{M_1}$ and $TS^{c^1}_{M_2}$ specifies the user's expectations on flatness and circularity after the component undergoes face milling and drilling operation respectively. The aggregate of information delineated in Table I allows manufacturers to estimate the resources (material, time etc.) required to manufacture the component, and accordingly prepare a production quote for this component. However, since the location of drilled holes is not disclosed in ECF_{c^1}, manufacturers will not be able to produce it as envisioned by the user at this stage. Hence, the user's IP is protected. All 7 ECFs, along with the component precedence constraints, constitute the EPF for the truck.

TABLE I

ECF FOR COMPONENT 1 (CHASSIS). *All dimensions in mm or mm^3.*

Encoded Component Features ECF_{c^1}	RM_{c^1}	Material				Aluminum
		Nominal Volume				76.2×50.8×20.0
	Process Specifications PS_{c^1}	Process M_1	Process Name m_1			Face Milling
			Process Dimensions d_1			76.2×50.8×5.0
			User Defined Toleranaces TS_{M_1}	Tolerance $\mathcal{T}_1$	ID	1
					ω	2
					$\mathcal{T}_{type}$	Flatness
					$\mathcal{T}_{dim}$	0.0
					$\mathcal{T}_{UD}$	+0.2
					$\mathcal{T}_{LD}$	-0.2
		Process M_2	Process Name m_2			Drilling
			Process Dimensions d_2			2× Φ5 × 50.8
			User Defined Tolerances TS_{M_2}	Tolerance $\mathcal{T}_2$	ID	2
					ω	3
					$\mathcal{T}_{type}$	Circularity
					$\mathcal{T}_{dim}$	0
					$\mathcal{T}_{UD}$	+0.3
					$\mathcal{T}_{LD}$	-0.3
	$\mathcal{P}_{PS_{c^1}}$	$\bar{I}_{c^1}$				-
		$\bar{E}_{c^1}$				-
		$\bar{M}_{c^1}$				-

B. Effect of Precedence

In this work, the optimization procedure introduced in [10] is updated with the use of a GA approach and the precedence constraint as explained in Section IV-B. The implementation of the precedence in the optimization algorithm presented in Section IV-C is realized using MATLAB® GA Solver. To demonstrate the effect of relaxing precedence constraints, a case with strict precedence (no flexibility) is compared with a relaxed precedence case. Fig. 5 shows the optimization output for different precedence constraints. The machining and AM manufacturers are marked with green triangles and red dots respectively. The user is marked with a black diamond. An optimization for the least total cost solution is evaluated in both cases. Solid blue lines, which show the optimal solution path, indicate that the optimal solutions in both cases use same quotes.

The strict order constraints based on the previous work [10] for the set of four components $H = \{c^1, c^2, c^3, c^4\}$ starts the production with the machining job of chassis (c^1), and continues with the three remaining AM components (c^2, c^3, c^4), thus $\mathcal{P}^{\text{str}}_H = (\bar{I}_H, \bar{E}_H, \bar{M}_H)$, where $\bar{I}_H = \{i^{c^1}\}$, $\bar{E}_H = \{\}$ and $\bar{M}_H = \{\bar{m}^{c^3}_{c^2,c^4}\}$. Note that the precedence is

Fig. 5. Strict precedence (left, Cost: $104.048), relaxed precedence (right, Cost: $103.638). Solid blue lines shows the optimal solution and dashed red lines show alternate (sub optimal) solutions. The sub-optimal solution on the right figure starts in the green triangle (machining mrf.)

defined on the components that are assumed to be assembled in a serial fashion, instead of the individual processes. This solution is identical to the exhaustive search solution presented in [10]. The distance $L(I_{AM}, user)$ between the AM manufacturer (I_{AM}) and the user is longer than the distance $L(I_{Mach}, user)$ between the machining manufacturer I_{Mach} and the user. As the hard ordered immediate precedence is relaxed to the loose constraints as $\mathcal{P}^{\text{rlx}}_H = (\{i^{c^1}, i^{c^2}\}, \{\bar{e}^{c^3}_{c^2,c^4}\}, \{\})$, the machining processes are allowed to appear in a different order and the resulting solution shown on the right in Fig. 5 yields a lower total cost solution due to the lowered logistics costs (actual values given in the figure).This elementary example exhibits the effect of the precedence on the optimal solution. As the precedence constraints are relaxed, the GA evaluates a greater number of possible solutions that may end up resulting in better solutions in terms of the optimized parameters.

The resulting optimal solution shown in the Fig. 5, uses one of the simulated machining manufacturers and an online 3D printing service. Although manufacturers located closer to the user than the optimal solution are present in the database, the evaluated solution uses a manufacturer approximately 20 miles away from the user. This is due to the low manufacturing cost in the quotes provided by that manufacturer. Also, note that all 3D printing processes are performed by a single manufacturer, which is an intuitive solution for the best cost scenario, since the manufacturers are expected to apply a discount as they submit for multiple processes. FDM is used in 3D printing by the specified manufacturer. Setting different transportation costs and times would yield varying results for the same set of quotes.

Fig. 6 shows the mean score of the solutions of the GA based on the fitness function (Eq. 1) for the two different precedence constraints. The value of the coefficient α_1 is varied while α_2 is kept constant. The mean values on the figure are the result of 15 consecutive runs for the specified α values. The strict precedence in Fig. 6 is given by $\mathcal{P}^{\text{str}}_H = (\{i^{c^2}\}, \{\}, \{\bar{m}^{c^3}_{c^2,c^4}\})$, while the relaxed precedence is given by $\mathcal{P}^{\text{rlx}}_H = (\{i^{c^1}, i^{c^2}\}, \{\bar{e}^{c^3}_{c^2,c^4}\}, \{\})$. Experimentation with the implemented optimization showed a 22-50% improvement of the fitness score of the final solution over the initial feasible solution. It is clear that the relaxed precedence results in solutions with lower scores (higher fitness). While the effect of precedence relaxation would be better pronounced with

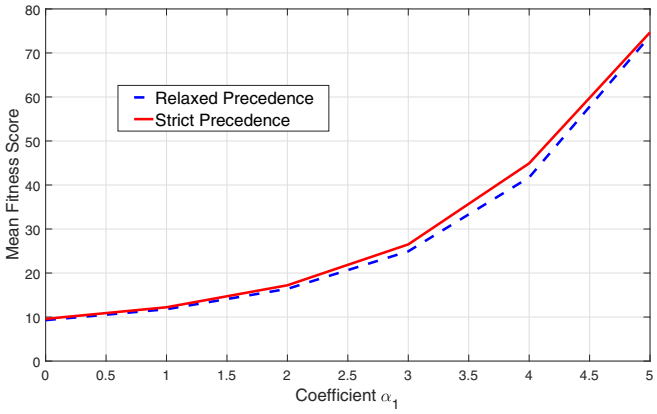

Fig. 6. Comparison of the evaluated value of the fitness function given in Eq. 1 for two cases of $\mathcal{P}_H$ given in Section V-B, versus the time function coefficient α_1. The second coefficient α_2 in Eq. 1 is kept constant at zero to evaluate the normalized effect of quality in the solution with varying α_1. Other constants in the optimization: $\beta_1 = 0.75\ (\$/mi),\ \beta_2 = 0.0029\ (day/mi),\ \gamma = 1$.

increased unit transportation cost and time, the unit cost and time given in the figure are used for a realistic case study.

VI. Conclusion

In this paper, we proposed a framework *Production as a Service* that facilitates the task of finding suitable manufacturers for product developers with small manufacturing needs. The minimum information required from users and manufacturers to request, and submit production quotes respectively is listed. The novel concept of features is also introduced. Features encode the required information in an abstracted manner, thereby preventing IP theft. In addition, a description of the data structures employed to record this information is also provided. The architecture of the framework in terms of its constituting interfaces, APIs and database is discussed. While the interfaces allow users and manufacturers to communicate with the framework, the APIs equip the framework with the capability to analyze RFQs and quotes, and find optimal production schemes. Finally, the potential benefits of relaxing precedence constraints on manufacturing processes are illustrated through a case study.

While the proposed structure of features is sufficient for relatively simple geometries in product designs, a more comprehensive formalism is needed for the abstraction of intricate geometries. A more comprehensive case study that utilizes various new parameters like customer satisfaction, setup costs, and design costs will be conducted in the future research to investigate the effect of different metrics. A functionality that the current framework design lacks is the ability to split the requested batch size across multiple manufacturers when needed. Incorporating this functionality in the framework is a subject for future research.

ACKNOWLEDGMENT

This work was funded in part by NSF 1546036. The authors would like to thank the UM3D Lab at the University of Michigan for providing quotes for the case study. The authors also thank the Fall 2015 Secure Cloud Manufacturing Multidisciplinary Design Project Team at the University of Michigan for designing the original case study model truck.

References

[1] S. J. Hu, "Evolving paradigms of manufacturing: from mass production to mass customization and personalization," *Procedia CIRP*, vol. 7, pp. 3–8, 2013.

[2] A. Ordanini and P. Pasini, "Service co-production and value co-creation: The case for a service-oriented architecture (soa)," *European Management Journal*, vol. 26, no. 5, pp. 289–297, 2008.

[3] L. v. Moergestel, E. Puik, D. Telgen, and J.-J. C. Meyer, "Implementing manufacturing as a service: A pull-driven agent-based manufacturing grid," in *Proceedings of the 11th International Conference on ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer*, vol. 1356. CEUR Workshop Proceedings, 2015, pp. 172–187.

[4] M. Moghaddam, J. R. Silva, and S. Y. Nof, "Manufacturing-as-a-service: From e-work and service-oriented architecture to the cloud manufacturing paradigm," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 828 – 833, 2015.

[5] D. Wu, D. W. Rosen, L. Wang, and D. Schaefer, "Cloud-based design and manufacturing: A new paradigm in digital manufacturing and design innovation," *Computer-Aided Design*, vol. 59, pp. 1 – 14, 2015.

[6] C. Oboulhas, X. Xiaofei, and Z. Dechen, "A model for multi-plant production planning coordination," *The International Arab Journal of Information Technology*, vol. 2, no. 3, pp. 177–182, 2005.

[7] 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 Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 36, no. 1, pp. 62–75, 2006.

[8] Y. Wang, J. Hulstijn, and Y.-H. Tan, "Data quality assurance in international supply chains: An application of the value cycle approach," in *Conference on e-Business, e-Services and e-Society*. Springer, 2015, pp. 467–478.

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

[10] M. Porter, V. Raghavan, Y. Lin, Z. M. Mao, K. Barton, and D. Tilbury, "Production as a service: Optimizing utilization in manufacturing systems," *ASME Dynamic Systems and Control Conference*, 2016.

[11] Mongodb. [Online]. Available: https://www.mongodb.com/

[12] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[13] C. Dimopoulos and A. M. Zalzala, "Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons," *IEEE transactions on evolutionary computation*, vol. 4, no. 2, pp. 93–113, 2000.

[14] N. Jasika, N. Alispahic, A. Elma, K. Ilvana, L. Elma, and N. Nosovic, "Dijkstra's shortest path algorithm serial and parallel execution performance analysis," in *MIPRO, 2012 proceedings of the 35th international convention*. IEEE, 2012, pp. 1811–1815.

[15] J. McCall, "Genetic algorithms for modelling and optimisation," *Journal of Computational and Applied Mathematics*, vol. 184, no. 1, pp. 205–222, 2005.

[16] A. R. Contreras, C. V. Valero, and J. A. Pinninghoff, "Applying genetic algorithms for production scheduling and resource allocation. special case: A small size manufacturing company," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2005, pp. 547–550.

[17] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE transactions on evolutionary computation*, vol. 6, no. 6, pp. 566–579, 2002.

[18] K. R. Antani, B. Pearce, L. Mears, R. Renu, M. E. Kurz, and J. Schulte, "Application of system learning to precedence graph generation for assembly line balancing," in *ASME 2014 International Manufacturing Science and Engineering Conference*. American Society of Mechanical Engineers, 2014, pp. V001T04A001–V001T04A001.

[19] J. Dou, X. Dai, and Z. Meng, "Precedence graph-oriented approach to optimise single-product flow-line configurations of reconfigurable manufacturing system," *International Journal of Computer Integrated Manufacturing*, vol. 22, no. 10, pp. 923–940, 2009.