Toward Cloud Manufacturing: A Decision Guidance Framework for Markets of Virtual Things

Xu Han[a] and Alexander Brodsky[b]
Department of Computer Science, George Mason University, 4400 University Drive, Fairfax, U.S.A.

Keywords: Markets of Virtual Things, Decision Guidance, Formal Framework, Cloud Manufacturing.

Abstract: In the value creation chain today, entrepreneurs have been faced with stiff hindrance in turning their innovative ideas into marketable products due to the manufacturing-entrepreneurship disconnect in terms of accessibility, predictability and agility. Toward bridging this gap, in this paper we develop a formal mathematical framework for markets of virtual things: parameterized products and services that can be searched, composed and optimized. The proposed framework formalizes the notions of virtual product and service designs and customer-facing specs as well as requirements' specs. Based on these formal concepts, the framework also formalizes the notions of search for and composition of virtual products and services that (1) are mutually consistent with the requirement specs, (2) are Pareto-optimal in terms of customer-facing metrics such as cost, product desirable characteristics and delivery terms; and (3) that are optimal in terms of customer utility function that is expressed in terms of customer facing metrics. We also propose the design of a repository of virtual things and their artifacts, to be used in support of the virtual things' markets. The proposed markets of virtual things can lead to democratizing innovation by allowing entrepreneurs without design and manufacturing expertise to bring their ideas to markets quickly.

1 INTRODUCTION

In the value creation chain today, entrepreneurs have been faced with stiff hindrance in turning their innovative ideas into marketable products due to the manufacturing-entrepreneurship disconnect in terms of accessibility, predictability and agility ((Brodsky et al., 2021)). In order for entrepreneurs to realize their innovative ideas, they have to either (1) have strong technical backgrounds and expansive knowledge about product and process design, development, manufacturing and testing, or (2) assemble a team of experts who can help translate and convey the ideas to the designers, developers, testers and manufacturers. That is, entrepreneurs lack accessibility to manufacturing capacity, predictability and agility to get to market. Furthermore, the current approaches typically lead to sub-optimal outcomes due to information loss during the ideation and communication processes.

For manufacturers, due to a lack of market information, significant amount of manufacturing capacity is underdeveloped, idle or wasted, missing out opportunities to reach higher levels of economies of scale to save cost and to create value. When entrepreneurs find a rising market trend for a certain product, the technical specs with all essential information that captures the complex features and properties of the product can hardly be generated and passed in time to the manufacturers that have the interest, availability and resources to produce it. Furthermore, when the information is passed from the entrepreneurs to the downstream channels, there is slow or no feedback. And there is little or no opportunity for optimizing the decision making and value creation processes. That is, while manufacturers may have predictability for products they know how to produce, they lack access to innovative product ideas and related higher-margin revenue opportunities, and the agility to respond to these innovative ideas.

Clearly, entrepreneurs are in need of a simple way of converting their product and service ideas to manufacturing-ready technical specs so that they can achieve easy accessibility, predictability and agility of the ideation-to-production process. Trying to bridge the gap, there has been significant research in parametric product and process design ((Gingold et al.,

[a] https://orcid.org/0000-0002-3347-3627
[b] https://orcid.org/0000-0002-0312-2105

2009), (Yu et al., 2011), (LaToza et al., 2013), (Shin et al., 2017)), analysis and optimization ((Egge et al., 2013), (Shao et al., 2018)). Parametric design seems to point out a promising direction as it offers the flexibility to the entrepreneurs - setting and changing the parameters enables high customizability, and a large set of design alternatives can be generated for the entrepreneurs to choose from. Recently, a number of start-ups, such as Xometry, Kerfed and Physna, have taken important complimentary steps to simplify the process from product specs to manufacturing orders.

However, none of the above-mentioned works has specified how to present the artifacts of the design in a way that the entrepreneurs would find straightforward and intuitive to understand and practically work on, while providing the manufacturers with all the product and process designs necessary for manufacturing. For different products, there is no consistent way to reuse the models of the parts that can be shared. Furthermore, when optimization is deployed to parametric designs, it is typically done in product and process silos, as opposed to optimizing holistically across the value chain, thus missing out the opportunity to reach Pareto-optimal solutions in creating the products.

In order to enable the entrepreneurs to conceive their ideas in a non-technical language but to express them in technically recognizable terms, we proposed the concept of markets of virtual products and services to support cloud manufacturing ((Brodsky et al., 2021)). However, the proposed concept lacked the mathematical formalization which is necessary to provide the theoretical basis for developing market systems. Bridging this gap is exactly the focus of this paper.

More specifically, in this paper we develop a formal mathematical framework for markets of virtual things: parameterized products and services that can be searched, composed and optimized. The proposed framework formalizes the notions of virtual product and service design and customer-facing specs as well as requirements' specs. Based on these formal concepts, the framework also formalizes the notions of search for and composition of virtual products and services that (1) are mutually consistent with the requirement specs, (2) are Pareto-optimal in terms of customer-facing metrics such as cost, product desirable characteristics and delivery terms; and (3) that are optimal in terms of customer utility function that is expressed in terms of customer facing metrics. We also propose the design of a repository of virtual things and their artifacts, to be used in support of the market.

The paper is organized as follows. In Section 2 we give an overview of the idea and associated key concepts. Then in Section 3 we present the mathematical formalization framework of the V-things. We illustrate with an example to show how various artifacts in the framework are organized for composition, analytical and optimization purposes in Section 4. And we conclude with possible directions for future works in Section 5.

2 OVERVIEW OF V-THING FRAMEWORK

In the traditional ideation to production value creation chain, the entrepreneurs conceive their ideas of certain products, work with designers to sketch out their ideas, build the prototypes, and then pass the technical specs onto the manufacturers for production. The process requires expertise in various capabilities, and is often inaccurate, slow and rigid. We desire to overcome these defects through the productivity framework of V-things that employees a fundamentally new approach of connecting entrepreneurs with manufacturers in a bootstrapped marketplace of virtual products and services (shown as the V-Things Markets on the right in Figure 1).

Within the marketplace of V-things, the virtual products can be presented by their parameterized CAD design, and searched by their parameters. The virtual services are the parameterized transformations of virtual products (assembly, transportation, etc.). All virtual products and services have their own analytical model respectively that describes their important characteristics ((Brodsky et al., 2017)). When the entrepreneurs are searching and composing the virtual products in the design environment, they can utilize the AI-powered decision guidance system to reach optimal decisions (shown as the V-Thing Decision Guidance System on the left in Figure 1). The designers and manufacturers can also proactively participate in the discovery of new market demands within the V-thing marketplace using the V-thing design tools. And insights can be drawn by analyzing the data in the system with the avail of Decision Guidance Analytics Language ((Brodsky and Luo, 2015)).

Take for example that an entrepreneur wants to make a new product of a special type of bicycle that is non-existent in the real world. She can start with searching for any virtual products that are similar to her idea in the V-things markets. If she finds none, she will consider building her own bike. First, she will initiate her virtual bike project, and start to describe her idea of the bike. She may use the design-by-sketch and design-by-example features to generate a design spec from scratch, or she can specify a

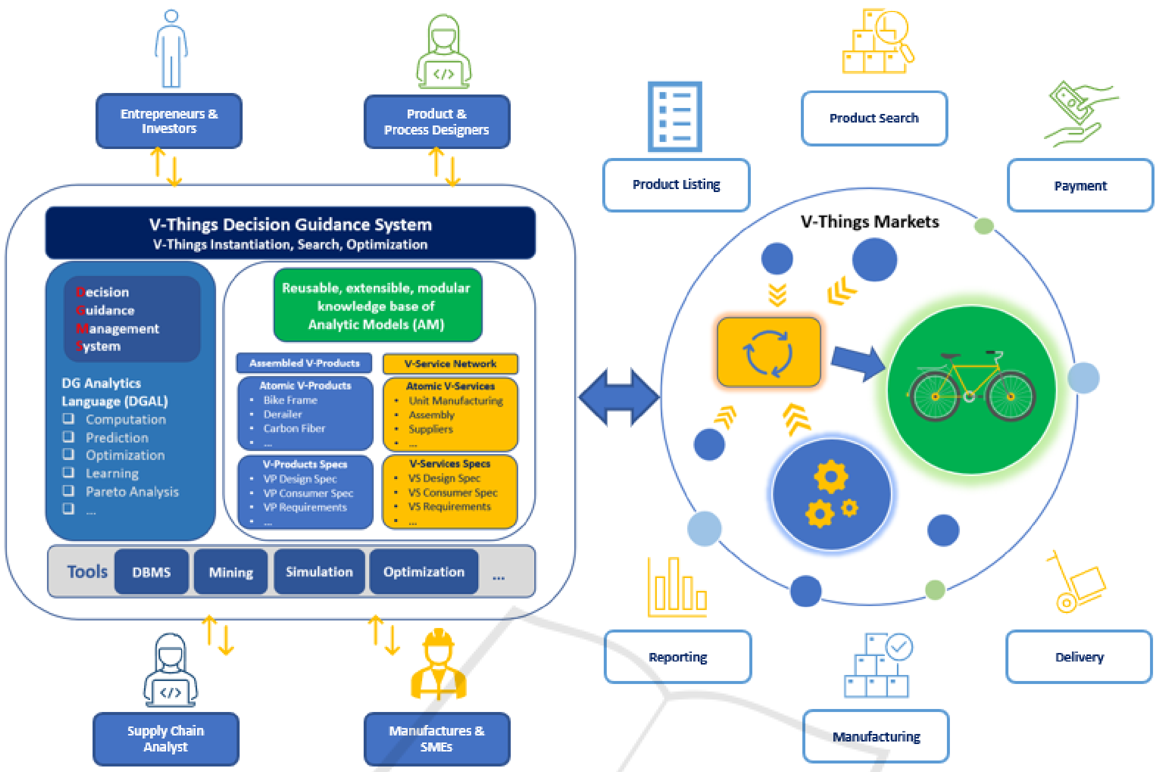


Figure 1: The V-Things Decision Guidance Ecosystem.

more definitive spec of the bike with its components (bike frame, bike chain, tires, etc.). Second, she can search again for the components of the bike as virtual products in the marketplace, and if she cannot find one, she can either design one or solicit for one with a spec. The system will make recommendations and guide her to make the best informed decisions along the way. The entrepreneur can recursively find the components or generate the specs of the components for the bike that would closely approximate her original idea. Third, she would solicit for the services through which the components can be manufactured, integrated and assembled into the final product. After all constraints in the specs are verified to be feasible by each virtual product component provider and service provider, the entrepreneur can see the metrics of manufacturing the bike that are generated and optimized by the analytical models (cost, time, carbon emission, etc.). And she can see if the bike reflects her original idea or not, then she can make a decision whether to list the bike in the virtual marketplace.

If the entrepreneur thinks the product is satisfactory and decides to take the go-to-market move, the special bike virtual product will be made available for the potential consumers in the marketplace. And the entrepreneur can make further business decisions on how to price, advertise, and deliver the product. The potential consumers can see this special type of bike in the V-things marketplace, what its physical properties and performance metrics are, how much it costs, and how long it takes to deliver - everything about a product that the consumers want to know except that the product is in a virtual state in the sense that it is "non-existent-yet" in the physical world.

If there is strong interest shown by the consumers in the product, whether it comes through the number of orders placed or the relevant discussions posted, the entrepreneur will give the green light to put the bike into actual production and start to ship it to the consumers as a real product. She can take advantage of the responsive market reaction as a pilot campaign. After collecting enough user feedback data from the marketing campaign and the product reviews, the entrepreneur can get informed of how well the consumers have received this new product. From there the entrepreneur can initiate a product launch with calculated risk by listing the bike in the real-world online or on-premise markets and sell the product to the targeted user groups.

3 MATHEMATICAL FORMALIZATION FOR VIRTUAL THINGS

3.1 Virtual Products

Intuitively, a virtual product, which is defined formally below, is a parameterized CAD design with an associated analytic model that defines all feasible parametric instantiations along with their consumer-facing metrics. For example, a virtual product can be a final product (a bike), a part of another product (a bike fork or a wheel), or a raw material (steel). The analytic models put the constraints on the virtual products (the dimensions of the bike fork must fit that of the wheels), and describe feasibility of the product based on the parameters and metrics (a bike weighing 1 ton will be infeasible as it violates the tolerance of the wheels).

Definition 3.1. *A virtual product design spec* (or VP for short) is a tuple

$$VP = < domain, parametersSchema, metricSchema, template, analyticModel, feasibilityMetric >$$

where

1. domain - is a set of all things under consideration. E.g., all components that can be specified using CAD design.
2. parametersSchema - is a set
$$PS = (p_1, D_1), \dots, (p_n, D_n)$$
of pairs, where for $i = 1, \dots, n$
   (a) p_i - a unique parameter name (e.g., geometric dimension, tolerance, density of material) (fixed parameters vs decision variables)
   (b) Di - the domain of p_i
3. metricSchema - is a set
$$MS = \{(m_1, M_1), \dots, (m_k, M_k)\}$$
of pairs, where for $i = 1, \dots, k$
   (a) m_i - a unique metric name (e.g., weight, strength, volume, floatable (Boolean))
   (b) M_i - the domain of m_i
4. template - is a function
$$T : D_1 \times \dots \times D_n \to Domain$$
which gives, for parameter instantiation $(p_1, \dots, p_n)$ in $D_1 \times \dots \times D_n$, a specific thing in the Domain of things.
5. analyticModel - is a function
$$AM\colon D_1 \times \dots \times D_n \to M_1 \times \dots \times M_n$$
which gives, for parameter instantiation $(p_1, \dots, p_n)$ in $D_1 \times \dots \times D_n$, a vector $(m_1, \dots, m_k)$, where $(m_1, \dots, m_k)$ are metric values in $M_1 \times \dots \times M_k$. We will denote resulting metric value m_i, $i = 1, \dots, n$, using the (.) dot notation as $AM(p_1, \dots, p_n).m_i$ or $m_i(p_1, \dots, p_n)$
6. feasibilityMetric - is a Boolean metric name C in $\{m_1, \dots, m_n\}$ such that its corresponding domain $D = \{T, F\}$ in metricSchema MS (T to indicate that parameter instantiation is feasible, and F otherwise)

3.2 Specs of Virtual Products

In order to communicate the information between multiple parties in the V-things marketplace, manufacturers may not want to disclose low-level design parameters (in the parametric schema) but only disclose product metrics (in the metric schema) which we assume contain all product information relevant to consumers. For example, a consumer purchasing a bike may only want to know the high-level commercial and performance information about the bike (such as price, weight, max speed and type) rather than of the more granular supply chain metrics (such as prices of the parts, dimensions of the bike pedals and strength of the steel.) This notion is captured in the concept of VP consumer-facing specs.

Definition 3.2. *VP consumer-facing spec* (or VP CFS for short) is a tuple

$$\delta = < domain, metricSchema, MFC >$$

where domain, metricSchema are as defined in VP, and $MFC\colon M_1 \times \dots \times M_k \to \{T, F\}$ is a set of feasibility constraints that tells, for every vector of metrics in $M_1 \times \dots \times M_k$, whether it is feasible or not.

Given a VP design spec, manufacturers may want to extract a VP consumer-facing spec from it, defined next.

Definition 3.3. *VP consumer-facing spec derived from VP design spec* vp *(denoted CFS(vp) for short).* Let

$$vp = (domain, parametersSchema, metricSchema, template, analyticModel, feasibilityMetric)$$

be a VP design spec. The derived consumer-facing spec CFS(vp) is a tuple $(domain, metricSchema, MFC)$, where MFC is defined as follows: $\forall (m_1, \dots, m_k) \in M_1 \times \dots \times M_k$, $MFC(m_1, \dots, m_k) = \exists (p_1, \dots, p_n) \in D_1 \times \dots \times D_n$ s.t. $(m_1, \dots, m_k) = AM(p_1, \dots, p_n) \wedge C(p_1, \dots, p_n) = T$

Note: CFS(vp) defines the set of all feasible metric vectors $\{(m_1,\dots,m_k)|(m_1,\dots,m_k) \in M_1 \times \cdots \times M_k \wedge MFC(m_1,\dots,m_k)\}$

In order to enable the users to search for the VPs, we introduce the VP Requirements spec. The purpose of the Requirements spec is for the users to get conditions for search as they put constraints on the VP space.

Definition 3.4. *VP (consumer) Requirements Spec (or RS for short)* is a tuple

$$RS =< domain, metricSchema, objectiveSchema, objectives, MC, OC >$$

where

1. $objectiveSchema$ is a set
$$OS = \{(o_1,O_1),\dots,(o_l,O_l)\}$$
of pairs, where for $i = 1,\dots,l$
   (a) o_i - a unique objective name (e.g., cost, risk, time, etc.)
   (b) O_i - the domain of o_i
2. objectives: $M_1 \times \cdots \times M_k \to O_1 \times \cdots \times O_l$ defines objectives as a function of metrics i.e., $objectives(m_1,\dots,m_k)$ is a vector of objective values $(O_1,\dots,O_l)$
3. $MC : M_1 \times \cdots \times M_k \to \{T,F\}$ define feasibility constraints in the metric space
4. $OC : O_1 \times \cdots \times O_l \to \{T,F\}$ define feasibility constraints in the objective space

The user search should only render the results that fall in the range of the constraints domains determined by the product characteristics and specified by the user. For example, a user searches for a bike with weight not greater than 10 kg, and price not greater than $200, then the result should only contain products that are feasible within the metrics ranges, and also comply to the user specified objective domains.

Definition 3.5. *VP Requirements Spec Constraints*

Overall constraints of VP Requirements Spec are the constraints $C : M_1 \times \cdots \times M_k \to \{T,F\}$ defined by: $C(m_1,\dots,m_n) = MC(m_1,\dots,m_n) \wedge OC(objectives(m_1,\dots,m_n))$

Note also that VP Requirements spec defines a set of all feasible metric vectors R:

$$R = \{(m_1,\dots,m_n)|(m_1,\dots,m_n) \in M_1 \times \cdots \times M_n \wedge C(m_1,\dots,m_n)\}$$

To search for a VP in the marketplace, the user specifies the VP Requirements Spec (RS), and the system checks if any VP consumer-facing Spec (CFS) matches the RS.

Definition 3.6. Search for VP

- Let
$$VP =< domain, parametersSchema, metricSchema, template, analyticModel, feasibilityMetric >$$
be a VP design spec. Out of parameter vector $(p_1,\dots,p_l,\dots,p_n)$ in parametersSchema, let $DV = (p_1,\dots,p_l)$ be designated as decision variables (without loss of generality). We use the term fixed parameters for the vector of remaining parameters $FP = (p_{(l+1)},\dots,p_n)$.
- Let $CFS =< domain, metricSchema, MFC >$ be a VP (consumer-facing) Spec
- Let
$$RS =< domain, metricSchema, objectiveSchema, objectives, MC, OC >$$
be a VP Requirements spec
- Let $U : O_1 \times \cdots \times O_l \to \mathbb{R}$ be a utility function

We say that:

- RS and CFS match (or CFS is feasible w.r.t. RS, or CFS and RS are mutually consistent) if
  1. they have identical domains and metric schema $(m_1,M_1),\dots,(m_k,M_k)$
  2. their joint constraint $C(m_1,\dots,m_k) = MC(m_1,\dots,m_n)$ and $OC(objectives(m_1,\dots,m_n))$ and $MFC(m_1,\dots,m_k)$ are satisfiable
- A metrics vector $(m_1^*,\dots,m_k^*) \in M_1 \times \cdots \times M_k$ is Pareto-optimal w.r.t. to RS and CFS if:
  1. the joint constraint $C(m_1^*,\dots,m_k^*)$ holds
  2. there does not exist a metrics vector $(m_1',\dots,m_k')$ that
     - satisfies the joint constraint C
     - for objective vectors $(o_1^*,\dots,o_l^*) = objective(m_1^*,\dots,m_k^*)$ and $(o_1',\dots,o_l') = objective(m_1',\dots,m_k')$, $(\forall i = 1,\dots,l, o_i' >= o_i^*)$ and $(\exists j, 1 <= j <= l)$ such that $o_j' > o_j^*$
- A metrics vector $(m_1^*,\dots,m_k^*) \in M_1 \times \cdots \times M_k$ is optimal w.r.t. RS, CFS and U if:
$$(m_1^*,\dots,m_k^*) \in argmax(m_1,\dots,m_k) \in M_1 \times \cdots \times M_k(U(objective(m_1,\dots,m_k)))$$
subject to $C(m_1,\dots,m_k)$
- Given a vector $FP = (v_{(l+1)},\dots,v_n)$ of fixed parameters, parameter vector $(p_1^*,\dots,p_n^*) \in P_1 \times \cdots \times P_n$ and the corresponding product $p = template(p_1^*,\dots,p_n^*)$ are Pareto-optimal w.r.t. RS, VP and FP if:

1. the constraint $C(p_1^*,\dots,p_n^*) = MFC(AM(p_1^*,\dots,p_n^*))$ and $\forall j = l+1,\dots,n$, $p_j^* = p_j$
2. there does not exist a parameter vector $(p_1',\dots,p_n')$ that
   - satisfies the constraint C, and
   - for objective vectors $(o_1^*,\dots,o_l^*) = objective(AM(p_1^*,\dots,p_n^*))$ and $(o_1',\dots,o_l') = objective(AM(p_1',\dots,p_n'))$, $(\forall i = 1,\dots,l) o_i' >= o_i^*)$ and $(\exists j, 1 <= j <= l)$ such that $o_j' > o_j^*$

- Given a vector $FP = (v_{(l+1)},\dots,v_n)$ of fixed parameters, a parameter vector $(p_1^*,\dots,p_n^*) \in P_1 \times \cdots \times P_n$ and the corresponding product $p = template(p_1^*,\dots,p_n^*)$ are optimal w.r.t. RS, VP, U and FP if:
  - $(p_1^*,\dots,p_n^*) \in argmax(p_1,\dots,p_n) \in P_1 \times \cdots \times P_n(U(objective(AM(p_1,\dots,p_n)))$ subject to $C(AM(p_1,\dots,p_n))$ and $\forall j = l+1,\dots,n, p_j^* = p_j$

Claim (follows directly from definitions): Given CFS derived from VP, RS, U and let $(m_1^*,\dots,m_k^*) = AM(p_1^*,\dots,p_n^*)$. Then:

- $(p_1^*,\dots,p_n^*)$ is Pareto-optimal w.r.t. RS and $VP \iff (m_1^*,\dots,m_k^*)$ are Pareto-optimal w.r.t. RS and CFS
- $(p_1^*,\dots,p_n^*)$ is optimal w.r.t. RS, VP and $U \iff (m_1^*,\dots,m_k^*)$ is optimal w.r.t. RS, CFS and U

3.3 Virtual Services

Intuitively, a virtual service is a parameterized transformation function of zero or more virtual things into zero or more virtual things. Virtual service is also associated with an analytic model, which gives metrics of interest and the feasibility Boolean value, for every instantiation of service parameters, which include parameters of input and output virtual things, as well as additional internal parameters.

For example, a supply service transforms zero input virtual things into output things that correspond to a planned purchase (bike supply). Its metrics may include total cost and delivery time, as a function of parameters that capture catalog prices and ordered quantities. Or, a manufacturing service transforms raw materials and parts (virtual things) into products (virtual things), like in the case of assembling bike parts into a bike. Or, a CNC machining service transforms an (virtual) input metal part into a processed (virtual) part, after a series of drilling and milling operations (turning steel material into a bike frame). Or, a transportation service transforms (virtual) items at some locations, into the same (virtual) items at some other locations (shipping a bike from a factory to a customer). Some services are associated with real-world brick-and-mortar services; some may be defined by a service network which involves sub-services. The following definitions formalize these concepts.

Definition 3.7. A virtual service or VS is a tuple

$$VS = (input, output, internalParametersSchema, internalMetricSchema, analyticModel, feasMetric)$$

where

1. input - is a set $\{VT_i | i \in I\}$ of input virtual things, where I is an input index set
2. $output$ - is a set $\{VT_i | i \in O\}$ of output virtual things, where O is an output index set
3. $internalParametersSchema$ - is a set
   $$IPS = \{(p_1,D_1),\dots,(p_n,D_n)\}$$
   of pairs, where for $i = 1,\dots,n$
   (a) p_i - a parameter name (e.g., quantities of input or output things, prices, coefficients in physics-based equations, control parameters of equipment)
   (b) Di - the domain of Pi
4. $internalMetricSchema$ - is a set
   $$IMS = \{(m_1,M_1),\dots,(m_k,M_k)\}$$
   of pairs, where for $i = 1,\dots,k$
   (a) m_i - a metric name (e.g., cost, delivery time, profit, carbon emissions)
   (b) M_i - the domain of m_i
   Let the parametersSchema
   $$PS = \{(v_1,V_1),\dots,(v_s,V_s)\}$$
   be the union of *parametersSchema* of input virtual products, output virtual products and $internalParametersSchemas$.
   Let the set
   $$MS = \{(mm_1,MM_1),\dots,(mm_r,MM_r)\}$$
   be the union of *metricSchemas* of input virtual things, output virtual things, and $internalMetricSchema$.
5. $analyticModel$ - is a function
   $$AM : V_1 \times \cdots \times V_s \to MM_1 \times \cdots \times MM_r$$
   which gives, for parameter instantiation $(v_1,\dots,v_s) \in V_1 \times \cdots \times V_s$, a vector $(mm_1,\dots,mm_r)$, where $(mm_1,\dots,mm_r)$ are metric values in $MM_1 \times \cdots \times MM_r$. We will denote resulting metric value $mm_i, i = 1,\dots,r$, using the (.) dot notation as $AM(v_1,\dots,v_s).m_i$ or $m_i(v_1,\dots,v_s)$

6. $feasMetric$ - is a Boolean metric name $C \in mm_1, \ldots, mm_r$ such that its corresponding domain $D = \{T, F\} \in MS$ (T to indicate that parameter instantiation is feasible, and F otherwise)

3.4 Specs of Virtual Services

Definition 3.8. *VS Consumer Spec* (projection on metric space of VS) that only reveals partially to the consumers is a tuple

$$< input, output, internalMetricSchema, MC >$$

where

- $input$ - is a set of VP (consumer-facing) specs $< domain, metricSchema, MFC >$
- $output$ - is a set of VP (consumer-facing) specs $< domain, metricSchema, MFC >$
- $internalMetricSchema$ is a set
$$IMS = \{(m_1, M_1), \ldots, (m_k, M_k)\}$$
of pairs, where for $i = 1, \ldots, k$
  1. m_i - a metric name (e.g., cost, delivery time, profit, carbon emissions)
  2. M_i - the domain of m_i

  Let metric schema
$$MS = \{(mm_1, MM_1), \ldots, (mm_r, MM_r)\}$$
be the union of *metricSchemas* of input virtual things, output virtual things, and $internalMetricSchema$.
- $MC : MM_1 \times \cdots \times MM_r \to \{T, F\}$ which tells, for every vector of metrics in $MM_1 \times \cdots \times MM_r$, whether it is feasible or not.

Definition 3.9. *VS Requirement Feasibility Spec* is a tuple

$$< input, output, internalMetricSchema, \\ objectiveSchema, objectives, MC, OC >$$

where

- $input$ is a set of (input) VP requirement specs
- $output$ is a set of (output) VP requirement specs
- $internalMetricSchema$ is a set
$$IMS = \{(m_1, M_1), \ldots, (m_k, M_k)\}$$
of pairs, where for $i = 1, \ldots, k$
  1. m_i - a metric name (e.g., cost, delivery time, profit, carbon emissions)
  2. M_i - the domain of m

  Let metric schema
$$MS = \{(mm_1, MM_1), \ldots, (mm_r, MM_r)\}$$
be the union of *metricSchemas* of input virtual things, output virtual things, and $internalMetricSchema$.
- $objectiveSchema$ is a set
$$OS = (o_1, O_1), \ldots, (o_l, O_l)$$
of pairs, where for $i = 1, \ldots, l$
  1. o_i - a unique objective name (e.g., cost, profit, time, carbon emissions)
  2. O_i - the domain of o_i
- $objectives : MM_1 \times \cdots \times MM_r \to O_1 \times \cdots \times O_l$ defines objectives as a function of metrics, i.e., $objectives(mm_1, \ldots, mm_r)$ is a vector of objective values $(o_1, \ldots, o_l) \in O_1 \times \cdots \times O_l$ associated with metrics $(mm_1, \ldots, mm_r)$
- $MC : MM_1 \times \cdots \times MM_r \to \{T, F\}$
- $OC : O_1 \times \cdots \times O_l \to \{T, F\}$

Definition 3.10. *VS Requirements Spec Constraints*

Overall constraints of VS Requirements Spec are the constraints

$$C : MM_1 \times \cdots \times MM_r \to \{T, F\}$$

defined by:

$$(mm_1, \ldots, mm_r) = MC(mm_1, \ldots, mm_r) \wedge \\ OC(objectives(mm_1, \ldots, mm_r))$$

Note also that VS Requirements Spec defines a set of all feasible metric vectors R:

$$R = \{(mm_1, \ldots, mm_r) | (mm_1, \ldots, mm_r) \in \\ MM_1 \times \cdots \times MM_r \wedge C(mm_1, \ldots, mm_r)\}$$

Definition 3.11. Search for VS

- Let
$$VS = (input, output, internalParametersSchema, \\ internalMetricSchema, analyticModel, feasMetric)$$
be a virtual service design spec.
- Let
$$PS = \{(v_1, V_1), \ldots, (v_s, V_s)\}$$
be the parameter schema of VS.
- Let
$$MS = \{(mm_1, MM_1), \ldots, (mm_r, MM_r)\}$$
be the metric schema of VS. Out of VS parameter vector $(v1, \ldots, v_h, \ldots, v_s)$ in internalParametersSchema, let $DV = (v_1, \ldots, v_h)$ be designated as decision variables (without loss of generality). We use the term fixed parameters for the vector of remaining parameters $FP = (v_{(l+1)}, \ldots, v_s)$.

- Let
$$CFS =< input, output, internalMetricSchema, MC >$$
be a virtual service consumer-facing spec.
- Let
$$MS = \{(mm_1, MM_1), \ldots, (mm_r, MM_r)\}$$
be the metric schema of CFS.
- Let
$$RS =< input, output, internalMetricSchema, objectiveSchema, objectives, MC, OC >$$
be a virtual service requirements spec.
- Let
$$MS = \{(mm_1, MM_1), \ldots, (mm_r, MM_r)\}$$
be the metric schema of RS.
- Let $U : O_1 \times \cdots \times O_l \to R$ (the set of reals) be a utility function, where $O_1, \ldots, O_n$ are the domains from the objective schema
$$OC = \{(o_1, O_1), \ldots, (o_l, O_l)\}$$

We say that:

- RS and CFS match (or CFS is feasible w.r.t. RS, or CFS and RS are mutually consistent) if
  1. they have the same input, output and metric schemata $\{(mm_1, MM_1), \ldots, (mm_r, MM_r)\}$
  2. the joint constraint $CC(mm_1, \ldots, mm_r)$ defined as the conjunction of $C(mm_1, \ldots, mm_r)$ and $MC(mm_1, \ldots, mm_r)$ is satisfiable, where $C(mm_1, \ldots, mm_r)$ is the overall constraint of the VS Requirements Spec, and $MC(mm_1, \ldots, mm_r)$ is the metric constraint of the VS consumer-facing spec
- A metrics vector $(mm_1^*, \ldots, mm_r^*) \in MM_1 \times \cdots \times MM_r$ is Pareto-optimal w.r.t. to RS and CFS if:
  1. the joint constraint $CC(mm_1^*, \ldots, mm_r^*)$ holds
  2. there does not exist a metrics vector $(mm_1', \ldots, mm_r')$ that
     - satisfies the joint constraint CC
     - for objective vectors $(o_1^*, \ldots, o_l^*) = objective(m_1^*, \ldots, m_k^*)$ and $(o_1', \ldots, o_l') = objective(m_1', \ldots, m_k')$ the following holds: $(\forall i = 1, \ldots, l, o_i' >= o_i^*)$ and $(\exists j, 1 <= j <= l)$ such that $o_j' > o_j^*$
- A metrics vector $(mm_1^*, \ldots, mm_r^*) \in MM_1 \times \cdots \times MM_r$ is optimal w.r.t. RS, CFS and U if: $(mm_1^*, \ldots, mm_r^*) \in argmax(mm_1, \ldots, mm_r) \in MM_1 \times \cdots \times MM_r(U(objective(mm_1, \ldots, mm_r)))$ subject to $CC(mm_1, \ldots, mm_r)$
- Given a vector $FP = (v_{(h+1)}, \ldots, v_s)$ of fixed parameters, parameter vector $(v_1^*, \ldots, v_s^*) \in V_1 \times \cdots \times V_s$ (and the corresponding product $p = template(v_1^*, \ldots, v_s^*)$) are Pareto-optimal w.r.t. VS, RS and FP if:
  - the following constraint is satisfied:
  $$CC(v_1^*, \ldots, v_h^*, \ldots, v_s^*) = MC(AM(v_1^*, \ldots, v_s^*))$$
  $\wedge(\forall j = h+1, \ldots, s, v_j^* = v_j)$ $\wedge C(v_1^*, \ldots, v_s^*)$, where MC is the metric constraint from RS, and C is the feasibility metric from VS.
  - there does not exist a parameter vector $(v_1', \ldots, v_s')$ that
    * satisfies the constraint CC, and
    * for objective vectors $(o_1^*, \ldots, o_l^*) = objective(AM(p_1^*, \ldots, p_n^*))$ and $(o_1', \ldots, o_l') = objective(AM(v_1', \ldots, v_s'))$, $(\forall i = 1, \ldots, l, o_i' >= o_i^*)$ and $(\exists j, 1 <= j <= l)$ such that $o_j' > o_j^*$
- Given a vector $FP = (v_{(h+1)}, \ldots, v_s)$ of fixed parameters, parameter vector $(v_1^*, \ldots, v_s^*) \in V_1 \times \cdots \times V_s$ and the corresponding product $p = template(v_1^*, \ldots, v_s^*)$ are optimal w.r.t. VS, RS, U and FP if:
  - $(v_1^*, \ldots, v_h^*, v_{(h+1)}^* \ldots, v_s^*) \in argmax(v_1^*, \ldots, v_s^*) \in V1 \times \cdots \times Vs$ $(U(objective(AM(v_1^*, \ldots, v_s^*))$ subject to $CC(v_1^*, \ldots, v_h^*, \ldots, v_s^*) = MC(AM(v_1^*, \ldots, v_s^*))$ and $(\forall j = h+1, \ldots, s, v_j^* = v_j)$ and $C(v_1^*, \ldots, v_s^*)$, where MC is the metric constraint from RS, and C is the feasibility metric from VS.

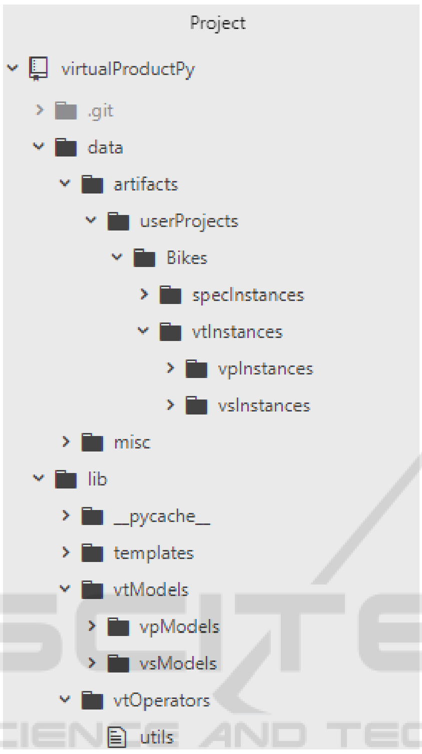
Claim (follows directly from definitions): Given CFS derived from VP, RS, U and let $(mm_1^*, \ldots, mm_r^*) = AM(v_1^*, \ldots, v_s^*)$, then:

- $(v_1^*, \ldots, v_s^*)$ is Pareto-optimal w.r.t. VS, RS, and FP if and only if $(mm_1^*, \ldots, mm_r^*)$ are Pareto-optimal wr.t. RS and CFS
- $(v_1^*, \ldots, v_s^*)$ is optimal w.r.t. VS, RS, FP and U if and only if $(mm_1^*, \ldots, mm_r^*)$ is optimal wr.t. RS, CFS and U

4 V-THING REPOSITORY

4.1 Architecture Overview

To enable rapid and convenient creation of V-things, we put the definitions into a more recognizable form of software implementation of the V-things marketplace. The challenge is how to organize the repository so that the artifacts can be stored hierarchically and connected seamlessly to the conceptual schema. In this section we show the organization of the fundamental artifacts (virtual products, virtual services, specs, analytical models, utility functions, etc.) and we demonstrate the composition of a V-thing with the example of a virtual bike product.



```
Project
v virtualProductPy
    > .git
    v data
        v artifacts
            v userProjects
                v Bikes
                    > specInstances
                    v vtInstances
                        > vpInstances
                        > vsInstances
        > misc
    v lib
        > __pycache__
        > templates
        v vtModels
            > vpModels
            > vsModels
        v vtOperators
            utils
```

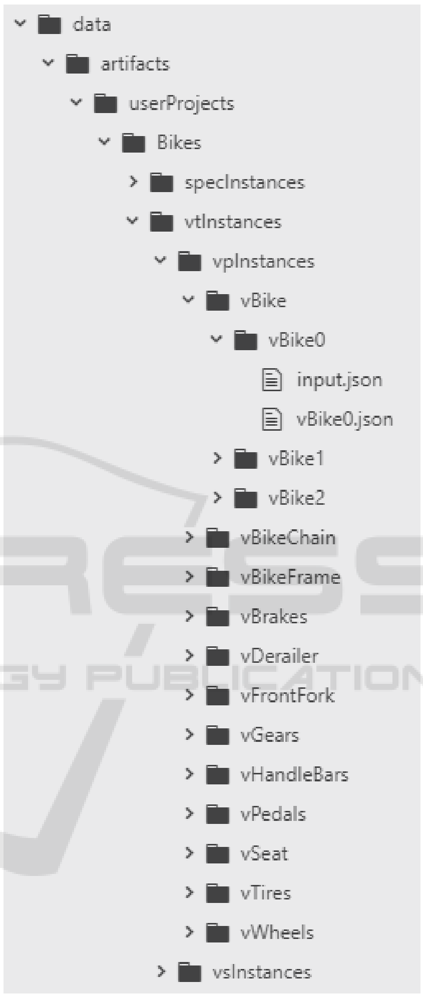
Figure 2: The Repository Design.

We create a software project and separate the abstractions of data objects and of model objects in the repository (shown in Figure 2). On the high level, the data folder contains the user-specified artifacts, including the instances of V-things, and the spec instances. These are all organized under the Bikes project associated with the user. In the lib folder we put the more functional elements including the templates, the analytical models and the V-thing operators which can be readily reused.

Each artifact is stored in its own file in the system, and has its unique file identifier in the entire system. The identifier consists of the directory path from the root to the folder containing the file plus the file name. To achieve referential integrity in the file system, it necessities that in the local environment, say, within the same folder, each file and folder must have a unique name, which is ensured by most of the modern operating systems.

4.2 Artifacts in the Repository

We view all the virtual things as the digital instances in the form of parametric data. For broad compatibility, the data is stored as JSON objects in the form of JSON files. And we differentiate the access rights to the data based on the types of the instances or the specific files which the user groups have interests in.



```
v data
    v artifacts
        v userProjects
            v Bikes
                > specInstances
                v vtInstances
                    v vpInstances
                        v vBike
                            v vBike0
                                input.json
                                vBike0.json
                            > vBike1
                            > vBike2
                        > vBikeChain
                        > vBikeFrame
                        > vBrakes
                        > vDerailer
                        > vFrontFork
                        > vGears
                        > vHandleBars
                        > vPedals
                        > vSeat
                        > vTires
                        > vWheels
                    > vsInstances
```

Figure 3: The Data Artifacts.

Under the userProjects directory, the users can initiate their virtual thing project ad hoc either by creating from scratch or by importing from a different source. A user can have arbitrarily many projects of interest created or imported. In our example, we have created the Bikes project (Figure 3).

For the instance data associated with the virtual bike product, there is a distinction between the V-thing objects and their specs. While the specInstances folder hosts all the specs (CFS, VP Spec, VS Spec, RS, RS Constraints, etc.) for the V-thing, the vtInstances folder hosts the actual instances of the V-things in the vpInstances folder for virtual products and the vsInstances folder for virtual services respectively.

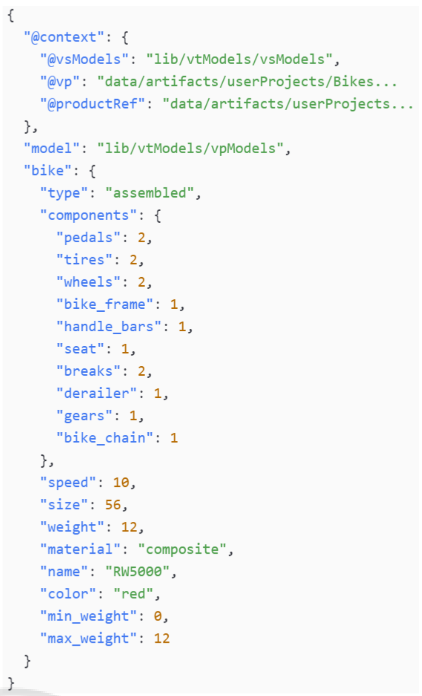
4.2.1 VP Instances

Specifically for a virtual product, vBike for example, it can have zero, one or multiple instances depending on how the user wants to instantiate the virtual product. The vpInstances folder holds the instances for the bike parts which are virtual products as well.

A virtual product instance is described by a JSON file named by the virtual product plus an ID. Each file contains one JSON object according to its schema in the VP Design Spec. A example of the data in vBike0.json file is shown in Figure 4.

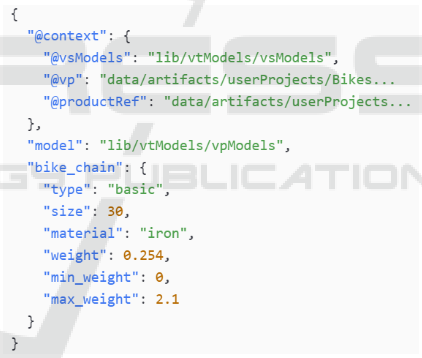
For each virtual product instance file, it can have an optional context header, marked by "@context" annotation. Each entry within the context object specifies a shortcut for a directory path which we call a context. Whenever the program processing data for a virtual product, it would recognize and convert the "@" annotated shortcut to the full path. The "model" entry links the data to the analytical model. And the "bike" sub-object contains the parametric data for the virtual product. Important information as whether it is an atomic product or a composite product is recorded. And if it is a composite product, which means it has composed of other virtual products, the components are also recorded. And key metrics information for the bike virtual product is attached.

Another example is shown below for an atomic virtual product in Figure 5. The vBikeChain0.json file contains the data of a virtual bike chain product. Instead of being composed by other products, it is an atomic product, which can be used in a stand-alone manner or as a component for other virtual products and services.



```
{
  "@context": {
    "@vsModels": "lib/vtModels/vsModels",
    "@vp": "data/artifacts/userProjects/Bikes...
    "@productRef": "data/artifacts/userProjects...
  },
  "model": "lib/vtModels/vpModels",
  "bike": {
    "type": "assembled",
    "components": {
      "pedals": 2,
      "tires": 2,
      "wheels": 2,
      "bike_frame": 1,
      "handle_bars": 1,
      "seat": 1,
      "breaks": 2,
      "derailer": 1,
      "gears": 1,
      "bike_chain": 1
    },
    "speed": 10,
    "size": 56,
    "weight": 12,
    "material": "composite",
    "name": "RW5000",
    "color": "red",
    "min_weight": 0,
    "max_weight": 12
  }
}
```

Figure 4: Composite VP Instance.



```
{
  "@context": {
    "@vsModels": "lib/vtModels/vsModels",
    "@vp": "data/artifacts/userProjects/Bikes...
    "@productRef": "data/artifacts/userProjects...
  },
  "model": "lib/vtModels/vpModels",
  "bike_chain": {
    "type": "basic",
    "size": 30,
    "material": "iron",
    "weight": 0.254,
    "min_weight": 0,
    "max_weight": 2.1
  }
}
```

Figure 5: Atomic VP Instance.

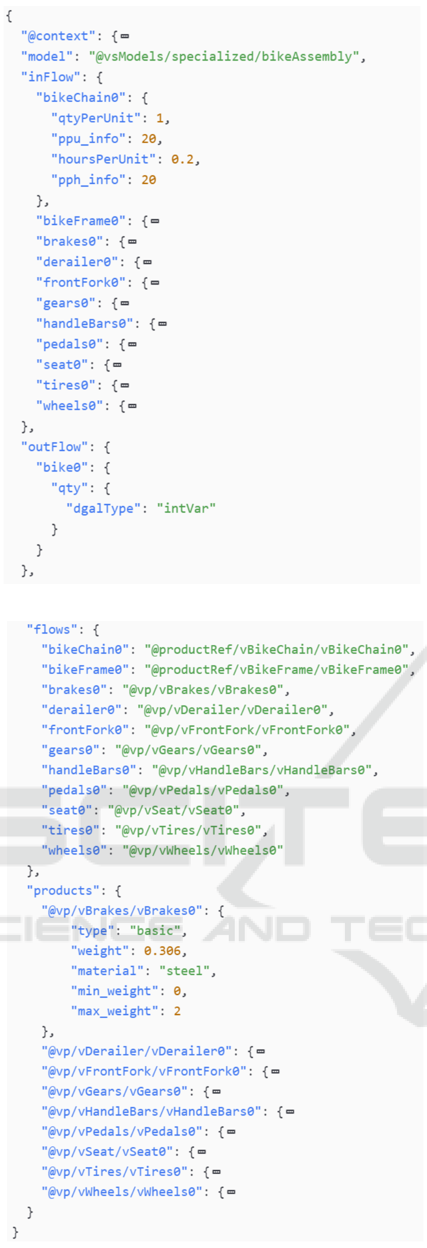
4.2.2 VS Instances

For virtual services, the data is organized differently than that of the virtual products. As the virtual services describe the flow of materials and labor, they are more process-based. While the users are interested in the metrics of a service, the essential information for the processes should also be captured by the virtual services. An example of a bike assembly service is shown in file bikeAssembly0.json (Figure 6).

The information about the flows is reflected in the schematic sections of inflow, outflow, flows and products respectively. The inflows direct what products and materials and how they are passed into the service. The outflows are the virtual products that will be rendered by the virtual service, and there can be none. The flows summarise how each product plays a part role in comprising the final outflow product. And the products section records the relevant parametric data of the inflow products that will be used for processing and analytical purposes in the service.

4.3 V-Thing Creation

While an atomic V-thing can be created from scratch with the parameters schema, a composite V-thing may have parts in it that are also V-things. A composite V-thing may already contain all the data from its sub-components, but if not, that means, some of the data is stored in the V-thing files that may not have been created, and thus instantiation is needed. References are used as placeholders to point each missing part



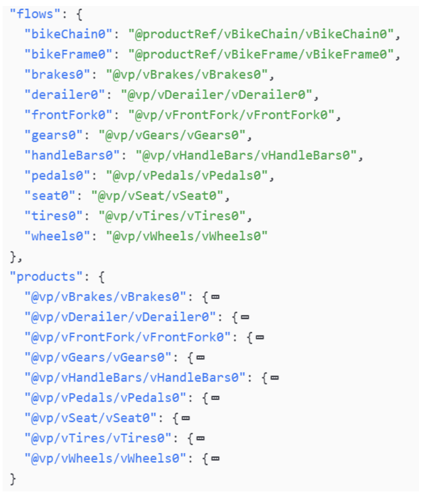
```
{
  "@context": {▭
  "model": "@vsModels/specialized/bikeAssembly",
  "inFlow": {
    "bikeChain0": {
      "qtyPerUnit": 1,
      "ppu_info": 20,
      "hoursPerUnit": 0.2,
      "pph_info": 20
    },
    "bikeFrame0": {▭
    "brakes0": {▭
    "derailer0": {▭
    "frontFork0": {▭
    "gears0": {▭
    "handleBars0": {▭
    "pedals0": {▭
    "seat0": {▭
    "tires0": {▭
    "wheels0": {▭
  },
  "outFlow": {
    "bike0": {
      "qty": {
        "dgalType": "intVar"
      }
    }
  },
```

```
  "flows": {
    "bikeChain0": "@productRef/vBikeChain/vBikeChain0",
    "bikeFrame0": "@productRef/vBikeFrame/vBikeFrame0",
    "brakes0": "@vp/vBrakes/vBrakes0",
    "derailer0": "@vp/vDerailer/vDerailer0",
    "frontFork0": "@vp/vFrontFork/vFrontFork0",
    "gears0": "@vp/vGears/vGears0",
    "handleBars0": "@vp/vHandleBars/vHandleBars0",
    "pedals0": "@vp/vPedals/vPedals0",
    "seat0": "@vp/vSeat/vSeat0",
    "tires0": "@vp/vTires/vTires0",
    "wheels0": "@vp/vWheels/vWheels0"
  },
  "products": {
    "@vp/vBrakes/vBrakes0": {
        "type": "basic",
        "weight": 0.306,
        "material": "steel",
        "min_weight": 0,
        "max_weight": 2
    },
    "@vp/vDerailer/vDerailer0": {▭
    "@vp/vFrontFork/vFrontFork0": {▭
    "@vp/vGears/vGears0": {▭
    "@vp/vHandleBars/vHandleBars0": {▭
    "@vp/vPedals/vPedals0": {▭
    "@vp/vSeat/vSeat0": {▭
    "@vp/vTires/vTires0": {▭
    "@vp/vWheels/vWheels0": {▭
  }
}
```

Figure 6: Bike Assembly VS Instance.

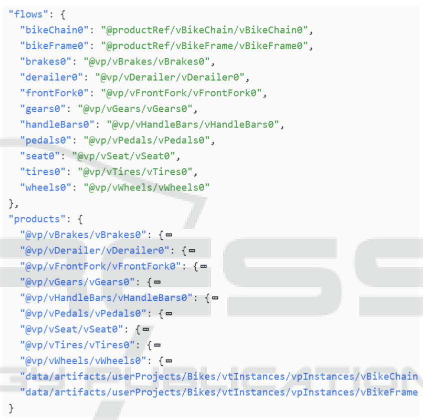
to a "yet-to-exist" V-thing which we call a partial V-thing. In that case, the entrepreneur should solicit in the V-thing Market for the missing parts with a spec. Once a V-thing is provided in accordance to the spec, an instantiation function will be applied that pulls in the data from the component V-thing file, replaces the reference, and renders a full data file for the composite V-thing.

For example, a virtual bike assembly service will create a virtual bike product out of its parts (bike chain, bike frame, brakes, derailers, front fork, gears,



```
"flows": {
  "bikeChain0": "@productRef/vBikeChain/vBikeChain0",
  "bikeFrame0": "@productRef/vBikeFrame/vBikeFrame0",
  "brakes0": "@vp/vBrakes/vBrakes0",
  "derailer0": "@vp/vDerailer/vDerailer0",
  "frontFork0": "@vp/vFrontFork/vFrontFork0",
  "gears0": "@vp/vGears/vGears0",
  "handleBars0": "@vp/vHandleBars/vHandleBars0",
  "pedals0": "@vp/vPedals/vPedals0",
  "seat0": "@vp/vSeat/vSeat0",
  "tires0": "@vp/vTires/vTires0",
  "wheels0": "@vp/vWheels/vWheels0"
},
"products": {
  "@vp/vBrakes/vBrakes0": {▭
  "@vp/vDerailer/vDerailer0": {▭
  "@vp/vFrontFork/vFrontFork0": {▭
  "@vp/vGears/vGears0": {▭
  "@vp/vHandleBars/vHandleBars0": {▭
  "@vp/vPedals/vPedals0": {▭
  "@vp/vSeat/vSeat0": {▭
  "@vp/vTires/vTires0": {▭
  "@vp/vWheels/vWheels0": {▭
}
```

Figure 7: Partial V-Thing.



```
"flows": {
  "bikeChain0": "@productRef/vBikeChain/vBikeChain0",
  "bikeFrame0": "@productRef/vBikeFrame/vBikeFrame0",
  "brakes0": "@vp/vBrakes/vBrakes0",
  "derailer0": "@vp/vDerailer/vDerailer0",
  "frontFork0": "@vp/vFrontFork/vFrontFork0",
  "gears0": "@vp/vGears/vGears0",
  "handleBars0": "@vp/vHandleBars/vHandleBars0",
  "pedals0": "@vp/vPedals/vPedals0",
  "seat0": "@vp/vSeat/vSeat0",
  "tires0": "@vp/vTires/vTires0",
  "wheels0": "@vp/vWheels/vWheels0"
},
"products": {
  "@vp/vBrakes/vBrakes0": {▭
  "@vp/vDerailer/vDerailer0": {▭
  "@vp/vFrontFork/vFrontFork0": {▭
  "@vp/vGears/vGears0": {▭
  "@vp/vHandleBars/vHandleBars0": {▭
  "@vp/vPedals/vPedals0": {▭
  "@vp/vSeat/vSeat0": {▭
  "@vp/vTires/vTires0": {▭
  "@vp/vWheels/vWheels0": {▭
  "data/artifacts/userProjects/Bikes/vtInstances/vpInstances/vBikeChain
  "data/artifacts/userProjects/Bikes/vtInstances/vpInstances/vBikeFrame
}
```

Figure 8: Instantiated V-Thing.

handle bars, pedals, seats, tires, and wheels). Among the parts, there is no suitable bike chain or bike frame virtual product that is readily available in the market. Then the entrepreneur lists her demand with the Requirements Specs for the bike chain and bike frame in the market, while she continues building the virtual bike product through the assembly service. The parts that await to be instantiated are marked with "@productRef" annotation indicating the data is not yet plugged in (under flows, bikeChain0 and bikeFrame0 have no correspondent entry under products as shown in Figure 7). Once the data is ready for the component V-things, the entrepreneur will go ahead and apply the instantiation function, and the full data file will be generated for subsequent processing. (V-thing data entries are attached under products for vBikeChain and vBikeFrame as shown in Figure 8).

4.4 Operators in the Repository

Operators are needed upon the repository in order to adequately manipulate the V-things. A V-thing can be created, updated, queried and deleted, and the operators should enable these operations. We implement the operators as utility functions and group them under the lib directory. When there is a need to conduct certain operations on the V-things data, the user can make use of the operators through the APIs or on the user interface to interact with the V-things artifacts in the repository.

In particular, we exemplify the instantiation operator which is of significance. The instantiation operator, or instantiator, is a function that recursively checks each component of a V-thing and instantiates with the referenced data, turning the partial V-thing into a full data file for further processing or analytical use. Please note that the instantiator should not modify the original input. The contract for the instantiator is as follows:

```
instantiator(input):
"""
Requires:   non-null non-empty V-thing input
Effects:    if input is an instantiated
            V-thing, return a copy of this;
            else recursively instantiate
            each referenced component
"""
```

5 CONCLUSIONS

We developed a formal mathematical framework for markets of virtual things or V-things: parameterized products and services that can be searched, composed and optimized. We also proposed a design of a repository of virtual things and their artifacts, to be used in support of the market. The proposed markets of virtual thing have the potential to make ideation-to-manufacturing process considerably more accessible, predictable and agile. This, in turn, can democratize innovation by allowing entrepreneurs without design and manufacturing expertise to bring their ideas to markets quickly.

Many research questions remain open, including how to develop a system for entrepreneurs to formulate their ideas in V-things, how to guide the users to make optimized decisions, and how effective the V-thing markets are in facilitating the ideation-to-production process.

REFERENCES

Brodsky, A., Gingold, Y. I., LaToza, T. D., Yu, L., and Han, X. (2021). Catalyzing the agility, accessibility, and predictability of the manufacturing-entrepreneurship ecosystem through design environments and markets for virtual things. In *Proceedings of the 10th International Conference on Operations Research and Enterprise Systems, ICORES 2021, Online Streaming, February 4-6, 2021*, pages 264–272. SCITEPRESS.

Brodsky, A., Krishnamoorthy, M., Nachawati, M. O., Bernstein, W. Z., and Menascé, D. A. (2017). Manufacturing and contract service networks: Composition, optimization and tradeoff analysis based on a reusable repository of performance models. In *2017 IEEE International Conference on Big Data (IEEE BigData 2017), Boston, MA, USA, December 11-14, 2017*, pages 1716–1725. IEEE Computer Society.

Brodsky, A. and Luo, J. (2015). Decision guidance analytics language (DGAL) - toward reusable knowledge base centric modeling. In *ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 1, Barcelona, Spain, 27-30 April, 2015*, pages 67–78. SciTePress.

Egge, N. E., Brodsky, A., and Griva, I. (2013). An efficient preprocessing algorithm to speed-up multistage production decision optimization problems. In *46th Hawaii International Conference on System Sciences, HICSS 2013, Wailea, HI, USA, January 7-10, 2013*, pages 1124–1133. IEEE Computer Society.

Gingold, Y. I., Igarashi, T., and Zorin, D. (2009). Structured annotations for 2d-to-3d modeling. *ACM Trans. Graph.*, 28(5):148.

LaToza, T. D., Shabani, E., and van der Hoek, A. (2013). A study of architectural decision practices. In *6th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2013, San Francisco, CA, USA, May 25, 2013*, pages 77–80. IEEE Computer Society.

Shao, G., Brodsky, A., and Miller, R. (2018). Modeling and optimization of manufacturing process performance using modelica graphical representation and process analytics formalism. *J. Intell. Manuf.*, 29(6):1287–1301.

Shin, S., Kim, D. B., Shao, G., Brodsky, A., and Lechevalier, D. (2017). Developing a decision support system for improving sustainability performance of manufacturing processes. *J. Intell. Manuf.*, 28(6):1421–1440.

Yu, L.-F., Yeung, S. K., Tang, C., Terzopoulos, D., Chan, T. F., and Osher, S. J. (2011). Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4):86.