An application of graph traversal algorithm to design task planning in model-based product development

Hitoshi KOMOTO *, Shinsuke KONDOH * and Keijiro MASUI*
*National Institute of Advanced Industrial Science and Technology
Namiki 1-2-1, Tsukuba 305-8564, Japan
E-mail: h.komoto@aist.go.jp

Revised 13 December 2015

Abstract
In model-based product development, engineering models representing specific parts a product from diverse aspects and resolutions are used to reduce cost and lead time while systematically reducing design alternatives. Designers in the product development processes will benefit from these models if they are timely available. In order to utilize these models collected from past development processes, we propose a product development process simulation method, which computationally supports planning of the current product development process. In the proposed method, these models are used to identify the types of design tasks and dynamically instantiate them so that they can satisfy given product specifications. The method is integrated with estimation of the degree of progress in product development by using a set-based approach. The simulation procedure performs iterative updates of the degree of progress with a traversal search on the graph, which is made by extracting the attributes and their relations defined in these models, and connecting these models at the attributes in common. The use of the proposed method is illustrated through its application to the development of a wind turbine.

Key words : Product development process modeling, System modeling, Graph traversal, Automation, Engineering design

1. Introduction

For the design and development of complex products like automobiles and medical instruments, designers and engineers perform various design tasks such as verification, validation, and testing (Shabi and Reich, 2012). For effective collaboration, knowledge from these aspects is codified and shared as models from various aspects (e.g., requirements, physical and logical models) (Tomiyama, et al., 2007). Model-based product development (MBPD) is a prescriptive framework promoting the development of these models, ultimately using them to more rapidly develop higher-quality products (Chandrasegaran et al., 2013).

Most of the existing literature on MBPD investigates the connectivity, traceability, and interoperability of product models from various aspects, or aspect models (e.g., (Tomiyama et al., 1989)). This type of study contributes to improving our understanding of product modeling for MBPD; however, few studies have been conducted on process modeling for MBPD, which involves predicting how effective product development processes are with the various aspect models available during product development (e.g., (Kondoh and Tezuka, 2014)).

Many approaches to product development process modeling have been studied to support managerial decisions (Smith and Morrow, 1999). The existing literature presents the advantages of process modeling by describing the specifics of process instances (e.g., concurrency and iteration) and identifying consistent and repeatable patterns that emerge in successive product development occurrences (Browning et al., 1996). The process models comprise a number of tasks (or activities) and their interdependencies, which are primitive model elements (e.g., (Eppinger et al., 1996)). Such primitive tasks are explicitly given for specific purposes like assembly planning (Lapperriere and ElMalaghy, 1992; Krause et al., 2004; Bley and Franke, 2004). However, in MBPD, the contexts of the usage of these models are diverse. For instance, an engineering model can be used for both design (model configuration to meet specifications) and analysis (simulation of the behavior of a model with given configuration against specifications). Furthermore, unlikely to
assembly planning of products with a fixed assembly structure, tasks in product development are flexibly formulated and instantiated dynamically (Giffin et al., 2009).

Here, we propose a method to simulate product development processes in MBPD, which computationally supports planning of product development processes with product models representing specific parts of the product from various aspects. In addition to the availability of these product models (e.g., mathematical formulae, simulation models, and experimental data) used in a product development process rather than explicit description of process model primitives with well-defined input and output information (e.g., calculation, simulation, and experiment), the authors made the following assumptions. First, these models are partly and explicitly described with the attributes (e.g., engineering parameters, characteristics of shapes and materials) of the product under development and the relations among the attributes. The assumption indicates a scope of the proposed method that it cannot use the other types of information defined in these models for product development process planning. Second, these models can be combined when they have common attributes. This assumption indicates these models can construct a large product model, which connects various parts across engineering domains. It also indicates that product development processes are planned considering dependency among product models in terms of the attributes in common. These are reasonable assumptions in the field of engineering design, because they are based on an essential abstraction of product models studied in engineering design literature (e.g., Metamodel in Tomiyama et al. 1989 and Yoshioka et al., 2004). While adopting these assumptions, the authors argue that graph traversal search (Skiena (2008)) is a reasonable choice as a basis of an algorithm for dynamic formation of design tasks as well as simulation product development progress considering the connectivity among product models available in product development.

In comparison with static methods for analyzing product development processes (e.g., (Shabi and Reich, 2012)), the proposed method aims to use simulations to elucidate the relation between the task type in product development and the quality of the product models employed for the tasks. This method is similar to the existing methods in literature on product development process analysis with quality confidence (e.g., (Clarkson and Hamilton, 2000)) and simulation (e.g., (Cho and Eppinger, 2001)). However, the proposed method regards product models, which represent specific parts and aspects of products and are available in prior to product development, as primitive objects that dynamically formulate the types of tasks during process simulation and generate the instances. Although such graph-structured models have been analyzed in various contexts of product development such as potential fault propagation (Kurtoglu and Tumer, 2008) and static formation of assembly processes (Lapperriere and ElMalaghy, 1992) and product development tasks (Cho and Eppinger, 2001), to the best of the authors’ knowledge, these models have not been analyzed regarding the capability to dynamically formulate and instantiate tasks according to the state of progress in product development.

Furthermore, the novelty of the proposed method resides in an indicator for quantifying the progress of product development processes in terms of a set of the ranges of design parameters refined in progress (relative to the ranges defined in the given specifications). The indicator is formulated based on a set-based approach, which has been applied to various activities in product development (e.g., conceptual design (Malak et al., 2009) and managerial decision making in concurrent engineering (Sobek et al., 1999)). With such an indicator, the degree of progress in product development is described by the probability of correctness of a set of design concepts regarding specific aspects (Hoyle et al., 2011).

Section 2 discusses the degree of progress in product development by using the set-based approach. Section 3 describes a method to construct a product development process model based on product models and run simulations with the model. Section 4 applies this method to the development of a wind turbine. Section 5 concludes the paper.

2. Set-based approach in product development

2.1 Representation of design concepts

The set-based approach in product development promotes the systematic generation of a design concept (e.g., a hybrid vehicle) by combining partial solutions from specific aspects (e.g., the chassis and speed control) and subsequently refining the overall concept by selecting and refining the partial solutions. Although known for its applications to concurrent engineering in the automobile industry (Sobek et al., 1999), the approach is also applied to various stages in product development like conceptual design (Malak et al., 2009).

The existing literature on the set-based approach explains it with set-based representations of design concepts (e.g., (Malak et al., 2009)). A design concept is defined by the value space of a set of product specifications and attributes.
Concrete implementations of the concept are defined by combinations of the specific values (or value ranges) assigned to each product attribute that jointly meet the given specifications. In the set-based approach, product models are regarded as descriptive (but not necessarily executable) relational knowledge about product attributes and specifications. Some product models describe relations between product attributes and specifications regarding value spaces, while others describe relations solely among product attributes, such as physical laws and phenomena.

2.2 Estimating progress in product development

In the set-based approach, design concepts are refined by refining individual product attributes in terms of value spaces. This study uses such value spaces to determine the progress of product development processes.

Focusing on a product attribute $i$, the degree of progress in product development $p_i$ is defined by the actual value range $x_i$ and the reference value range $\overline{x}_i$. The actual value range includes all values of attribute $i$ obtained from design alternatives considered by designers in product development. As a result of the generation and selection of such design alternatives, the actual value range is updated with product development progress. The target value range includes the target values of attribute $i$ that the designers define at a specific time of a product development process. For instance, in context of a development stage of home appliances, the maximum energy consumption of design alternatives is measured or estimated in product development (the actual value range), and it is compared with the targets specified at discrete time points (e.g., the date of design reviews) in order to evaluate the product development progress (the reference value range).

$$p_i = \frac{x_i \cap \overline{x}_i}{x_i} \quad (1)$$

The generic form of this formulation is the probability of correctness of a set of design concepts (Hoyle et al., 2011). Thus, the degree of progress $p_i$ is interpreted as the probability of the design concept satisfying the given targets defined at arbitral time points in product development (or specifications at the end of the product development) from the perspective of attribute $i$, or the quantified confidence of the design concept (Clarkson and Hamilton, 2000). Fig. 1(a) shows how $p_i$ varies with $x_i$ in product development with constant $\overline{x}_i$. $p_i$ converges to 1 when the actual value space is equivalent to the reference value space. Furthermore, $p_i$ is 0 when the two value spaces do not overlap.

Fig. 1 (a) Representation and computation of the degree of progress in product development (b) How to compute $p_A$ given $p_B$ and $p_C$
Product models play a role in the computation of the value space of one model element (i.e., an attribute or specification) on the basis of that of another element. To compute the degree of progress regarding an element \(i\) of the product model \(M\) based on the degree of progress of the other elements, the following multiplication relation is defined in a manner analogous to the joint probability of probabilistic variables:

\[
 p_i = \prod_{j \in M \setminus \#i} p_j
\]  

(2)

This relation assumes that the value spaces of all elements in a product model are mutually orthogonal (i.e., mutually independent), just as the joint probability of probabilistic variables is computable. Fig. 1(b) illustrates how to compute \(p_A\), given \(p_B\) and \(p_C\), where \(A, B,\) and \(C\) are related elements within the same product model. In this study, the assumption is employed for the sake of simplicity of the computation of the degree of product development process progress. In some situations, the degree of progress of individual product models are dependent one another. In this case, Equation 2 does not have a form of simple multiplication. However, Equation 2 can still deal with the connectivity among product models.

3. Simulation of product development processes

3.1 Integration of product models

In this paper, product models representing specific parts and aspects in product development are used to generate individual tasks and compose a complete product development process model. Examples of these product models are mathematical formulae of product attributes, simulation models from specific domains (e.g., CAD models and signal flow diagrams), sets of measurement data derived from experiments, evaluation models of products from users’ perspectives (for market surveys), and roadmaps of technology related to specific product attributes (e.g., the innovation of materials with advanced properties). In product development, specific relations between product attributes and specifications are extracted from these product models with diverse levels of abstraction. Abstract product models described beyond the scope of individual aspects are called Metamodel (Tomiyama et al., 1989; Yoshioka et al., 2004).

In order to computationally support planning of product development processes with these product models, these products models are combined at the attributes in common. In this study, it is formalized as an undirected graph whose nodes are classified as specifications (S), attributes (A), and relations (R). Assuming that each product model consists of a relation, the elements of the graph are product models consisting of a relation and its neighboring attributes and specifications. As shown in Section 3.2, the degree of progress is determined considering all specifications and attributes, and the model quality is assigned to all relations.

Fig. 2 shows the class diagram of product models in general, and an example of an integrated product model, which consists of eight product models (with unique relations). Connections between specifications and relations are one-to-one mappings, meaning one product model is prepared for the evaluation of one specification. Connections between relations and attributes are many-to-many mappings, allowing the inclusion of attributes in multiple product models and the existence of product models comprising multiple attributes. In order to distinguish the integrated product model from other product models, which constitute the integrated model, the paper calls these product models partial product models.

3.2 Product development process models and their tasks

Product development process models derived from an integrated product model comprise tasks, which are intervals defined by the timing of execution and termination. Tasks are classified into two types: model quality improvement (MQI) tasks and process progress (PP) tasks. MQI tasks are the design tasks that update mathematical formula, simulation models, experiment settings, each of which individually represents the relation in a partial product model used in product development processes. PP tasks are the design tasks that use mathematical formula, simulation model, and experiment setting for the computation of the value spaces (ranges) obtained from currently considered design alternatives using partial product models updated by MQI tasks.

An MQI task is denoted by MQI(R), which increases the quality of a partial product model, which is a part of the integrated product model, whose development plan is simulated with the proposed method. Formally, the quality of a partial product model is represented by a relation \(j\) at time \(t\) in product development \(Q_j(t, \tau)\) is defined as
where $f_j(t)$ is the reference model quality and $\tau$ is the most recent time at which MQI($j$) is executed. The reference model quality indicates the external progress of the quality of each partial product model. The MQI value range is between 0 and 1. The actual model quality $Q_j(t, \tau)$ is updated when the corresponding MQI tasks are executed in product development. Figure 3 illustrates the relation between $Q_j(t, \tau)$ and $f_j(t)$. The model quality is a score representing the degree of model detail (e.g., numerical precision) about relations among its related attributes. In product development, designers first employ a model with low quality. Then the quality is gradually increased (e.g., a FEM model) by executing MQI tasks. For instance, designers first employs a beam model for relating force with reflection. Then, they introduce an FEM model for computing the above relation more precisely. The FEM model is verified based on the numerical data obtained by experiments (i.e., MQI tasks).

A PP task updates the degree of progress in product development regarding a specification S or attribute A (the OUT node) in a partial product model represented by a relation R. The update is based on the degree of progress of other attributes (IN nodes) in the same partial product model. According to the type of OUT node, the task is denoted by either PP(S, R) or PP(A, R). Figure 4 shows possible partial patterns of the integrated product model regarding a PP task. Figure 4(a) shows a pattern without IN nodes, where the OUT node is independent because it is independent of other attributes.
Conversely, the OUT node attribute shown in Fig. 4(b) is dependent. Specifications are always OUT nodes of partial product models and depend on the attributes of these models (Fig. 4(c)).

In executing a PP task, the product development degree of progress of the OUT node $p_{\text{out}}(t)$ is updated. The updated value is the product of the degree of all IN nodes and the quality of the partial product model represented by relation $j$ used for the update. Equation 4 is used in cases with the pattern shown in Fig. 4(a), and Equation 5 with the patterns shown in Figs. 4(b) and (c).

$$p_{\text{out}}(t) = Q_j(t, \tau)$$  \hspace{1cm} (4)

$$p_{\text{out}}(t) = Q_j(t, \tau) \prod_{i \in \text{IN}} p_i(t)$$  \hspace{1cm} (5)

### 3.3 Simulation procedure

This section describes a procedure to compute the execution of the above MQI and PP tasks based on a given (integrated) product model containing multiple attribute relations. The procedure is interpreted as a series of dynamic interactions between the task generator and the task scheduler (Fig. 5). Briefly speaking, the task generator periodically looks at the state of the product model in terms of MQI and PP values, and generates appropriate tasks (which are classified into four types as shown below). The task scheduler is responsible for the execution and termination of the generated tasks and update the state of the product model.

![Fig. 5 Interaction between the task generator and scheduler](image)

The data prepared prior to executing the simulation procedure are as follows: (1) the target values of the degree of progress for all specifications (each denoted by $p^*_S$ in Table 1), (2) the cost and duration of tasks, and (3) the reference model quality $f_j(t)$ of all partial product models, while $Q_j(t, \tau)$ is computed based on the occurrences of MQI tasks during simulation. The initial value of $p_A$ is defined by evaluating the value of attribute A of the initial set of design alternatives. The target values $p^*_S$ at specific time points in product development are defined considering the expected distribution of the value of specification with reference to the initial set design alternatives.

After defining the abovementioned data, the simulator periodically triggers the task generator, which traverses the entire product model following a width-first search algorithm in Skiena (2008). Each search is initiated at a specification. The search path is decomposed into a number of directed edges, each of which is connected with one of the elements of the product model. As shown by the arrows in Fig. 5, edges are classified into four types (I–IV) based on the source and target node types. According to the edge types, the task generator formulates tasks to be initiated and selects nodes from the neighbor nodes to continue the search.

Table 1 shows the detailed procedure followed by the task generator depending on the edge type. When the search starts (type I), the actual degree of progress regarding the specification $p_S$ is compared with that of the target $p^*_S$. If $p_S$ is smaller than $p^*_S$, the search goes to the neighbor relation.

In the next step (type II), the generator evaluates whether the PP(S, R) task should be executed. If so, the degree of progress regarding the specification is computed using Equation 5. If the computed value $p^*_S$ is bigger than the actual value $p_S$, the generator schedules the task PP(S, R). If $p^*_S$ is smaller than the target $p^*_S$, the task MQI(R) is requested. Otherwise, the search continues to the neighbor attributes. As a result of the execution of the PP(S, R) task, it is allowed for $p^*_S$ to get larger than the target $p^*_S$ (but the maximum is 1). The PP(S, R) task may be executed again when the updated $p^*_S$ becomes bigger than $p^*_S$.

Table 1: Procedure followed by the task generator depending on the edge type.

| Edge Type | Task Description |
|-----------|------------------|
| I         | MQI(R)           |
| II        | PP(S, R)         |
| III       | MQI(R)           |
| IV        | MQI(R)           |

[DOI: 10.1299/jamdsm.2016jamdsm0092] © 2016 The Japan Society of Mechanical Engineers
As long as the search continues, sub-procedures corresponding to edge types III (R–A) and IV (A–R) are performed iteratively. In type III’s case, if the degree of progress can be improved (i.e., the computed value $p_A^*$ is smaller than 1), all relations connected to A are added to the search path. In type IV’s case, the model first checks whether the task PP(A, R) should be executed by computing $p_A^*$ by using Equation 4 (if A is independent) or 5 (if A is dependent). Then, the degree of progress can be improved with respect to A (i.e., $p_A < p_A^*$), and all attributes connected to R are added to the search path. The task MQI(R) is also requested for the increase of $p_A$. When $p_A^*$ becomes larger than 1, $p_A^*$ is set to 1 and no additional tasks will be executed regarding the attribute A.

In response to the task requests from the task generator, the task scheduler initiates the requested tasks by defining the execution and termination timing.

### Table 1. Sub-procedures of the task generator for all edge types

| Type     | Tasks requests and search continuation                                                                 |
|----------|-------------------------------------------------------------------------------------------------------|
| I: O to S| If $p_S < p_S^*$ then Nodes = [R] connected with S.                                                    |
| II: S to R| Compute $p_S^*$ (with Equation 5).                                                                  |
|          | If $p_S < p_S^*$ then request PP(S, R).                                                                |
|          | If $p_S^* < p_S^*$ then request MQI(R) AND Nodes = all As connected with R.                            |
| III: R to A| If $p_A < 1$ then Nodes = all Rs connected with A                                                    |
| IV: A to R| Compute $p_A^*$ (with Equation 4 or 5).                                                               |
|          | If $p_A < p_A^*$ then request PP(A, R).                                                                |
|          | If $p_A^* < 1$ then request MQI(R) AND Nodes = all As connected with R.                                |

S: Specification node, R: Relation node, A: Attribute node, O: None (indicating the root of the search), Nodes: Nodes added to the search path

### 4. An example

This section describes an example of the proposed process model simulation method in the context of developing a wind turbine. To clearly present the method within the limited space, an integrated product model (Fig. 6) is prepared comprising 21 partial product models, each corresponding to a relation (with connected specifications and attributes). These partial product models, in total, include 3 specifications and 15 attributes. For instance, a partial product model at the bottom left in Fig. 6 shows a partial product model about Betz’s law, which relates four attributes, Mechanical_Power, Radius (both of a Rotorblade), Speed, and Density (both of Wind). The entire product model was developed on Draw.io (Draw.io, 2014) and converted into a simulation model. The rest describes how to evaluate the degree of progress of an wind turbine development process regarding five attributes (colored ovals in Fig. 6) including Radius and Bladenumber of Rotorblade, compared with the target degree of progress corresponding to all the specifications like Electrical_Power, Cost, and Strength (colored parallelograms in Fig. 6).

#### 4.1 Preparation

Before executing the simulation, quantitative information was given to the model.

The target degree of progress regarding all the specifications was defined as a three-step function (e.g., Fig. 6(b)). Improvements to the partial product models’ quality (e.g., a computational fluid dynamics (CFD) model to predict wind speed from geometry) were considered external constraints (Fig. 6(c)). The hypothesized degree of progress regarding the above five attributes was modeled as the reference model quality of the five relations labeled as Selection (e.g., Fig. 6(d)).

Including the above mentioned information, other quantitative information supplied to the entire product model is shown in Tables 2 and 3. Table 2 shows execution costs, execution durations, and the reference model quality of each partial product model. Execution costs and execution durations of each partial product model are defined with regard to all the specifications and attributes connected with the corresponding relation. For instance, the execution duration time related with the partial product model about CFD is defined by the string “CFD : 2 ; Geometry : 1 ; Speed : 3”. This
means that it takes 2 time steps (weeks) to execute the MQI task to improve the CFD model itself, and it takes 1 or 3 weeks to execute a PP task compute the geometry or speed with the CFD model. Whether these tasks are executable or not depend on the magnitude relations between the actual and target values formulated in Table 1. The progress of the reference model quality of the CFD model shown in Fig. 6 (c) corresponds to the list of time-value pairs divided with semicolons “0 : 0 ; 20 : 0.4 ; 40 : 0.7 ; 60 : 0.9” shown as a list in Table 2.

Table 3 shows the target value of the degree of progress with respect to the specifications. The target degree of progress regarding the specification Strength shown in Fig. 6 (b) corresponds to the list divided with semicolons “0 : 0.2 ; 25 : 0.5 ; 50 : 0.7” shown in Table 3.

| ID | Name | Cost | Time | Reference model quality |
|----|------|------|------|-------------------------|
| 45 | Betz’s law | CFDb | 0.5 | Betz’s law:1:Density:1:Mod |
| 56 | CFD | CFDb:1:Geometry:1:Speed:1 | 0.2 | CFDb:2:Geometry:1:Speed:1 |
| 106 | Catalogue | Catalogue:1:Efficiency:1:8 | 0.1 | Catalogue:1:Efficiency:1:8 |
| 103 | Constant | Constant:1:Density:1 | 0 | Constant:1:Density:1 |
| 36 | Database | Database:1:Speed:1 | 0.6 | Database:1:Speed:1 |
| 48 | Evaluation | Evaluation:1:Efficiency:1:3 | 0.1 | Evaluation:1:Efficiency:1:3 |
| 95 | Evaluation | Evaluation:1:Bladenumber:1 | 0.1 | Evaluation:1:Bladenumber:1 |
| 90 | Experiment | Experiment:1:Angular_Vel | 0 | Experiment:1:Angular_Vel |
| 111 | Formula | Formula:1:Angular_Vel | 0 | Formula:1:Angular_Vel |
| 85 | InvProportional | InvProportional:1:Angular | 0 | InvProportional:1:Angular |
| 44 | InvSquareProportional | InvSquareProportional:1:Sp | 0 | InvSquareProportional:1:Sp |
| 60 | Measurement | Measurement:1:Height:1 | 0 | Measurement:1:Height:1 |
| 82 | Proportional | Proportional:1:Bladenumber:1 | 0 | Proportional:1:Bladenumber:1 |
| 79 | Proportional | Proportional:1:Bladenumber:1 | 0 | Proportional:1:Bladenumber:1 |
| 116 | Selection | Selection:1:Weight:1 | 0.5 | Selection:1:Weight:1 |
| 118 | Selection | Selection:1:Size:1 | 0.5 | Selection:1:Size:1 |
| 122 | Selection | Selection:1:Radius:1 | 0.5 | Selection:1:Radius:1 |
| 124 | Selection | Selection:1:Bladenumber:1 | 0.5 | Selection:1:Bladenumber:1 |
| 120 | Selection | Selection:1:Diameter:1 | 0.5 | Selection:1:Diameter:1 |
| 122 | Simulation | Simulation:1:Diameter:1 | 0.8 | Simulation:1:Diameter:1 |

Table 3. Target value of the degree of progress with respect to the specifications.

| ID | Name     | Target values |
|----|----------|---------------|
| 17 | Cost     | 0.225 | 0.5/0.7 |
| 9  | Electrical_Power | 0.225 | 0.5/0.7 |
| 11 | Strength | 0.225 | 0.5/0.7 |

4.2 Simulation results and interpretation

Figure 7 shows the simulation results regarding the occurrence and termination of instantiated tasks. They are vertically ordered (from top to down) according to the timing of instantiation. Those located at the same horizontal position are the same tasks instantiated at different timings. The results show some of the complex behaviors of the product development process as follows.

First, multiple partial product models with different quality were used for PP tasks regarding the same attribute. For instance, Speed of the wind was estimated with Database at the beginning of the process and was then estimated by CFD based on Geometry of the site (Fig. 7(a)). Similarly, Mechanical_Power was initially estimated using Betz’s law and subsequently experimentally verified using Angular_Velocity of the turbine obtained by the prototype Experiment (Fig. 7(b)).

Second, each partial product model was used for multiple tasks. For instance, an partial product model representing Formula relating Mechanical_Power to Angular_Velocity was used first to derive its Angular_Velocity based on the Mechanical_Power estimated using Betz’s law and then to estimate Mechanical_Power on the basis of Angular_Velocity (Fig. 7(c)). As these observation shows, the input and output of each task are defined by the attributes of the corresponding partial product model, and they are dynamically determined based on relative differences among the value of \( p_A \) of its attributes. In other words, these inputs and outputs are not a static property of the integrated product model shown in Fig.
6. Figure 8 shows the simulated degree of progress of the product development process regarding three specifications, whose related tasks are shown as labels with parallelograms in Fig. 7. The results show that the simulated progress is greater than the given target near $t = 70$ weeks, which indicates that the hypothetical plan of the degree of refinement of the value space regarding the selected five attributes is sufficient to meet the target. In particular, the target regarding Electrical Power was satisfied around $t = 55$ weeks (Fig. 7(d)), after which tasks to evaluate the degree of progress regarding Electric Power did not occur.

![Diagram of product model](attachment:product_model.png)

Fig. 6. The entire product model representing a wind turbine under development

5. Summary and conclusions

The study has proposed a method to generate and simulate tasks in product development in MBPD, where various product models representing specific parts, aspects, abstraction levels, and quality, are available. The method has dealt
with the degree of progress in product development by using the set-based approach. As an example, the development of a wind turbine was demonstrated using the proposed method. The simulation results are useful for understanding complex relations between tasks and product models with their attributes and specifications as intermediate concepts. Further studies on this method will focus on overcoming its limitations and extending its advantages. For example, the simple graph search employed in the simulation procedure is a limitation preventing the model from dealing with tasks with multiple inputs and outputs.

![Fig. 7 Simulation of the product development process model](image)

![Fig. 8 The degree of progress regarding the target specifications](image)
Acknowledgements

This research is partially supported by the Japanese Society for Promotion of Science (JSPS), Ministry of Education, Culture, Sports, Science and Technology, Japan, under a Grant-in-Aid for Scientific Research (No. 26820035) and by Council for Science, Technology and Innovation, Cross-Ministerial Strategic Innovation Promotion Program (Innovative design/manufacturing technologies), and New Energy and Industrial Technology Development Organization. The authors acknowledge Prof. Fumihiko Kimura at Hosei University for insightful suggestions about the direction of the study.

References

Bley, H. and Franke, C., Integration of product design and assembly planning in the digital factory, CIRP Annals–Manufacturing Technology, Vol. 53, No. 1 (2004), pp. 25–30.
Browning, T.R., Frische, E., Negele, H., Key concepts in modelling product development processes, Systems Engineering, Vol. 9, No. 2 (1996), pp. 104–128.
Chandrasekar, S.K., Ramani, K., Sriram, R.D., Horvath, I., Bernard, A., Harik R.F., Gao, W., The evolution, challenges, and future of knowledge representation in product design systems, Computer-aided Design, Vol. 45, No. 2 (2013), pp. 204-228.
Cho, S. and Eppinger, S.D, Product development process modeling using advanced simulation, in ASME Proceedings of DETC2001, DTM, Paper No. 21691 (2001).
Clarkson, P.J. and Hamilton, J.R., “Signposting,” A parameter-driven task-based method of the design process, Research in Engineering Design, Vol. 12, No. 1 (2000), pp. 18–38.
Draw.io, Free Online Diagram, J Graph Ltd., http://www.draw.io, visited on 2014 July.
Eppinger, S. D., Whitney, D. E., Smith, R., Gebala, D., A model-based method for organizing tasks in product development, Research in Engineering Design, Vol. 6, No. 1 (1996), pp.1–13.
Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C., Clarkson, J.P., Change propagation analysis in complex technical systems, Journal of Mechanical Design, Vol. 131, No. 8 (2009), pp.1-14.
Hoyle, C., Tumer, I.Y., Kuroglu, T., C. Wei, Multi-stage uncertainty quantification for verifying the correctness of complex system designs, in ASME Proceedings of DETC2011, Paper No. 47888 (2011).
Kondoh S. and Tezuka, A., Synthesis of verification models in multidisciplinary design of complex engineered systems, CIRP Annals–Manufacturing Technology, Vol. 63. No. 1 (2014), pp.145–148.
Krause, F.L., Kind, C., Voigtberger, J., Adaptive modelling and simulation for product development processes, CIRP Annals–Manufacturing Technology, Vol. 53, No. 1 (2004), pp. 135–181.
Kurtoglu, T. and Tumer, I.Y., A graph-based fault identification and propagation framework for functional design of complex systems, Journal of Mechanical Design, Vol. 130, No. 5 (2008), pp.1-8.
Lapperriere, L. and ElMalaghy, H.A, Planning of Products Assembly and Disassembly, CIRP Annals–Manufacturing Technology, Vol. 41, No. 1 (1992), pp.5–9.
Malak, R. M., Aughenbaugh, J.M., Paredis, C.J.J., Multi-attribute utility analysis in set-based conceptual design, Computer-aided Design, Vol. 41, (2009), pp. 214–227.
Shabi, J. and Reich, Y., Developing an analytical model for planning systems verification, validation and testing processes, Advanced Engineering Informatics, Vol. 26, (2012), pp.429-438.
Skiena, S.S., the algorithm design manual (2008), 2nd edition, Springer.
Smith, R.P. and Morrow, J.A., Product development process modeling, Design Studies, Vol. 20, No. 3 (1999), pp. 237–261.
Sobek II, D.K., Ward, A.C., Liker, J.K., Toyota’s principles of set-based concurrent engineering, Sloan Management Review, Vol. 40, (1999), pp. 67-83.
Tomiyama, T., D’Amelio, V., Urbanic, J., ElMaragh, W., Complexity of Multi-Disciplinary Design, CIRP Annals–Manufacturing Technology, Vol. 56. No. 1 (2007), pp.185–188.
Tomiyama, T., Kiriya, T., Takeda, H., Xue, D., Yoshikawa, H., Metamodal: a key to intelligent CAD systems, Research in Engineering Design, Vol. 1, No. 1 (1989), pp.19-34.
Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., Tomiyama, T., Physical concept ontology for the knowledge intensive engineering framework. Advanced Engineering Informatics, Vol. 18, (2004), pp. 95-113.