

Modeling Process Platforms based on An Object-Oriented Visual Diagrammatic Modeling Language

Linda Zhang

► To cite this version:

Linda Zhang. Modeling Process Platforms based on An Object-Oriented Visual Diagrammatic Modeling Language. International Journal of Production Research, 2009, 47 (16), pp.4413-4435. 10.1080/00207540801950144 . hal-00513029

HAL Id: hal-00513029 https://hal.science/hal-00513029

Submitted on 1 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Modeling Process Platforms based on An Object-Oriented Visual Diagrammatic Modeling Language

Journal:	International Journal of Production Research				
Manuscript ID:	TPRS-2007-IJPR-0717.R1				
Manuscript Type:	Original Manuscript				
Date Submitted by the Author:	05-Jan-2008				
Complete List of Authors:	Zhang, Linda; University of Groningen, Operations				
Keywords:	PRODUCTION MANAGEMENT, PRODUCTION MODELLING				
Keywords (user):	Process platform, high variety production				





MODELING PROCESS PLATFORMS BASE ON AN OBJECT-ORIENTED VISUAL DIAGRAMMATIC MODELING LANGUAGE

Lianfeng Zhang^{*}[†]

[†]Department of Operations, University of Groningen, Landleven 5, 9747 AD Groningen, The Netherlands

ABSTRACT

Process platforms have been recognized as a promising means of dealing with product variety while achieving a near mass production efficiency. To assist practitioners to better understand, implement and use process platforms, this study addresses the underlying logic of coping with the challenges in high variety production by adopting process platforms. Accordingly, this paper proposes to model process platforms with focus on the application processes. In view of the significance of dynamic modeling and visualization in shedding light on the logic of any processes, this study introduces a visual diagrammatic modeling language based on object-oriented (OO) techniques, named as OOVDML. With the graphical notations, uniquely shaped symbols, syntax and semantics, control mechanisms and arrangement rules, the OOVDML not only captures the logic of process platform's application but also provides a visualization of their behaviors in a holistic view. Moreover, incorporating OO modeling allows readers to focus on their own interests. This study approaches to modeling process platform's application with respect to activities pertaining to customer order processing, engineering change control and production job planning. Also reported is an industrial example of electronics products. The results of the case study not only show the suitability of the OOVDML but also shed light on the dynamic behaviors of process platforms.

Key Words: *Process platform, high variety production, diagrammatic modeling, objectoriented methods, visualization.*

^{*} Corresponding author. Email: <u>L.Zhang@rug.nl</u>

1. INTRODUCTION

 Manufacturing environments nowadays are characterized by a high variety of customized products, often coupled with small quantities and short delivery lead-times. To survive the resulting intense global competition by pleasing their customers, manufacturing companies struggle to provide quickly high product variety at low costs. The key for companies to achieve efficiency in producing large numbers of customized products lies in an ability to maintain the resulting high variety production to be as stable as possible (Schierholt, 2001; Williams et al., 2007; Zhang, 2007). In this respect, process configuration contributes to manufacturing stability by generating similar processes for part families (Schierholt, 2001). It is an alternative of computer-aided process planning per se.

Similar as the process configuration thinking, a concept of process platforms has been recognized as a promising means for companies to achieve a near mass production efficiency by managing high variety production, wherein the complex products consisting of assemblies and parts are involved (Zhang, 2007). The rationale is to plan and utilize similar, yet optimal, production processes as these existing on shop floors to fulfill diverse customized products. Current research efforts have approached process platforms from several aspects (Zhang, 2007), including conceptual formulation (Jiao et al., 2007a), structural representation (Zhang et al., 2007), construction (Jiao et al., 2007b) and identification of mapping relationships within a process platform (Jiao et al., 2008). Besides the insight provided by the above works, a good understanding of the underlying logic of coping with difficulties in high variety production by adopting process platforms is necessary for companies to design, develop and apply process platforms.

With an attempt to assist practitioners to better understand, implement and use process platforms, this study addresses the underlying logic of process platform's behaviors. In view of the significance of dynamic modeling and visualization in shedding light on the

 logic of any processes, this paper, accordingly, proposes to model process platforms with focus on the application processes, i.e., dynamic modeling. Along with the complexities in fulfilling high product variety, the resulting difficulties in modeling process platform's application behaviors have been recognized, as elaborated below.

1.1. Difficulties in modeling process platforms

Modeling process platforms intends to visualize their application behaviors in high variety production in a holistic view. The diverse customized products along with the resulting large number of constituent items impose many complexities in production. For example, many personal, activities, data, information, etc. are involved in different phases of production. The same data may be manipulated by many different activities, which, in their turn, may manipulate other data as well. Personal may carry out different activities and activities may be carried out by different personal. Activities also have mutual dependencies. They may access the same data at the same time or they may be ordered in complex ways. In this regard, modeling process platforms is expected to capture the inherent complexities in one dimension rather than many dimensions, e.g., activities, processes, personal. Describing the different types of system elements in more than one dimension incurs difficulties in providing readers with an overall picture of process platform's application in a holistic view.

Second, process platform modeling is to shed light on the underlying logic of coping with difficulties by adopting process platforms in high variety production. Incorporating process platforms when fulfilling various products has brought about many additional events, activities, information and data in production. Moreover, it is not uncommon that any processes involve a number of activities that are concurrent, dependent and parallel. Because of these activities/events and their complicated relationships, reasoning about process platform's application behaviors is important for the interested readers to gain an

insight in the logic. Therefore, the process platform models to be constructed must be able to reveal the underlying reasoning of process platform's application.

Third, in a company, people at different levels view the same thing from different aspects and have different focuses. This is also true for process platform's application. For example, while management people may care about the functional areas involved and their relationships, production personal are more concerned with the activity details in his/her own area. In accordance with the different views, the process platform models to be constructed should present the corresponding application processes at different levels of abstraction, which allows different readers to pay attention to their own interested areas.

Furthermore, the ultimate goal of process platform modeling is to help practitioners better understand and grasp the essence of process platform's application. Thus, it raises the importance in an unambiguous modeling. In other words, process platform models should be constructed to allow different readers to interpret the corresponding application uniquely and in an exactly same way. Besides, the models are expected to provide an easyunderstandability and readability.

1.2. Strategy for solution

To cope with the modeling difficulties, this study puts forward an OOVDML (objectoriented (OO) visual diagrammatic modeling language) by integrating the principles of a number of well-defined modeling techniques.

Graphical notations and diagrams are incorporated in the OOVDML to provide a visualization of process platform's application in a holistic view. Besides graphical representations, textual representations in the natural language are introduced. Textual representations are able to capture the complicated data, information, personal, activities and their relationships and further map them in one dimension. In the OOVDML, they are used to model the detailed activities, involved personal, inputs and outputs. They are also

 used to denote the names and attributes of graphical notations defined. To deal with the issues regarding different levels of abstractions, the class concept in OO modeling is incorporated. Both the class attributes and/or detailed class operations can be hidden, when necessary. As a result, the readers are able to focus on their own interests. Moreover, a number of arrangement rules and control mechanisms along with predicate formulas are defined in the OOVDML to capture and model the reasoning of process platform's application behaviors.

Rather than all processes in high variety production, this study approaches process platform modeling with focus on activities pertaining to production job planning, engineering change control and customer order processing. The reason is that these processes are fundamental to efficient high variety production.

The rest of the paper is structured as follows: The work regarding high variety production management and dynamic modeling languages is present in Section 2, following which the fundamentals of process platforms are introduced in Section 3. The OOVDML is detailed in Section 4. Section 5 reports an industrial example involving vibration motors for mobile phones. Section 6 concludes the paper by discussing advantages and disadvantages of the proposed OOVDML and by identifying future research.

2. RELATED WORK

In the past several decades, a large body of literature has been reported to assist companies to provide various customized products as expected by customers. Such research efforts have been made mainly in the areas of product platform development (Simpson, 2004), product family design (Sanderson and Uzumeri, 1995), outsourcing (Harland et al., 2005) and supply chain management (Min and Zhou, 2002). While the resulting methodologies/system prototypes/frameworks, to some degree, help companies

 achieve design efficiency and effectiveness and selection of proper supply chain partners, they lack an explicit consideration of issues regarding production of high product variety based on companies existing manufacturing resources (Lu and Botha, 2006). Schierholt (2001) introduces the concept of process configuration, which combines the principles of product configuration and process planning, to deal with manufacturing process generation. Williams et al. (2007) put forward process parameter platforms for developing manufacturing processes by taking into account the non-uniform market demand. In essence, both process configuration and process parameter platforms to assist companies to achieve a near mass production efficiency in high variety production. Unlike process configuration and process parameter platforms intend to facilitate production process generation for complicated end-products, wherein both parts and assemblies are involved.

The conceptual models of process platforms are formulated with respect to basic constructs, definitions, relationships and functionalities using set theory and OO techniques (Jiao et al., 2007a). Also discussed are process platform's concept implications: generic variety representation, generic structures and generic planning. The large number of different types of product items (including end-products) and the corresponding process elements involved in a process platform have been identified and described using unified modeling language (Zhang et al., 2007). To assist companies to benefit from the past production practice, Jiao et al. (2007b) put forward a data mining methodology to identify and form process families in relation to product families. Similarly, to facilitate configuration rule construction in process platforms, an approach based on association rule mining is discussed to obtain mapping relationships between product and process variety from large volumes of existing production data (Jiao et al., 2008).

Page 7 of 36

Dynamic modeling is to model the relevant processes involving objects being studied. As the core of process modeling, modeling techniques or languages have been investigated. There are two classes of process modeling languages using formal notations: visual diagrammatic process modeling languages and programming process modeling languages. Since the programming process modeling languages (Vernadat, 1993; Sutton et al., 1990) have strong limitations in readability and understandability, most process modeling has employed diagrams. Diagrammatic representation that combines graphic notations and natural language-like strings as the formalism can facilitate readers' understanding.

The major diagrammatic process modeling languages can be summarized as simple process diagrams, advanced process diagrams and functional description diagrams. The simple process diagrams, such as traditional flow charts, material flow diagrams (Baudin, 1990) and acyclic networks (Hajdu, 1997), are simple, easy to use and understand. However, they are unable to express complicated execution relationships and have limited capacity in representing behavioral aspect of a process. With an attempt to include more information about processes, advanced process diagrams, e.g., process flow entity diagrams (Grabowski et al., 1996), event-driven process chains (Zukunft and Rump, 1996), IDEF3 (http://www.idef.com/Home.htm), event diagrams (Martin and Odell, 1992), Petri nets (Peterson, 1981), role activity diagrams (Ould, 1995), have been developed. Compared with simple process diagrams, the advanced process diagrams can capture more information about processes from, e.g., organizational perspective, behavioral perspective. Nevertheless, most of such advanced process diagrams are unable to handle the exceptional and non-deterministic execution relationships. Furthermore, they lack the capability to represent processes from the informational and functional perspectives. While functional description diagrams, including IDEF0 (http://www.idef.com/Home.htm), object flow diagrams (Martin and Odell, 1992) and use case diagrams (Jacobson, 1992),

are well-defined mechanisms to represent functional, informational and organizational perspectives of processes, they are unable to express the behavioral perspective.

OO modeling notions and methods (Rumbaugh et al., 1991; Booch, 1994; Martin and Odell, 1996; Harmon, 1998) are powerful for dealing with modeling complexity in the real world and building a model in a comprehensive, expressive, understandable and structured formalism. The notions and technique are believed to provide an innovative and effective way of modeling a variety of production activities in relation to diverse products as well. Recognizing the limitations of the diagrammatic process modeling languages and the power of OO notations and methods, Ma (1999) puts forward a language including a customer process flow diagram and an entity representation diagram to model service product design. With reference to his work, the OOVDML is devised in this study to model process platform's application processes.

3. FUNDAMENTALS OF A PROCESS PLATFORM

Current practice in design, be it product configuration or platform-based product family design, leads to the concept of product families. A product family consists of a set of customized, yet related, products which perform a same basic function. While customized products belonging to the same family assume a common product structure, they differ with one another in optional features and functionalities. The design changes among family members impose necessary variations in converting abstract design concepts into physical products. On the other hand, the similarity and commonality inherent in product families, exhibited by similar and/or same raw materials, components, subassemblies and assemblies, makes it possible for companies to utilize similar production processes to fulfill product family members. Lu and Botha (2006) point out that unlike product development, process development has not received too much attention in today's manufacturing environment because of the technical difficulties and managerial

challenges to be considered. In response to the lack of research in process development, Zhang (2007) proposes process platforms to assist companies to plan similar, yet optimal, production processes for product families. The ultimate goal is to help companies achieve a near mass production efficiency in producing high product variety.

A process platform entails a conceptual structure and overall logical organization of production processes to produce a product family. It provides a generic umbrella to capture and utilize commonality, within which each new product fulfillment is instantiated and extended, and thereby anchoring process planning to a common structure. A process platform contains all data pertaining to the product family, e.g., items, quantity-per, parent-child relationships, and all production process data of the corresponding process family, e.g., operations types and precedence, work centers, machines, tools, standard cycle times, setup activities. With an attempt to include all product and process family data in a single structure without data redundancy, the concept of generic variety representation (van Veen, 1992) is adopted to organize data in process platforms.

In a process platform, the specific data pertaining to individual products and production processes is organized as a generic product structure of the product family and as a generic process structure of the process family in relation to the product family, respectively. As a result, the two generic structures are common to all members in product and process families. Further, they are integrated into a single structure, called generic product-process structure, which is fundamental to the process platform. The integration is achieved by adopting the mapping relationships between two sets of family data. Figures 1 and 2 show conceptually the generic process structure, respectively.

As shown in the figures, each node in the generic structures is either a generic product item or a generic process in relation to a generic product item. While a generic product item represents a family of item variants (or individual items) of the same type, a generic process refers to the set of production processes to produce variants belonging to the corresponding item family. For example, A1 (in both figures) is a family of assemblies of type A1 and AP1 represents the set of assembly processes producing A1. A generic process is further decomposed into a set of ordered generic operations, as shown by the decomposition of, for example, AP4 in the Generic Process Structure in Figure 2. Each generic operation, in turn, is described by generic process elements such as generic work centers (including tools, fixtures, jigs, etc.), cycle times and setups. Similarly, each such generic process element represents a family of process elements of the same type.

The way of organizing product family data and process family data into a unified generic structure enables the simultaneous derivation of a bill of materials (BOM) and the corresponding bill of manufacturing resources and operations (BOMfrO) for a new design. While a BOM represents a product design and describes items level by level along the product hierarchy, a BOMfrO represents a production process and reveals operations, operations precedence and the set of manufacturing resources, e.g., machines, tools, fixtures, in accordance with items in the BOM.

With focus on the development of an appropriate language to model process platform's application in high variety production, this study assumes the interested readers can get comprehensive and detailed information/knowledge about process platforms by referring to (Zhang, 2007).

4. THE OBJECT-ORIENTED VISUAL DIAGRAMMATIC MODELING LANGUAGE

In most situations, a process flow consists of more than one activity. These activities interact with one another through a number of temporal and logical relationships. In line with activities and their relationships, execution transitions are defined as follows:

Definition 1: An execution transition refers to a set of temporal and logical relationships among the executions of activities in a process flow.

In the domain of production, the temporal and logical relationships can be classified into three basic types, including

- *S*(*ai_aj*): the temporal and logical relationship between activities *ai* and *aj* is *sequential*. It means that *aj* is performed only after the completion of *ai*. In other words, the execution of one activity depends on the completion of another.
- *P*(*ai_aj*): the temporal and logical relationship between *ai* and *aj* is *parallel*. It means that *ai* and *aj* are executed either synchronously or asynchronously and there is no logical constraints on their execution.
- *O*(*ai_aj*): the temporal and logical relationship between *ai* and *aj* is *optional*. It means that either *ai* or *aj* rather than both can be executed under certain conditions.

4.1. Primary constructs, notations and semantics

The OOVDML consists of 5 building blocks: *StartPoint, AchievementPoint, ProductionActivityClass, ArrowLink* and *SequentialJunction*.

(1) ProductionActivityClass (PAC)

It is common that similar production activities are always performed for some similar objectives. From the OO point of view, the set of specific production activities can be generalized into a class. By incorporating OO concepts, a *PAC* is defined to represent a class of similar production activity instances. Such activity instances not only possess the similar structural properties but also are performed by the responsible functions or personal by following a common pattern. Figure 3 shows the notation of a *PAC* and an example.

A *PAC*, drawn as a rectangle with 3 compartments, has a name, attributes and operations, as shown in Figure 3(a). Activity attributes describe the structural properties of

a class of production activities. The activity attributes common to most production activity classes are identified as follows:

- responsible functions: indicating the functions executing these activities;
- processing time: showing the duration time of an activity;
- frequency of occurrence: describing the frequency of an activity; and
- constraints: concerning with the conditions that need to be met for an occurrence of an activity to start, continue, or stop.

Activity operations describe the behavioral property of a class of production activities, more specifically the processing steps. An activity starts when its operation is triggered; an activity terminates when its operation stops.

The syntax of the production activity names, attributes and operations is defined as follows:

ActivityName ::= Ipa: Gerund-phrase // Gerund-phrase: a gerund phrase string

Ipa ::= **PA** IntegerNumber

IntegerNumber $::= 1 \mid 2 \mid 3 \mid \dots$

ActivityAttribute ::= ActivityAttributeName = AttributeValue ActivityAttributeName ::= String // String: a string

AttributeValue ::= STF // STF: a written representation in strings, tables, figures, or the combination of them

ActivityOperation ::= *ActivityOperationName* {*ActivityScript*}

ActivityOperationName ::= Gerund-phrase () | Verb-phrase ()

ActivityScript ::= Script // Script: a script written in the natural language

An *Ipa* is a code of a production activity instance of a class. A code is formed by a reserved word **PA** (production activity) and an integer number. A *Gerund-phrase* is a gerund phrase and is used as a part of a name string that can be understood literally. An *ActivityAttributeName* denotes the name of an activity attribute and an *AttributeValue* represents a value of an attribute. An *ActivityOperationName* and an *ActivityScript* define the name and specification of an activity operation, respectively. An

ActivityOperationName can be either a *Gerund-phrase* or a *Verb-phrase*. An *ActivityScript* is written in the mode of a script in the natural language.

A *Script* describes the prescriptive actions of the responsible function or personal, the critical conditions under which the actions happen and the involved inputs/outcomes. An example of a script of a *PAC*: Managing demand is as follows:

After completing the calculation of the forecasted product demand, they submit the result to the coordinator. The coordinator analyses all the received results to assure the completeness of demand information, e.g., the sources and quantity. Subsequently, he communicates the obtained final demand information to planners responsible for production planning and master production planning...

(2) ArrowLink and SequentialJunction

They are two building blocks representing precedence execution transitions in the process flow of production activities. Figure 4 shows an *ArrowLink* and a *SequentialJunction*.

An *ArrowLink* indicates one-way traffic to "transmit" an execution control flow. It paves a section of road for execution transition from the completion of some activities to the starting of others. A *SequentialJunction* acts as a "transformer" and transforms its incoming execution control flows into its outgoing flows. While the incoming execution control flows refer to the control flows on the *ArrowLinks* that connect to the *SequentialJunction* by their arrow ends, or head ends, the outgoing execution control flows are the flows on the *ArrowLinks* that connect to the *SequentialJunction* by their tail ends. A set of control conditions, under which the incoming execution control flows are transformed into outgoing flows, make up the control mechanism.

An *ArrowLink* is represented by an adorned arrow line with an arrow end and a tail end. The adornment in the form of a string sits near the arrow line and identifies the *ArrowLink*. A *SequentialJunction* is represented by an adorned solid bar and a dashed-rectangle tag. Similarly, the adornment in the form of a string is placed near the solid bar and identifies the *SequentialJunction*. The strings on the tag describe the set of control conditions. When a *SequentialJunction* connects only two *ArrowLinks*, the tag is allowed to be omitted. In this case, the default control condition is when the incoming *ArrowLink* receives execution control flow, the outgoing *ArrowLink* does. The strings are written in predicate formulas. Such formulas are able to provide a convenient and flexible mode to define the control conditions in concise propositions. The basic components of strings include two predicates, three connectives and related semantics conventions as follows:

Predicates:

- (1) **signal**(*x*): *ArrowLink x* has an execution control flow; and
- (2) event(e): event e occurs.

Connectives:

- (1) \wedge : (conjunction) means "and";
- (2) \lor : (disjunction) indicates "or"; and
- (3) \rightarrow : (conditional symbol) means "if... then...".

Related semantics conventions:

- For *ArrowLink x* that connects to a *PAC* by the head end, when **signal**(*x*) becomes true, an activity defined by the connected *PAC* will eventually happen, i.e., start.
- When an activity defined by a *PAC* completes or terminates, **signal**(*x*) becomes true for *ArrowLink x* connecting to the *PAC* by its tail end.
- When a *StartPoint* is initiated, **signal**(*x*) becomes true for *ArrowLink x* that connects to the *StartPoint* by its tail end.
- For *ArrowLink x* that connects to an *AchievementPoint*, when **signal**(*x*) becomes true, a process of production activities ends.

With the above definitions and semantics, the OOVDML is able to represent three types of activities bearing the three temporal and logical relationships:

(1) the activities that will start after one or more activities complete, i.e., *S*(*ai_aj*);

(2) the activities that are performed in parallel, i.e., $P(ai_aj)$; and

(3) the activities that are performed optionally, i.e., $O(ai_aj)$.

The following examples explain how strings on the tags model the precedence execution transitions.

Example 1. Figure 5 gives a fragment of a simplified production process flow with three production activities: PA1, PA2 and PA3. The control condition represented by the tag attached to J1 describes the execution transition as follows: After the completion of the activities preceding *ArrowLink* ar1, all three activities: PA1, PA2 and PA3 are performed eventually. The control condition represented by the tag attached to J2 models the following execution transition. The necessary condition, under which activities following ar8 will be carried out, is the completion of PA1 and PA2, or PA3.

Along with the completion of activities some events may occur. Hence, such events are considered as the cause of triggering certain activities. In such cases, predicate event(e) can be used to define the relevant control condition.

Example 2. The control condition represented by the tag attached to J2 in Figure 6 models the below execution transition.

- (1) if event e3*i*: option A specified by the customer happens, then PA3*i* will be performed;
- (2) if event e3*j*: option B specified by the customer happens, then PA3*j* will be performed; and
- (3) if event e3k: option C specified by the customer happens, then PA3k will be performed.

In Figure 6, the tag attached to J1 is omitted and the default control condition is that when ar1 receives an execution control flow, ar2 obtains the execution control flow as well.

(3) *StartPoint* and *AchievementPoint*: A *StartPoint* denotes a start point. It indicates the initiation of a process flow of production activities. A start point attribute can be used to represent the name of the process flow. An *AchievementPoint* denotes the ending point of a process flow of production activities. When necessary, an achievement point attribute is used to indicate the noticeable "achievements". Figure 7 shows the notation examples of a *StartPoint* and an *AchievementPoint*. A *StartPoint* is represented by a circle with strings in the centre indicating the start point attribute. An *AchievementPoint* is represented by a rounded rectangle. When necessary, strings can be put in the centre to denote its attribute.

4.2. Arrangement rules

The following rules are defined to connect rationally the primary constructs, thus modeling process platform's behaviors.

R1. For a StartPoint,

• there is only one ArrowLink that connects to the StartPoint by its tail end.

R2. For an AchievementPoint,

• there is only one ArrowLink that connects to the AchievementPoint by its head end.

R3. For a PAC,

• there exist two different *ArrowLinks*. One connects to the *PAC* by its head end and the other by its tail end.

R4. For an *ArrowLink*, any one of the following cases is valid.

- Its tail end connects to a SequentialJunction and the head end connects to a PAC.
- Its tail end connects to a *PAC* and the head end connects to a *SequentialJunction*.

- Its tail end connects to a *SequentialJunction* and the head end connects to another *SequentialJunction*.
- R5. For a Sequential Junction,
 - there is one or more *ArrowLinks* that connect to the *SequentialJunction* by their respective head ends; and
 - there is one or more *ArrowLinks* that connect to the *SequentialJunction* by their respective tail ends.
- R6. For two different Sequential Junctions,
 - there exists either only one or no *ArrowLink* that directly connects to the two *SequentialJunctions* by its two ends.

5. INDUSTRIAL EXAMPLE

The industrial example adopted is the production of vibration motors for mobile phones in an electronics company. While vibration motors for mobile phones are not so complicated compared with others, the degree of their product complexity allows the modeling of process platform's application by reflecting real world situations as more as possible. Due to the frequent design changes to mobile phones, motors must be customized to match the requirements of these diverse mobile phones. In spite of the fact that such individual motors have certain items in common, they have their distinct design specifications, thus imposing different production requirements. Different from the mass production environment in the past, the current manufacturing environment in the company is characterized by diverse customized motors, low production volumes, short delivery lead-times and increasingly reduced costs. Together with the limited manufacturing resources, these characteristics complicate the company's production from every aspect: planning, scheduling, execution, etc. Taking production process planning as an example, the production process for producing a standardized motor in the mass

production environment may not be optimal when various motors are to be fulfilled using the same set of manufacturing resources within a same time period.

For illustrative simplicity, modeling process platforms using the proposed OOVDML has been applied to a motor family. Figure 8 shows the common product structure of the motor family and, Figure 9 shows a simplified version of the generic product-process structure underpinning the process platform of the motor family. Furthermore, this case study focuses on process platform modeling with respect to activities pertaining to production job planning, customer order processing and engineering change control.

5.1. Customer order processing

While viewed from a broad viewpoint, customer order processing encompasses the entire process ranging from customer order entry, production, to the shipping of endproducts, it relates to activities pertaining to customer order entry in a narrow aspect.

The company adopts various functional requirements to constitute product catalogues representing the motors that they can offer. Such product catalogues are expected to facilitate customer order entry since customers are more familiar with general functions fulfilled by motors rather than technical details. The company accommodates the functional representation of individual motors through identifying specific items by variety parameters and the corresponding value instances. In this regard, functional requirements are associated with options, i.e., generic items, in their process platforms.

Since functional requirements relate to generic items in a process platform, each functional requirement specified in a customer order can be described by an instance of a generic item with respect to specific parameter values, i.e., $\{FR\} \sim \{Generic item.$ Parameter = Parameter value}, that is, a specific item. Hence, variety parameters coherently link functional requirements in a customer order to the corresponding motors

 and constitute items. By incorporating production job planning (to be introduced below), the company derives both BOMs and BOMfrOs from the process platform and further generates the set of production job data. Therefore, the process platform performs as a well-structured mechanism that ties customer orders to the corresponding BOMs, BOMfrOs and production job data, and thus facilitate the management of customer orders in their production fulfillment.

In line with the fact that the company has applied various functional requirements at different levels to represent their motor offerings, Table 1 shows a customer order, represented by a list of functional requirements, with a lost size: 50 units. Figure 10 shows the process of processing the customer order.

5.2. Engineering change control

An engineering change (EC) refers to a change and/or a modification in material, dimension, form, fit or function of an item, be it a part, an assembly, or an end-product, after the design is released (Maull et al., 1992). It is common that each EC causes a series of down-stream changes along the product development process across a company where multi-disciplines work together dealing with these induced changes. As a result, various functions across a company have to adjust their activities in order to deal with ECs and their impacts. Hence, for the purpose of production management, ECs must be made to relevant BOM files and be managed as a part of the engineering change control (ECC) process. This coincides with the observations from 1) Terwiesch and Loch (1999): ECs consume one third to one half of the total engineering capacity and represent 20-50% of total tool costs; and 2) McIntosh (1995): ECs occur frequently for continuous improvement and determine as much as 70 to 80% of the final cost of a product. As

 claimed by Tavcar and Duhovnik (2005), companies' ability to manage engineering changes efficiently reflects their agility. Failure to control ECs and manage the corresponding ECC process inevitably leads to the production of low quality (sometimes even unwanted) products, higher production costs and delayed order deliveries.

Since the company's production involves a high variety of customized motors coupled with small quantities and short delivery lead times, ECs become more serious due to the various design changes. With the process platform, the company can manage ECs effectively, through defining both BOMs and BOMfrOs based on different values of variety parameters in accordance with design changes.

For any EC proposal, the systematic analysis of possible implementations of the change, i.e., new item design, is required. In their previous production, such analysis was time and effort-consuming. With the process platform, the company can perform the analysis effectively. First, the generic product and process structures eliminate the time to analyze the costly and/or technically infeasible product items to implement the proposal. The reason is that the two generic structures provide all the technically and economically feasible designs for both motors and the constituent items. Consequently, the process platform confines the analysis of infeasible implementation by considering the company's existing design and manufacturing capabilities. Second, the simultaneous derivation of BOMs and the corresponding BOMfrOs in response to possible design proposals enables multiple functions in the company to assess concurrently the impacts with respect to time and cost of changes on activities to be carried out in their own functional areas.

The combination of the functional requirements in the order in Table 1 necessitate new bracket assembly. Thus the company has performed ECC on the process platform, as shown in Figure 11. After processing the order and conducting engineering changes, the motor in terms of BOM is obtained, as shown in Table 2.

5.3. Production job planning

A production job encompasses a series of tasks to be finished for producing an item, be it a part, an assembly, or an end-product. A production job may not directly correspond to a customer order nor does it necessarily have to be related to a particular one. They are either to satisfy needs that arise from customer orders or to simply serve for inventory purpose. On the other hand, any production jobs relate to known product design, more specifically BOMs which are a common representation format of design. Accordingly, production job planning is to specify due dates, lot sizes, a picking list of all required material items and a corresponding list of manufacturing resources and associated operations, i.e., a production process.

With the integrated generic product-process structure, the process platform not only provides the company with a planning standard, according to which all reporting formats of the traditional BOMs and production processes for an item can be generated but also facilitates the company to create jobs and to plan their variations. According to the production need, the company derives the BOM of an item to be produced from the process platform by instantiating a set of variety parameters with respect to the set of value instances that define the item. Interested readers can refer to the approach to deriving BOMs from a process platform in (Jiao et al., 2007a; Zhang, 2007).

Since in the process platform, the company has integrated both the generic and specific product and process data by a set of variety parameters and their value instances, for any customer order, the company derives the proper BOM in conjunction with the corresponding BOMfrO level by level along the generic product-process structure embedded in the process platform. In addition, the company adopts the concept of generic

planning (Jiao et al., 2007a; Zhang, 2007) to ensure the selection of proper product data and production process data.

The company generates the complete production job data by considering the corresponding due date and lot size. This is accomplished by associating the BOM data with the corresponding BOMfrO data based on the materials-operations links embedded in the process platform. In materials-operations links, product items of the proceeding operations become the material items of the immediately following operations. The company also adopts some constraints of job sequence when generating the set of production job data. One example of sequence constraints is that the job making a parent item cannot be started until the jobs producing its child items are completed or partially finished. The obtained production job data not only are necessary for the company to carry out production execution activities but also are essential for the company to perform production control activities, e.g., checking the availability of materials and manufacturing resources so as to ensure the completion of the relevant jobs within the due dates.

In addition, the company has improved the maintenance and modification of production job data based on the process platform. The reason is that the only required work of modifying an existing set of production job data in accordance with new customer orders is to copy the relevant data and paste them in another file rather than modifying the process platform. Table 3 shows the corresponding production job data in accordance with the motor in Table 2 obtained through the process of planning production jobs, shown in Figure 12.

Page 23 of 36

 While the above three processes have their different focus areas, they have certain overlap due to the adoption of the process platform in Figure 9. The common sub-process is circled by red double lines in the three processes.

6. CONCLUSIONS

In view of the significance of process platforms for managing production of high product variety in manufacturing environments nowadays, this study proposed to model process platforms with focus on the application processes. Accordingly, the OOVDML was put forward to cope with the modeling difficulties. The industrial example has revealed several advantages of the OOVDML for modeling process platform's application. First, the diagrammatic representation visualizes an overall picture of process platform's application process in a holistic view. This, in turn, provides a starting point for companies to understand, analyze and further improve the process in consideration. Second, the well-defined syntax and semantics offer a thorough understanding and rigorous interpretation of activities, personal, their relationships, etc. in process platform's application processes. Consequently, in conjunction with the diagrammatic representation they enable companies to be aware of the impacts of any changes to activities, personal and their relationships on the application processes. Third, the OO modeling techniques along with the texts allow the application processes to be documented easily at different levels of granularity, which provides different people with their own interests.

Nevertheless, the OOVDML has its limitations. With focus on the modeling of process platform's application processes, the OOVDML does not pay attention to a process platform itself. Consequently, it cannot shed light on the variety of constituent elements and their complicated relationships inherent in a process platform. This may provide an opportunity to extend this study. In future research, a comprehensive modeling language/formalism may be developed based on several well-defined modeling techniques. Such a comprehensive modeling language should be able to capture and model both process platforms and their application processes. The resulting models can, thus, be expected to reflect the interactions between process platforms and their application, based on which companies can make improvements on process platforms and/or application processes eventually.

Another avenue for directing future research would be to develop a computer-aided system incorporating the OOVDML. Companies' legacy systems may be integrated with such a system so that the system is able to capture real-time data and information, thus providing more accurate information for companies to make right decisions in adding, removing, modifying activities in the application processes.

ACKNOWLEDGEMENTS

The author would like to thank two anonymous reviewers and the editor for their insightful and constructive comments on the earlier version of this paper.

References

Baudin M., Manufacturing Systems Analysis. 1990, Yourdon Press Computing Series.

Booch G., *Object-Oriented Analysis and Design*. 1994, Benjamin/Cummings Publishing Company, Inc.

Grabowski H., Furrer M., Renner D. and Schmid C., Implementation of information system supporting engineering process based on World Wide Web. In: Scholz-Reiter, B., Stickel, E. (Eds.), 1994, *Business Process Modeling*, Springer, Berlin.

Hajdu M., Networks Scheduling Techniques for Construction Project Management. 1997,Kluwer Academic Publishers, Dordrecht.

Harland C., Knight L., Lamming R. and Walker H., Outsourcing: Assessing the risks and benefits for organizations, sectors and nations. *International Journal of Operations & Production Management*, 2005, 25, pp. 831-850.

 Harmon P., Understanding UML: The Developer's Guide: With a Web-based Application in Java. 1998, Morgan Kaufmann Publishers, San Francisco.

http://www.idef.com/Home.htm

- Jacobson I., *The Object-Oriented Software Engineering*. 1992, Addsion-Wesley Publishing Company.
- Jiao J., Zhang L. and Pokharel S., Process platform planning for variety coordination from design to production in mass customization manufacturing. *IEEE Transactions on Engineering Management*, 2007a, **54**, pp. 112-129.
- Jiao J., Zhang L., Pokharel S. and He Z., Identifying generic routings for product families based on text mining and tree matching. *Decision Support Systems*, 2007b, **43**, pp. 866-883.
- Jiao J., Zhang L., Zhang Y. and Pokharel S., Association rule mining for product and process variety mapping. *International Journal of Computer Integrated Manufacturing*, 2008, **21**, pp. 111-124.
- Kolisch R., Integration of assembly and fabrication for make-to-order production. *International Journal of Production Economics*, 2000, **68**, pp. 287-306.
- Lu Q. and Botha B., Process development: a theoretical framework. *International Journal of Production Research*, 2006, **44**, pp. 2977-2996.
- Ma Q., An Approach to the Representation of Service Process Design. PhD dissertation, 1999, Department of Industrial Engineering and Logistics Management, Hong Kong University of Science and Technology.
- Martin J. and Odell J., *Object-Oriented Analysis and Design*. 1992, Prentice Hall, Englewood Cliffs, NJ.
- Martin J. and Odell J., *Object-Oriented Method Pragmatic Considerations*. 1996, Prentice Hall, Englewood Cliffs, NJ.

- Maull R., Hughes D. and Bennett J., The role of the bill-of-materials as a CAD/CAPM interface and the key importance of engineering change control. *Computing & Control Engineering Journal*, 1992, **3**, pp. 63-70.
- McIntosh K.G., Engineering Data Management: A Guide to Successful Implementation, 1995, McGraw-Hill, New York.
- Min H. and Zhou G., Supply chain modeling: Past, present and future. *Computers & Industrial Engineering*, 2002, **43**, pp. 231-249.

Ould M.A., Business Processes. 1995, Wiley, New York.

- Peterson J.L., *Petri Net Theory and The Modeling of Systems*. 1981, Prentice-Hall, Englewood Cliffs, NJ.
- Rumbaugh J., Blaha M., Premerlani W., Eddy F. and Lorensen W., *Object-Oriented Modeling and Design*. 1991, Prentice Hall, Englewood Cliffs, NJ.
- Sanderson S. and Uzumeri M., Managing product families: The case of the Sony Walkman. *Research Policy*, **24**, pp. 761-782.
- Schierholt K., Process configuration: Combining the principles of product configuration and process planning. *Aritificial Intelligence for Engineering Design, Analysis and Manufacturing*, 2001, **15**, pp. 411-424.
- Simpson T.W., Product platform design and customization: Status and promise. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 2004, **18**, pp. 3-20.
- Sutton S., Heimbigner D. and Osterweil L.J., Language constructs for managing change in processes centered environments. *Proceedings of the Fourth SIGSOFT Symposium on Software Development Environments. Software Engineering Notes*, 1990, **15**, 206–217.
- Taal M. and Wortmann J.C., Integrating MRP and finite capacity planning. *Production Planning & Control*, 1997, 8, pp. 245-254.

- Tavcar J. and Duhovnik J., Engineering change management in individual and mass production. *Robotics and Computer-Integrated Manufacturing*, 2005, **21**, pp. 205-215.
- Terwiesch C. and Loch C.H., Managing the process of engineering change orders: The case of the climate control system in automobile development. *Journal of Product Innovation Management*, 1999, **16**, pp. 160-172.
- van Veen E.A., *Modeling Product Structures by Generic Bills-of-Materials*, 1992, Elsevier, New York.
- Vernadat F., CIMOSA: enterprise modeling and integration using a process based approach. In: Yoshikawa H. and Goossenaerts J. (Eds.) Information Infrastructure Systems for Manufacturing, 1993, North-Holland, Amsterdam.
- Williams C.B., Allen J.K., Rosen D.W. and Mistree F., Designing platforms for customizable products and processes in markets of non-uniform demand. *Current Engineering: Research & Applications*, 2007, **15**, pp. 201-216.
- Wortmann J.C., Muntslag D.R. and Timmermans P.J.M., *Customer-Driven Manufacturing*, 1997, Chapman and Hall, London.
- Yeh C.H., Production data modeling: An integrated approach. *International Journal of Operations & Production Management*, 1995, **15**, pp. 52-62.
- Zhang L., *Process platform based-production configuration for mass customization*. PhD Dissertation, 2007, Division of Systems and Engineering Management, Nanyang Technological University, Singapore.
- Zhang L., Jiao J. and Helo P., Process platform representation based on unified modeling language. *International Journal of Production Research*, 2007, **45**, pp. 323-350.
- Zukunft O. and Rump F., From business process modeling to workflow management: An integrated approach. In: Scholz-Reiter B. and Stickel E. (Eds.) *Business Process Modeling*, 1996, Springer, Berlin.









ActivityName	PA2: Preparing materials
ActivityAttribute	Responsible person = material handler processing time = range from <i>x</i> hours to <i>y</i> hours
ActivityOperation	Prepare() { prepare the required materials according to the work order}



(b) A production activity example





Figure 4: An ArrowLink and a SequentialJunction



Figure 5: A fragment of a production process flow



Figure 7: Example notations of StartPoint and AchievementPoint



Figure 8: The common product structure of a motor family





Figure 9: The process platform of the motor family





Figure 10: Processing the customer order in Table 1





Figure 11: An ECC process in relation to the customer order in Table 1



Figure 12: Planning production jobs for the customer order in Table 1

Table 1: A customer order

Order #:xxxxx Due date:xxxxx	Customer Info.xxxxxx Delivery:xxxxxx	Volume: <u>50</u> Description:xxxxxxxxxx						
Functional Requirements:								
Rubber Holder. Color = Red								
Weight. Width = 5mm								
Shaft. Diameter = 3mm								
Shaft. Length = 16mm								
Frame. Thickness = 3mm								
Frame. Length = 20mm								
Magnet. Length =	: 14mm							
Bracket A. Shape = L								
Bracket A. Color = Red								
Bracket A. Width	= 6mm							
Bracket B. Shape = L								
Bracket B. Color	= Red							
Bracket B. Width = 6mm								
Terminal. length =	Terminal. length = 6mm							
Terminal. Width = 5mm								
Tape. Color = Red								
Tape. Width = 3 mm								
Commutator. Thickness = 2mm								

Table 2: The motor specified in the customer order in Table 1

Hierarchy Level	Generic Item	Parameter	Value	Parameter Value	Quantity per
1	Rubber holder	Color	Red	C_R	1
1	Weight	Width	5mm	W_5	1
1	Mainbody				1
. 2	Armature Assy				1
. 2	Frame Assy				1
. 2	Bracket Assy				1
3	Shaft	Diameter	3mm	D_3	1
3	Coil Assy	Length	16mm	L_16	1
3	Frame	Thickness	3mm	Т 3	1
		Length	20mm	L_20	
3	Magnet	Length	14mm	L_14	1
3	Bracket A	Shape	L	SL	1
		Color	Red	C_R	
		Width	6mm	W_6	
3	Bracket B	Shape	L	S_L	1
		Color	Red	C_R	
		Width	6mm	W_6	
3	Terminal	Length	6mm	L_6	1
		Width	5mm	W_5	
4	Coil	Diameter	1mm	D_1	1
4	Tape	Width	3mm	W_3	1
4	Commutator	Thickness	2mm	T_2	1

Table 3: The PJD for producing the motor in Table 2

Job no.	Sequence no.	Operation	Work center	Machine	Tool	Fixture	Labor No./wc	Cycle time (Sec./item x Lot Size)	Component/Material (Quantity)	Product (Lot size)	Component job no.
20	50	Vibration motor assembly	WC-Avm	WcaulkingMcI		WsittingjigI WcaulkingheadI	1	9.25 x 50	Weight (1x50) Rubber Holder (1x50) MainbodyAssy(1x50)	Vibration Motor (1x50)	N/A N/A 19
19	40	Mainbody assembly	WC-Amb	FcaulkingMcHS AinsertingMcV		Caulking bladeII BracketholderL	1	9.25 x 50	Armature Assy (1x50) Frame Assy (1x50) Bracket Assy (1x50)	Mainbody Assy (1x50)	18 15 13
18	30	Armature assembly	WC-A aa	Sinserting and Soldering Mc.		Supporting holderI PalletII	1	5.00 x 50	Coil Assy (1x50) Shaft (1x50)	Armature Assy (1x50)	17
17	20	Coil assembly	WC-A ca			GuidingjigIII	1	5.00 x 50	Coil (1x50) Tape (1x50) Commuter (1x50)	Coil Assy (1x50)	16
16	30	Coil fabrication	WC-M c	Cwinding McV5		TrayII	1	4.50 x 50	Raw Material (1set x50)	Coil (1x50)	N/A
15	20	Frame assembly	WC-Afa	FMpressing Mc.		FholderI MholderVII	1	4.50 x 50	Magnet (1x50) Frame (1x50)	Frame Assy (1x50)	14
14	10	Frame fabrication	WC-Mf	Fstamping McI	Die	Holder01	1	5.00 x 50	Raw Material (1set x50)	Frame (1x50)	N/A
13	20	Bracket assembly	WC-A ba	Bfusing McII.	BinserterV	Bpressing jigAI	1	4.00 x 50	Bracket A (1x50) Bracket B (1x50) Terminal (1x50)	Bracket Assy (1x50)	12
	10	Bracket A fabrication	WC-M ba	BinjectionMc04	BadjustorA	BlocatorA	1	5.25 x 50		Bracket A (1x50)	
12	10	Bracket B fabrication	WC-M bb	BinjectionMc04	BadjustorA	Bprealignment jigI	1	5.25 x 50	Raw Material (1set x50)	Bracket B (1x50)	N/A
	10	Terminal fabrication	WC-M t	Tcutting McI	DieII	TholderIII	1	5.00 x 50		Terminal (1x50)	